Towards Complete Methods for Automatic Complexity and Termination Analysis of (Probabilistic) Programs

Marcel Hark

Towards Complete Methods for Automatic Complexity and Termination Analysis of (Probabilistic) Programs

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften
der RWTH Aachen University zur Erlangung des akademischen
Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Marcel Hark, Master of Science

aus

Düsseldorf

Berichter: Prof. Dr. Jürgen Giesl

Prof. Dr. Laura Kovács

Tag der mündlichen Prüfung: 18.06.2021

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.

Marcel Hark. *Towards Complete Methods for Automatic Complexity and Termination Analysis of (Probabilistic) Programs.* June 22, 2021

Abstract

The increasing importance of computer programs in our everyday life has led to more and more complex software systems. To prove correctness of such a system, formal verification is the standard methodology. Two of the most important properties of a program are its termination behavior and its efficiency.

Moreover, in recent years *randomization* in programming has gained a lot of interest. For example, to model non-deterministic behavior in real-world applications, probabilistic concepts have proved very successful.

In this thesis, we investigate formal verification of programs which also feature assignments via discrete probability distributions. In particular, we are interested in proving (non-)termination and inferring bounds on the (expected) worst-case runtime of such programs.

In general, formal verification of programs is undecidable. Still, whenever possible, *complete* approaches for verifying certain properties on (sub-)classes of programs are preferable to incomplete ones since they *always* yield definite results, i.e., either a proof or a counterexample. Hence, we characterize sub-classes of programs for which we can present complete approaches for analyzing termination and runtime complexity.

To analyze systems arising from real-world applications, formal verification has to proceed *automatically*. Thus, we also investigate the automation of our results.

The contributions of this thesis are as follows:

(i) We characterize classes of non-probabilistic programs where both termination on a *single* input as well as *all* inputs are decidable, witnesses for non-termination are recursively enumerable, and upper bounds on the runtime can always be computed.

(ii) We connect the semantics of probabilistic programs that is defined via so-called expectation transformers to the semantics defined via so-called stochastic processes. This leads to the first inductive proof rules for lower bounds on expected outcomes and expected runtimes of probabilistic programs.

(iii) We present a class of probabilistic programs where not only probabilistic termination is decidable and both upper and lower bounds on the expected runtime can easily be inferred but also a closed form for the *exact* expected runtime can always be computed.

(iv) We develop an approach for deriving upper bounds on the expected runtime of general probabilistic programs. To this end, we use non-probabilistic runtime and size bounds as well as an alternating computation of *expected* runtime and size bounds. Our experiments demonstrate the performance and the applicability of this approach.

Zusammenfassung

Die steigende Zahl von Computerprogrammen in unserem Alltag hat zu immer umfangreicheren Softwaresystemen geführt. Die klassische Herangehensweise, um die Korrektheit solcher Systeme zu garantieren, ist formale Verifikation. Zwei der wichtigsten Eigenschaften von Programmen sind Terminierung und Effizienz.

Außerdem wird in den letzten Jahren *Randomisierung* in der Programmierung immer wichtiger. So haben probabilistische Konzepte sich als sehr erfolgreich erwiesen, um Systeme aus praxisnahen Anwendungen zu modellieren.

Diese Dissertation untersucht die formale Verifikation von Programmen, welche auch randomisierte Zuweisungen durch diskrete Zufallsverteilungen unterstützen. Hierbei sind wir besonders daran interessiert, Terminierung bzw. Nichtterminierung zu beweisen und Schranken für die (erwartete) Laufzeit zu berechnen.

Verifikation von Programmen ist im Allgemeinen unentscheidbar. Dennoch sind wann immer möglich *vollständige* Ansätze zur Verifikation bestimmter Eigenschaften von (Teil-)Klassen von Programmen unvollständigen vorzuziehen, da sie *immer* ein eindeutiges Ergebnis liefern, also entweder einen Beweis oder ein Gegenbeispiel. Ein wichtiges Ziel dieser Arbeit ist daher die Charakterisierung von Teilklassen von Programmen, für die vollständige Techniken zur Analyse von Terminierung und Laufzeitkomplexität existieren.

Um Systeme zu analysieren, die aus praxisnahen Anwendungen hervorgehen, muss formale Verifikation *automatisch* ablaufen. Daher studieren wir auch die Automatisierung der hier vorgestellten Techniken.

Die Beiträge dieser Arbeit sind im Folgenden aufgelistet.

(i) Wir beschreiben Klassen nicht-probabilistischer Programme, für die sowohl Terminierung auf einer gegebenen Eingabe als auch Terminierung auf allen Eingaben entscheidbar sind, nichtterminierende Eingaben rekursiv aufgezählt und obere Schranken für die Laufzeit immer berechnet werden können.

(ii) Wir verknüpfen die Semantik probabilistischer Programme, die durch sogenannte Erwartungs-Kalküle gegeben ist, mit jener, die durch sogenannte stochastische Prozesse definiert ist. So erhalten wir die ersten induktiven Beweisregeln zur Bestimmung unterer Schranken für die erwarteten Ausgaben bzw. Laufzeiten probabilistischer Programme.

(iii) Wir präsentieren eine Klasse probabilistischer Programme, für welche nicht nur Terminierung entscheidbar ist und sowohl obere als auch untere Schranken für die erwartete Laufzeit leicht hergeleitet werden können, sondern auch eine geschlossene Form für die *exakte* erwartete Laufzeit immer berechnet werden kann.

(iv) Wir entwickeln einen Ansatz zur Bestimmung oberer Schranken für die erwartete Laufzeit allgemeiner probabilistischer Programme. Hierfür benutzen wir sowohl Schranken für die klassische Laufzeit bzw. Größe von Variablenwerten als auch eine abwechselnde Berechnung von Schranken für die *erwartete* Laufzeit bzw. Größe. Unsere experimentelle Auswertung zeigt die große Leistungsfähigkeit dieses Ansatzes.

Contents

I Preface **1**

1 Introduction **5**
1.1 Motivation 5
1.2 Structure of This Thesis 7

2 Contributions and Publications **9**
2.1 Part II and [FHG20; HFG20] 10
2.1.1 Content and Connection to [FHG20; HFG20] 10
2.1.2 Contribution to [FHG20; HFG20] 11
2.2 Part III and [GGH19a; Har+20; MHG21b] 12
2.2.1 Content and Connection to [GGH19a; Har+20; MHG21b] . . 12
2.2.2 Contribution to [GGH19a; Har+20; MHG21b] 13

II Polynomial Loops **15**

3 Introduction **17**
3.1 Motivation 17
3.2 Structure of This Part 19

4 Preliminaries **21**

5 Reducing Termination of *twn*-Loops to Termination of *tnn*-Loops **27**

6 Reducing Termination of *tnn*-Loops to $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathrm{A})$ **31**

7 The Halting Problem **39**

8 Runtime Bounds **47**
8.1 Computing Concrete Runtime Bounds . . . 47
8.2 Computing Asymptotic Runtime Bounds . . . 55
8.3 Linearizing *tnn*-Loops . . . 58

9 Transformation to Triangular Weakly Non-Linear Form **69**
9.1 Transforming Loops . . . 69
9.2 Finding Automorphisms to Transform Loops into *twn*-Form . . . 75
9.3 Analyzing *twn*-transformable loops . . . 78
9.3.1 Deciding Termination . . . 79
9.3.2 Deciding the Halting Problem . . . 80
9.3.3 Computing Runtime Bounds . . . 81

10 Complexity of Deciding Termination **85**
10.1 Linear-Update Loops . . . 86
10.2 Arbitrary *twn*-Loops over $\mathcal{S} = \mathbb{R}_\mathrm{A}, \mathbb{R}$ . . . 88
10.3 Uniform Loops . . . 91
10.3.1 Problem Statement . . . 91
10.3.2 Hierarchical Expressions . . . 94
10.3.3 Hierarchical Partitions . . . 99
10.3.4 Sign Conditions . . . 103
10.3.5 The Algorithm . . . 113
10.4 Termination of Uniform Loops Over the Integers . . . 118
10.4.1 Removing the Zero-Constraints . . . 121
10.4.2 Handling the Inequations . . . 125

11 Related Work **129**
11.1 Work Related to Ch. 6 and 9 . . . 129
11.2 Work Related to Ch. 7 and 8 . . . 131
11.3 Work Related to Ch. 10 . . . 133
11.4 Further Reading . . . 133

12 Conclusion and Future Work **135**

III Probabilistic Programs **137**

13 Introduction **139**
13.1 Motivation . . . 139
13.2 Structure of This Part . . . 142

14 Syntax and Semantics of Probabilistic Programs **145**
14.1 Syntax of Probabilistic Programs via pGCL . . . 145
14.2 An Operational Semantics for pGCL-Programs . . . 149
14.3 Denotational Semantics for pGCL-Programs . . . 164
14.3.1 The Weakest Preexpectation Transformer wp . . . 164
14.3.2 The Expected Runtime Transformer ert . . . 169
14.4 Denotational Semantics for Probabilistic Loops via Stochastic Processes 174
14.4.1 Canonical Probability Space . . . 175
14.4.2 Canonical Stochastic Process . . . 181

15 Lower Bounds in Probabilistic Program Verification **185**
15.1 Bounds on Weakest Preexpectations . . . 185
15.1.1 Upper Bounds . . . 186
15.1.2 Lower Bounds . . . 188
15.1.3 Problem Statement . . . 191
15.1.4 Uniform Integrability . . . 191
15.2 Expectations and the Stopped Canonical Process . . . 193
15.3 Inductive Proof Rules for Lower Bounds on Weakest Preexpectations 195
15.3.1 Uniform Integrability . . . 196
15.3.2 The Optional Stopping Theorem of Weakest Preexpectations 197
15.4 Lower Bound Rules by McIver and Morgan . . . 208
15.5 Upper Bounds and the Lemma of Fatou . . . 211
15.6 Lower Bounds on the Expected Runtime . . . 213
15.6.1 Case Study: Coupon Collector's Problem . . . 214
15.6.2 Case Study: Refuting PAST . . . 219
15.7 Discussion of Non-Determinism . . . 221

16 Computing Exact Expected Looping Times of *cpa*-Loops **227**
16.1 Probabilistic Addition Loops . . . 227
16.1.1 One-Dimensional *pa*-Loops . . . 231
16.2 Expected Looping Time of *pa*-Loops with Direct Termination . . . 238
16.3 Expected Looping Times of *cpa*-Loops . . . 240
16.3.1 Reduction to Random Walk Loops . . . 240

16.3.2 Deciding Termination . . . 243
16.3.3 Computing Asymptotic Expected Looping Times . . . 244
16.4 Computing Exact Expected Looping Times . . . 245
16.4.1 Finding All Solutions of the Recurrence Equation . . . 247
16.4.2 Finding the Smallest Solution of the Recurrence Equation . . . 249
16.4.3 Implementation . . . 255

17 Upper Bounds via Probabilistic Transition Systems 257
17.1 Probabilistic Transition Systems . . . 258
17.1.1 Syntax . . . 258
17.1.2 Semantics . . . 261
17.1.3 Comparison of PITSs and Probabilistic Programs . . . 265
17.2 Complexity Bounds . . . 267
17.2.1 Runtime and Size Bounds . . . 268
17.2.2 Expected Runtime and Size Bounds . . . 269
17.3 Computing Expected Runtime Bounds . . . 273
17.3.1 Probabilistic Linear Ranking Functions . . . 273
17.3.2 Inferring Expected Runtime Bounds . . . 275
17.4 Computing Expected Size Bounds . . . 281
17.4.1 Local Change Bounds and the General Result Variable Graph 281
17.4.2 Inferring Expected Size Bounds . . . 284
17.5 Evaluation . . . 287
17.5.1 Implementation . . . 288
17.5.2 Experiments . . . 289

18 Related Work 295
18.1 Work Related to Ch. 14 and 15 . . . 295
18.2 Work Related to Ch. 16 . . . 297
18.3 Work Related to Ch. 17 . . . 299

19 Conclusion and Future Work 301

IV Postface 305

20 Outlook 307

Glossary 309

Index 313

Bibliography **317**

A Details on Probability Theory **337**
A.1 σ-Fields . . . 337
A.2 Probability Spaces . . . 338
A.3 Integrals of Arbitrary Measures . . . 339
A.4 Random Variables . . . 341
A.5 Uniform Integrability . . . 342
A.6 Conditional Expected Values . . . 343
A.7 Martingale Theory . . . 346

B Missing Proofs **349**
B.1 Proof of Thm. 5.1.2 . . . 349
B.2 Proof of Lemma 6.1.4 . . . 350
B.3 Proof of Equation (6.1.6) . . . 350
B.4 Proof of Equation (6.1.7) . . . 351
B.5 Proof of Thm. 6.1.9 . . . 351
B.6 Proof of Thm. 6.1.10 . . . 353
B.7 Proof of Cor. 6.1.11 . . . 353
B.8 Proof of Lemma 6.1.12 . . . 353
B.9 Proof of Lemma 8.1.5 . . . 353
B.10 Proof of Lemma 9.1.6 . . . 354
B.11 Proof of Lemma 9.1.7 . . . 355
B.12 Proof of Thm. 9.1.10 . . . 355
B.13 Proof of Lemma 9.1.12 . . . 356
B.14 Proof of Lemma 9.1.14 . . . 357
B.15 Proof of Thm. 9.1.16 . . . 357
B.16 Proof of Lemma 9.2.2 . . . 358
B.17 Proof of Thm. 9.2.3 . . . 359
B.18 Proof of Lemma 10.3.5 . . . 359
B.19 Proof of Lemma 10.4.6 . . . 360
B.20 Proof of Lemma 14.4.23 . . . 361
B.21 Proof of Thm. 14.4.25 . . . 361
B.22 Proof of Cor. 14.4.27 . . . 363
B.23 Proof of Lemma 15.2.3 . . . 364
B.24 Proof of Thm. 15.2.4 . . . 365
B.25 Proof of Cor. 15.3.3 . . . 365
B.26 Proof of Lemma 15.3.4 . . . 366
B.27 Proof of Thm. 15.3.10 . . . 366

B.28 Proof of Thm. 15.3.11 . . . 367
B.29 Proof of Thm. 15.4.1 . . . 371
B.30 Proof of Thm. 15.6.1 . . . 374
B.31 Proof of Thm. 15.7.3 . . . 376
B.32 Proof of Lemma 15.7.4 . . . 379
B.33 Proof of Thm. 15.7.8 . . . 379
B.34 Proof of Thm. 15.7.10 . . . 381
B.35 Proof of Cor. 16.1.5 . . . 383
B.36 Proof of Thm. 16.2.1 . . . 383
B.37 Proof of Thm. 16.3.4 . . . 384
B.38 Proof of Cor. 16.3.9 . . . 387
B.39 Proof of Thm. 16.3.10 . . . 388
B.40 Proof of Lemma 16.4.7 . . . 390
B.41 Proof of Lemma 16.4.9 . . . 392
B.42 Proof of Thm. 16.4.10 . . . 394
B.43 Proof of Cor. 16.4.12 . . . 394
B.44 Proof of Thm. 17.2.10 . . . 395
B.45 Proof of Thm. 17.3.8 . . . 396
B.46 Proof of Lemma 17.3.12 . . . 429
B.47 Proof of Thm. 17.4.3 . . . 430
B.48 Proof of Thm. 17.4.7 . . . 430
B.49 Proof of Thm. 17.4.10 . . . 433

Part I

Preface

> Gratitude is one of the least articulate of the emotions, especially when it is deep.
>
> Felix Frankfurter

Acknowledgments

First of all, I would like to thank my supervisor Jürgen Giesl. When I applied for a PhD position in his group in the beginning of 2017, he hired me although he knew me only vaguely from some lectures. In my research, he provided guidance whenever necessary but also gave me unlimited freedom in the choice of my research projects and topics even if I was responsible for a sharp rise in rejected drafts in our group.

During all those years, I enjoyed working at Research Group Computer Science 2, Chair 2, and UnRAVeL which was to a large extent due to my colleagues, e.g., Kevin Batz, Stefan Dollase, Rebecca Haehn, David Keller, Lutz Klinkenberg, Gereon Kremer, Peter Lindner, Christoph Matheja, Stefan Schupp, and Tobias Winkler, to mention only some of them, with whom I had many fruitful and joyful discussions on (un)related research topics, and whom I could ask for advise on software development and even server maintenance.

With my office mates Daniel Cloerkes, Jera Hensel, and Dominik Meier I always had good-natured companions during the day with whom I enjoyed uncountable conversations during (lunch) breaks. I hope I will never forget the many discussions on random topics I had with Christian Hensel. I would also like to thank Christian for providing the LaTeX-template of his thesis which saved me a lot of time when I started writing my thesis.

During my PhD I was responsible for the development of a new version of the tool KoAT. Without my students Fabian Böller, Tom Küspert, Nils Lommen, and Fabian Meyer, who significantly contributed to this project with their Bachelor's and Master's theses and did most of the implementation, this project would have never been in a state comparable to its current one.

I would also like to thank my co-authors Florian Frohn (who also read parts of this thesis and provided helpful feedback), Peter Giesl, Benjamin Kaminski, and Joost-Pieter Katoen for the many long discussions and their advise during our projects. Furthermore, I am grateful to Laura Kovács, one of the most renowned researchers from our community, for agreeing to be my second supervisor.

Doubtless, without my family, in particular my parents Monika and Rüdiger and my sister Nadja, my PhD would not have been possible. During my entire life and especially during my years at RWTH Aachen they supported me in every aspect with

all their love and care. Moreover, my father also designed the amazing cover of this thesis.

Finally, I am deeply thankful to my wife Carolin for all her love and support. She had to put up with me when things did not work out so well and always cheered me up with tireless patience. I doubt that I would have completed my PhD without her.

Chapter 1

Introduction

1.1 Motivation

Formal Verification

We are living in the age of information technology and computer science. Almost every part of our life is connected with software: shopping, banking, and social networking are only some examples. As of today, five out of the ten biggest companies in the world are software companies and three out of the remaining five obtain at least huge revenues from their software departments [Com21]. However, as the use of software systems has increased so has their complexity. Unfortunately, it is almost impossible to keep such systems error-free. Thus, the list of famous bugs and errors in software which have led not only to the loss of billions of dollars but also to the loss of lives is long [Wik21].

The two most widespread ways to tackle this problem are as follows. The first one is to provide a variety of *test cases* where the desired outcome is known, e.g., inputs which are notorious for causing exceptional behavior. This is especially useful for finding errors. However, it is *infeasible* to test all possible inputs and thus, testing is usually unsuitable for proving the *absence* of errors.

The second one is *formal verification*. The basic idea is to provide a mathematical proof that (an abstraction of) the software under consideration satisfies a certain property, e.g., that the output always has a certain form w.r.t. the input or that no execution of the software takes forever. Nevertheless, if such a proof fails, a witness for the failed attempt should be generated whenever possible, i.e., a counterexample to the desired property [God05].

Among the most important properties of software are termination and runtime

complexity. For example, non-termination leads to vulnerability to attacks since an adversary can force the system to run forever by providing a certain input. However, in practical applications non-termination is indistinguishable from enormous worst-case runtimes. For example, if a program always terminates but can (unintentionally) take up to several years to finish its computation, it is as vulnerable to attacks as a non-terminating program. Furthermore, inefficient software might lead to a waste of energy and thus a waste of resources. So analyzing termination and runtime complexity of an implementation with a particular focus on inferring upper *and* lower bounds on its (worst-case) runtime is of great interest for proving correctness of software or detecting bugs in it: while small upper bounds *exclude* inefficiency, large lower bounds expose bugs.

Still, such an analysis should not need to be performed manually but proceed *automatically* – most desirably by a simple push-button technique – to be able to analyze systems arising from real-world applications. For example, Facebook already uses (a restricted) static analysis to detect certain types of inefficient code to give feedback to the developers fully automatically [Çiç+20]. Moreover, the automatic analysis of the resource consumption of smart contracts (certain kinds of digital protocols which are used in, e.g., cryptocurrencies) has recently attracted a lot of attention [Alb+19b; Alb+20; Das+21; Pér+20].

In recent years, *randomization* in programming has gained a lot of interest. Primality testing of natural numbers, a crucial ingredient of public key cryptography, and sorting data are only two examples where the introduction of randomness, e.g., the use of random number generators, has led to a vast increase in efficiency for solving these problems, see [MU05] for a survey on this topic. While the gain of randomization in algorithms has been well-known for many years, the recent increase of interest stems from so-called cyber-physical systems, i.e., systems which model real-world scenarios. Here, non-deterministic behavior of the system can often be simulated best by *probabilistic* constructs. Moreover, the rise of machine learning has led to a vast interest in probabilistic programming [Gor+14].

In this thesis, we study formal verification of programs featuring not only standard deterministic constructs but also random assignments and non-determinism. In particular, we investigate termination and runtime analysis with a focus on automation and completeness.

Practical Applicability

Indeed, many of the results presented in this thesis are of a theoretical nature. We will not analyze real-world systems or high-level programming languages such as C++ or Java directly. In contrast, we focus on so-called *backend-languages*. These

are simple mathematical formalisms which are in general Turing complete and thus allow for modeling arbitrary programs. For example, the backend-language of integer transition systems – tail-recursive programs whose only data types are Boolean values and integers – have been used successfully to analyze termination of LLVM programs [Hen+18] and the runtime complexity of Java programs (which indeed contain more sophisticated data structures) fully automatically [FG17].

We consider theoretical foundations of termination and complexity analysis of (probabilistic) programs. To this end, we use different semantics which involve deep results from mathematics, but lead to conceptually simple approaches for analyzing these properties. In this way, our results contribute to the automatic runtime and termination analysis of real-world systems.

Completeness

Soundness is the fundamental requirement for verification techniques. However, completeness, meaning that an approach *always* yields a *definite* answer, is a double-edged sword in verification.

Of course, verification of non-trivial characteristics is undecidable in general. The most famous example is termination on a given input also known as the halting problem [Tur37]. This means that in general, the *automation* of a complete approach for verification of such characteristics has to be *incomplete*.

However, undecidability just means that there is no algorithm for solving *every* instance of the problem, i.e., there can still be restricted but non-trivial classes of programs where an *automatic and complete* verification of certain characteristics is indeed possible. The characterization of such classes is of great interest. The reason is that they may appear as sub-programs of larger programs and thus, such techniques can be incorporated into a modular analysis of programs.

In this thesis, we present classes of non-probabilistic programs where both termination on a given input as well as termination on all inputs are decidable and upper bounds on the runtime can always be computed. Moreover, we characterize a class of probabilistic programs where not only termination on all inputs is decidable and upper and lower bounds on its runtime can easily be inferred but also a *closed form* of its exact expected runtime is computable.

1.2 Structure of This Thesis

First of all, Ch. 2 discusses the contributions contained in this thesis and their connection to prior publications of the author.

In Part II, we investigate so-called polynomial loops. These are non-probabilistic loops whose condition is built from polynomial inequations and in whose body the variables are updated by polynomials (Ch. 4). For different classes of these loops, we study complete approaches for termination and complexity analysis. We present new decidability results for termination on all inputs (Ch. 5 and 6) as well as the halting problem (Ch. 7). Moreover, we develop a complete technique for inferring upper bounds on the runtime (Ch. 8).

Part III is devoted to probabilistic programs, i.e., programs which also feature probabilistic concepts, e.g., random assignments. We introduce different semantics for these programs and present new results on the connections between those semantics (Ch. 14). We use these connections to develop new inductive proof rules for the inference of lower bounds on expected outcomes and expected runtimes (Ch. 15). In Ch. 16, we discuss decidability of termination for a certain class of probabilistic programs. We show that for this class upper and lower bounds on the expected runtime can always be inferred, and the expected runtime can be computed exactly. Finally, we present an approach for computing upper bounds on the expected runtime of general probabilistic programs (Ch. 17).

Part II and Part III are independent of each other. Nevertheless, both contain classes of programs where termination is decidable. We compare these classes briefly at the end of Ch. 16.

Parts II and III have separate introductions (Ch. 3 and Ch. 13, respectively). Here, we motivate the contributions contained in the corresponding part and discuss its structure. Furthermore, both parts contain separate overviews on the state-of-art and works related to the results presented there (Ch. 11 and Ch. 18, respectively) as well as separate conclusions (Ch. 12 and Ch. 19, respectively).

This thesis is concluded with an outlook on future work in Ch. 20. In contrast to the specialized conclusions in Parts II and III, we recapitulate our results in terms of the high-level motivation given in Sect. 1.1.

Finally, the appendix provides details on probability theory (App. A) which are used in the proofs of Part III. Moreover it contains all proofs which have been omitted in Parts II and III in App. B.

Chapter 2

Contributions and Publications

I now discuss the contributions of my thesis and their connection to my already published papers which I list below. I especially highlight which yet unpublished results are contained in this thesis. Moreover, I comment on my contributions to these papers. I would like to point out that all of my publications are the result of an intensive cooperation between all of the listed authors.

[FHG20] F. Frohn, M. Hark, and J. Giesl. "Termination of Polynomial Loops". In: *Proc. SAS*. Lecture Notes in Computer Science Vol. 12389. 2020, pp. 89–112.

[GGH19a] J. Giesl, P. Giesl, and M. Hark. "Computing Expected Runtimes for Constant Probability Programs". In: *Proc. CADE*. Lecture Notes in Computer Science Vol. 11716. 2019, pp. 269–286.

[Har+20] M. Hark, B. L. Kaminski, J. Giesl, and J.-P. Katoen. "Aiming Low is Harder: Induction for Lower Bounds in Probabilistic Program Verification". In: *Proc. ACM Program. Lang.* 4.POPL (2020), 37:1–37:28.

[HFG20] M. Hark, F. Frohn, and J. Giesl. "Polynomial Loops: Beyond Termination". In: *Proc. LPAR*. EPiC Series in Computing Vol. 73. 2020, pp. 279–297.

[MHG21b] F. Meyer, M. Hark, and J. Giesl. "Inferring Expected Runtimes of Probabilistic Integer Programs Using Expected Sizes". In: *Proc. TACAS (1)*. Lecture Notes in Computer Science Vol. 12651. 2021, pp. 250–269.

2.1 Part II and [FHG20; HFG20]

2.1.1 Content and Connection to [FHG20; HFG20]

In Part II, we discuss termination of a class of polynomial loops, so-called *twn*-loops. In these loops, the update is mildly restricted to be "triangular" and "weakly non-linear" (see Ch. 4 for a formal definition of these terms). While in automated termination analysis one is concerned with sound approaches to prove or disprove termination, we are interested in a reduction of termination to a suitable theory which is both sound *and* complete. To this end, we make use of the computability of closed forms for the iterated update (Ch. 5). In this way, we characterize a novel class of loops where termination is decidable if the variables range over the real or the real algebraic numbers and non-termination is semi-decidable if the variables range over the rationals or the integers (Ch. 6). This result has been published in [FHG20].

In case we can disprove termination, an input on which the loop does not terminate is of high interest. Therefore, we prove decidability of the halting problem for *twn*-loops over the real algebraic numbers (and thus also over all its subrings, e.g., the integers and the rationals). In this way, we obtain a method to enumerate all witnesses for non-termination recursively (Ch. 7).

In case we can prove termination, such a proof is only of limited use without any bound on the runtime of the loop, i.e., a bound on the number of loop iterations until termination is reached depending on the *size* of the input. Hence, we present a technique to compute an over-approximation of the runtime of a *twn*-loop over the integers. In contrast to automated complexity analysis, where one is interested in developing sound approaches to analyze the runtime of programs, our approach *always* yields a bound on the runtime of a *twn*-loop over the integers. To the best of our knowledge, this is the first positive result regarding the computability of upper bounds on the runtime complexity of a class of polynomial integer loops (Ch. 8). This approach and the result on the decidability of the halting problem have been published in [HFG20].

In addition to [HFG20], in Sect. 8.2 we present the yet unpublished result on how to compute the *asymptotic* runtime of a *twn*-loop. While the technique for computing upper bounds on the runtime from [HFG20] relied on computing a closed form for the iterated update, our new technique for computing an asymptotic upper bound only uses elementary arithmetic and syntactic properties of the loop. This is achieved by a detailed analysis of the so-called *linearization* of *twn*-loops as presented in [OBP16a] (Sect. 8.3). However, we also present insights on linearization which exceed the ones from [OBP16a]. These results enable a new complexity result for deciding termination of *twn*-loops in Sect. 10.2.

Nevertheless, *twn*-loops are a restricted class of polynomial loops. Therefore, we present a transformation which allows us to transform non-*twn* loops into *twn*-form (Ch. 9). Of course, we also discuss how to determine whether a loop can be transformed into *twn*-form. In this way, we can generalize the results presented so far to a broader class of polynomial loops, so-called *twn*-transformable loops (Sect. 9.3). The results of Sect. 9.1 and 9.2 are published in [FHG20], whereas the results of Sect. 9.3 are published partly in [FHG20] and partly in [HFG20].

Finally, we also study the complexity of deciding termination (Ch. 10). As published in [FHG20], in Sect. 10.1 we show that deciding termination of linear loops with real spectrum over the integers, the rationals, the real algebraic numbers, and the reals is Co-NP-complete and deciding termination of linear-update loops with real spectrum over the real algebraic numbers and the reals is $\forall\mathbb{R}$-complete, where $\forall\mathbb{R}$ is the complexity class of decidability problems which can be reduced to validity of a universally quantified formula of polynomial inequations over the reals. While a restricted version of our result on the complexity of deciding termination for *twn*-loops (Thm. 10.2.2) which only considers deciding termination in case of a *bounded* number of variables is also stated in [FHG20, Thm. 45], the version presented here considers the general case of deciding termination where the number of variables is *not* restricted. Note that both Thm. 10.2.2 and [FHG20, Thm. 45] consider arbitrary *twn*-loops, i.e., they also cover loops with non-linear updates.

In addition, we also classify a class of polynomial loops in which deciding termination can be interpreted as a parameterized decision problem and is decidable in polynomial time when fixing such a parameter (Sect. 10.3 and 10.4). This result is based on discussions with Florian Frohn and Gereon Kremer and unpublished so far.

2.1.2 Contribution to [FHG20; HFG20]

I contributed to [FHG20] by generalizing an earlier version of our (semi-)decision procedure in Ch. 6 to *twn*-loops which can also contain disjunctions in their condition. Furthermore, I developed the transformation of polynomial loops using polynomial automorphisms and the corresponding extension of the (semi-)decision procedure from Ch. 6 to so-called update invariant and $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathrm{A})$-definable sets (see Sect. 9.3.1). Additionally, I was significantly involved in inferring the tight complexity results on deciding termination of linear(-update) loops which make both use of my transformation of polynomial loops. Moreover, I proved the complexity result on deciding termination for *twn*-loops with a bounded number of variables ([FHG20, Thm. 45]).

In [HFG20], I developed the technique to compute an upper bound on the runtime of a *twn*-loop and proved its correctness. The ideas of my proof gave rise to the general framework of *stabilization* thresholds. This framework allowed for a combination

of inferring upper runtime bounds and deciding the halting problem in [HFG20]. Furthermore, this combination simplified our decision procedure for the halting problem substantially. In addition, I also generalized our decision procedure for the halting problem and the computability result on upper runtime bounds via the transformation from [FHG20] to certain non-*twn* loops.

In both [FHG20; HFG20], I was heavily involved in structuring and writing preliminary versions of the papers and illustrating the main results by means of running examples. I contributed significantly to polishing and proofreading both articles until they were ready for publication.

2.2 Part III and [GGH19a; Har+20; MHG21b]

2.2.1 Content and Connection to [GGH19a; Har+20; MHG21b]

In Part III, we consider the formal verification of probabilistic programs. In contrast to a non-probabilistic program, a probabilistic program computes a *(sub)distribution* over possible outputs for any input. Thus, instead of "the outcome", "termination", or "the runtime" one is interested in the *expected* outcome, the *probability* of termination, and the *expected* runtime.

In Ch. 14 and 15, we connect two denotational semantics of probabilistic loops, one based on so-called expectation transformers and one based on so-called stochastic processes. In this way, we obtain the first inductive proof rules for verifying lower bounds on expected outcomes and expected runtimes of probabilistic loops. These results have been published in [Har+20]. In addition to [Har+20], we explain how we can use these rules to show that the expected runtime is infinite which is based on discussions with Benjamin Lucien Kaminski (Sect. 15.6.2). Additionally, we compare our rules to recent results in related work (Ex. 15.3.14). Moreover, in Sect. 15.7 we add a thorough discussion on the presence of (finite) non-determinism: we adapt some of the results from [Har+20] (Thm. 14.4.25 and 15.2.4 and Cor. 14.4.26 to 14.4.28) and explain for the remaining results why such an adaption is beyond the scope of this thesis. Nevertheless, Sect. 15.7 shows that our main results from Ch. 15 remain valid in the non-deterministic setting provided a conjecture holds whose justification we explain in detail. The results of Sect. 15.6.2 and Sect. 15.7 as well as the generalized versions of Thm. 14.4.25 and 15.2.4 and Cor. 14.4.26 to 14.4.28 presented here are unpublished so far.

In Ch. 16, we study so-called *pa*-loops. These loops are probabilistic loops where we only allow the addition of constants to the variables in the body of the loop. We develop a simple procedure for deciding termination of *cpa*-loops (*pa*-loops in which all occurring probabilities are independent of the variables in the loop, i.e., the

probabilities are constant) and show that their expected runtime is either infinite or asymptotically linear by providing easy-to-derive lower and upper bounds. Using these insights, we develop an algorithm for computing a closed form for the exact expected runtime of *cpa*-loops. This approach has been published in [GGH19a] and is implemented in the automatic complexity analysis tool KoAT [Bro+16; KoAT].

In contrast to [GGH19a], we handle *pa*-loops via the semantics of probabilistic loops from Ch. 14 and 15. In this way, we avoid the connection to random walk theory used in [GGH19a] and thus obtain much more concise proofs for the decision procedure and the bounds on the expected runtime of *cpa*-loops provided in Sect. 16.3.

In addition to [GGH19a], we also discuss sufficient criteria for termination of one-dimensional *pa*-loops in Sect. 16.1. Here, we also discuss results presented in [Küs19], the Bachelor's thesis of my student Tom Küspert which I have supervised. These criteria then boil down to the decision procedure for *cpa*-loops given in Sect. 16.3 and thus we embed this procedure into a more general concept. So in Ch. 16 we also study a class of loops where termination is decidable. Nevertheless, this class is *not* the probabilistic counterpart of *twn*-loops which we study in Part II but only of a restricted sub-class of *twn*-loops. We comment on this at the end of Ch. 16.

In Ch. 17, we consider so-called probabilistic transition systems. Given such a system, we present an (incomplete) approach for computing upper bounds on their expected runtimes. To this end, we use existing concepts of non-probabilistic runtime and size, and develop a notion of *expected* size. This approach is also implemented in the automatic complexity analysis tool KoAT [Bro+16; KoAT]. In Sect. 17.5, we demonstrate the power of this approach in extensive experiments. The results of Ch. 17 are published in [MHG21b].

Probabilistic transition systems provide a different syntax for probabilistic programs than the one used in Ch. 14 to 16. Thus, in addition to [MHG21b], we include a high-level comparison of the concepts we introduce for probabilistic transition systems to similar (or seemingly similar) concepts from Ch. 14 to 16. In this way, we justify the introduction of probabilistic transition systems and link Ch. 17 to Ch. 14 to 16, i.e., we illustrate how any probabilistic program which is given via the syntax in Ch. 14 to 16 can be transformed into an equivalent probabilistic transition system. Thus, we can indeed use the approach presented in Ch. 17 to infer upper bounds on the expected runtime of general probabilistic programs.

2.2.2 Contribution to [GGH19a; Har+20; MHG21b]

I contributed to [Har+20] by developing and proving the connection between expectation transformers and stochastic processes. I formulated and proved the first version of our Optional Stopping Theorem for expected outcomes of probabilistic

programs, one of the main results in [Har+20] (see Sect. 15.3), and was highly involved in refining it into its form presented there. Furthermore, I generalized the existing rules for lower bounds on *bounded* expected outcomes in Sect. 15.4 to a *sound and complete* characterization of lower bounds by using our connection of expectation transformers to stochastic processes and non-trivial results from martingale theory. I also developed the explanation for the simplicity of deriving upper bounds via Fatou's Lemma in Sect. 15.5 which can therefore be interpreted as the probability-theory-counterpart of so-called Park induction. Finally, I used our proof rule for lower bounds on expected outcomes to derive and prove our rule for lower bounding expected runtimes of probabilistic programs (see Sect. 15.6).

In [GGH19a], I connected the expected runtime of *cpa*-loops to the smallest solutions for recurrence relations by using existing results on expectation transformers. Additionally, I developed the reduction of *cpa*-loops to random walk loops which transferred our results on random walk loops to a significantly larger class of probabilistic loops. Furthermore, I connected random walk loops to random walks from probability theory which then gave rise to the decision procedure in Sect. 16.3.2 and the bounds in Sect. 16.3.3. These results enabled the computation of the exact expected runtime in Sect. 16.4. Moreover, I coordinated the implementation of this approach in our tool KoAT and the experimental evaluation [Gie+19a].

The foundations of [MHG21b] have been laid in the Bachelor's thesis of Fabian Meyer and his project as a student assistant in our group which I have supervised both. I developed the concepts of (probabilistic) size and runtime (see Sect. 17.2.2). Moreover, I translated the notion of "local size" from non-probabilistic integer transition systems to expected local change bounds (see Sect. 17.4.1) which then enabled our methods for computing global size bounds (see Sect. 17.4.2). Furthermore, I contributed significantly to the proofs of our main results for computing expected runtime and expected size bounds by formalizing them in probability theory. Finally, I coordinated the implementation of this approach in our tool KoAT and was significantly involved in the experimental evaluation [MHG21a].

In all of [GGH19a; Har+20; MHG21b], I contributed significantly to illustrating *all* results by means of a running example. Moreover, I was heavily involved in writing the first versions of the articles, proofreading earlier versions, and polishing these papers until they were ready for publication.

Part II

Polynomial Loops

Chapter 3

Introduction

3.1 Motivation

Termination and *runtime complexity* are two of the most important properties of programs. For example, termination on all inputs is crucial for proving total correctness of programs, whereas non-termination indicates unintended behavior or bugs. Moreover, as argued in the introduction of this thesis, non-terminating programs are hardly distinguishable from highly inefficient programs. Thus, whenever termination of a program can be proven one should also be able to infer an upper bound on the runtime to exclude inefficiency. Of course, the analysis of loops is essential for reasoning about termination and runtime complexity of general programs.

In this part, we study *complete* approaches for analyzing termination and runtime complexity of certain classes of loops. A complete approach advances over one which is only sound since it always yields a definite result, e.g., if a complete approach for proving termination on all inputs fails, it has in fact refuted termination. In particular, we investigate *decidability* of termination and *computability* of upper bounds on the runtime for our classes of loops.

Since in general termination is undecidable [Her71; Tur37], such decidability results can only be obtained for restricted classes of loops. We restrict ourselves to certain classes of polynomial loops in this part. There already exist several decidability results for the termination of linear loops [BIK14; Bra06; FG19a; HOW19; Li14; Li17b; OPW15; Tiw04; Xia+11] but only few results on the decidability of termination for certain forms of non-linear loops [Li16; Li17a; NOW20; Wu+10; XZ10]. However, all of these previous works only deal with conjunctive loops besides [NOW20] which restricts its analysis to non-linear loops with conditions defining compact sets. In this

part, we regard both linear and non-linear loops with arbitrary conditions, i.e., they may also contain disjunctions and define non-compact sets. Furthermore, we study non-linear loops over the integers, the rationals, the real algebraic numbers, and the reals, whereas the existing decidability results for non-linear loops are restricted to loops over the reals.

Most techniques for proving termination of loops rely on so-called *ranking functions*. While ranking functions are complete for termination, they are in general *not* computable. Thus, techniques using ranking functions are usually restricted to (lexicographic combinations of) polynomial ranking functions. There exists many computability results for polynomial ranking functions, see, e.g., [Ali+10; BDG19; BG14; BG17; BMS05a; BMS05b; BMS05d; PR04]. Still, these ranking functions are only *sound* for proving termination so in particular they are unsuited for refuting termination. To the best of our knowledge, the only results regarding completeness of polynomial ranking functions are [Dom21; NOW20]. Here, [Dom21] deals with a sub-class of linear loops whose update-matrix has only the eigenvalues 0 and 1 and [NOW20] handles non-linear loops whose condition defines a compact set. However, both works do not address the computability of such ranking functions. In contrast to these approaches, we use the computability of so-called *closed forms* to reduce termination of a loop from our classes to validity of a certain formula. We show that this reduction is sound and complete, i.e., validity of the resulting formula proves termination, whereas invalidity of this formula proves non-termination. Moreover, we prove that this reduction is computable. By using existing results on the decidability of certain logics, we then obtain our decidability results on termination. Additionally, our classes of loops are characterized by simple syntactic restrictions so in particular we do not require that the condition of the loop defines a set with certain properties such as compactness and our classes also contain non-linear loops.

Moreover, we also use concepts from algebra to enlarge the classes of loops to which our reduction is applicable. This is done by transforming polynomial loops via so-called polynomial automorphisms in such a way that this transformation does not affect the termination behavior, i.e., the transformed loop terminates if and only if the original loop terminates.

Additionally, we also investigate the complexity of our reduction. Here, we rely on the just mentioned transformation. In this way, we obtain (tight) complexity results for deciding termination of certain classes of polynomial loops.

However, a (non-)termination proof alone is often of limited use. In particular, non-termination proofs should be accompanied by a *witness for non-termination*, i.e., a non-terminating input that can, e.g., be used for debugging. Nevertheless, most existing (semi-)decision procedures for (non-)termination do not yield such witnesses [Bra06; FG19a; HOW19; NOW20; OPW15; Tiw04; Wu+10; Xia+11; XZ10]. Thus,

we also investigate decidability of the halting problem for our classes of loops, i.e., termination on a given input. The basic idea is to compute a threshold for the given input such that if the loop has not terminated on this input within the iterations up to the threshold, it never will. In this way, we obtain a technique to compute witnesses for non-termination. Moreover, this technique is able to enumerate *all* witnesses for non-termination recursively.

Finally, many automated techniques to derive bounds on the runtime complexity of programs have been proposed [Alb+11; Alb+19a; Bro+16; Car+17; Flo16; Fro+20; HAH12; HDW17; HS15; SSH17; SZV17]. However, these techniques are usually incomplete. In contrast, we present a complete approach for computing an upper bound on the runtime, i.e., we show that for a certain class of loops upper bounds on the runtime are always computable. This approach can also handle loops which are out of reach for existing techniques for inferring upper bounds on the runtime complexity.

3.2 Structure of This Part

We investigate so-called *twn-loops* which we introduce in Ch. 4. We prove that termination of *twn*-loops can be reduced to termination of so-called *tnn*-loops (Ch. 5). In this way, we show that termination of *twn*-loops over the real algebraic numbers and the reals is decidable and that non-termination over the integers and the rationals is semi-decidable (Ch. 6). Thus, we identify classes of loops where (non-)termination is (semi-)decidable.

Moreover, we prove decidability of the *halting problem* for *twn*-loops by using so-called *stabilization thresholds* (Ch. 7). As already mentioned, this approach enables us to compute witnesses for non-termination and even to enumerate all witnesses for non-termination recursively.

In Ch. 8, we present our complete technique to derive polynomial upper bounds on the runtime complexity of *twn*-loops over the integers which is also based on stabilization thresholds. In particular, this implies that triangular *linear* loops have at most linear runtime complexity.

To increase the applicability of our approaches, we introduce an (incomplete) transformation *Tr* from non-*twn*-loops to *twn*-loops (Ch. 9). In particular, we show that it is semi-decidable whether a given loop can be transformed into *twn*-form and discuss classes of loops which can *always* be transformed into *twn*-form. We then show that our techniques presented in Ch. 6 to 8 can also be used to analyze *twn*-transformable loops.

Finally, we also analyze the complexity of deciding termination for *twn*-loops. More precisely, we prove Co-NP-completeness of termination of linear loops with real spectrum over the integers, the rationals, the real algebraic numbers, and the reals, and $\forall\mathbb{R}$-completeness of termination of linear-update loops with real spectrum over the real algebraic numbers and the reals (Ch. 10). Additionally, we present a special class of linear loops where the termination problem can be *parameterized* and has polynomial runtime when fixing this parameter, i.e., it is in XP.

Finally, we discuss related work (Ch. 11) and conclude this part with some remarks on future work (Ch. 12). All missing proofs can be found in App. B.

Chapter 4

Preliminaries

Let $\mathbb{R}_A$ denote the real algebraic numbers. In this part, we consider polynomial loops.

Definition 4.1.1 (Polynomial Loop)**.** A *(polynomial) loop* has the form in Fig. 4.1(a). Here, $\vec{x}$ is a vector of $d \geq 1$ pairwise different variables that range over a ring $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_A$ where $\leq$ denotes the subring relation. To improve readability, we use row- and column-vectors interchangeably. Moreover, $\vec{u} \in \left(\mathcal{S}[\vec{x}]\right)^d$ where $\mathcal{S}[\vec{x}]$ is the set of all polynomials over $\vec{x}$ with coefficients from $\mathcal{S}$. The condition φ is an arbitrary propositional formula over the atoms $\{p \triangleright 0 \mid p \in \mathcal{S}[\vec{x}], \triangleright \in \{\geq, >\}\}$.

Note that negation is syntactic sugar in our setting, as, e.g., $\neg(p > 0)$ is equivalent to $-p \geq 0$. So w.l.o.g. the condition φ of a loop is built from atoms, $\wedge$, and $\vee$.

while (φ) **do** {
$\quad \vec{x} \leftarrow \vec{u}$
}

(a) Standard Form

while $(x_1 + x_2^2 > 0)$ **do** {
$\quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 + x_2^2 \cdot x_3 \\ x_2 - 2 \cdot x_3^2 \\ x_3 \end{bmatrix}$
}

(b) Loop $\mathbf{L}_{ex}$

Figure 4.1: Polynomial Loops

We require $\mathcal{S} \leq \mathbb{R}_{\mathbb{A}}$ instead of $\mathcal{S} \leq \mathbb{R}$, as it is unclear how to represent transcendental numbers on computers. However, in Ch. 6 we will see that the loops considered in this thesis terminate over $\mathbb{R}$ iff they terminate over $\mathbb{R}_{\mathbb{A}}$. Thus, our results immediately carry over to loops where the variables range over $\mathbb{R}$. Hence, we sometimes also consider loops over $\mathcal{S} = \mathbb{R}$. However, even then we restrict ourselves to loops where all constants in φ and $\vec{u}$ are algebraic.

We often represent a loop by the tuple $(\varphi, \vec{u})$ of the *condition* φ and the *update* $\vec{u} = (u_1, \dots, u_d)$. Unless stated otherwise, $(\varphi, \vec{u})$ is always a loop on $\mathcal{S}^d$ using the variables $\vec{x} = (x_1, \dots, x_d)$ where $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathbb{A}}$.

Notation 4.1.2. *We use the following notions for certain classes of loops.*

- *A* linear-update loop *has the form* $(\varphi, A \cdot \vec{x} + \vec{b})$ *and it has* real spectrum *if A has real eigenvalues only.*
- *A* linear loop *is a linear-update loop where φ is linear (i.e., its atoms are only linear inequations). Here, "linear" refers to "linear polynomials", i.e., of degree at most* 1*, so it includes* affine *functions.*
- *A* conjunctive loop *is a loop $(\varphi, \vec{u})$ where φ does not contain disjunctions.*

For any entity s, $s[x/t]$ is the entity that results from s by replacing all free occurrences of x by t. Similarly, if $\vec{x} = (x_1, \dots, x_d)$ and $\vec{t} = (t_1, \dots, t_d)$, then $s[\vec{x}/\vec{t}]$ results from s by replacing all free occurrences of x_i by t_i, for each $1 \leq i \leq d$.

Any vector of polynomials $\vec{u} \in \left(\mathcal{S}[\vec{x}]\right)^d$ can also be regarded as a function $\vec{u} : \left(\mathcal{S}[\vec{x}]\right)^d \to \left(\mathcal{S}[\vec{x}]\right)^d$ where for any $\vec{p} \in \left(\mathcal{S}[\vec{x}]\right)^d$, $\vec{u}(\vec{p}) = \vec{u}[\vec{x}/\vec{p}]$ results from *applying* the polynomials $\vec{u}$ to the polynomials $\vec{p}$. In a similar way, we can also apply a formula to polynomials $\vec{p} \in \left(\mathcal{S}[\vec{x}]\right)^d$. To this end, we define $\psi(\vec{p}) = \psi[\vec{x}/\vec{p}]$ for first-order formulas ψ with free variables $\vec{x}$. As usual, function application associates to the left, i.e., $\vec{u}(\vec{b})(\vec{p})$ stands for $(\vec{u}(\vec{b}))(\vec{p})$. However, we obviously have $(\vec{u}(\vec{b}))(\vec{p}) = \vec{u}(\vec{b}(\vec{p}))$ since applying polynomials only means that one instantiates variables. This will become important in Ch. 9

Def. 4.1.3 formalizes the intuitive notion of termination for a loop $(\varphi, \vec{u})$. Here, $\vec{u}^n$ denotes the n-fold application of $\vec{u}$, i.e., $\vec{u}^0(\vec{c}) = \vec{c}$ and $\vec{u}^{n+1}(\vec{c}) = \vec{u}(\vec{u}^n(\vec{c}))$.

Definition 4.1.3 (Termination, [Bra06; Tiw04])**.** Let $(\varphi, \vec{u})$ be a loop over $\mathcal{S}$ and $\vec{e} \in \mathcal{S}^d$. If

$$\forall n \in \mathbb{N}.\ \varphi(\vec{u}^n(\vec{e})),$$

then $\vec{e}$ is a *witness for non-termination*. Otherwise, $(\varphi, \vec{u})$ *terminates on* $\vec{e}$. If $(\varphi, \vec{u})$ does not have any witnesses for non-termination in $\mathcal{S}^d$, it *terminates* over $\mathcal{S}$.

If $\vec{u}^{n_0}(\vec{e})$ is a witness for non-termination for some $n_0 \in \mathbb{N}$, $\vec{e}$ is called a *witness for eventual non-termination*. $(E)NT_{(\varphi,\vec{u})}$ denotes the set of witnesses for (eventual) non-termination of $(\varphi, \vec{u})$ and we define $T_{(\varphi,\vec{u})} = \mathcal{S}^d \setminus NT_{(\varphi,\vec{u})}$.

Note that Def. 4.1.3 introduces two different notions of termination: termination on all inputs (by means of the absence of non-terminating inputs) and termination on a fixed input $\vec{e}$. The former notion relates to the *universal halting problem* which will be studied in Ch. 6, while the latter relates to the classical *halting problem* which we will study in Ch. 7.

For any entity s, let $\mathcal{V}(s)$ be the set of all free variables that occur in s. Given an assignment $\vec{x} \leftarrow \vec{u}$, the relation $\succ_{\vec{u}} \in \mathcal{V}(\vec{u}) \times \mathcal{V}(\vec{u})$ is the transitive closure of

$$\{(x_i, x_j) \mid i, j \in \{1, \dots, d\}, i \neq j, x_j \in \mathcal{V}(u_i)\},$$

i.e., $x_i \succ_{\vec{u}} x_j$ means that x_i depends on x_j.

Definition 4.1.4 (Triangular Weakly Non-Linear Loops)**.** A loop $(\varphi, \vec{u})$

- is *triangular* if $\succ_{\vec{u}}$ is well founded.
- is *weakly non-linear* if there is no $1 \leq i \leq d$ such that x_i occurs in a non-linear monomial of u_i, i.e., $u_i = c_i \cdot x_i + pol_i$ where $c_i \in \mathcal{S}$ and $pol_i \in \mathcal{S}[\vec{x}]$ does *not* contain x_i.
- is a *twn-loop* if it is triangular and weakly non-linear.
- is *non-negative* if it is weakly non-linear and the coefficient c_i of the monomial x_i in u_i is non-negative for all $1 \leq i \leq d$.
- A *tnn-loop* is triangular and non-negative.

Note that a *tnn*-loop is by definition a special form of a *twn*-loop.

The restriction to triangular loops prohibits "cyclic dependencies" of variables (e.g., where the new values of x_1 and x_2 both depend on the old values of x_1 and x_2). For example, a loop whose body consists of the assignment $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 + x_2^2 \\ x_2 + 1 \end{bmatrix}$ is triangular since $\succ = \{(x_1, x_2)\}$ is well founded, whereas a loop with the body $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 + x_2^2 \\ x_1 + 1 \end{bmatrix}$ is not triangular. Triangularity is used to compute a *closed form* (see

Def. 4.1.5) for the n-fold application of the update $\vec{u}$ by handling one variable after the other. From a practical point of view, the restriction to triangular loops seems quite natural. For example, in [Fro20], 1511 polynomial loops were extracted from the *Termination Problems Data Base* [TPDB], the benchmark collection which is used at the annual *Termination and Complexity Competition* [Gie+19b], and only 26 of them were non-triangular.

A loop with the body $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 + x_2^2 \\ x_2 + 1 \end{bmatrix}$ is weakly non-linear, whereas a loop with the body $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 \cdot x_2 \\ x_2 + 1 \end{bmatrix}$ is not. By Def. 4.1.4, the update of every *twn*-loop can be transformed to the form

$$\begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \leftarrow \begin{bmatrix} c_1 \cdot x_1 + pol_1 \\ \vdots \\ c_d \cdot x_d + pol_d \end{bmatrix}$$

where $c_i \in \mathcal{S}$ and $pol_i \in \mathcal{S}[x_{i+1}, \dots, x_d]$ by permuting variables. Together with triangularity, weak non-linearity enables us to compute a closed form for the n-fold application of the update.

Definition 4.1.5 (Closed Form)**.** A vector $\vec{q}$ of d expressions over the variables $\vec{x}$ and a distinguished variable n is a *closed form* for $\vec{u} \in (\mathcal{S}[\vec{x}])^d$ if $\vec{q}[\vec{x}/\vec{e}, n/n'] = \vec{u}^{n'}(\vec{e})$ for all $\vec{e} \in \mathcal{S}^d$ and $n' \in \mathbb{N}$, i.e., both expressions evaluate to the same element of $\mathcal{S}$.

Our *twn*-loops are a special case of *solvable loops* [RK04].

Definition 4.1.6 (Solvable Loops)**.** A loop $(\varphi, \vec{u})$ is *solvable* if there is a partitioning $\mathcal{J} = \{J_1, \dots, J_k\}$ of $\{1, \dots, d\}$ such that for each $1 \leq i \leq k$ we have

$$\vec{u}_{J_i} = A_i \cdot \vec{x}_{J_i} + \vec{p}_i,$$

where $\vec{u}_{J_i}$ is the vector of all u_j with $j \in J_i$ (and $\vec{x}_{J_i}$ is defined analogously), $d_i = |J_i|$, $A_i \in \mathcal{S}^{d_i \times d_i}$, and $\vec{p}_i \in (\mathcal{S}[\vec{x}_{J_{i+1}}, \dots, \vec{x}_{J_k}])^{d_i}$. The eigenvalues of a solvable loop are defined as the union of the eigenvalues of all A_i.

So solvable loops allow for blocks of variables with linear dependencies, and *twn*-loops correspond to the case that each such block has size 1. While our approach could easily be generalized to solvable loops with *real* eigenvalues, in Ch. 9 we show that such a generalization does not increase its applicability.

For our decidability results in Ch. 6, we will reduce termination to the so-called *existential fragment* of the first-order theory.

Definition 4.1.7 (Existential Fragment of First-Order Theory [Ren92; Sch09])**.** Let $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathrm{A}}$ be a ring. The *existential fragment of the first-order theory of* $\mathcal{S}$ is the set $\mathrm{Th}_{\exists}(\mathcal{S})$ of all formulas $\exists \vec{y} \in \mathcal{S}^k.\ \psi$ where ψ is a propositional formula over the atoms $\{p \rhd 0 \mid p \in \mathbb{Q}[\vec{y}, \vec{z}],\ \rhd \in \{\geq, >\}\}$ and $k \in \mathbb{N}$. Here, $\vec{y}$ and $\vec{z}$ are pairwise disjoint vectors of variables (i.e., the variables $\vec{z}$ are *free*).

The existential fragment of the first-order theory of $\mathcal{S}$ and $\mathbb{R}_{\mathrm{A}}$ is the set $\mathrm{Th}_{\exists}(\mathcal{S}, \mathbb{R}_{\mathrm{A}})$ of all formulas $\exists \vec{y}' \in \mathbb{R}_{\mathrm{A}}^{k'}, \vec{y} \in \mathcal{S}^k.\ \psi$, with a propositional formula ψ over $\{p \rhd 0 \mid p \in \mathbb{Q}[\vec{y}', \vec{y}, \vec{z}],\ \rhd \in \{\geq, >\}\}$ where $k', k \in \mathbb{N}$ and the variables $\vec{y}'$, $\vec{y}$, and $\vec{z}$ are pairwise disjoint. A formula without free variables is *closed*.

In the following, we also consider formulas over inequations $p \rhd 0$ where p's coefficients are from $\mathbb{R}_{\mathrm{A}}$ to be elements of $\mathrm{Th}_{\exists}(\mathbb{R}_{\mathrm{A}})$ (resp. $\mathrm{Th}_{\exists}(\mathcal{S}, \mathbb{R}_{\mathrm{A}})$). The reason is that real algebraic numbers are $\mathrm{Th}_{\exists}(\mathbb{R}_{\mathrm{A}})$-definable. In this case, we also refer to $p \rhd 0$ as *linear* if p is linear, e.g., $\sqrt{2} \cdot x_1 - \sqrt{3} \cdot x_2 > 0$ is a linear inequation in our setting.

A formula $\xi \in \mathrm{Th}_{\exists}(\mathbb{R}_{\mathrm{A}}, \mathcal{S})$ is *satisfiable* if there exists an assignment I of the free variables in ξ with real algebraic numbers such that $I(\xi)$ holds when quantified variables range over their specified domain and the mathematical operations are interpreted in the standard way. In this case, I satisfies ξ. A formula ξ is *valid* if it is satisfied by *all* assignments. Hence, validity and satisfiability coincide for closed formulas. Thus, for closed formulas we only consider validity. Moreover, satisfiability of a non-closed formula ξ is equivalent to validity of the formula ξ' which also existentially quantifies the free variables of ξ. Thus, for formulas from $\mathrm{Th}_{\exists}(\mathbb{R}_{\mathrm{A}}, \mathcal{S})$ we can w.l.o.g. consider validity of closed formulas.

Note that validity of formulas from $\mathrm{Th}_{\exists}(\mathcal{S})$ or $\mathrm{Th}_{\exists}(\mathcal{S}, \mathbb{R}_{\mathrm{A}})$ is decidable if $\mathcal{S} \in \{\mathbb{R}_{\mathrm{A}}, \mathbb{R}\}$ and semi-decidable if $\mathcal{S} \in \{\mathbb{Z}, \mathbb{Q}\}$ [Coh69; Tar98]. By undecidability of Hilbert's Tenth Problem [Mat70], validity is undecidable for $\mathcal{S} = \mathbb{Z}$. While validity of full first-order formulas (i.e., also containing universal quantifiers) over $\mathcal{S} = \mathbb{Q}$ is undecidable [Rob49], it is still open whether validity of formulas from $\mathrm{Th}_{\exists}(\mathbb{Q})$ or $\mathrm{Th}_{\exists}(\mathbb{Q}, \mathbb{R}_{\mathrm{A}})$ is decidable. However, validity of *linear* formulas from $\mathrm{Th}_{\exists}(\mathcal{S})$ or $\mathrm{Th}_{\exists}(\mathcal{S}, \mathbb{R}_{\mathrm{A}})$ is decidable for all $\mathcal{S} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}_{\mathrm{A}}, \mathbb{R}\}$ [Dan47; Gom58; Gom60; Kan60; LD60].

Example 4.1.8. Consider the loop $\mathbf{L}_{ex}$ in Fig. 4.1(b). This loop is triangular since $>_{\vec{u}^=} = \{(x_1, x_2), (x_1, x_3), (x_2, x_3)\}$ is well founded. Moreover, it is weakly non-linear. Since the coefficient of x_i is 1 in the update of x_i for all $1 \leq i \leq 3$, this loop is also non-negative, i.e., $\mathbf{L}_{ex}$ is *tnn*. We consider this loop over $\mathbb{Z}$.

Chapter 5

Reducing Termination of *twn*-Loops to Termination of *tnn*-Loops

In this chapter, we show that for analyzing termination of *twn*-loops we can restrict ourselves to *tnn*-loops, because each *twn*-loop can be transformed into a *tnn*-loop via *chaining*. Moreover, we discuss closed forms for the n-fold application of the update of *tnn*-loops which can be represented as a vector of *poly-exponential expressions*.

Definition 5.1.1 (Chaining). *Chaining* a loop $(\varphi, \vec{u})$ yields $(\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$.

Clearly, $(\varphi, \vec{u})$ terminates on $\vec{e} \in \mathcal{S}^d$ iff $(\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$ terminates on $\vec{e}$. Moreover, if $(\varphi, \vec{u})$ is a *twn*-loop, then $(\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$ is a *tnn*-loop, i.e., the coefficient of each x_i in $u_i(\vec{u})$ is non-negative. Thus, analogous to [FG19a], we obtain the following theorem.

Theorem 5.1.2 (Soundness of Chaining). *Let $(\varphi, \vec{u})$ be a twn-loop on $\mathcal{S}^d$.*

(i) *$(\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$ is tnn.*

(ii) *$(\varphi, \vec{u})$ terminates on $\vec{e} \in \mathcal{S}^d$ iff $(\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$ terminates on $\vec{e}$.*

As chaining is clearly computable, we obtain the following corollary (see [FG19a]).

Corollary 5.1.3. *Termination of twn-loops is reducible to termination of tnn-loops.*

It is well known that closed forms for *tnn*-loops are computable, see, e.g., [Kin+19]. The reason is that the bodies of *tnn*-loops correspond to a special case of *C-finite*

recurrences, which are known to be solvable [KP11]. The resulting closed forms may contain polynomial arithmetic and exponentiation w.r.t. n (as, e.g., $x_1 \leftarrow 2 \cdot x_1$ has the closed form $x_1 \cdot 2^n$) as well as certain piecewise defined functions. For example, the closed form $x_1^{(n)}$ of $x_1 \leftarrow 1$ is $x_1^{(0)} = x_1$ and $x_1^{(n)} = 1$ for all $n \in \mathbb{N}$, $n > 0$.

We use *poly-exponential expressions* [FG19a] to represent closed forms, where piecewise defined functions are simulated via *Iverson brackets*. Here we slightly generalize the definition of poly-exponential expressions from [FG19a, Def. 9] by allowing arbitrary polynomials over the variables $\vec{x}$ as coefficients. Given a formula ψ over n, its Iverson bracket $[\psi] : \mathbb{N} \to \{0, 1\}$ evaluates to 1 iff ψ is satisfied (i.e., $[\psi](c) = 1$ if $\psi[n/c]$ holds and $[\psi](c) = 0$, otherwise). In this way, we avoid handling piecewise defined functions via disjunctions (as done in the closed form of [Kin+19]). Poly-exponential expressions are sums of arithmetic terms over the variables $\vec{x}$ and the additional designated variable n, where it is always clear which addend determines the asymptotic growth of the whole expression when increasing n. This is crucial for our reducibility proof in Ch. 6. In the following, for any set $X \subseteq \mathbb{R}$, any $k \in X$, and $\triangleright \in \{\geq, >\}$, let $X_{\triangleright k} = \{x \in X \mid x \triangleright k\}$.

Definition 5.1.4 (Poly-Exponential Expressions)**.** Let C be the set of all finite conjunctions over $\{n = c, n \neq c \mid c \in \mathbb{N}\}$ where n is a designated variable. The set of all *poly-exponential expressions* with the variables $\vec{x}$ over $\mathcal{S}$ is

$$\mathbb{PE}_{\mathcal{S}}[\vec{x}] = \left\{ \sum_{j=1}^{\ell} \left[\psi_j\right] \cdot \alpha_j \cdot n^{a_j} \cdot b_j^n \;\middle|\; \ell, a_j \in \mathbb{N},\ \psi_j \in C,\ \alpha_j \in \mathcal{Q}_{\mathcal{S}}[\vec{x}],\ b_j \in \mathcal{S}_{>0} \right\},$$

where $\mathcal{Q}_{\mathcal{S}} = \left\{ \frac{r}{s} \;\middle|\; r \in \mathcal{S}, s \in \mathcal{S}_{>0} \right\}$ is the quotient field of $\mathcal{S}$.

An example for a poly-exponential expression over $\mathbb{Z}$ (with quotient field $\mathcal{Q}_{\mathbb{Z}} = \mathbb{Q}$) is

$$[n \neq 0 \wedge n \neq 1] \cdot \left(\tfrac{1}{2} \cdot {x_1}^2 + \tfrac{3}{4} \cdot x_2 - 1\right) \cdot n^3 \cdot 3^n + [n = 1] \cdot (x_1 - x_2).$$

Note that the restriction to triangular loops ensures that the closed form consists indeed of elements of $\mathbb{PE}_{\mathcal{S}}[\vec{x}]$. For example, for arbitrary matrices $A \in \mathbb{R}_{\mathrm{A}}^{d \times d}$, the update $\vec{x} \leftarrow A \cdot \vec{x}$ is known to admit a closed form as in Def. 5.1.4 with complex b_j's, whereas real numbers suffice for triangular matrices. Moreover, non-negativity is required to ensure $b_j > 0$ (e.g., the non-*tnn* loop $x_1 \leftarrow -x_1$ has the closed form $x_1 \cdot (-1)^n$). So together with triangularity, weak non-linearity ensures that for every *tnn*-loop over $\mathcal{S}$, one can compute a closed form $\vec{q} \in \left(\mathbb{PE}_{\mathcal{S}}[\vec{x}]\right)^d$.

Example 5.1.5. Reconsider the loop $\mathbf{L}_{ex}$ in Fig. 4.1(b). The closed form for the update after $n \in \mathbb{N}$ loop iterations is given by

$$\vec{q} = \begin{bmatrix} \frac{4}{3} \cdot x_3^5 \cdot n^3 + \left(-2 \cdot x_3^5 - 2 \cdot x_2 \cdot x_3^3\right) \cdot n^2 + \left(x_2^2 \cdot x_3 + \frac{2}{3} \cdot x_3^5 + 2 \cdot x_2 \cdot x_3^3\right) \cdot n + x_1 \\ -2 \cdot x_3^2 \cdot n + x_2 \\ x_3 \end{bmatrix} \in \left(\mathbb{PE}_{\mathbb{Z}}[\vec{x}]\right)^3$$

Chapter 6

Reducing Termination of *tnn*-Loops to $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$

It is known that the bodies of *tnn*-loops can be linearized [OBP16a] (see Sect. 8.3 for a discussion of [OBP16a]), i.e., one can reduce termination of a *tnn*-loop $(\varphi, \vec{u})$ to termination of a linear-update *tnn*-loop $(\varphi', \vec{u}')$, where φ' may be *non*-linear. Moreover, [Wu+10; XZ10] showed decidability of termination for certain classes of conjunctive linear-update loops over $\mathbb{R}$, which cover conjunctive linear-update *tnn*-loops. So, by combining the results of [OBP16a] and [Wu+10; XZ10], one can conclude that termination for *conjunctive tnn*-loops over $\mathbb{R}$ is decidable.

However, we now present a reduction of termination of *tnn*-loops to $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$ which applies to *tnn*-loops over *any* ring $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}$ and can also handle *disjunctions* in the condition of the loop. Moreover, our reduction yields tight complexity results on termination of linear loops over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}_\mathbb{A}$, and $\mathbb{R}$, and on termination of linear-update loops over $\mathbb{R}_\mathbb{A}$ and $\mathbb{R}$ (see Ch. 10).

The idea of this reduction is similar to [FG19a], where the authors only considered conjunctive linear loops over $\mathbb{Z}$. In contrast, we analyze termination of a loop $(\varphi, \vec{u})$ on $\mathcal{S}^d$ for any $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_\mathbb{A}$ and allow arbitrary propositional formulas and *non*-linearity in the condition. So the correctness proofs differ substantially from [FG19a].

In the following, let $(\varphi, \vec{u})$ be *tnn* and let $\vec{q} \in \left(\mathbb{PE}_\mathcal{S}[\vec{x}]\right)^d$ be the closed form for $\vec{u}$. We now show how to encode termination of $(\varphi, \vec{u})$ on $\mathcal{S}^d$ into a $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$-formula. More precisely, we show that there is a function with the following specification that

is computable in polynomial time:

$$\begin{aligned} \text{Input}: &\quad (\varphi, \vec{u}) \text{ and } \vec{q} \text{ as above} \\ \text{Result}: &\quad \text{a closed formula } \chi \in \mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A}) \text{ such that} \\ &\quad \chi \text{ is valid iff } (\varphi, \vec{u}) \text{ does not terminate on } \mathcal{S}^d \end{aligned} \tag{6.1.1}$$

We use the concept of *eventual non-termination*, i.e., the condition of the loop may be violated finitely often, see Def. 4.1.3. Clearly, $(\varphi, \vec{u})$ is non-terminating iff it is eventually non-terminating [OPW15]. The formula χ in (6.1.1) will encode the existence of a witness for eventual non-termination.

By the definition of $\vec{q}$, eventual non-termination of $(\varphi, \vec{u})$ on $\mathcal{S}^d$ is equivalent to validity of

$$\exists \vec{x} \in \mathcal{S}^d,\ n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ \varphi(\vec{q}). \tag{6.1.2}$$

Example 6.1.1. Continuing Ex. 5.1.5, $\mathbf{L}_{ex}$ is eventually non-terminating on $\mathbb{Z}$ iff there is a corresponding witness $\vec{e} \in \mathbb{Z}^3$, i.e., iff

$$\exists x_1, x_2, x_3 \in \mathbb{Z},\ n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ p > 0 \tag{6.1.3}$$

is valid where

$$\begin{aligned} p = &\left(x_1 + x_2^2\right) + \left(x_2^2 \cdot x_3 + \tfrac{2}{3} \cdot x_3^5 + 2 \cdot x_2 \cdot x_3^3 - 4 \cdot x_2 \cdot x_3^2\right) \cdot n \\ &+ \left(-2 \cdot x_3^5 - 2 \cdot x_2 \cdot x_3^3 + 4 \cdot x_3^4\right) \cdot n^2 + \left(\tfrac{4}{3} \cdot x_3^5\right) \cdot n^3. \end{aligned} \tag{6.1.4}$$

Let $\vec{q}_{norm}$ be like $\vec{q}$, but each factor $[\psi]$ is replaced by 0 if it contains an equation and by 1, otherwise. The reason is that for large enough n, equations in ψ become false and negated equations become true. Thus, (6.1.2) is equivalent to

$$\exists \vec{x} \in \mathcal{S}^d,\ n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ \varphi(\vec{q}_{norm}). \tag{6.1.5}$$

In this way, we obtain *normalized* poly-exponential expressions.

Definition 6.1.2 (Normalized $\mathbb{PE}$s)**.** We call $p \in \mathbb{PE}_\mathcal{S}[\vec{x}]$ *normalized* if it is in

$$\mathbb{NPE}_\mathcal{S}[\vec{x}] = \left\{ \sum_{j=1}^{\ell} \alpha_j \cdot n^{a_j} \cdot b_j^n \;\middle|\; \ell, a_j \in \mathbb{N},\ \alpha_j \in \mathcal{Q}_\mathcal{S}[\vec{x}],\ b_j \in \mathcal{S}_{>0} \right\},$$

where w.l.o.g. $(b_i, a_i) \neq (b_j, a_j)$ if $i \neq j$. We define $\mathbb{NPE}_\mathcal{S} = \mathbb{NPE}_\mathcal{S}[\emptyset]$.

As φ is a propositional formula over $\mathcal{S}[\vec{x}]$-inequations, $\varphi(\vec{q}_{norm})$ is a propositional formula over $\mathbb{NPE}_\mathcal{S}[\vec{x}]$-inequations. By (6.1.5), we need to check whether $\exists \vec{x} \in \mathcal{S}^d$. $\varphi(\vec{q}_{norm})$ is valid for large enough n. To do so, we generalize [FG19a, Lemma 24].

Definition 6.1.3 (Asymptotic Domination, [Lan09])**.** A function $g : \mathbb{N} \longrightarrow \mathbb{R}$ *dominates* a function $f : \mathbb{N} \longrightarrow \mathbb{R}$ asymptotically ($f \in o(g)$) if for all $m > 0$ there is an $n_0 \in \mathbb{N}$ such that $|f(n)| < m \cdot |g(n)|$ for all $n \in \mathbb{N}_{>n_0}$.

With this terminology at hand, we can now state the following lemma.

Lemma 6.1.4. *Let* $b_1, b_2 \in (\mathbb{R}_\mathbb{A})_{>0}$ *and* $a_1, a_2 \in \mathbb{N}$. *If* $(b_2, a_2) >_{lex} (b_1, a_1)$, *then* $n^{a_1} \cdot b_1^n \in o(n^{a_2} \cdot b_2^n)$, *where* $(b_2, a_2) >_{lex} (b_1, a_1)$ *iff* $b_2 > b_1$ *or* $b_2 = b_1 \wedge a_2 > a_1$.

In the following, let $p \geq 0$ or $p > 0$ occur in $\varphi(\vec{q}_{norm})$. Then we can order the coefficients of p according to the asymptotic growth of their addends w.r.t. n.

Definition 6.1.5 (Ordering Coefficients)**.** *Marked coefficients* are of the form $\alpha^{(b,a)}$ where $\alpha \in \mathcal{Q}_\mathcal{S}[\vec{x}]$, $b \in \mathcal{S}_{>0}$, and $a \in \mathbb{N}$. We define unmark $\left(\alpha^{(b,a)}\right) = \alpha$ and $\alpha_2^{(b_2,a_2)} >_{coef} \alpha_1^{(b_1,a_1)}$ if $(b_2, a_2) >_{lex} (b_1, a_1)$. Let $p = \sum_{j=1}^{\ell} \alpha_j \cdot n^{a_j} \cdot b_j^n \in \mathbb{NPE}_\mathcal{S}[\vec{x}]$, where $\alpha_j \neq 0$ for all $1 \leq j \leq \ell$. Then the marked coefficients of p are

$$\mathrm{coefs}(p) = \{0^{(1,0)}\} \text{ if } \ell = 0 \quad \text{and} \quad \mathrm{coefs}(p) = \left\{ \alpha_j^{(b_j,a_j)} \;\middle|\; 1 \leq j \leq \ell \right\}, \text{ otherwise.}$$

Example 6.1.6. Continuing Ex. 6.1.1, $\mathrm{coefs}(p)$ is $\{\alpha_1^{(1,0)}, \alpha_2^{(1,1)}, \alpha_3^{(1,2)}, \alpha_4^{(1,3)}\}$ where

$$\alpha_1 = x_1 + x_2^2 \qquad \alpha_2 = x_2^2 \cdot x_3 + \tfrac{2}{3} \cdot x_3^5 + 2 \cdot x_2 \cdot x_3^3 - 4 \cdot x_2 \cdot x_3^2$$
$$\alpha_3 = -2 \cdot x_3^5 - 2 \cdot x_2 \cdot x_3^3 + 4 \cdot x_3^4 \qquad \alpha_4 = \tfrac{4}{3} \cdot x_3^5$$

Note that $p(\vec{e}) \in \mathbb{NPE}_\mathcal{S}$ for any $\vec{e} \in \mathcal{S}^d$, i.e., the only variable in $p(\vec{e})$ is n. Now the $>_{coef}$-maximal addend determines the asymptotic growth of $p(\vec{e})$:

$$o(p(\vec{e})) = o(k \cdot n^a \cdot b^n) \quad \text{where } k = k^{(b,a)} = \max_{>_{coef}} \left(\mathrm{coefs}\left(p(\vec{e})\right)\right). \tag{6.1.6}$$

Note that (6.1.6) would be incorrect for the case $k = 0$ if we replaced $o(p(\vec{e})) = o(k \cdot n^a \cdot b^n)$ with $o(p(\vec{e})) = o(n^a \cdot b^n)$ as $o(0) = \emptyset \neq o(1)$. Obviously, (6.1.6) implies

$$\exists n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ \operatorname{sign}\left(p(\vec{e})\right) = \operatorname{sign}(k), \tag{6.1.7}$$

where $\operatorname{sign}(0) = 0$, $\operatorname{sign}(k) = 1$ if $k > 0$, and $\operatorname{sign}(k) = -1$ if $k < 0$. This already allows us to reduce eventual non-termination to $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$ if φ is an atom.

Lemma 6.1.7. *Given $p \in \mathbb{NPE}_\mathcal{S}[\vec{x}]$ and $\triangleright \in \{\geq, >\}$, one can reduce validity of*

$$\exists \vec{x} \in \mathcal{S}^d,\ n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ p \triangleright 0 \tag{6.1.8}$$

to validity of a closed formula from $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$. This reduction takes polynomially many steps in the sum of the sizes of the coefficients of p.

Proof. We have $p \in \mathbb{NPE}_\mathcal{S}[\vec{x}]$, so $p(\vec{e}) \in \mathbb{NPE}_\mathcal{S}$ for any $\vec{e} \in \mathcal{S}^d$. Hence,

$$\exists n_0 \in \mathbb{N}. \forall n \in \mathbb{N}_{>n_0}. p(\vec{e}) \triangleright 0 \quad \text{iff} \quad \operatorname{unmark}\left(\max_{>_{coef}}\left(\operatorname{coefs}\left(p(\vec{e})\right)\right)\right) \triangleright 0 \text{ (by (6.1.7)).}$$

Let $\operatorname{coefs}(p) = \{\alpha_1^{(b_1,a_1)}, \ldots, \alpha_\ell^{(b_\ell,a_\ell)}\}$, where $\alpha_i^{(b_i,a_i)} \prec_{coef} \alpha_j^{(b_j,a_j)}$ for all $1 \leq i < j \leq \ell$. If $p(\vec{e}) = 0$, then $\alpha_1(\vec{e}) = \ldots = \alpha_\ell(\vec{e}) = 0$ and thus $\operatorname{coefs}\left(p(\vec{e})\right) = \{0^{(1,0)}\}$ and $\operatorname{unmark}\left(\max_{>_{coef}}\left(\operatorname{coefs}\left(p(\vec{e})\right)\right)\right) = 0$. Otherwise, there is a $1 \leq i \leq \ell$ with $\operatorname{unmark}\left(\max_{>_{coef}}\left(\operatorname{coefs}\left(p(\vec{e})\right)\right)\right) = \alpha_i(\vec{e}) \neq 0$ and $\alpha_j(\vec{e}) = 0$ for all $i+1 \leq j \leq \ell$. Now define

$$\operatorname{red}(p > 0) = \bigvee_{i=1}^{\ell}\left(\alpha_i > 0 \wedge \bigwedge_{j=i+1}^{\ell} \alpha_j = 0\right)$$

$$\text{and} \qquad \operatorname{red}(p \geq 0) = \operatorname{red}(p > 0) \vee \bigwedge_{j=1}^{\ell} \alpha_i = 0.$$

Obviously,

$$\operatorname{unmark}\left(\max_{>_{coef}}\left(\operatorname{coefs}\left(p(\vec{e})\right)\right)\right) \triangleright 0 \quad \text{iff} \quad (\operatorname{red}(p \triangleright 0))(\vec{e}) \text{ holds.}$$

Hence, (6.1.8) is equivalent to

$$\exists \vec{x} \in \mathcal{S}^d.\ \operatorname{red}(p \triangleright 0). \tag{6.1.9}$$

We can compute and sort $\operatorname{coefs}(p)$ in polynomial time in the sum of the sizes of the coefficients of p. Furthermore, $\operatorname{red}(p \triangleright 0)$ is a disjunction of at most $\ell + 1$ subformulas, where each subformula consists of at most ℓ (in-)equations over $\operatorname{coefs}(p)$. As ℓ is bounded by the sum of the sizes of the coefficients of p, $\operatorname{red}(p \triangleright 0)$ can be computed in polynomial time. Thus, it follows that (6.1.9) can be computed in polynomial time. □

Example 6.1.8. We continue Ex. 6.1.6. Here, unmark $\left(\max_{>_{coef}}(\mathrm{coefs}(p))\right) = \frac{4}{3} \cdot x_3^5$. Thus, (6.1.3) is valid iff $\exists x_1, x_2, x_3 \in \mathbb{Z}.\ \mathrm{red}(p > 0)$ is valid where

$$\begin{aligned}\mathrm{red}(p > 0) \;=\; & (\alpha_1 > 0 \wedge \alpha_2 = 0 \wedge \alpha_3 = 0 \wedge \alpha_4 = 0) \;\vee\; (\alpha_2 > 0 \wedge \alpha_3 = 0 \wedge \alpha_4 = 0) \\ & \vee\; (\alpha_3 > 0 \wedge \alpha_4 = 0) \qquad\qquad \vee\; \alpha_4 > 0.\end{aligned}$$

Then $[x_1/-4, x_2/2, x_3/1]$ satisfies $\alpha_4 > 0$ as $\left(\frac{4}{3} \cdot x_3^5\right)[x_1/-4, x_2/2, x_3/1] > 0$. Thus, $(-4, 2, 1)$ witnesses eventual non-termination of $\mathbf{L}_{ex}$ on $\mathbb{Z}^3$.

Now we lift our reduction to propositional formulas. Note that a version of the following Thm. 6.1.9 that only covers conjunctions is a direct corollary of Lemma 6.1.7. To handle disjunctions, the proof of Thm. 6.1.9 exploits the crucial additional insight that a *tnn*-loop $(\varphi \vee \varphi', \vec{u})$ terminates iff $(\varphi, \vec{u})$ and $(\varphi', \vec{u})$ terminate, which is not true in general (as, e.g., witnessed by the loop $(x > 0 \vee -x > 0, -x)$).

Theorem 6.1.9 (Reducing Eventual Non-Termination to $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$)**.** *Given a propositional formula ξ over the atoms $\{p \triangleright 0 \mid p \in \mathbb{N}\mathbb{P}\mathbb{E}_\mathcal{S}[\vec{x}], \triangleright \in \{\geq, >\}\}$, one can reduce validity of*

$$\exists \vec{x} \in \mathcal{S}^d, n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ \xi \tag{6.1.10}$$

to validity of a formula $\exists \vec{x} \in \mathcal{S}^d.\ \mathrm{red}(\xi) \in \mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$. This reduction takes polynomially many steps in the sum of the sizes of the atoms in ξ.

Here, the formula $\mathrm{red}(\xi)$ results from ξ by replacing each atom $p \triangleright 0$ in ξ by $\mathrm{red}(p \triangleright 0)$.

Thm. 6.1.9 shows that the function (6.1.1) is computable (in polynomial time): $\vec{q}_{norm}$ can clearly be computed in polynomial time from $\vec{q}$ and we can then apply Thm. 6.1.9 to $\varphi(\vec{q}_{norm})$. Combining Cor. 5.1.3, (6.1.5), and Thm. 6.1.9 then leads to the main result of this chapter.

Theorem 6.1.10 (Reducing Termination)**.** *Termination of tnn-loops (resp. twn-loops) on $\mathcal{S}^d$ is reducible to* $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$.

However, in general this reduction is not computable by using only polynomial space so in particular, it is not computable in polynomial time. The reason is that the closed form $\vec{q}$ for $\vec{u}$ cannot be computed by using only polynomial space if the update $\vec{u}$ contains *non-linear* terms. For example, consider the *tnn*-loop $\mathbf{L}_{np}$ in Fig. 6.1. The closed form for x_i is $q_i = d^{(d^{i-1} \cdot (n-i+1))} \cdot x_1^{(d^{i-1})}$ for all $n \geq d$. Note that $\log\left(d^{(d^{d-1})}\right)$ grows faster in d than any polynomial in d. Thus, the closed form q_d of x_d contains

while (true) **do** {

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{d-1} \\ x_d \end{bmatrix} \leftarrow \begin{bmatrix} d \cdot x_1 \\ x_1^d \\ \vdots \\ x_{d-2}^d \\ x_{d-1}^d \end{bmatrix}$$

}

Figure 6.1: *tnn*-Loop $\mathbf{L}_{np}$

constants whose logarithm grows faster than any polynomial in d. Hence, q_d cannot be computed by using only polynomial space, in particular, it cannot be computed in polynomial time. As mentioned at the beginning of this chapter, the bodies of *tnn*-loops could also be linearized [OBP16a]. However, since the linearization of $\mathbf{L}_{np}$ contains these constants as well, it cannot be computed in polynomial time, either (we refer to Sect. 10.2 for more details).

Note that our reduction also works if $\mathcal{S} = \mathbb{R}$, i.e., termination over $\mathbb{R}$ is reducible to $\mathrm{Th}_\exists(\mathbb{R}, \mathbb{R}_\mathbb{A})$. As $\mathbb{R}$ and $\mathbb{R}_\mathbb{A}$ are *elementary equivalent* (see, e.g., [BPR06]), i.e., a first-order formula is valid over $\mathbb{R}$ iff it is valid over $\mathbb{R}_\mathbb{A}$, we get the following corollary.

Corollary 6.1.11 ((Semi-)Decidability of (Non-)Termination)**.** *Let $(\varphi, \vec{u})$ be a twn-loop.*

(i) The loop $(\varphi, \vec{u})$ terminates over $\mathbb{R}_\mathbb{A}$ iff it terminates over $\mathbb{R}$.

(ii) Termination of $(\varphi, \vec{u})$ on $\mathcal{S}^d$ is decidable if $\mathcal{S} = \mathbb{R}_\mathbb{A}$ or $\mathcal{S} = \mathbb{R}$.

(iii) Non-termination of $(\varphi, \vec{u})$ on $\mathcal{S}^d$ is semi-decidable if $\mathcal{S} = \mathbb{Z}$ or $\mathcal{S} = \mathbb{Q}$.

For *conjunctive twn*-loops, Cor. 6.1.11 (ii) also follows from combining [OBP16a] and [Wu+10; XZ10] (see Lemma 8.3.6).

Our technique does not yield witnesses for non-termination, but the formula constructed by Thm. 6.1.9 describes *all* witnesses for *eventual* non-termination. So, the set of witnesses for *eventual* non-termination is $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$-definable (see Def. 9.1.13), while in general the set of witnesses for non-termination is not (see [DX12]).

Lemma 6.1.12. *Let $\xi = \varphi(\vec{q}_{norm})$. Then $\vec{e} \in \mathbb{R}_\mathbb{A}^d$ witnesses eventual non-termination of $(\varphi, \vec{u})$ on $\mathcal{S}^d$ iff* $\mathrm{red}(\xi)(\vec{e})$.

We will show how to compute witnesses for non-termination from witnesses for eventual non-termination of *twn*-loops in Ch. 7. Thus, Lemma 6.1.12 combined with Ch. 7 yields a technique to enumerate all witnesses for non-termination recursively, i.e., the set $NT_{(\varphi,\vec{u})}$ is recursively enumerable for *twn*-loops over $\mathcal{S}$.

Algorithm 6.1.1 summarizes our technique to check termination of *twn*-loops.

Algorithm 6.1.1: Checking Termination

Input: a *twn*-loop $(\varphi, \vec{u})$
Result: $\top$ resp. $\bot$ if (non-)termination of $(\varphi, \vec{u})$ on $\mathcal{S}^d$ is proven, ? otherwise
if $(\varphi, \vec{u})$ *is not tnn* **then** $(\varphi, \vec{u}) \leftarrow (\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$
$\vec{q} \leftarrow$ closed form for $\vec{u}$
if *(in)validity of* $\mathrm{red}(\varphi(\vec{q}_{norm}))$ *cannot be proven* **then return** ?
if $\mathrm{red}(\varphi(\vec{q}_{norm}))$ *is valid* **then return** $\bot$ **else return** $\top$

Chapter 7

The Halting Problem

We now consider the *halting problem* for *twn*-loops. In contrast to termination, i.e., to the question whether a loop terminates on *all* $\vec{e} \in \mathcal{S}^d$, the halting problem asks whether a loop terminates on a *given* $\vec{e} \in \mathcal{S}^d$. In this chapter, we prove the following theorem for any ring $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathbb{A}}$.

Theorem 7.1.1 (Decidability of the Halting Problem)**.** *The halting problem for twn-loops is decidable.*

Clearly, if the halting problem is decidable for $\mathcal{S} = \mathbb{R}_{\mathbb{A}}$, it is also decidable for all subrings of $\mathbb{R}_{\mathbb{A}}$. Thus, w.l.o.g. we restrict ourselves to loops over $\mathbb{R}_{\mathbb{A}}$ throughout this chapter.

In Ch. 6, we presented (semi-)decision procedures for (non-)termination of *twn*-loops. However, the approach from Ch. 6 only yields witnesses for *eventual* non-termination of a loop $(\varphi, \vec{u})$. If one applies the update $\vec{u}$ sufficiently often to $\vec{e} \in ENT_{(\varphi,\vec{u})}$, one obtains a witness $\vec{u}^{n_0}(\vec{e})$ for non-termination. However, the number of required updates, i.e., the value of n_0, is not obvious. So in general, up to now it was unclear how to find *witnesses* for non-termination of *twn*-loops. By proving Thm. 7.1.1, i.e., by showing that the set $NT_{(\varphi,\vec{u})}$ is decidable, we fill this gap.

To prove decidability of $NT_{(\varphi,\vec{u})}$, we show how to compute a natural number n_0 such that the truth value of the condition φ *stabilizes*, i.e., such that $\varphi(\vec{u}^n(\vec{e})) \iff \varphi(\vec{u}^{n_0}(\vec{e}))$ for all $n \geq n_0$. Thus, given the closed form $\vec{q}$ for $\vec{u}$ and a witness $\vec{e}$ for eventual non-termination (which can, e.g., be computed as in Ch. 6), a witness for non-termination can efficiently be computed by evaluating $\vec{q}[\vec{x}/\vec{e}, n/n_0]$. Hence, one can also enumerate all (possibly infinitely many) witnesses for non-termination recursively.

We now prove Thm. 7.1.1, i.e., we show that for any $\vec{e} \in \mathbb{R}_{\mathbb{A}}^d$, it is decidable whether $\vec{e}$ is a witness for non-termination. As chaining preserves NT (see Thm. 5.1.2), w.l.o.g. we assume that

$$(\varphi, \vec{u}) \text{ is a } \textit{tnn}\text{-loop.} \qquad \text{(Simplification 1)}$$

As mentioned above, our proof is based on the concept of *stabilization*.

Definition 7.1.2 (Stabilization)**.** We say that $(\varphi, \vec{u})$ *stabilizes* on $\vec{e} \in \mathbb{R}_{\mathbb{A}}^d$ after $n_0 \in \mathbb{N}$ iterations if

$$\forall n \geq n_0.\ \varphi(\vec{u}^n(\vec{e})) \iff \varphi(\vec{u}^{n_0}(\vec{e})).$$

The smallest value n_0 for which $(\varphi, \vec{u})$ stabilizes on $\vec{e}$ is called the *stabilization threshold* of $(\varphi, \vec{u})$ on $\vec{e}$ (written $sth_{(\varphi,\vec{u})}(\vec{e})$).

Clearly, $(\varphi, \vec{u})$ stabilizes on $\vec{e}$ after n_0 iterations iff

$$\forall n \geq n_0.\ \varphi(\vec{q}[\vec{x}/\vec{e}]) \iff \varphi(\vec{q}[\vec{x}/\vec{e}, n/n_0]),$$

where $\vec{q}$ is the closed form for $\vec{u}$. For convenience, we sometimes also say that $\varphi(\vec{q})$ stabilizes on $\vec{e}$. If $\varphi(\vec{q}) \equiv p \rhd 0$ for $p \in \mathbb{PE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}]$ and $\rhd \in \{\geq, >\}$, we also just say that p stabilizes on $\vec{e}$.

For any $p \in \mathbb{PE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}]$ and $\vec{e} \in \mathbb{R}_{\mathbb{A}}^d$, let $sth_p(\vec{e})$ be the smallest value n_0 such that

$$\forall n \geq n_0.\ \operatorname{sign}\left(p[\vec{x}/\vec{e}]\right) = \operatorname{sign}\left(p[\vec{x}/\vec{e}, n/n_0]\right).$$

So if $\varphi(\vec{q}) \equiv p \rhd 0$, we have $sth_{(\varphi,\vec{u})} \leq sth_p$. Here, we compare functions pointwise, i.e., $f \leq g$ iff $f(\vec{e}) \leq g(\vec{e})$ for all $\vec{e} \in \operatorname{dom}(f) = \operatorname{dom}(g)$.

The following lemma is an immediate consequence of Ch. 5 and the fact that every poly-exponential expression is weakly monotonic w.r.t. n for large enough values of n. In other words, for large enough n, every $p \in \mathbb{PE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}]$ is weakly monotonically increasing (or decreasing), i.e., $n' \geq n$ implies validity of $p[n/n'] \geq p$ (or of $p[n/n'] \leq p$).

Lemma 7.1.3. *Every tnn-loop stabilizes on each* $\vec{e} \in \mathbb{R}_{\mathbb{A}}^d$.

So for *tnn*-loops $(\varphi, \vec{u})$, stabilization generalizes eventual non-termination: For every $\vec{e} \in ENT_{(\varphi,\vec{u})}$ there exists an n_0 such that $\forall n \geq n_0.\ \varphi(\vec{u}^n(\vec{e}))$, and for every $\vec{e} \in \mathbb{R}_{\mathbb{A}}^d \setminus ENT_{(\varphi,\vec{u})}$ there exists an n_0 such that $\forall n \geq n_0.\ \neg\varphi(\vec{u}^n(\vec{e}))$.

Example 7.1.4. We consider the *tnn*-loop $\mathbf{L}_{ex}$ in Fig. 4.1(b) over $\mathbb{R}_{\mathbb{A}}$. It stabilizes on any input. Its condition is $x_1 + x_2^2 > 0$ and for all $x_1, x_2, x_3 \in \mathbb{R}_{\mathbb{A}}$ we have $\varphi(\vec{q}) = p > 0$, where p is stated in (6.1.4).

To decide the halting problem, it suffices to find a computable upper bound n_0 on $sth_{(\varphi,\vec{u})}(\vec{e})$. Then $(\varphi, \vec{u})$ diverges on $\vec{e}$ iff $\forall n \leq n_0.\ \varphi(\vec{u}^n(\vec{e}))$. Let $pol_1 \rhd_1 0, \dots, pol_r \rhd_r 0$ be all inequations occurring in φ. Then we have

$$sth_{(\varphi,\vec{u})} \leq \max\left\{ sth_{(pol_i \rhd_i 0,\vec{u})} \;\middle|\; 1 \leq i \leq r \right\} \tag{7.1.1}$$

and thus, we can assume

$$\varphi \equiv pol \rhd 0 \qquad \text{where } \rhd \in \{\geq, >\} \text{ and } pol \in \mathbb{R}_{\mathbb{A}}[\vec{x}]. \tag{Simplification 2}$$

So in the following, let $\varphi(\vec{q}) \equiv pol(\vec{q}) \rhd 0 \equiv p \rhd 0$ for some $p \in \mathbb{PE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}]$. Then our goal is to over-approximate the stabilization threshold of p on $\vec{e}$. Here, we may assume $p \in \mathbb{NPE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}]$ without loss of generality. To see why, let $c \in \mathbb{N}$ be larger than any constant that occurs in a factor $[\psi]$ in p. (If there is no such factor, we choose $c = 0$.) Then we can compute the stabilization threshold for p_{norm} instead of p on $\vec{e}$, where p_{norm} results from p by replacing all factors $[\psi]$ that contain positive literals with 0 and all other such factors with 1. Afterwards, we can easily lift the resulting bound on $sth_{p_{norm}}(\vec{e})$ to a bound on $sth_p(\vec{e})$. The reason is that we have $p = p_{norm}$ for all $n \geq c$ (since then, positive literals become false and negative literals become true). Thus,

$$sth_{(\varphi,\vec{u})}(\vec{e}) \leq sth_p(\vec{e}) \leq \max\{sth_{p_{norm}}(\vec{e}), c\}. \tag{7.1.2}$$

Hence, in the following we assume

$$\varphi(\vec{q}) \equiv pol(\vec{q}) \rhd 0 \equiv p \rhd 0 \qquad \text{where } p \in \mathbb{NPE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}]. \tag{Simplification 3}$$

To infer a bound on $sth_p(\vec{e})$, we rely on the concept of *monotonicity thresholds*.

Definition 7.1.5 (Monotonicity Threshold)**.** Let $a_1, a_2 \in \mathbb{N}$ and $b_1, b_2 \in (\mathbb{R}_{\mathbb{A}})_{>0}$ with $(b_1, a_1) <_{lex} (b_2, a_2)$. Moreover, let $k \in \mathbb{R}_{\mathbb{A}}$. Then the smallest $n_0 \in \mathbb{N}$ such that $k \cdot n^{a_1} \cdot b_1^n < n^{a_2} \cdot b_2^n$ for all $n \geq n_0$ is called the *k-monotonicity threshold* of (b_1, a_1) and (b_2, a_2).

Example 7.1.6. As an example, consider $(b_1, a_1) = (1, 2) <_{lex} (1, 3) = (b_2, a_2)$. The 3-monotonicity threshold of (b_1, a_1) and (b_2, a_2) is $n_0 = 4$, as $3 \cdot n^{a_1} \cdot b_1^n = 3 \cdot n^2 \cdot 1^n < n^3 \cdot 1^n = n^{a_2} \cdot b_2^n$ for all $n \geq n_0 = 4$, but $3 \cdot 3^{a_1} \cdot b_1^3 = 3 \cdot 3^2 \cdot 1^3 = 27 \geq 27 = 3^3 \cdot 1^3 = 3^{a_2} \cdot b_2^3$.

Lemma 7.1.7 (Computing Monotonicity Thresholds)**.** *For any $a_1, a_2 \in \mathbb{N}$, $b_1, b_2 \in (\mathbb{R}_{\mathbb{A}})_{>0}$ with $(b_1, a_1) <_{lex} (b_2, a_2)$, and any $k \in \mathbb{R}_{\mathbb{A}}$, the k-monotonicity threshold of (b_1, a_1) and (b_2, a_2) is computable.*

Proof. If $k \leq 0$, the k-monotonicity threshold of (b_1, a_1) and (b_2, a_2) is 1. We now regard the case $k > 0$. To prove the lemma, we show that if both

$$a_1 \leq a_2 \qquad \text{or} \qquad \left(\tfrac{m}{m+1}\right)^{a_1 - a_2} \cdot \tfrac{b_2}{b_1} \geq 1 \qquad \text{and} \tag{7.1.3}$$

$$k \cdot m^{a_1} \cdot b_1^m < m^{a_2} \cdot b_2^m \tag{7.1.4}$$

hold, m is an upper bound on the k-monotonicity threshold of (b_1, a_1) and (b_2, a_2). Then the claim immediately follows, as $\lim_{m \to \infty} \left(\tfrac{m}{m+1}\right)^{a_1 - a_2} \cdot \tfrac{b_2}{b_1} = \tfrac{b_2}{b_1} > 1$ if $a_1 > a_2$ (and thus $b_1 < b_2$), i.e., (7.1.3) and (7.1.4) hold for large enough m. So to compute monotonicity thresholds, one initializes m to 0 and increments it until (7.1.3) and (7.1.4) are satisfied. As (7.1.3) and (7.1.4) are only a sufficient criterion for upper bounds on the k-monotonicity threshold, we now have $k \cdot n^{a_1} \cdot b_1^n < n^{a_2} \cdot b_2^n$ for all $n \geq m$, but m is not necessarily the smallest such number. Hence, one then has to decrement m again until $k \cdot (m-1)^{a_1} \cdot b_1^{m-1} \geq (m-1)^{a_2} \cdot b_2^{m-1}$ holds.

So assuming (7.1.3) and (7.1.4), we need to prove

$$k \cdot n^{a_1} \cdot b_1^n < n^{a_2} \cdot b_2^n \qquad \text{for all } n \geq m. \tag{7.1.5}$$

Since $n \geq m > 0$ due to (7.1.4), we have

$$k \cdot n^{a_1} \cdot b_1^n < n^{a_2} \cdot b_2^n \qquad \text{iff} \qquad k < n^{a_2 - a_1} \cdot \left(\frac{b_2}{b_1}\right)^n.$$

Case $a_1 \leq a_2$: Then $n^{a_2 - a_1} \cdot \left(\frac{b_2}{b_1}\right)^n$ is monotonically increasing and thus, (7.1.5) follows from (7.1.4).

Case $a_1 > a_2$: Let

$$f(n) = n^{a_2 - a_1} \cdot \left(\frac{b_2}{b_1}\right)^n \qquad \text{and} \qquad f^{\Delta}(n) = f(n+1) - f(n).$$

We prove that (7.1.3) implies $f^{\Delta}(n) \geq 0$ for all $n \geq m$. Here, we use f^{Δ} instead of f's first derivative since we consider monotonicity w.r.t. $n \in \mathbb{N}$ (as opposed to $n \in \mathbb{R}$). As (7.1.4) is equivalent to $f(m) > k$, our claim (7.1.5) then follows by induction on n.

For all $n \geq m$, we have

$$
\begin{aligned}
& f^{\Delta}(n) \geq 0 \\
\iff & (n+1)^{a_2-a_1} \cdot \left(\frac{b_2}{b_1}\right)^{n+1} - n^{a_2-a_1} \cdot \left(\frac{b_2}{b_1}\right)^{n} \geq 0 && \text{(by definition of } f^{\Delta}\text{)} \\
\iff & \left(\left(\tfrac{n+1}{n}\right)^{a_2-a_1} \cdot \tfrac{b_2}{b_1} - 1\right) \cdot n^{a_2-a_1} \cdot \left(\tfrac{b_2}{b_1}\right)^{n} \geq 0 \\
\iff & \left(\tfrac{n+1}{n}\right)^{a_2-a_1} \cdot \tfrac{b_2}{b_1} - 1 \geq 0 && \text{(as } n^{a_2-a_1} \cdot \left(\tfrac{b_2}{b_1}\right)^{n} > 0 \text{ for all } n \geq m > 0\text{)} \\
\iff & \left(\tfrac{n}{n+1}\right)^{a_1-a_2} \cdot \tfrac{b_2}{b_1} - 1 \geq 0 \\
\iff & \left(\tfrac{n}{n+1}\right)^{a_1-a_2} \cdot \tfrac{b_2}{b_1} \geq 1 \\
\impliedby & \left(\tfrac{m}{m+1}\right)^{a_1-a_2} \cdot \tfrac{b_2}{b_1} \geq 1 && \text{(as } n \geq m \text{ implies } \left(\tfrac{n}{n+1}\right)^{a_1-a_2} \cdot \tfrac{b_2}{b_1} \geq \left(\tfrac{m}{m+1}\right)^{a_1-a_2} \cdot \tfrac{b_2}{b_1}\text{)} \\
\iff & \text{(7.1.3)} && \text{(as } a_1 > a_2\text{)}
\end{aligned}
$$

□

Lemma 7.1.8 shows that Lemma 7.1.7 allows us to over-approximate the stabilization threshold of p on $\vec{e}$. This finishes the proof of Thm. 7.1.1.

Lemma 7.1.8 (Over-Approximating Stabilization Thresholds). *For any $p \in \mathbb{NPE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}]$ and any $\vec{e} \in (\mathbb{R}_{\mathbb{A}})^d$, one can compute an $m \in \mathbb{N}$ with $m \geq sth_p(\vec{e})$.*

Proof. Let $p[\vec{x}/\vec{e}] = \sum_{j=1}^{\ell} k_j \cdot n^{a_j} \cdot b_j^n$ with $k_j \neq 0$ for all $1 \leq j \leq \ell$, and $(b_1, a_1) <_{lex} \ldots <_{lex} (b_\ell, a_\ell)$. If $\ell = 1$, we trivially have $\operatorname{sign}\left(p[\vec{x}/\vec{e}]\right) = \operatorname{sign}(k_\ell)$ for all $n \geq 1$, which implies $1 \geq sth_p(\vec{e})$.

If $\ell \geq 2$, for each addend $k_j \cdot n^{a_j} \cdot b_j^n$ with $1 \leq j < \ell - 1$, we can compute the $\frac{|k_j|}{|k_{\ell-1}|}$-monotonicity threshold mt_j of (b_j, a_j) and $(b_{\ell-1}, a_{\ell-1})$ by Lemma 7.1.7. Thus, $|k_j| \cdot n^{a_j} \cdot b_j^n < |k_{\ell-1}| \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n$ holds for all $n \geq mt_j$. Let $mt = \max\left\{ mt_j \;\middle|\; 1 \leq j < \ell - 1 \right\}$ (where we use the convention $\max \emptyset = 0$, i.e., we have $mt = 0$ if $\ell = 2$). Then for all $n \geq mt$ we obtain

$$
\sum_{j=1}^{\ell-1} k_j \cdot n^{a_j} \cdot b_j^n \leq \sum_{j=1}^{\ell-1} |k_j| \cdot n^{a_j} \cdot b_j^n \leq \sum_{j=1}^{\ell-1} |k_{\ell-1}| \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n = (\ell - 1) \cdot |k_{\ell-1}| \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n.
$$

Algorithm 7.1.1: Deciding the Halting Problem

Input: a *twn*-loop $(\varphi, \vec{u})$ and $\vec{e} \in \mathbb{R}_A^d$
Result: $\top$ if $(\varphi, \vec{u})$ terminates on $\vec{e}$, $\bot$ otherwise

```
if (φ, u⃗) is not tnn then (φ, u⃗) ⟵ (φ ∧ φ(u⃗), u⃗(u⃗))
q⃗ ⟵ closed form for u⃗ with q⃗ ∈ (PE_{R_A}[x⃗])^d
n_0 ⟵ 0
foreach inequation p ▷ 0 occurring in φ(q⃗) do
    p ⟵ p_norm
    compute m ≥ sth_p(e⃗) as in the proofs of Lemma 7.1.7 and Lemma 7.1.8
    n_0 ⟵ max{n_0, m}
c ⟵ 1 + the maximal constant that occurs in any factor [ψ] in φ(q⃗)
n_0 ⟵ max{n_0, c}
if φ(q⃗[x⃗/e⃗]) holds for all 0 ≤ n ≤ n_0 then return ⊥ else return ⊤
```

Moreover, by Lemma 7.1.7 we can also compute the $\frac{(\ell-1)\cdot|k_{\ell-1}|}{|k_\ell|}$-monotonicity threshold mt' of $(b_{\ell-1}, a_{\ell-1})$ and (b_ℓ, a_ℓ). Hence, $(\ell-1)\cdot|k_{\ell-1}|\cdot n^{a_{\ell-1}}\cdot b_{\ell-1}^n < |k_\ell|\cdot n^{a_\ell}\cdot b_\ell^n$ holds for all $n \geq mt'$. Let $m = \max\{mt, mt'\}$. Then for all $n \geq m$ we have

$$\sum_{j=1}^{\ell-1} k_j \cdot n^{a_j} \cdot b_j^n \leq (\ell-1)\cdot|k_{\ell-1}|\cdot n^{a_{\ell-1}}\cdot b_{\ell-1}^n < |k_\ell|\cdot n^{a_\ell}\cdot b_\ell^n,$$

i.e., for all $n \geq m$ we have $\text{sign}\left(p[\vec{x}/\vec{e}]\right) = \text{sign}(k_\ell)$, which implies $m \geq sth_p(\vec{e})$. □

Example 7.1.9. We show how to decide the halting problem for the loop $\mathbf{L}_{ex}$ on the witness for eventual non-termination which we computed in Ex. 6.1.8: $x_1 = -4$, $x_2 = 2$, and $x_3 = 1$. By Ex. 6.1.1, we have $\varphi(\vec{q}) \equiv p > 0$, i.e., $sth_{(\varphi,\vec{u})}(-4,2,1) \leq sth_p(-4,2,1)$ where p is given in (6.1.4). So our goal is to compute an upper bound on $sth_p(-4,2,1)$. We have

$$p[x_1/-4, x_2/2, x_3/1] = \tfrac{4}{3}\cdot n^3 - 2\cdot n^2 + \tfrac{2}{3}\cdot n.$$

Thus, $\ell = 3$, $k_1 = \frac{2}{3}$, $k_2 = -2$, and $k_3 = \frac{4}{3}$. As in the proof of Lemma 7.1.8, we first compute the $\frac{|k_1|}{|k_2|}$-monotonicity threshold, i.e., the $\frac{1}{3}$-monotonicity threshold of $(1,1)$ and $(1,2)$, which is $mt = 1$. Next, we compute the $\frac{(\ell-1)\cdot|k_2|}{|k_3|}$-monotonicity

threshold, i.e., the 3-monotonicity threshold of $(1,2)$ and $(1,3)$, which is $mt' = 4$, see Ex. 7.1.6. Hence, $\max\{mt, mt'\} = 4$ is an upper bound on $sth_p(-4,2,1)$. While $\bigwedge_{n=2}^{4} \left(\frac{4}{3} \cdot n^3 - 2 \cdot n^2 + \frac{2}{3} \cdot n > 0\right)$ holds, $\left(\frac{4}{3} \cdot 1^3 - 2 \cdot 1^2 + \frac{2}{3} \cdot 1 > 0\right)$ is false. Thus, $(-4,2,1)$ is not a witness for non-termination of $\mathbf{L}_{ex}$. However, we have just proven that $\vec{u}^2((-4,2,1)) = (0,-2,1)$ witnesses non-termination of $\mathbf{L}_{ex}$.

Algorithm 7.1.1 summarizes our technique to decide the halting problem for *twn*-loops. If necessary, the loop is chained in Line 1 to enforce Simplification 1. Then n_0 is initialized to 0 and afterwards, we process each inequation $p \triangleright 0$ in $\varphi(\vec{q})$ separately, see Simplification 2. Line 5 implements Simplification 3. Then n_0 is set to an upper bound on $sth_{p_{norm}}(\vec{e})$, for all inequations $p_{norm} \triangleright 0$, see (7.1.1). Finally, in Line 8 and 9 we ensure that n_0 is large enough to justify normalizing p to p_{norm}, see (7.1.2).

Chapter 8

Runtime Bounds

In this chapter, we discuss *runtimes* of *twn*-loops over the *integers*. We first show how to obtain upper bounds on their concrete runtimes in Sect. 8.1. Then we present a technique to compute bounds on asymptotic runtimes in Sect. 8.2. While our approach for computing concrete bounds requires the computation of closed forms, we show that an asymptotic bound can be read off the loop directly.

8.1 Computing Concrete Runtime Bounds

We now show how to compute an upper bound on the concrete runtime of a *twn*-loop over the integers. Def. 8.1.1 introduces this notion formally, which is only defined for inputs where the loop terminates.

Definition 8.1.1 (Runtime)**.** For a loop $(\varphi, \vec{u})$ over $\mathbb{Z}$, we define its *runtime* $rt_{(\varphi,\vec{u})} : T_{(\varphi,\vec{u})} \longrightarrow \mathbb{N}$ on $\vec{e} \in \mathbb{Z}^d$ by

$$rt_{(\varphi,\vec{u})}(\vec{e}) = \min \left\{ n \in \mathbb{N} \;\middle|\; \neg(\varphi(\vec{u}^n(\vec{e}))) \right\}.$$

So $(\varphi, \vec{u})$ terminates on $\vec{e} \in T_{(\varphi,\vec{u})}$ after $rt_{(\varphi,\vec{u})}(\vec{e})$ iterations. In practice, it is usually infeasible to compute the runtime exactly, so that state-of-the-art complexity analyzers rely on approximations, see e.g., [Alb+11; Alb+19a; Bro+16; Car+17; Flo16; Fro+20; HAH12; HDW17; HS15; SSH17; SZV17]. In this spirit, we prove that the runtime of a *twn*-loop on $\mathbb{Z}^d$ is bounded (from above) by a polynomial in the 1-norm $\|\vec{x}\| = \sum_{j=1}^{d} |x_j|$

of $\vec{x}$. Moreover, this polynomial is *computable*. This constitutes the first computability result for the runtime complexity of non-linear programs that we are aware of.

Remark 8.1.2. Measuring vectors of integers via the 1-norm is standard in *automated complexity analysis* where one is interested in analyzing the efficiency of *implementations*. This is orthogonal to *complexity theory*, where the size of numbers is measured logarithmically and the goal is to characterize decision problems by proving membership or hardness w.r.t. complexity classes like P or NP (see Ch. 10).

This section is devoted to proving the following theorem.

Theorem 8.1.3 (Polynomial Loops Have Polynomial Runtime)**.** *Let $(\varphi, \vec{u})$ be a tnn-loop over $\mathbb{Z}$ and let $\vec{q} \in (\mathbb{PE}_{\mathbb{Z}}[\vec{x}])^d$ be the closed form for $\vec{u}$. Then there is a polynomial $f \in \mathbb{N}[y]$ such that*

$$rt_{(\varphi,\vec{u})}(\vec{e}) \leq f(\|\vec{e}\|) \qquad \text{for all } \vec{e} \in T_{(\varphi,\vec{u})},$$

where $\deg(f)$ is bounded by the maximal degree of the coefficients in $\varphi(\vec{q})$.

Note that Thm. 8.1.3 does not hold for loops over $\mathbb{Q}$ or $\mathbb{R}_{\mathbb{A}}$, as demonstrated by the following example.

Counterexample 8.1.4. Consider the loop $(x_1 \geq 0 \wedge x_2 > 0, (x_1 - x_2, x_2))$ over $\mathbb{Q}$ or $\mathbb{R}_{\mathbb{A}}$. It terminates after $\max\left\{0, \left\lceil \frac{x_1}{x_2} \right\rceil + 1\right\}$ iterations, but its runtime is not bounded by any continuous function $f : \mathbb{R} \longrightarrow \mathbb{R}$ in $\|(x_1, x_2)\|$. To see this, for any $k \in \mathbb{N}$ let $\vec{e}_k = \left(1, \frac{1}{k}\right)$ and let $f : \mathbb{R} \longrightarrow \mathbb{R}$ be continuous. Then $\lim_{k\mapsto\infty} \|\vec{e}_k\| = \lim_{k\mapsto\infty}(|1| + |\frac{1}{k}|) = 1$ and thus, $\lim_{k\mapsto\infty} f(\|\vec{e}_k\|) = f(1) \in \mathbb{R}$ by continuity. However, for any $k \in \mathbb{N}$ we have

$$rt_{(x_1\geq 0\wedge x_2>0,(x_1-x_2,x_2))}(\vec{e}_k) = \max\{0, \lceil k \rceil + 1\} = k + 1.$$

Hence,

$$\lim_{k\mapsto\infty} rt_{(x_1\geq 0\wedge x_2>0,(x_1-x_2,x_2))}(\vec{e}_k) = \lim_{k\mapsto\infty} (k + 1) = \infty.$$

Thus, we have

$$rt_{(x_1\geq 0\wedge x_2>0,(x_1-x_2,x_2))}(\vec{e}_k) > f(\|\vec{e}_k\|)$$

for large enough k. Hence, there is no continuous function $f : \mathbb{R} \longrightarrow \mathbb{R}$ with

$$rt_{(x_1\geq 0\wedge x_2>0,(x_1-x_2,x_2))}(\vec{e}_k) = k + 1 \leq f(\|\vec{e}_k\|)$$

for all $k \in \mathbb{N}$. Essentially, the problem is that measuring the "size" of non-integer inputs via the 1-norm $\|\cdot\|$ is not suitable for analyzing runtime complexity. The same problem arises with any other continuous function $\|\cdot\|' : \mathbb{R}^d \to \mathbb{R}$, e.g., any norm on $\mathbb{R}^d$, because then we have $\lim_{k\mapsto\infty} \|\vec{e}_k\|' = \|\lim_{k\mapsto\infty}(1, \frac{1}{k})\|' = \|(1,0)\|'$ and thus, for any continuous function f, $\lim_{k\mapsto\infty} f(\|\vec{e}_k\|')$ is a constant from $\mathbb{R}$. For that reason, we restricted ourselves to loops over $\mathbb{Z}$ in this section.

While Thm. 8.1.3 only applies to *tnn*-loops, it can also be used to obtain a bound on the complexity of other *twn*-loops due to the following lemma.

Lemma 8.1.5 (Chaining Preserves Asymptotic Runtime)**.** *Let $(\varphi, \vec{u})$ be a loop over $\mathbb{Z}$ and let $(\varphi_{\text{ch}}, \vec{u}_{\text{ch}}) = (\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$ be the corresponding chained loop. We have*

$$2 \cdot rt_{(\varphi_{\text{ch}},\vec{u}_{\text{ch}})}(\vec{e}) \;\le\; rt_{(\varphi,\vec{u})}(\vec{e}) \;\le\; 2 \cdot rt_{(\varphi_{\text{ch}},\vec{u}_{\text{ch}})}(\vec{e}) + 1 \qquad \text{for all } \vec{e} \in \mathbb{Z}^d.$$

From Thm. 8.1.3 and Lemma 8.1.5, we get the following corollary.

Corollary 8.1.6 (Linear Loops Have Linear Runtime)**.** *Let $(\varphi, \vec{u})$ be a triangular* linear *loop over $\mathbb{Z}$. Then there is a polynomial $f \in \mathbb{N}[y]$ with $\deg(f) = 1$ such that*

$$rt_{(\varphi,\vec{u})}(\vec{e}) \le f(\|\vec{e}\|) \qquad \text{for all } \vec{e} \in T_{(\varphi,\vec{u})}.$$

The reason is that chaining preserves linearity, i.e., if $(\varphi, \vec{u})$ is linear, $(\varphi_{\text{ch}}, \vec{u}_{\text{ch}})$ is linear, too. Hence, all coefficients of the closed form $\vec{q}$ of $\vec{u}^n_{\text{ch}}$ are also linear [FG19a].

We now prove Thm. 8.1.3. As we clearly have

$$rt_{(\varphi,\vec{u})}(\vec{e}) \le sth_{(\varphi,\vec{u})}(\vec{e}) \qquad \text{for all } \vec{e} \in T_{(\varphi,\vec{u})},$$

it suffices to find an upper bound on $sth_{(\varphi,\vec{u})}$ to derive an upper bound on $rt_{(\varphi,\vec{u})}$. Moreover, due to (7.1.1), w.l.o.g. we may restrict ourselves to loops of the form $(pol \rhd 0, \vec{u})$. More precisely, if $pol_1 \rhd_1 0, \dots, pol_r \rhd_r 0$ are all inequations occurring in φ, we compute a polynomial upper bound $f_i \in \mathbb{N}[y]$ on $sth_{(pol_i \rhd_i 0,\vec{u})}$ for each $1 \le i \le r$. Then $f_1 + \dots + f_r$ is an upper bound on $sth_{(\varphi,\vec{u})}$ since $f_1(y) + \dots + f_r(y) \ge \max\{f_1(y), \dots, f_r(y)\}$ for all $y \in \mathbb{N}$.

As in Ch. 7, let $\vec{q} \in (\mathbb{PE}_{\mathbb{Z}}[\vec{x}])^d$ be the closed form for $\vec{u}$ and let $p = pol(\vec{q})$. Moreover, let $c \in \mathbb{N}$ again be larger than any constant that occurs in a factor $[\psi]$ in p and let $p_{norm} \in \mathbb{NPE}_{\mathbb{Z}}[\vec{x}]$ be defined as in Ch. 7. As in (7.1.2), we therefore have

$$rt_{(pol \rhd 0,\vec{u})}(\vec{e}) \le sth_p(\vec{e}) \le \max\{sth_{p_{norm}}(\vec{e}), c\}.$$

Note that the degrees of the coefficients of p_{norm} are bounded by the degrees of the coefficients of p. Thus, to prove Thm. 8.1.3, it suffices to find a polynomial bound f

whose degree is bounded by the maximal degree of the coefficients in p_{norm}. So from now on, we assume $p \in \mathbb{NPE}_{\mathbb{Z}}[\vec{x}]$.

Furthermore, w.l.o.g. we may assume that

$$p = \sum_{j=1}^{\ell} \alpha_j \cdot n^{a_j} \cdot b_j^n \qquad \text{where } \alpha_j \in \mathbb{Z}[\vec{x}] \text{ (instead of } \alpha_j \in \mathbb{Q}[\vec{x}] = Q_{\mathbb{Z}}[\vec{x}]).$$

(Simplification 4)

The reason is that by multiplying with the denominators of all fractions that occur in some α_j, we obtain a $p' \in \mathbb{NPE}_{\mathbb{Z}}[\vec{x}]$ with $sth_p = sth_{p'}$ where the coefficients of p' are elements of $\mathbb{Z}[\vec{x}]$.

Example 8.1.7. Again, we consider the loop $\mathbf{L}_{ex}$ in Fig. 4.1(b) over $\mathbb{Z}$, where $\varphi(\vec{q}) \equiv p > 0$ with p as in (6.1.4). To compute a bound on sth_p, we can regard the expression $p > 0$ over $\mathbb{Z}$ instead, where

$$\begin{aligned} p = {} & \left(3 \cdot x_1 + 3 \cdot x_2^2\right) + \left(3 \cdot x_2^2 \cdot x_3 + 2 \cdot x_3^5 + 6 \cdot x_2 \cdot x_3^3 - 12 \cdot x_2 \cdot x_3^2\right) \cdot n \\ & + \left(-6 \cdot x_3^5 - 6 \cdot x_2 \cdot x_3^3 + 12 \cdot x_3^4\right) \cdot n^2 + \left(4 \cdot x_3^5\right) \cdot n^3. \end{aligned} \tag{8.1.1}$$

The following lemma allows us to compute a bound on the stabilization threshold $sth_p(\vec{e})$. More precisely, the lemma shows that $sth_p(\vec{e})$ can be over-approximated by a polynomial in the 1-norm of $\vec{e} \in \mathbb{Z}^d$, provided that $\alpha_\ell(\vec{e}) \neq 0$. Here, we (again) use the convention $\max \emptyset = 0$.

Lemma 8.1.8 (Bound on Stabilization Thresholds). *Let $p \in \mathbb{NPE}_{\mathbb{Z}}[\vec{x}]$ with $p = \sum_{j=1}^{\ell} \alpha_j \cdot n^{a_j} \cdot b_j^n$ and $(b_1, a_1) <_{lex} \ldots <_{lex} (b_\ell, a_\ell)$, where $\alpha_j \in \mathbb{Z}[\vec{x}]$ for all $1 \leq j \leq \ell$. Then there is a polynomial $f \in \mathbb{N}[y]$ with $\deg(f) = \max\left\{\deg(\alpha_j) \mid 1 \leq j < \ell\right\}$ such that*

$$sth_p(\vec{e}) \leq f(\|\vec{e}\|) \qquad \text{for all } \vec{e} \in \mathbb{Z}^d \text{ with } \alpha_\ell(\vec{e}) \neq 0.$$

Proof. If $\ell = 1$, the claim is trivial by choosing $f(y) = 1$. The reason is that for each $\vec{e}$ we have $p(\vec{e}) = \alpha_1(\vec{e}) \cdot n^{a_1} \cdot b_1^n$, whose sign is the sign of $\alpha_1(\vec{e}) \neq 0$ for every $n \in \mathbb{N}$ with $n \geq 1$.

Otherwise, let

$$\begin{aligned} m &= \max\left\{\deg(\alpha_j) \mid 1 \leq j < \ell\right\}, \\ k_{\max} &= \max\left\{|k| \mid 1 \leq j < \ell,\ k \text{ is a coefficient of } \alpha_j\right\}, \text{ and} \\ \alpha_{\max}(\vec{x}) &= k_{\max} \cdot \textstyle\sum_{i=0}^{m} \|\vec{x}\|^i. \end{aligned}$$

Then for all $1 \le j < \ell$ and all $\vec{e} \in \mathbb{Z}^d$ we have

$$\alpha_j(\vec{e}) \le \left|\alpha_j(\vec{e})\right| \le \max\left\{ |k| \;\middle|\; k \text{ is a coefficient of } \alpha_j \right\} \cdot \sum_{i=0}^{\deg(\alpha_j)} \|\vec{e}\|^i \le \alpha_{\max}(\vec{e}).$$

From now on, we fix $\vec{e} \in \mathbb{Z}^d$ with $\alpha_\ell(\vec{e}) \neq 0$.

If $\ell \ge 3$, let

$$mt = \max\left\{ \text{1-monotonicity threshold of } (b_j, a_j) \text{ and } (b_{\ell-2}, a_{\ell-2}) \;\middle|\; 1 \le j < \ell - 2 \right\},$$

let mt' be the $(\ell-2)$-monotonicity threshold of $(b_{\ell-2}, a_{\ell-2})$ and $(b_{\ell-1}, a_{\ell-1})$, and let $N = \max\{mt, mt'\}$. So for all $n \ge N$ we have

$$\sum_{j=1}^{\ell-2} n^{a_j} \cdot b_j^n \le \sum_{j=1}^{\ell-2} n^{a_{\ell-2}} \cdot b_{\ell-2}^n = (\ell-2) \cdot n^{a_{\ell-2}} \cdot b_{\ell-2}^n < n^{a_{\ell-1}} \cdot b_{\ell-1}^n.$$

If $\ell = 2$, we define $N = 1$. Thus, we get:

$$\begin{aligned}
&\left| \sum_{j=1}^{\ell-1} \alpha_j(\vec{e}) \cdot n^{a_j} \cdot b_j^n \right| \\
&\le \sum_{j=1}^{\ell-1} \left|\alpha_j(\vec{e})\right| \cdot n^{a_j} \cdot b_j^n \\
&\le \sum_{j=1}^{\ell-1} \alpha_{\max}(\vec{e}) \cdot n^{a_j} \cdot b_j^n \\
&= \alpha_{\max}(\vec{e}) \cdot \sum_{j=1}^{\ell-1} n^{a_j} \cdot b_j^n \\
&= \alpha_{\max}(\vec{e}) \cdot \left(\sum_{j=1}^{\ell-2} n^{a_j} \cdot b_j^n + n^{a_{\ell-1}} \cdot b_{\ell-1}^n \right) \\
&< 2 \cdot \alpha_{\max}(\vec{e}) \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n \qquad \text{for all } n \ge N
\end{aligned}$$

We proceed by a case analysis.

Case $b_\ell = b_{\ell-1}$: As $(b_{\ell-1}, a_{\ell-1}) <_{lex} (b_\ell, a_\ell)$ we have $a_{\ell-1} < a_\ell$ and hence

$$\begin{aligned}
&2 \cdot \alpha_{\max}(\vec{e}) \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n \\
&\le n^{a_{\ell-1}+1} \cdot b_\ell^n \qquad \text{for all } n \ge 2 \cdot \alpha_{\max}(\vec{e}) \\
&\le n^{a_\ell} \cdot b_\ell^n.
\end{aligned}$$

Thus, we obtain

$$\left|\sum_{j=1}^{\ell-1} \alpha_j(\vec{e}) \cdot n^{a_j} \cdot b_j^n\right| < n^{a_\ell} \cdot b_\ell^n \qquad \text{for all } n \geq \max\{N, 2 \cdot \alpha_{\max}(\vec{e})\}. \tag{8.1.2}$$

Case $b_{\ell-1} < b_\ell$: Then $(b_{\ell-1}, a_{\ell-1}+1) <_{lex} (b_\ell, a_\ell)$. Let $M \in \mathbb{N}$ be the 1-monotonicity threshold of $(b_{\ell-1}, a_{\ell-1}+1)$ and (b_ℓ, a_ℓ). Then we have

$$2 \cdot \alpha_{\max}(\vec{e}) \cdot n^{a_{\ell-1}} \cdot b_{\ell-1}^n \leq n^{a_{\ell-1}+1} \cdot b_{\ell-1}^n < n^{a_\ell} \cdot b_\ell^n$$

for all $n \geq \max\{M, 2 \cdot \alpha_{\max}(\vec{e})\}$ and thus, we obtain

$$\left|\sum_{j=1}^{\ell-1} \alpha_j(\vec{e}) \cdot n^{a_j} \cdot b_j^n\right| < n^{a_\ell} \cdot b_\ell^n \qquad \text{for all } n \geq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\}. \tag{8.1.3}$$

From (8.1.2) and (8.1.3), we get

$$\forall n \geq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\}. \quad \left|\sum_{j=1}^{\ell-1} \alpha_j(\vec{e}) \cdot n^{a_j} \cdot b_j^n\right| < n^{a_\ell} \cdot b_\ell^n$$

(where we define $M = 0$ if $b_\ell = b_{\ell-1}$).

Recall that we restricted ourselves to loops over $\mathbb{Z}$ (i.e., we have $\vec{e} \in \mathbb{Z}^d$) and to poly-exponential expressions where the polynomials α_j only have integer coefficients (see Simplification 4). Thus, if $\alpha_\ell(\vec{e}) > 0$, $\alpha_\ell \in \mathbb{Z}[\vec{x}]$ and $e \in \mathbb{Z}^d$ imply $\alpha_\ell(\vec{e}) \geq 1$. Hence, if $n \geq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\}$, then $p[\vec{x}/\vec{e}] > (\alpha_\ell(\vec{e}) - 1) \cdot n^{a_\ell} \cdot b_\ell^n \geq 0$ and hence $\text{sign}\left(p[\vec{x}/\vec{e}]\right) = 1$. Similarly, if $\alpha_\ell(\vec{e}) < 0$, then $\alpha_\ell \in \mathbb{Z}[\vec{x}]$ and $e \in \mathbb{Z}^d$ imply $\alpha_\ell(\vec{e}) \leq -1$. So if $n \geq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\}$, then $p[\vec{x}/\vec{e}] < (\alpha_\ell(\vec{e}) + 1) \cdot n^{a_\ell} \cdot b_\ell^n \leq 0$ and hence, $\text{sign}\left(p[\vec{x}/\vec{e}]\right) = -1$. This implies

$$sth_p(\vec{e}) \leq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\} \qquad \text{for all } \vec{e} \in \mathbb{Z}^d \text{ with } \alpha_\ell(\vec{e}) \neq 0. \tag{8.1.4}$$

It remains to show that there is a polynomial $f \in \mathbb{N}[y]$ with $\deg(f) = m$ such that $f(\|\vec{e}\|) \geq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\}$ for all $\vec{e} \in \mathbb{Z}^d$. We define

$$f(y) = 2 \cdot k_{\max} \cdot \sum_{i=0}^{m} y^i + \max\{N, M\},$$

where we have $f \in \mathbb{N}[y]$ as $\alpha_j \in \mathbb{Z}[\vec{x}]$ implies $k_{\max} \in \mathbb{N}$. Then for any $\vec{e} \in \mathbb{Z}^d$ we get

$$\begin{aligned} f(\|\vec{e}\|) &= 2 \cdot k_{\max} \cdot \sum_{i=0}^{m} \|\vec{e}\|^i + \max\{N, M\} \\ &= 2 \cdot \alpha_{\max}(\vec{e}) + \max\{N, M\} && \text{(by definition of } \alpha_{\max}) \\ &\geq \max\{N, M, 2 \cdot \alpha_{\max}(\vec{e})\}. \end{aligned}$$

□

Example 8.1.9. We continue Ex. 8.1.7. To compute a bound on the runtime of $\mathbf{L}_{ex}$, we construct a polynomial upper bound f on sth_p. Following the proof of Lemma 8.1.8, we have $\ell = 4$, $m = 5$, and $k_{\max} = 12$ since

$$\alpha_1 = 3 \cdot x_1 + 3 \cdot x_2^2, \qquad \alpha_2 = 3 \cdot x_2^2 \cdot x_3 + 2 \cdot x_3^5 + 6 \cdot x_2 \cdot x_3^3 - 12 \cdot x_2 \cdot x_3^2,$$
$$\alpha_3 = -6 \cdot x_3^5 - 6 \cdot x_2 \cdot x_3^3 + 12 \cdot x_3^4, \qquad \text{and} \qquad \alpha_4 = 4 \cdot x_3^5,$$

Moreover, we have $mt = 2$ (as the 1-monotonicity threshold of $(1, 0)$ and $(1, 1)$ is 2) and $mt' = 3$ (as the 2-monotonicity threshold of $(1, 1)$ and $(1, 2)$ is 3). Thus, we get $N = \max\{mt, mt'\} = 3$. Furthermore, we have $M = 0$, as $b_4 = b_3 = 1$. Hence,

$$f(y) = 2 \cdot k_{\max} \cdot \sum_{i=0}^{m} y^i + \max\{N, M\} = 24 \cdot \left(1 + \sum_{i=1}^{5} y^i\right) + 3 = 24 \cdot \sum_{i=1}^{5} y^i + 27.$$

Note that $\alpha_\ell(\vec{e}) = \alpha_4(\vec{e}) \neq 0$ for all $\vec{e} \in \mathbb{Z}^3$ with $e_3 \neq 0$. Thus, $sth_p(e_1, e_2, e_3)$ is bounded by $f(\|(e_1, e_2, e_3)\|) = 24 \cdot \sum_{i=1}^{5} (|e_1| + |e_2| + |e_3|)^i + 27$ for all $(e_1, e_2, e_3) \in \mathbb{Z}^3$ with $e_3 \neq 0$. However, if $e_3 = 0$, then $p(e_1, e_2, e_3) = 3 \cdot e_1 + 3 \cdot e_2^2$ is constant in n, i.e., in this case $sth_p(e_1, e_2, e_3) = 0$. Thus, $sth_p(e_1, e_2, e_3)$ is bounded by $f(\|(e_1, e_2, e_3)\|)$ for all $(e_1, e_2, e_3) \in \mathbb{Z}^3$.

Consequently, the runtime of $\mathbf{L}_{ex}$ is also bounded by $f(\|(e_1, e_2, e_3)\|)$ for all $(e_1, e_2, e_3) \in T_{\mathbf{L}_{ex}}$.

Counterexample 8.1.10. The loop from Counterex. 8.1.4 also shows that Lemma 8.1.8 does *not* hold for rings like $\mathbb{Q}$ or $\mathbb{R}_{\mathbb{A}}$. Here, the closed form of the n-fold update is $\vec{q} = (x_1 - n \cdot x_2, x_2)$. Thus, $\varphi(\vec{q})$ is $x_1 - n \cdot x_2 \geq 0 \wedge x_2 > 0$. When considering $p = x_1 - n \cdot x_2$, then we have $sth_p(\vec{e}_k) = k + 1$ for $\vec{e}_k = \left(1, \frac{1}{k}\right)$. But

as shown in Counterex. 8.1.4, there is no continuous function $f : \mathbb{R} \longrightarrow \mathbb{R}$ with $sth_{pe}(\vec{e}_k) = k + 1 \leq f(\|\vec{e}_k\|) = f\left(1 + \frac{1}{k}\right)$ for all $k \in \mathbb{N}$.

Indeed, the construction of f in the proof of Lemma 8.1.8 would no longer yield an upper bound on sth_{pe}. In our example, we have $\ell = 2$, $\alpha_1 = x_1$, and $\alpha_2 = -x_2$. Hence, $m = 1$ and $k_{\max} = 1$. Finally, since $\ell = 2$, we have $N = 1$ and since $b_1 = b_2 = 1$, we have $M = 0$. Thus, if we construct f as in the proof of Lemma 8.1.8, we get $f(y) = 2 \cdot k_{\max} \cdot \sum_{i=0}^{m} y^i + \max\{N, M\} = 2 \cdot y + 3$. So $f(\|\vec{e}_k\|) = 5 + \frac{2}{k}$, but $f(\|\vec{e}_5\|) = 5 + \frac{2}{5} < 6 = sth_p(\vec{e}_5)$, i.e., f is not an upper bound on sth_p.

According to the preconditions of Lemma 8.1.8, the resulting bound f is only valid for all $\vec{e} \in \mathbb{Z}^d$ with $\alpha_\ell(\vec{e}) \neq 0$. However, for every $\vec{e} \in \mathbb{Z}^d$, we either have $p[\vec{x}/\vec{e}] = 0$ (and thus $sth_p(\vec{e}) = 0$) or there is an $1 \leq \ell_{\vec{e}} \leq \ell$ such that $|\alpha_{\ell_{\vec{e}}}(\vec{e})| \geq 1$ and $\alpha_j(\vec{e}) = 0$ for all $\ell_{\vec{e}} < j \leq \ell$. So by applying Lemma 8.1.8 to $\sum_{j=1}^{\ell_{\vec{e}}} \alpha_j \cdot n^{a_j} \cdot b_j^n$, we obtain a polynomial $f_{\ell_{\vec{e}}}$ such that $sth_p(\vec{e}) \leq f_{\ell_{\vec{e}}}(\|\vec{e}\|)$. Thus, by applying Lemma 8.1.8 to each expression $\sum_{j=1}^{\ell'} \alpha_j \cdot n^{a_j} \cdot b_j^n$ with $1 \leq \ell' \leq \ell$, we obtain polynomials $f_1, \ldots, f_\ell$ such that for each $\vec{e} \in \mathbb{Z}^d$, there is *some* f_j with $sth_p(\vec{e}) \leq f_j(\|\vec{e}\|)$. Hence, for $f = \sum_{j=1}^{\ell} f_j$ we obtain $sth_p(\vec{e}) \leq f(\|\vec{e}\|)$ since $f_j \in \mathbb{N}[y]$ for all $1 \leq j \leq \ell$. As the definition of f is independent of $\vec{e}$, this finishes the proof of Thm. 8.1.3.

Our technique to compute runtime bounds via Thm. 8.1.3 and Lemma 8.1.5 can be implemented as in Algorithm 8.1.1. For Simplification 1, the loop is again chained in Line 1 if necessary. In that case, the difference between the complexity of the original and the chained loop is taken into account in Line 13, where the result is adapted according to Lemma 8.1.5. Next, f is initialized to the polynomial 1 and afterwards, we again process each inequation $p \triangleright 0$ in $\varphi(\vec{q})$ separately, see Simplification 2. Line 5 implements Simplification 3 and Line 6 corresponds to Simplification 4. As explained after its proof, we have to apply Lemma 8.1.8 to each expression $\sum_{j=1}^{\ell'} \alpha_j \cdot n^{a_j} \cdot b_j^n$ with $1 \leq \ell' \leq \ell$ and the obtained bounds have to be added in order to get a bound on sth_p. As in the proof of Lemma 8.1.8, we can skip poly-exponential expressions with just a single addend (i.e., where $\ell' = 1$). Thus, in Line 10, f is updated to the sum of its previous value and the upper bound on $\sum_{j=1}^{\ell'} \alpha_j \cdot n^{a_j} \cdot b_j^n$. By adding the bounds on sth_p for all $p \triangleright 0$ occurring in $\varphi(\vec{q})$, Line 10 also reflects the polynomial approximation of the max-expressions from (7.1.1) via a sum. Similarly, Line 12 approximates the max-expression from (7.1.2) via summation.

When comparing Algorithm 7.1.1 and Algorithm 8.1.1, note that Algorithm 7.1.1 computes an upper bound on $sth_{(\varphi,\vec{u})}(\vec{e})$ for a *given* input $\vec{e}$, whereas Algorithm 8.1.1 computes an upper bound on the stabilization threshold $sth_{(\varphi,\vec{u})}$ *for all* inputs $\vec{e}$. More precisely, for *twn*-loops over $\mathbb{Z}$, Algorithm 8.1.1 computes a polynomial f such that

Algorithm 8.1.1: Computing Runtime Bounds

Input: a *twn*-loop $(\varphi_{in}, \vec{u}_{in})$ on $\mathbb{Z}^d$
Output: a polynomial $f \in \mathbb{N}[y]$ with $rt_{(\varphi_{in},\vec{u}_{in})}(\vec{e}) \leq f(\|\vec{e}\|)$ for all $\vec{e} \in T_{(\varphi_{in},\vec{u}_{in})}$

1 **if** $(\varphi_{in}, \vec{u}_{in})$ *is tnn* **then** $(\varphi, \vec{u}) \leftarrow (\varphi_{in}, \vec{u}_{in})$ **else** $(\varphi, \vec{u}) \leftarrow (\varphi_{in} \wedge \varphi_{in}(\vec{u}_{in}), \vec{u}_{in}(\vec{u}_{in}))$
2 $\vec{q} \leftarrow$ closed form for $\vec{u}$ with $\vec{q} \in (\mathbb{PE}_{\mathbb{Z}}[\vec{x}])^d$
3 $f \leftarrow 1$
4 **foreach** *inequation* $p \rhd 0$ *occurring in* $\varphi(\vec{q})$ **do**
5 $\quad p \leftarrow p_{norm}$
6 $\quad p \leftarrow p \cdot \prod_{k \text{ occurs as denominator in } p} k$
7 $\quad$ let $p = \sum_{j=1}^{\ell} \alpha_j \cdot n^{a_j} \cdot b_j^n$ with $(b_1, a_1) <_{lex} \ldots <_{lex} (b_\ell, a_\ell)$
8 $\quad$ **foreach** ℓ' *with* $2 \leq \ell' \leq \ell$ **do**
9 $\quad\quad$ compute m, $k_{\max}$, N, and M for $\sum_{j=1}^{\ell'} \alpha_j \cdot n^{a_j} \cdot b_j^n$ as in the proof of Lemma 8.1.8
10 $\quad\quad f \leftarrow f + 2 \cdot k_{\max} \cdot \sum_{i=0}^{m} y^i + \max\{N, M\}$
11 $c \leftarrow 1 +$ the maximal constant that occurs in any factor $[\psi]$ in $\varphi(\vec{q})$
12 $f \leftarrow f + c$
13 **if** $(\varphi_{in}, \vec{u}_{in})$ *is tnn* **then return** f **else return** $2 \cdot f + 1$

$sth_{(\varphi,\vec{u})}(\vec{e}) \leq f(\|\vec{e}\|)$ holds for all $\vec{e} \in \mathbb{Z}^d$. So for such loops, Algorithm 8.1.1 could also be used to decide the halting problem since $(\varphi, \vec{u})$ is non-terminating on $\vec{e}$ iff $\varphi(\vec{q}[\vec{x}/\vec{e}])$ holds for all $0 \leq n \leq f(\|\vec{e}\|)$. However, in contrast to Algorithm 8.1.1, Algorithm 7.1.1 can be used to decide the halting problem for loops on *arbitrary* rings $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathrm{A}}$. The reason is that Lemma 7.1.7 (which is the basis of Algorithm 7.1.1) holds for $\mathbb{R}_{\mathrm{A}}$ (and therefore also for its subrings), whereas Lemma 8.1.8 (which is the basis of Algorithm 8.1.1) only holds for the ring $\mathbb{Z}$.

8.2 Computing Asymptotic Runtime Bounds

In this section, we present an additional insight concerned with the computation of an *asymptotic runtime bound* for a *tnn*-loop $(\varphi, \vec{u})$, which has not been published yet. By Lemma 8.1.8, the degree of the upper bound on the runtime of a *tnn*-loop which is computed by Algorithm 8.1.1 is bounded by the maximal degree of the coefficients of $\varphi(\vec{q}_{norm})$, where $\vec{q}_{norm}$ is the normalized closed form for $\vec{u}$. Thus, this maximal

degree induces an *asymptotic* upper bound on the runtime. So far, we have to compute $\varphi(\vec{q}_{norm})$ to read off this maximal degree from the coefficients. In this section, we show how to obtain an upper bound on this degree directly from the loop, i.e., *without* computing the closed form $\vec{q}_{norm}$. Thus, if one is interested only in an asymptotic runtime bound, this can be computed more efficiently than with Algorithm 8.1.1 (see Cor. 8.2.4). Note that Cor. 8.2.4 also applies to *twn*-loops by chaining the loop since by Lemma 8.1.5 this does not affect the asymptotic complexity.

However, it is worth noting that to reduce termination of a *tnn*-loop over $\mathbb{Z}$ to a $\mathrm{Th}_\exists(\mathbb{Z}, \mathbb{R}_\mathbb{A})$-formula with Algorithm 6.1.1, the closed forms have to be computed anyway. Thus, the result presented here is more a structural result on *tnn*-loops since an asymptotic upper bound only applies to *terminating* inputs. Nevertheless, the insights we gain when proving this result are crucial for our result on the complexity of deciding termination for arbitrary *twn*-loops in Sect. 10.2.

For any polynomial $r \in \mathbb{R}_\mathbb{A}[\vec{x}]$ we denote by $\deg_{x_j}(r)$ the highest power of x_j occurring in a monomial of the polynomial r, i.e., the degree of r when interpreting all variables besides x_j as constants.

Definition 8.2.1 (Dependency Degree)**.** Let $(\varphi, \vec{u})$ be *twn*. We define the *dependency degree* w.r.t. $\vec{u}$ for a variable x_i with $1 \leq i \leq d$ by

$$\mathrm{depdeg}_{\vec{u}}(x_i) = \max\left(\{\deg_{x_i}(u_i)\} \uplus D_{\vec{u}}(x_i)\right)$$

where

$$D_{\vec{u}}(x_i) = \left\{ \sum_{x_j \in \mathcal{V}(m)} \deg_{x_j}(m) \cdot \mathrm{depdeg}_{\vec{u}}(x_j) \;\middle|\; m \text{ monomial in } u_i, x_i \neq m \right\}.$$

Since $\succ_{\vec{u}}$ is well founded, $\mathrm{depdeg}_{\vec{u}}$ is well defined: for the variables x_i which are minimal elements w.r.t. $\succ_{\vec{u}}$ we have $D_{\vec{u}}(x_i) = \emptyset$ and thus, $\mathrm{depdeg}_{\vec{u}}(x_i) = \deg_{x_i}(u_i)$. It is worth noting that the dependency degree of a variable is an *elementary* property of the loop, i.e., it can be obtained from the loop using only elementary arithmetic operations. Moreover, Def. 8.2.1 applies to arbitrary *twn*-loops. We will use this notion also in Sect. 10.2.

Intuitively, the dependency degree $\mathrm{depdeg}_{\vec{u}}(x_i)$ describes the degree of the update of x_i in terms of the variables which are the minimal elements w.r.t. $\succ_{\vec{u}}$, where we take into account whether x_i occurs in u_i.

Example 8.2.2. Again, we consider the loop $\mathbf{L}_{ex}$ from Fig. 4.1(b) with its update $\vec{u} = (x_1 + x_2^2 \cdot x_3, x_2 - 2 \cdot x_3^2, x_3)$. Then, $\mathrm{depdeg}_{\vec{u}}(x_3) = 1$, $\mathrm{depdeg}_{\vec{u}}(x_2) = \max\left\{\deg_{x_2}(u_2), 2 \cdot \mathrm{depdeg}_{\vec{u}}(x_3)\right\} = \max\{1, 2\} = 2$, and $\mathrm{depdeg}_{\vec{u}}(x_1) = \max\left\{1, 2 \cdot \mathrm{depdeg}_{\vec{u}}(x_2) + 1 \cdot \mathrm{depdeg}_{\vec{u}}(x_3)\right\} = 2 \cdot 2 + 1 \cdot 1 = 5$. So intuitively, the update of x_1 has degree 5 in terms of x_3 since the update of x_2 is quadratic in x_3 and $x_2^2 \cdot x_3$ then has degree 5 in this sense.

However, if the update of x_3 were the constant $u_3 = 1$, we would have $\mathrm{depdeg}\, x_3 = 0$, $\mathrm{depdeg}_{\vec{u}}(x_2) = 1$, and $\mathrm{depdeg}_{\vec{u}}(x_2) = 2$.

We prove the following lemma in Sect. 8.3 which shows that when computing the degree of the condition φ in such a way that we measure the degree of a variable x_i not with 1 but with its dependency degree then we obtain the maximal degree of the coefficients of $\varphi(\vec{q}_{norm})$. Here, the monomials of a polynomial $r \in \mathbb{R}_A[\vec{x}]$ are the monomials occurring with a non-zero coefficient in r.

Lemma 8.2.3 (Dependency Degrees Induce Asymptotic Upper Bound)**.** *Let $(\varphi, \vec{u})$ be a tnn-loop and let $\vec{q}_{norm}$ be a normalized closed form for $\vec{u}$. Then the maximal degree of the coefficients of $\varphi(\vec{q}_{norm})$ is bounded by*

$$\max\left\{ \sum_{i=1}^{d} \deg_{x_i}(m) \cdot \mathrm{depdeg}_{\vec{u}}(x_i) \;\middle|\; r \trianglerighteq 0 \text{ atom in } \varphi,\ m \text{ monomial in } r \right\}.$$

From Lemmas 8.2.3 and 8.1.8 we directly obtain the following corollary.

Corollary 8.2.4. *Let $(\varphi, \vec{u})$ be a tnn-loop over $\mathbb{Z}$. Then*

$$\max\left\{ \sum_{i=1}^{d} \deg_{x_i}(m) \cdot \mathrm{depdeg}_{\vec{u}}(x_i) \;\middle|\; r \trianglerighteq 0 \text{ atom in } \varphi,\ m \text{ monomial in } r \right\}$$

is an upper bound on the degree of the polynomial computed by Algorithm 8.1.1.

Example 8.2.5. Reconsider the setting from Ex. 8.2.2. The condition of the loop $\mathbf{L}_{ex}$ consists of a single atom containing the monomials x_1 and x_2^2. Thus,

$$\max\left\{ \sum_{i=1}^{3} \deg_{x_i}(m) \cdot \mathrm{depdeg}_{\vec{u}}(x_i) \;\middle|\; r \trianglerighteq 0 \text{ atom in } \varphi,\ m \text{ monomial in } r \right\}$$
$$= \max\{1 \cdot 5,\ 2 \cdot 2\} = 5.$$

Indeed, the polynomial bound for the runtime of $\mathbf{L}_{ex}$ which we have computed in Ex. 8.1.9 has degree 5.

We will prove Lemma 8.2.3 by *linearizing* the *twn*-loop and using some structural insights on the linearization procedure in the next section.

8.3 Linearizing *tnn*-Loops

We consider the procedure from [OBP16a] to linearize a *tnn*-loop over any ring $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathrm{A}}$. However, [OBP16a] only provides a sketch of this procedure. Thus, we explain it in more detail and adapt it to our notation. The statements which are taken from or based on [OBP16a] are clearly marked. All other results are novel structural insights. These yield a proof for Lemma 8.2.3 and enable an analysis of the complexity of deciding termination for arbitrary *twn*-loops in Sect. 10.2. Before we go into detail, let us define the term *linearization* formally.

Definition 8.3.1 (Linearization of a Vector of Polynomials, [OBP16a])**.** Let $\vec{u} \in \mathcal{S}[\vec{x}]^d$ and $\vec{x}\,'$ be a vector of s fresh variables. If there is a *linear* $\vec{u}\,' \in \mathcal{S}[\vec{x}, \vec{x}\,']^{d+s}$ and a $\vec{w} \in \mathcal{S}[\vec{x}]^s$ such that for all $\vec{e} \in \mathcal{S}^d$ we have

$$(\vec{u}(\vec{e}), \vec{w}(\vec{u}(\vec{e}))) = \vec{u}\,'(\vec{e}, \vec{w}(\vec{e})),$$

we call $\vec{u}\,'$ a *linearization via* $\vec{w}$ of $\vec{u}$.

So intuitively, the fresh variables $\vec{x}\,'$ are used to mimic the non-linear part from $\vec{u}$ in a linear way in $\vec{u}\,'$. This non-linear behavior is captured by the polynomials in $\vec{w}$.

Example 8.3.2. Consider $\vec{u} = (x_2^2, x_3^2, x_3) \in \mathbb{Z}[x_1, x_2, x_3]^3$. Then,

$$\vec{u}\,' = (x_1',\ x_3',\ x_3,\ x_2',\ x_2',\ x_3')$$

over the variables $(x_1, x_2, x_3, x_1', x_2', x_3')$ is a linearization of $\vec{u}$ via $\vec{w} = (x_2^2, x_3^4, x_3^2)$ since we have for all $\vec{e} \in \mathbb{Z}^3$

$$(\vec{u}(\vec{e}), \vec{w}(\vec{u}(\vec{e}))) = (e_2^2, e_3^2, e_3, e_3^4, e_3^4, e_3^2) = \vec{u}\,'(\vec{e}, \vec{w}(\vec{e})).$$

Here, the non-linear part of $\vec{u}$ is mimicked by the variables $\vec{x}\,'$ in combination with the vector $\vec{w}$.

By Def. 8.3.1, we observe the following.

Corollary 8.3.3. *Let $\vec{u} \in \mathcal{S}[\vec{x}]^d$ and $\vec{u}' \in \mathcal{S}[\vec{x}, \vec{x}']^{d+s}$ be its linearization via $\vec{w} \in \mathcal{S}[\vec{x}]^s$. Then for all $\vec{e} \in \mathcal{S}^d$ and $n \in \mathbb{N}$ we have*

$$(\vec{u}^n(\vec{e}), \vec{w}(\vec{u}^n(\vec{e}))) = (\vec{u}')^n(\vec{e}, \vec{w}(\vec{e})).$$

We can now formally define the linearization of a loop to be a linearization of its update enriched with a "consistency check" in the condition of the resulting loop which ensures that the fresh variables indeed mimic the non-linear behavior.

Definition 8.3.4 (Linearization of a Loop)**.** Let $(\varphi, \vec{u})$ be a loop on $\mathcal{S}^d$ over the variables $\vec{x}$. A loop $(\varphi', \vec{u}')$ on $\mathcal{S}^{d+s}$ over the $d + s$ variables $\vec{y} = (\vec{x}, \vec{x}')$ is a *linearization of* $(\varphi, \vec{u})$ *via* $\vec{w} \in \mathcal{S}[\vec{x}]^s$ if

(i) $\vec{u}'$ is a linearization via $\vec{w}$ of $\vec{u}$

(ii) $\varphi' = \varphi \wedge \bigwedge_{j=1}^{s} \left(x_j' - w_j = 0\right)$.

Example 8.3.5. Let $(\varphi, \vec{u})$ be any loop on $\mathbb{Z}^3$ with $\vec{u} = (x_2^2, x_3^2, x_3)$. Then, the linearization of $(\varphi, \vec{u})$ via $\vec{w}$ as in Ex. 8.3.2 is $(\varphi', \vec{u}')$ where $\vec{u}'$ is given in Ex. 8.3.2 and

$$\varphi' = \varphi \wedge (x_1' - x_2^2) = 0 \wedge (x_2' - x_3^4) = 0 \wedge (x_3' - x_3^2) = 0.$$

We now state a few observations. Note that Lemma 8.3.6 (i) and (ii) justify why [OBP16a] and [Wu+10; XZ10] indeed prove a restricted version of Cor. 6.1.11 (ii) (see Ch. 6). Moreover, Lemma 8.3.6 (i) and (ii) will also be used to analyze the complexity of deciding termination of *twn*-loops in Sect. 10.2. Finally, Lemma 8.3.6 (iii) is stated as a "warning", which we briefly discuss after its proof.

Lemma 8.3.6. *Let $(\varphi, \vec{u})$ be a loop on $\mathcal{S}^d$ over the d variables $\vec{x}$ and let the loop $(\varphi', \vec{u}')$ on $\mathcal{S}^{d+s}$ over the variables $\vec{y} = (\vec{x}, \vec{x}')$ be a linearization via $\vec{w}$ of $(\varphi, \vec{u})$.*

(i) If for $\vec{e}' \in \mathcal{S}^{d+s}$ there is no $\vec{e} \in \mathcal{S}^d$ such that $\vec{e}' = (\vec{e}, \vec{w}(\vec{e}))$, the loop $(\varphi', \vec{u}')$ terminates on $\vec{e}'$.

(ii) The loop $(\varphi, \vec{u})$ terminates on $\vec{e} \in \mathcal{S}^d$ iff $(\varphi', \vec{u}')$ terminates on $(\vec{e}, \vec{w}(\vec{e}))$.

(iii) If $(\varphi, \vec{u})$ is a loop over $\mathbb{Z}$ and $\vec{w} \in \mathbb{Z}[\vec{x}]^s$, then $(\varphi', \vec{u}')$ is a loop over $\mathbb{Z}$. Furthermore,

$$rt_{(\varphi,\vec{u})}(\vec{e}) = rt_{(\varphi',\vec{u}')}(\vec{e}, \vec{w}(\vec{e})) \text{ for all } \vec{e} \in \mathbb{Z}^d.$$

Proof. We prove the statements separately.

(i) If for $\vec{e}\,' \in \mathcal{S}^{d+s}$ there is no $\vec{e} \in \mathcal{S}^d$ such that $\vec{e}\,' = (\vec{e}, \vec{w}(\vec{e}))$, then $\varphi'[\vec{y}/\vec{e}\,']$ is false by Def. 8.3.4 (ii).

(ii) We have

$$\begin{aligned}
&\varphi'[\vec{y}/(\vec{u}\,')^n(\vec{e}, \vec{w}(\vec{e}))] \\
\iff &\varphi'[\vec{x}/\vec{u}^n(\vec{e}), \vec{x}\,'/\vec{w}(\vec{u}^n(\vec{e}))] && \text{(by Cor. 8.3.3)} \\
\iff &\varphi[\vec{x}/\vec{u}^n(\vec{e})] \wedge \bigwedge_{j=1}^{s} \left(x_j' - w_j = 0\right)[\vec{x}/\vec{u}^n(\vec{e}), \vec{x}\,'/\vec{w}(\vec{u}^n(\vec{e}))] && \text{(by Def. 8.3.4 (ii))} \\
\iff &\varphi[\vec{x}/\vec{u}^n(\vec{e})] \wedge \bigwedge_{j=1}^{s} (0 = 0) \\
\iff &\varphi[\vec{x}/\vec{u}^n(\vec{e})],
\end{aligned}$$

for any $n \in \mathbb{N}$. Hence, $(\varphi, \vec{u})$ terminates on $\vec{e} \in \mathcal{S}^d$ iff $(\varphi', \vec{u}\,')$ terminates on $(\vec{e}, \vec{w}(\vec{e}))$.

(iii) If $(\varphi, \vec{u})$ and $\vec{w} \in \mathbb{Z}[\vec{x}]^s$, then $(\varphi', \vec{u}\,')$ is a loop over $\mathbb{Z}$ by definition. By the proof of the previous statement,

$$rt_{(\varphi,\vec{u})}(\vec{e}) = rt_{(\varphi',\vec{u}\,')}(\vec{e}, \vec{w}(\vec{e})) \text{ for all } \vec{e} \in \mathbb{Z}^d,$$

follows as well.

□

Thus, a linearization of a polynomial loop has identical termination behavior and even the runtimes coincide on certain (non-trivial) inputs. However, note that in general $\|(\vec{e}, w(\vec{e}))\|$ is *polynomial* in $\|\vec{e}\|$, i.e., while the runtimes in the sense of the actual values coincide, this relationship has to be taken into account when measuring the runtime in the *size* of the input.

Example 8.3.7. Consider the setting from Ex. 8.3.5. By Lemma 8.3.6 (iii),

$$rt_{(\varphi,\vec{u})}(e_1, e_2, e_3) = rt_{(\varphi,\vec{u})}(\vec{e}) = rt_{(\varphi',\vec{u}\,')}(\vec{e}, \vec{w}(\vec{e})) = rt_{(\varphi',\vec{u}\,')}(e_1, e_2, e_3, e_2^2, e_3^4, e_3^2).$$

Still, $\|(e_1, e_2, e_3, e_2^2, e_3^4, e_3^2)\|$ is polynomial in terms of $\|(e_1, e_2, e_3)\| = \|\vec{e}\|$. This has

to be taken into account when deriving the runtime of $(\varphi, \vec{u})$ in the size of the input $\vec{e}$ from the runtime of $(\varphi', \vec{u}')$ measured in the size of the input $(\vec{e}, \vec{w}(\vec{e}))$.

Finally, we can state the first fundamental property which is needed for the proof of Lemma 8.2.3.

Lemma 8.3.8. *Let $(\varphi, \vec{u})$ be a tnn-loop over $\mathcal{S}$ with linearization $(\varphi', \vec{u}')$ via $\vec{w}$ such that $(\varphi', \vec{u}')$ is a linear-update tnn-loop. Let $\vec{q}_{norm} \in \mathbb{NPE}_{\mathcal{S}}[\vec{x}]^d$ be the normalized closed form for $\vec{u}$. For all $1 \leq i \leq d$ the maximal degree occurring in a coefficient in the normalized closed form $(\vec{q}_{norm})_{x_i}$ of x_i is bounded by*

$$\max\left(\left\{\operatorname{depdeg}_{\vec{u}'}(x_s) \mid x_i \succeq_{\vec{u}'} x_s\right\} \uplus \left\{\deg(w_j) \mid x_i \succ_{\vec{u}'} x'_j\right\}\right),$$

where we again use $\max \emptyset = 0$.

Proof. Since $\vec{u}'$ is linear, its normalized closed form $\vec{q}'_{norm} \in \mathbb{NPE}_{\mathcal{S}}[\vec{x}, \vec{x}']^{d+s}$ only contains poly-exponential expressions involving linear coefficients (see [FG19a, Thm. 17]). By Cor. 8.3.3 we have

$$(\vec{u}^n(\vec{e}), \vec{w}(\vec{u}^n(\vec{e}))) = (\vec{u}')^n(\vec{e}, \vec{w}(\vec{e}))$$

for all $n \in \mathbb{N}$ and $\vec{e} \in \mathcal{S}^d$. As the normalized closed forms for $\vec{u}$ and $\vec{u}'$ are normalized poly-exponential expressions, we have

$$(\vec{q}_{norm}, \vec{w}(\vec{q}_{norm})) = (\vec{q}'_{norm})[\vec{x}'/\vec{w}(\vec{x})].$$

Since $(\vec{q}'_{norm})_{x_i}$, the closed form for x_i w.r.t. $\vec{u}'$, only contains linear coefficients, $\left(\vec{q}'_{norm}\right)_{x_i}[\vec{x}'/\vec{w}(\vec{x})])$ contains coefficients of at most degree

$$\max\left(\left\{\operatorname{depdeg}_{\vec{u}'}(x_s) \mid x_s \in \mathcal{V}\left((\vec{q}'_{norm})_{x_i}\right)\right\} \uplus \left\{\deg(w_j) \mid x'_j \in \mathcal{V}\left((\vec{q}'_{norm})_{x_i}\right)\right\}\right).$$

As $\vec{u}'$ is linear and triangular and thus, $\operatorname{depdeg}_{\vec{u}'}$ is either 1 or 0 (dependency degree 0 means that the closed form for such a variable is a constant), we have

$$\mathcal{V}\left((\vec{q}'_{norm})_{x_i}\right) \subseteq \left\{x_s \,\middle|\, x_i \succeq_{\vec{u}'} x_s,\ \operatorname{depdeg}_{\vec{u}'}(x_s) = 1\right\} \uplus \left\{x'_j \mid x_i \succ_{\vec{u}'} x'_j\right\} \uplus \{n\},$$

where n is the distinguished variable representing the number of iterations. Note that here we have used that for every variable x_i which occurs in its update $\vec{u}'_{x_i}$ and thus also occurs in its closed form $(\vec{q}'_{norm})_{x_i}$ its dependency degree is 1. Thus, the claim follows. □

Example 8.3.9. Reconsider the setting from Ex. 8.3.5. Here, $(\varphi, \vec{u})$ is *tnn* and $(\varphi', \vec{u}\,')$ is a linear-update *tnn*-loop and $\vec{w} = (x_2^2, x_3^4, x_3^2)$. The normalized closed form for $\vec{u}$ is given by

$$\vec{q}_{norm} = (x_3^4, x_3^2, x_3).$$

For example, the normalized closed form for x_1 contains coefficients of degree 4, and we also have

$$\begin{aligned} & \max\left(\left\{\operatorname{depdeg}_{\vec{u}\,'}(x_s) \mid x_1 \succeq_{\vec{u}\,'} x_s\right\} \uplus \left\{\deg(w_j) \mid x_1 \succ_{\vec{u}\,'} x_j'\right\}\right) \\ = \ & \max(\deg(w_1), \deg(w_2), \deg(w_3)) \\ = \ & 4. \end{aligned}$$

So far, we do not have a bound on the value $\max\{\deg(w_j \mid x_i \succ_{\vec{u}\,'} x_j'\}$ in Lemma 8.3.8. The procedure from [OBP16a] enables us to bound this value by $\operatorname{depdeg}_{\vec{u}}(x_i)$ (see Cor. 8.3.15). We now present this technique. To this end, let $(\varphi, \vec{u})$ be a *tnn*-loop over $\mathcal{S}$ and let $mdepdeg = \max_{1 \le i \le d} \operatorname{depdeg}_{\vec{u}}(x_i)$. W.l.o.g. we may assume that $x_i \succ_{\vec{u}} x_j$ implies $i > j$ for all $1 \le i, j \le d$.

The basic idea of the algorithm in [OBP16a] is to add variables which represent monomials of the form $\prod_{i=1}^{d} x_i^{z_i}$ in the variables $x_1, \dots, x_d$, where $z_1, \dots, z_d \in \mathbb{N}$. Here, we only consider monomials with positive degree, i.e, we leave out the constant monomial $m = 1$. To ease notation, we abbreviate $\prod_{i=1}^{d} x_i^{z_i}$ to $\vec{x}^{\vec{z}}$ where $\vec{z} = (z_1, \dots, z_d) \in \mathbb{N}^d$. We denote the variable representing such a monomial m in the linearized system by y_m. So in particular, y_{x_i} represents x_i in the linearized update. In the sense of Def. 8.3.1, the variables $\vec{x}\,'$ are then the variables y_m for a monomial m with $\deg(m) > 1$.

Moreover, if $m = \vec{x}^{\vec{z}}$ and $m' = \vec{x}^{\vec{z}\,'}$ are two monomials, we define the lexicographic ordering $\succ$ on monomials, where $m \succ m'$ holds if $\vec{z} \succ_{lex} \vec{z}\,'$. As usual, we write $m \succeq m'$ to denote $m \succ m'$ or $m = m'$.

The procedure from [OBP16a] now computes a *linear-update tnn*-loop $(\varphi', \vec{u}\,')$ over the variables y_m which is a linearization of $(\varphi, \vec{u})$ via some $\vec{w}$, where $\deg(m) \le mdepdeg$ and $y_{x_i} = x_i$. Here,

$$\varphi' = \varphi[x_i/y_{x_i}] \wedge \bigwedge_{\substack{m = \vec{x}^{\vec{z}}, \\ \deg(m) \le mdepdeg}} \left(y_m - \prod_{i=1}^{d} y_{x_i}^{z_i} = 0\right),$$

according to Def. 8.3.4.

For each monomial m with $\deg(m) \le mdepdeg$, the entry u'_{y_m} of $\vec{u}\,'$ corresponding to the update of y_m will only contain the variables $y_{m'}$ with $m \succeq m'$. The computation

of the entries in $\vec{u}'$ works according to the lexicographic ordering $\succ$, i.e., when computing u'_{y_m}, we can assume that we have already computed $u'_{m'}$ for all m' with $m \succ m'$. More precisely, when computing the update $u'_{m_1 \cdot m_2}$ for a monomial $m_1 \cdot m_2$, we can assume that we already know the updates u'_{m_1} and u'_{m_2} for the monomials m_1 and m_2. Let $u'_{m_i} = \sum_{m_i \geq m'_i} r_{m'_i} \cdot y_{m'_i} + C_i$ for $i \in \{1, 2\}$ and $r_{m'_i}, C_i \in \mathcal{S}$. Then we have

$$\begin{aligned} u'_{m_1} \cdot u'_{m_2} &= \left(\sum_{m_1 \geq m'_1} r_{m'_1} \cdot y_{m'_1} + C_1 \right) \cdot \left(\sum_{m_2 \geq m'_2} r_{m'_2} \cdot y_{m'_2} + C_2 \right) \\ &= \sum_{\substack{m_1 \geq m'_1, \\ m_2 \geq m'_2}} \left(r_{m'_1} \cdot r_{m'_2} \right) \cdot y_{m'_1} \cdot y_{m'_2} \\ &\quad + \sum_{m_2 \geq m'_2} (C_1 \cdot r_{m'_2}) \cdot y_{m'_2} + \sum_{m_1 \geq m'_1} (C_2 \cdot r_{m'_1}) \cdot y_{m'_1} + C_1 \cdot C_2. \end{aligned}$$

Since the update $u'_{m_1 \cdot m_2}$ for the monomial $m_1 \cdot m_2$ should have the same value as $u'_{m_1} \cdot u'_{m_2}$, the technique from [OBP16a] defines the following:

$$\begin{aligned} u'_{m_1 \cdot m_2} &= \sum_{\substack{m_1 \geq m'_1, \\ m_2 \geq m'_2}} \left(r_{m'_1} \cdot r_{m'_2} \right) \cdot y_{m'_1 \cdot m'_2} \\ &\quad + \sum_{m_2 \geq m'_2} (C_1 \cdot r_{m'_2}) \cdot y_{m'_2} + \sum_{m_1 \geq m'_1} (C_2 \cdot r_{m'_1}) \cdot y_{m'_1} + C_1 \cdot C_2. \end{aligned}$$

Here, instead of the non-linear monomial $y_{m'_1} \cdot y_{m'_2}$, one uses the variable $y_{m'_1 \cdot m'_2}$. Hence, the resulting linear update $u'_{m_1 \cdot m_2}$ indeed only contains variables $y_{m'}$ where $m_1 \cdot m_2 \succeq m'$ since $m_1 \succeq m'_1$ and $m_2 \succeq m'_2$ implies $m_1 \cdot m_2 \succeq m'_1 \cdot m'_2$. So for these variables $y_{m'}$, the linear update $u'_{m'}$ has already been computed.

The linearization procedure from [OBP16a] is summarized in Algorithm 8.3.1. If the input is a linear-update loop, it is immediately returned (Line 1). Then, Algorithm 8.3.1 proceeds in a "bottom-up" fashion: We start with $V = \{x_i \mid x_i \text{ is minimal w.r.t. } \succ_{\vec{u}}\}$ in Line 2. Then we compute the linearized updates for the variables y_m, where m is some monomial in the variables in V of degree at most $mdepdeg$. Here, we iterate over the monomials in *lexicographic order* until we can linearize the update of a variable $x_j \notin V$ in Lines 5 to 8. Then, we add x_j to V in Line 10. We repeat this procedure until we have linearized the updates of all variables $x_1, \ldots, x_d$.

Remark 8.3.10. We stated a version of Algorithm 8.3.1 which suffices for our needs. For example, we have not explicitly stated that each monomial should be processed only once, i.e., one should keep track of the already handled monomials. Nevertheless, when analyzing the complexity of the algorithm in Sect. 10.2 we will only take the computation of the new updates into account and implicitly assume that each monomial is processed only once. The computation of the new updates is the bottleneck of the complexity of Algorithm 8.3.1 since the other two "repeat ... until"-loops are just used for bookkeeping.

Algorithm 8.3.1: Linearizing *tnn*-Loops

Input: *tnn*-loop $(\varphi, \vec{u})$ over $\vec{x}$
Output: linear-update *tnn*-loop $(\varphi', \vec{u}')$

if $\vec{u}$ *is linear* **then return** $(\varphi, \vec{u})$
$V \longleftarrow \{x_i \mid x_i \text{ is minimal w.r.t. } \succ_{\vec{u}}\}$
$mdepdeg \longleftarrow \max_{1 \le i \le d} \operatorname{depdeg}_{\vec{u}}(x_i)$
repeat
  repeat
    foreach *monomial* m *with* $\mathcal{V}(m) \subseteq V$ *and* $\deg(m) \le mdepdeg$ **do**
      compute u'_{y_m}
  until *there is* $x_j \in \{x_1, \dots, x_d\} \setminus V$ *whose update can be linearized*
  compute $u'_{y_{x_j}}$
  $V \longleftarrow V \uplus \{x_j\}$
until $V = \{x_1, \dots, x_d\}$
return $\left(\varphi[x_i/y_{x_i}] \wedge \bigwedge_{m=\vec{x}^{\vec{z}},\, \deg(m) \le mdepdeg} \left(y_m - \prod_{i=1}^{d} y_{x_i}^{z_i} = 0\right), \vec{u}'\right)$

Example 8.3.11. Again, consider the setting from Ex. 8.3.5. We linearize $\vec{u}$ by using Algorithm 8.3.1.

At the beginning of the algorithm, $V = \{x_3\}$ since x_3 is the unique minimal element w.r.t. $\succ_{\vec{u}}$. Then, Algorithm 8.3.1 processes the monomial x_3^2 and computes the update $u'_{y_{x_3^2}} = y_{x_3^2}$. Afterwards, Algorithm 8.3.1 linearizes the update of x_2, i.e., $u'_{y_{x_2}} = y_{x_3^2}$ and x_2 can be added to V. Thereafter, Algorithm 8.3.1 processes the monomial x_3^4 which yields $u'_{x_3^4} = y_{x_3^4}$. Then, it continues with the

monomial x_2^2 and computes $u'_{y_{x_2^2}} = y_{x_3^4}$. Finally, Algorithm 8.3.1 linearizes the update of x_1, i.e., $u'_{y_{x_1}} = y_{x_2^2}$ and x_1 can be added to V and thus $V = \{x_1, x_2, x_3\}$. Here, we skipped monomials which are not needed for linearizing the loop (e.g., x_3^3) to ease readability.

To sum up, Algorithm 8.3.1 computes the linearized update

$$\left(y_{x_2^2},\ y_{x_3^4},\ y_{x_3^2},\ y_{x_3^4},\ y_{x_3^2},\ y_{x_3}\right)$$

over the variables $(y_{x_1}, y_{x_2^2}, y_{x_2}, y_{x_3^4}, y_{x_3^2}, y_{x_3})$.

Here, in the terminology of Ex. 8.3.2, we have $x'_1 = y_{x_2^2}$, $x'_2 = y_{x_3^4}$, and $x'_3 = y_{x_3^2}$.

We now prove that it is indeed enough to consider only monomials up to degree *mdepdeg* to linearize a *tnn*-loop.

Theorem 8.3.12 (Dependency Degree Suffices for Linearization)**.** *The linearization $u'_{y_{x_i}}$ of u_i only contains variables y_m where m is a monomial of at most degree* $\mathrm{depdeg}_{\vec{u}}(x_i)$.

Proof. We prove the claim by Noetherian induction on the order $\succ_{\vec{u}}$. If x_i is a minimal element w.r.t. $\succ_{\vec{u}}$, u_i is a linear polynomial, i.e., it is not affected by the linearization procedure. So, $u_i = u'_{y_{x_i}}$. This proves the base case.

Now let x_i be some variable such that x_i is not a minimal element w.r.t. $\succ_{\vec{u}}$. Then, let $m \neq x_i$ be some monomial occurring in u_i. Thus, we need the variable y_m in the linearized system and compute its linear update. By the induction hypothesis, for all variables x_j occurring in m, we know that $u'_{y_{x_j}}$ only contains variables y_{m_j} where m_j has at most degree $\mathrm{depdeg}_{\vec{u}}(x_j)$. Thus, we need the updates of all the variables $y_{m[x_j/m_j]}$, where m_j is some monomial such that y_{m_j} occurs in $u'_{y_{x_j}}$. Now, we have

$$\begin{aligned}
&\deg(m[x_j/m_j \mid x_j \in \mathcal{V}(m)]) \\
=\ &\sum_{x_j \in \mathcal{V}(m)} \deg_{x_j}(m) \cdot \deg(m_j) \\
\leq\ &\sum_{x_j \in \mathcal{V}(m)} \deg_{x_j}(m) \cdot \mathrm{depdeg}_{\vec{u}}(x_j) && \text{(by induction hypothesis)} \\
\leq\ &\mathrm{depdeg}_{\vec{u}}(x_i). && \text{(by Def. 8.2.1)}
\end{aligned}$$

This proves the claim. □

The following theorem captures the properties of Algorithm 8.3.1. Note, that this theorem is stated in the technical report [OBP16b] of [OBP16a] in a slightly different manner. Here, instead of *tnn*-loops, solvable loops are considered (Def. 4.1.6). Moreover, [OBP16a; OBP16b] do not consider the concept of non-negativity.

Theorem 8.3.13 (Soundness of Algorithm 8.3.1, [OBP16a])**.** *Given a tnn-loop* $(\varphi, \vec{u})$*, Algorithm 8.3.1 computes a linearization* $(\varphi', \vec{u}')$ *of* $(\varphi, \vec{u})$ *via* $\vec{w} = (m)_{y_m \in \vec{y},\, \deg(m)>1}$ *such that* $(\varphi', \vec{u}')$ *is a linear-update tnn-loop. If* $(\varphi, \vec{u})$ *is a loop over* $\mathcal{S}$*, so is* $(\varphi', \vec{u}')$*.*

Proof. The only statement which is not immediately clear from Algorithm 8.3.1 is the statement regarding non-negativity.

Again, we prove this claim by Noetherian induction on the order $>_{\vec{u}'}$. If y_m is a minimal element w.r.t. $>_{\vec{u}'}$, $m = x_i$ for some minimal variable x_i w.r.t. $>_{\vec{u}}$. Here, u_i is a linear polynomial, i.e., it is not affected by the linearization procedure. So, $u_i = u'_{y_{x_i}}$. This proves the base case since $\vec{u}$ is non-negative.

Now let y_m be some variable that is not a minimal element w.r.t. $>_{\vec{u}'}$. Then, $m = m_1 \cdot m_2$ for some non-constant monomials m_1, m_2 such that u'_{y_m} was obtained by multiplying the updates $u'_{y_{m_1}}$ and $u'_{y_{m_2}}$. But by induction hypothesis, the coefficient of y_{m_i} in $u'_{y_{m_i}}$ is non-negative for $i \in \{1, 2\}$. Hence, their product is non-negative. This product is the coefficient of y_m in u'_{y_m} which proves the claim. □

Example 8.3.14. Reconsider Ex. 8.3.11. The vector $\vec{w}$ as stated in Thm. 8.3.13 is exactly the one from Ex. 8.3.2, i.e.,

$$\vec{w} = (x_2^2, x_3^4, x_3^2,).$$

Now we combine Thm. 8.3.12 and 8.3.13 and obtain the following corollary.

Corollary 8.3.15. *Let* $(\varphi, \vec{u})$ *be a tnn-loop. Then* $(\varphi, \vec{u})$ *can be linearized to a linear-update tnn-loop* $(\varphi', \vec{u}')$ *via a vector* $\vec{w}$ *such that for all* $1 \leq i \leq d$

$$\max_{x_i >_{\vec{u}} x'_j} \deg(w_j) \leq \operatorname{depdeg}_{\vec{u}}(x_i).$$

So, the degree of $\vec{w}$ is at most $\max_{1 \leq i \leq d} \operatorname{depdeg}_{\vec{u}}(x_i)$. The following example shows that we indeed need to consider the dependency degree.

Example 8.3.16. We really need to consider the maximal dependency degree. Reconsider the setting from Ex. 8.3.5, i.e., $\vec{u} = (x_2^2, x_3^2, x_3)$. If it were possible to linearize a loop with this update to a linear-update *tnn*-loop via a vector $\vec{w}$ of degree < 4, Lemma 8.3.8 would imply that the closed form for $\vec{u}$ contains only coefficients of degree < 4. However, we have seen in Ex. 8.3.9 that it contains the coefficient x_3^4 of degree 4. Thus, this loop cannot be linearized via a vector $\vec{w}$ with $\deg(\vec{w}) < 4$. However,

$$\begin{aligned}
&\operatorname{depdeg}_{\vec{u}}(x_1)\\
&= \max\left\{\deg_{x_1}(u_1),\ 2 \cdot \operatorname{depdeg}_{\vec{u}}(x_2)\right\}\\
&= 2 \cdot \operatorname{depdeg}_{\vec{u}}(x_2) = 2 \cdot \max\left\{\deg_{x_2}(u_2),\ 2 \cdot \operatorname{depdeg}_{\vec{u}}(x_3)\right\}\\
&= 2 \cdot 2 \cdot 1\\
&= 4,
\end{aligned}$$

and we have seen in Ex. 8.3.11 and 8.3.14 that Algorithm 8.3.1 linearizes such a loop via a vector $\vec{w}$ of degree 4.

Finally, we can prove Lemma 8.2.3.

Proof of Lemma 8.2.3. Let $(\varphi, \vec{u})$ be a *tnn*-loop over $\mathbb{Z}$. By Cor. 8.3.15, it can be linearized via a vector $\vec{w}$ to a linear *tnn*-loop $(\varphi', \vec{u}\,')$ over $\mathbb{Z}$ such that

$$\max_{x_i >_{\vec{u}'} x_j'} \deg(w_j) \leq \operatorname{depdeg}_{\vec{u}}(x_i).$$

Moreover, we have

$$\max\left\{\operatorname{depdeg}_{\vec{u}'}(x_s) \;\middle|\; x_i \geq_{\vec{u}'} x_s\right\} \leq \operatorname{depdeg}_{\vec{u}}(x_i)$$

since $x_i \geq_{\vec{u}'} x_s$ implies $x_i \geq_{\vec{u}} x_s$ and $\operatorname{depdeg}_{\vec{u}'}(x_s) \leq \operatorname{depdeg}_{\vec{u}}(x_s)$.

Now, consider an atom $r \triangleright 0$ in φ. By Lemma 8.3.8, the coefficients in $r(\vec{q}_{norm})$ have degree at most

$$\max\left\{\sum_{i=1}^{d} \operatorname{depdeg}_{\vec{u}}(x_i) \cdot \deg_{x_i}(m) \;\middle|\; m \text{ monomial in } r\right\}.$$

Since the maximal degree occurring in a coefficient of $\varphi(\vec{q}_{norm})$ is bounded by the maximal degree occurring in a coefficient in an atom $r \triangleright 0$ in φ, this proves the claim. □

Note that we have used Algorithm 8.3.1 for proving Lemma 8.2.3. However, the asymptotic bound can be computed by elementary arithmetic from the *original* loop, i.e., we only use the linearization for the proof.

This section indicates why linearization of a *twn*-loop is costly in terms of complexity (see the discussion of the loop in Fig. 6.1 before Cor. 6.1.11): we have to handle monomials up to degree $mdepdeg$. We will discuss this in more detail in Ch. 10.

Chapter 9

Transformation to Triangular Weakly Non-Linear Form

In this chapter, we show how to handle loops that are not yet *twn*. To this end, we introduce a transformation of loops via *polynomial automorphisms* in Sect. 9.1 and show that our transformation preserves (non-)termination (Thm. 9.1.10). In Sect. 9.2, we use results from algebraic geometry to show that the question whether a loop can be transformed into *twn*-form is reducible to validity of $\mathrm{Th}_\exists(\mathbb{R}_\mathrm{A})$-formulas (Thm. 9.2.3). Moreover, we show that it is decidable whether a *linear* automorphism can transform a loop into a special case of the *twn*-form (Thm. 9.2.6). Finally, we show how to generalize our results from Ch. 6 to 8 to certain non-*twn* loops in Sect. 9.3.

9.1 Transforming Loops

Clearly, the *polynomials* $x_1, \ldots, x_d$ are *generators* of the $\mathcal{S}$-algebra $\mathcal{S}[\vec{x}]$, i.e., every polynomial from $\mathcal{S}[\vec{x}]$ can be obtained from $x_1, \ldots, x_d$ and the operations of the algebra (i.e., addition and multiplication). So far, we have implicitly chosen a special "representation" of the loop based on the generators $x_1, \ldots, x_d$.

We now change this representation, i.e., we use a different set of d polynomials which are also generators of $\mathcal{S}[\vec{x}]$. Then the loop has to be modified accordingly in order to adapt it to this new representation. This modification does not affect the loop's termination behavior, but it may transform a non-*twn*-loop into *twn*-form. The desired change of representation is described by $\mathcal{S}$*-automorphisms* of $\mathcal{S}[\vec{x}]$.

Definition 9.1.1 (Ring Endomorphisms)**.** A mapping $\eta : \mathcal{S}[\vec{x}] \longrightarrow \mathcal{S}[\vec{x}]$ is an *$\mathcal{S}$-endomorphism* of $\mathcal{S}[\vec{x}]$ if it is $\mathcal{S}$-linear and multiplicative, i.e., for all $c, c' \in \mathcal{S}$ and all $p, p' \in \mathcal{S}[\vec{x}]$

- $\eta(c \cdot p + c' \cdot p') = c \cdot \eta(p) + c' \cdot \eta(p')$,
- $\eta(1) = 1$, and
- $\eta(p \cdot p') = \eta(p) \cdot \eta(p')$.

The set of all $\mathcal{S}$-endomorphisms of $\mathcal{S}[\vec{x}]$ is denoted by $\mathrm{End}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$. The set $\mathrm{Aut}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$ of *$\mathcal{S}$-automorphisms of $\mathcal{S}[\vec{x}]$* consists of those $\eta \in \mathrm{End}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$ which are *invertible*, i.e., there exists an $\eta^{-1} \in \mathrm{End}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$ such that $\eta \circ \eta^{-1} = \eta^{-1} \circ \eta = id_{\mathcal{S}[\vec{x}]}$, where $id_{\mathcal{S}[\vec{x}]}$ is the identity function on $\mathcal{S}[\vec{x}]$. $\mathrm{Aut}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$ is a group under function composition $\circ$ with neutral element $id_{\mathcal{S}[\vec{x}]}$.

Example 9.1.2. Let $\eta \in \mathrm{End}_{\mathcal{S}}\left(\mathcal{S}[x_1, x_2]\right)$ with $\eta(x_1) = x_2$, $\eta(x_2) = x_1 - x_2^2$. Then $\eta \in \mathrm{Aut}_{\mathcal{S}}\left(\mathcal{S}[x_1, x_2]\right)$, where $\eta^{-1}(x_1) = x_1^2 + x_2$ and $\eta^{-1}(x_2) = x_1$.

As $\mathcal{S}[\vec{x}]$ is free on the generators $\vec{x}$, an endomorphism $\eta \in \mathrm{End}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$ is uniquely determined by the images of the variables, i.e., by $\eta(x_1), \ldots, \eta(x_d)$. Hence, we have a one-to-one correspondence between elements of $\left(\mathcal{S}[\vec{x}]\right)^d$ and $\mathrm{End}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$. In particular, every tuple $\vec{u} = (u_1, \ldots, u_d) \in \left(\mathcal{S}[\vec{x}]\right)^d$ corresponds to the unique endomorphism $\tilde{u} \in \mathrm{End}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$ with $\tilde{u}(x_i) = u_i$ for all $1 \leq i \leq d$. So for any $p \in \mathcal{S}[\vec{x}]$ we have $\tilde{u}(p) = p(\vec{u})$. Thus, the update of a loop induces an endomorphism which operates on polynomials.

Example 9.1.3. Consider the loop $\mathbf{L}_{aut}$ in Fig. 9.1, i.e., $\varphi = (x_2^3 + x_1 - x_2^2 > 0)$ and $\vec{u} = (u_1, u_2)$. Then $\vec{u}$ induces the endomorphism $\tilde{u}$ with $\tilde{u}(x_1) = u_1$ and $\tilde{u}(x_2) = u_2$. So we have $\tilde{u}(2 \cdot x_1 + x_2^3) = (2 \cdot x_1 + x_2^3)(\vec{u}) = 2 \cdot u_1 + u_2^3$.

For tuples of numbers (e.g., $\vec{e} = (5, 2)$), the endomorphism $\tilde{e}$ is $\tilde{e}(x_1) = 5$ and $\tilde{e}(x_2) = 2$. Thus, we have $\tilde{e}(x_2^3 + x_1 - x_2^2) = (x_2^3 + x_1 - x_2^2)(5, 2) = 2^3 + 5 - 2^2 = 9$.

We extend the application of endomorphisms $\eta : \mathcal{S}[\vec{x}] \longrightarrow \mathcal{S}[\vec{x}]$ to vectors of polynomials $\vec{u} = (u_1, \ldots, u_d)$ by defining $\eta(\vec{u}) = (\eta(u_1), \ldots, \eta(u_d))$ and to formulas $\varphi \in \mathrm{Th}_{\exists}(\mathcal{S}, \mathbb{R}_{\mathrm{A}})$ by defining $\eta(\varphi) = \varphi(\eta(\vec{x}))$, i.e., $\eta(\varphi)$ results from φ by applying η to

$$\textbf{while } \left(x_2^3 + x_1 - x_2^2 > 0 \right) \textbf{ do } \{ (x_1, x_2) \leftarrow (u_1, u_2) \}$$

$$\begin{aligned} \text{with} \quad u_1 &= ((-x_2^2 + x_1)^2 + x_2)^2 - 2 \cdot x_2^2 + 2 \cdot x_1 \\ u_2 &= (-x_2^2 + x_1)^2 + x_2 \end{aligned}$$

Figure 9.1: Loop $\mathbf{L}_{aut}$

all polynomials that occur in φ. This allows us to transform $(\varphi, \vec{u})$ into a new loop $Tr_\eta(\varphi, \vec{u})$ using any automorphism $\eta \in \mathrm{Aut}_S\left(S[\vec{x}]\right)$.

Definition 9.1.4 (Tr)**.** Let $\eta \in \mathrm{Aut}_S\left(S[\vec{x}]\right)$. We define $Tr_\eta(\varphi, \vec{u}) = (\varphi', \vec{u}\,')$ where

$$\varphi' = \eta^{-1}(\varphi) \qquad \text{and} \qquad \vec{u}\,' = (\eta^{-1} \circ \tilde{u} \circ \eta)(\vec{x}).$$

In other words, we have $\vec{u}\,' = (\eta(\vec{x}))\,(\vec{u})\,(\eta^{-1}(\vec{x}))$ since $(\eta^{-1} \circ \tilde{u} \circ \eta)(\vec{x}) = \eta^{-1}(\eta(\vec{x})[\vec{x}/\vec{u}]) = \eta(\vec{x})[\vec{x}/\vec{u}][\vec{x}/\eta^{-1}(\vec{x})] = (\eta(\vec{x}))(\vec{u})(\eta^{-1}(\vec{x}))$.

Example 9.1.5. We transform the loop $\mathbf{L}_{aut}$ in Fig. 9.1 with the automorphism η from Ex. 9.1.2. We obtain $Tr_\eta(\varphi, \vec{u}) = (\varphi', \vec{u}\,')$ where

$$\begin{aligned} \varphi' &= \eta^{-1}(\varphi) = ((\eta^{-1}(x_2))^3 + \eta^{-1}(x_1) - (\eta^{-1}(x_2))^2 > 0) \\ &= (x_1^3 + x_1^2 + x_2 - x_1^2 > 0) = (x_1^3 + x_2 > 0) \qquad \text{and} \\ \vec{u}\,' &= ((\eta^{-1} \circ \tilde{u} \circ \eta)(x_1), (\eta^{-1} \circ \tilde{u} \circ \eta)(x_2)) = (\eta^{-1}(\tilde{u}(x_2)), \eta^{-1}(\tilde{u}(x_1 - x_2^2))) \\ &= (\eta^{-1}(u_2), \eta^{-1}(u_1 - u_2^2)) = (x_1 + x_2^2, 2 \cdot x_2). \end{aligned}$$

So the resulting transformed loop is $(x_1^3 + x_2 > 0,\ (x_1 + x_2^2, 2 \cdot x_2))$. Note that while the original loop $(\varphi, \vec{u})$ is neither triangular nor weakly non-linear, the resulting transformed loop is *twn*. Also note that we used a *non-linear* automorphism with $\eta(x_2) = x_1 - x_2^2$ for the transformation.

While the above example shows that our transformation can indeed transform non-*twn*-loops into *twn*-loops, it remains to prove that this transformation preserves

(non-)termination. Then we can use our techniques for termination analysis of *twn*-loops for *twn-transformable*-loops as well, i.e., for all loops $(\varphi, \vec{u})$ where $Tr_\eta(\varphi, \vec{u})$ is *twn* for some automorphism η. (The question how to find such automorphisms will be addressed in Sect. 9.2.)

As a first step, by Lemma 9.1.6, our transformation is "compatible" with the operation $\circ$ of the group $\mathrm{Aut}_S\left(S[\vec{x}]\right)$, i.e., it is an *action*.

Lemma 9.1.6. *Tr is an* action *of* $\mathrm{Aut}_S\left(S[\vec{x}]\right)$ *on loops, i.e., for* $\eta_1, \eta_2 \in \mathrm{Aut}_S\left(S[\vec{x}]\right)$

$$Tr_{id_{S[\vec{x}]}}(\varphi, \vec{u}) = (\varphi, \vec{u}) \quad \text{and} \quad Tr_{\eta_1 \circ \eta_2}(\varphi, \vec{u}) = Tr_{\eta_2}(Tr_{\eta_1}(\varphi, \vec{u})).$$

With this information at hand, we can prove a crucial result which will enable us to generalize our results on witnesses for (eventual) non-termination, the halting problem, and the runtime to loops which can be transformed into *twn*-form.

Lemma 9.1.7. *Let* $(\varphi, \vec{u})$ *be a loop and let* $Tr_\eta(\varphi, \vec{u}) = (\varphi', \vec{u}')$. *For* $\vec{e} \in S^d$ *and* $n \in \mathbb{N}$ *we have*

$$\varphi(\vec{u}^n(\vec{e})) \iff \varphi'((\vec{u}')^n(\hat{\eta}(\vec{e}))).$$

Here, $\hat{\eta} : S^d \to S^d$ *maps* $\vec{e}$ *to* $\hat{\eta}(\vec{e}) = \tilde{e}(\eta(\vec{x})) = (\eta(\vec{x}))(\vec{e})$.

From Lemma 9.1.7, we obtain the following corollary which shows that a witness for (eventual) non-termination of $(\varphi, \vec{u})$ is transformed by $\eta(\vec{x})$ into a witness for (eventual) non-termination of $Tr_\eta(\varphi, \vec{u})$.

Corollary 9.1.8. *If* $\vec{e}$ *witnesses (eventual) non-termination of* $(\varphi, \vec{u})$, $\hat{\eta}(\vec{e})$ *witnesses (eventual) non-termination of* $Tr_\eta(\varphi, \vec{u})$.

Example 9.1.9. For the tuple $\vec{e} = (5, 2)$ from Ex. 9.1.3 and the automorphism η from Ex. 9.1.2 with $\eta(x_1) = x_2$ and $\eta(x_2) = x_1 - x_2^2$, we obtain

$$\hat{\eta}(\vec{e}) = (\eta(x_1), \eta(x_2))\,(\vec{e}) = (2, 5 - 2^2) = (2, 1).$$

As $\vec{e} = (5, 2)$ witnesses non-termination of the loop $\mathbf{L}_{aut} = (\varphi, \vec{u})$ in Fig. 9.1, $\hat{\eta}(\vec{e}) = (2, 1)$ witnesses non-termination of $Tr_\eta(\varphi, \vec{u})$ due to Cor. 9.1.8.

Finally, Thm. 9.1.10 states that transforming a loop preserves (non-)termination.

Theorem 9.1.10 (*Tr* Preserves Termination). *If $\eta \in \mathrm{Aut}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$, $\hat{\eta}$ is a bijection between the respective sets of witnesses for (eventual) non-termination, i.e.,*

$$(E)N\,T_{(\varphi,\vec{u})} \overset{\hat{\eta}}{\longleftrightarrow} (E)N\,T_{(\varphi',\vec{u}\,')} \quad \textit{and thus also} \quad T_{(\varphi,\vec{u})} \overset{\hat{\eta}}{\longleftrightarrow} T_{(\varphi',\vec{u}\,')}.$$

Therefore, $(\varphi, \vec{u})$ terminates iff $Tr_\eta(\varphi, \vec{u}) = (\varphi', \vec{u}\,')$ terminates.

Up to now, we only transformed a loop $(\varphi, \vec{u})$ on $\mathcal{S}^d$ using elements of $\mathrm{Aut}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$. However, we can also transform it into the loop $Tr_\eta(\varphi, \vec{u})$ on $\mathbb{R}_{\mathrm{A}}^d$ using an automorphism $\eta \in \mathrm{Aut}_{\mathbb{R}_{\mathrm{A}}}\left(\mathbb{R}_{\mathrm{A}}[\vec{x}]\right)$. Nevertheless, our goal remains to prove termination on $\mathcal{S}^d$ instead of $\mathbb{R}_{\mathrm{A}}^d$, which is not equivalent in general. Thus, in Sect. 9.3 we will show how to analyze termination of loops on certain subsets F of $\mathbb{R}_{\mathrm{A}}^d$. This allows us to analyze termination of $(\varphi, \vec{u})$ on $\mathcal{S}^d$ by checking termination of $Tr_\eta(\varphi, \vec{u})$ on the subset $\hat{\eta}(\mathcal{S}^d) \subseteq \mathbb{R}_{\mathrm{A}}^d$ instead.

By our definition of loops over a ring $\mathcal{S}$, we have $\vec{u}(\vec{e}) \in \mathcal{S}^d$ for all $\vec{e} \in \mathcal{S}^d$, i.e., $\mathcal{S}^d$ is *$\vec{u}$-invariant*. This property is preserved by our transformation.

Definition 9.1.11 ($\vec{u}$-Invariance). Let $(\varphi, \vec{u})$ be a loop on $\mathcal{S}^d$ and let $F \subseteq \mathcal{S}^d$. We call F *$\vec{u}$-invariant* or *update-invariant* if for all $\vec{e} \in F$ we have $\vec{u}(\vec{e}) \in F$.

Lemma 9.1.12. *Let $(\varphi, \vec{u})$ be a loop on $\mathcal{S}^d$, let $F \subseteq \mathcal{S}^d$ be $\vec{u}$-invariant, and let $\eta \in \mathrm{Aut}_{\mathbb{R}_{\mathrm{A}}}\left(\mathbb{R}_{\mathrm{A}}[\vec{x}]\right)$. Furthermore, let $Tr_\eta(\varphi, \vec{u}) = (\varphi', \vec{u}\,')$. Then $\hat{\eta}(F)$ is $\vec{u}\,'$-invariant.*

Recall that our goal is to reduce termination to a $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_{\mathrm{A}})$-formula. Clearly, *termination on F* cannot be encoded with such a formula if F cannot be defined via $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_{\mathrm{A}})$. Thus, we require that F is $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_{\mathrm{A}})$-*definable*.

Definition 9.1.13 ($\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_{\mathrm{A}})$-Definability). A set $F \subseteq \mathbb{R}_{\mathrm{A}}^d$ is *$\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_{\mathrm{A}})$-definable* if there is a $\psi \in \mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_{\mathrm{A}})$ with free variables $\vec{x}$ such that for all $\vec{e} \in \mathbb{R}_{\mathrm{A}}^d$

$$\vec{e} \in F \quad \text{iff} \quad \psi(\vec{e}) \text{ is valid.}$$

We say that ψ *defines* F.

An example for a $\mathrm{Th}_\exists(\mathbb{Z}, \mathbb{R}_{\mathrm{A}})$-definable set is $\{(a, 0, a) \mid a \in \mathbb{Z}\}$, which is characterized by the formula $\exists a \in \mathbb{Z}.\ x_1 = a \wedge x_2 = 0 \wedge x_3 = a$.

To analyze termination of $(\varphi, \vec{u})$ on $\mathcal{S}^d$, we can analyze termination of $Tr_\eta(\varphi, \vec{u})$ on $\hat{\eta}(\mathcal{S}^d) \subseteq \mathbb{R}_A^d$ instead. The reason is that $\vec{e} \in \mathcal{S}^d$ witnesses (eventual) non-termination of $(\varphi, \vec{u})$ iff $\hat{\eta}(\vec{e})$ witnesses (eventual) non-termination of $Tr_\eta(\varphi, \vec{u})$ due to Cor. 9.1.8, i.e., $\mathcal{S}^d$ contains a witness for (eventual) non-termination of $(\varphi, \vec{u})$ iff $\hat{\eta}(\mathcal{S}^d)$ contains a witness for (eventual) non-termination of $Tr_\eta(\varphi, \vec{u})$. While $\mathcal{S}^d$ is clearly $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_A)$-definable, the following lemma shows that $\hat{\eta}(\mathcal{S}^d)$ is $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_A)$-definable, too. More precisely, $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_A)$-definability is preserved by polynomial endomorphisms.

Lemma 9.1.14. *Let $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_A$ and let $\eta \in \mathrm{End}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$. If $F \subseteq \mathbb{R}_A^d$ is $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_A)$-definable then so is $\hat{\eta}(F)$.*

Example 9.1.15. The set $\mathbb{Z}^2$ is $\mathrm{Th}_\exists(\mathbb{Z}, \mathbb{R}_A)$-definable as we have $(x_1, x_2) \in \mathbb{Z}^2$ iff

$$\exists a, b \in \mathbb{Z}.\ x_1 = a \wedge x_2 = b.$$

Let $\eta \in \mathrm{End}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$ with $\eta(x_1) = \frac{1}{2} \cdot x_1^2 + x_2^2$ and $\eta(x_2) = x_2^2$. Then $\hat{\eta}(\mathbb{Z}^2)$ is also $\mathrm{Th}_\exists(\mathbb{Z}, \mathbb{R}_A)$-definable because for $x_1, x_2 \in \mathbb{R}_A$ we have $(x_1, x_2) \in \eta(\mathbb{Z}^2)$ iff

$$\exists y_1, y_2 \in \mathbb{R}_A,\ a, b \in \mathbb{Z}.\ \ y_1 = a \wedge y_2 = b \ \wedge\ x_1 = \tfrac{1}{2} \cdot y_1^2 + y_2^2 \wedge x_2 = y_2^2.$$

The following theorem shows that instead of regarding *solvable loops* [RK04], w.l.o.g. we can restrict ourselves to *twn*-loops. The reason is that every solvable loop with real eigenvalues can be transformed into a *twn*-loop by a *linear* automorphism η, i.e., the degree $\deg(\eta)$ of η is 1, where $\deg(\eta) = \max_{1 \leq i \leq d} \deg(\eta(x_i))$.

Theorem 9.1.16 (*twn*-Transformability of Solvable Loops)**.** *Let $(\varphi, \vec{u})$ be a solvable loop with real eigenvalues. Then one can compute a linear automorphism $\eta \in \mathrm{Aut}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$ such that $Tr_\eta(\varphi, \vec{u})$ is twn.*

We recapitulate our most important results on *Tr* in the following corollary. Here, we generalize the result of Thm. 9.1.10 to the setting where we consider termination on some update-invariant and $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_A)$-definable set.

Corollary 9.1.17 (Properties of *Tr*)**.** *Let $(\varphi, \vec{u})$ be a loop, $\eta \in \mathrm{Aut}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$, $Tr_\eta(\varphi, \vec{u}) = (\varphi', \vec{u}\,')$, and $F \subseteq \mathcal{S}^d$ be $\vec{u}$-invariant and $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_A)$-definable. Then*

(i) $\hat{\eta}(F) \subseteq \mathbb{R}_A^d$ is $\vec{u}\,'$-invariant and $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_A)$-definable,

(ii) $(\varphi, \vec{u})$ terminates on F iff $(\varphi', \vec{u}\,')$ terminates on $\hat{\eta}(F)$, and

(iii) $\vec{e} \in F$ *witnesses (eventual) non-termination of* $(\varphi, \vec{u})$ *iff* $\hat{\eta}(\vec{e}) \in \hat{\eta}(F)$ *witnesses (eventual) non-termination of* $(\varphi', \vec{u}\,')$.

9.2 Finding Automorphisms to Transform Loops into *twn*-Form

The goal of the transformation from Sect. 9.1 is to transform $(\varphi, \vec{u})$ into *twn*-form such that our techniques presented in Ch. 6 to 8 can also be used to decide termination, to decide the halting problem, and to compute an upper bound on the runtime of loops which can be transformed into *twn*-form. Hence, the two remaining challenges are to find a suitable automorphism $\eta \in \mathrm{Aut}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$ such that $Tr_\eta(\varphi, \vec{u})$ is *twn* and to adapt our techniques from Ch. 6 to 8 such that they apply to loops which can be transformed into *twn*-form. The latter challenge is discussed in Sect. 9.3. In this section, we present two techniques to check the existence of such automorphisms *constructively*, i.e., these techniques can also be used to compute such automorphisms.

Note that the search for suitable automorphisms is closely related to the question if a polynomial automorphism can be conjugated into a "de Jonquiéres"-automorphism, a difficult question from algebraic geometry [Ess00]. So future advances in this field may help to improve the results of the current section.

The first technique (Thm. 9.2.3) reduces the search for a suitable automorphism *of bounded degree* to $\mathrm{Th}_\exists(\mathbb{R}_A)$. It is known that for any automorphism the degree of its inverse has an upper bound in terms of the length d of $\vec{x}$, see [Ess00, Cor. 2.3.4].

Theorem 9.2.1 (Bounding Degree of Inverse Automorphism, [Ess00])**.** *Let* $\eta \in \mathrm{Aut}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$. *Then we have* $\deg(\eta^{-1}) \leq (\deg(\eta))^{d-1}$.

By Thm. 9.2.1, checking if an endomorphism is indeed an automorphism can be reduced to $\mathrm{Th}_\exists(\mathbb{R}_A)$. To do so, one encodes the existence of suitable coefficients of the polynomials $\eta^{-1}(x_1), \ldots, \eta^{-1}(x_d)$, which all have at most degree $(\deg(\eta))^{d-1}$.

Lemma 9.2.2. *Let* $\eta \in \mathrm{End}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$. *Then the question whether* $\eta \in \mathrm{Aut}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$ *holds is reducible to* $\mathrm{Th}_\exists(\mathbb{R}_A)$.

Based on Lemma 9.2.2, we now present our first technique to find an automorphism η that transforms a loop into *twn*-form.

Theorem 9.2.3 (*Tr* With Automorphisms of Bounded Degree)**.** *For any* $\delta \geq 0$, *the question whether there exists an* $\eta \in \mathrm{Aut}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$ *with* $\deg(\eta) \leq \delta$ *such that* $Tr_\eta(\varphi, \vec{u})$ *is twn is reducible to* $\mathrm{Th}_\exists(\mathbb{R}_A)$.

So if the degree of η is bounded a priori, it is decidable whether there exists an $\eta \in \mathrm{Aut}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$ such that $Tr_\eta(\varphi, \vec{u})$ is *twn* since $\mathrm{Th}_\exists(\mathbb{R}_A)$ is decidable.

We call a loop *twn-transformable* if there is an $\eta \in \mathrm{Aut}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$ such that $Tr_\eta(\varphi, \vec{u})$ is *twn*. By Thm. 9.2.3, *twn*-transformability is *semi-decidable* since one can increment δ until a suitable automorphism is found. So in other words, any loop which is transformable to a *twn*-loop can be transformed via Thm. 9.2.3.

We call our transformation *Tr complete* for a class of loops if *every* loop from this class is *twn*-transformable. For such classes of loops, a suitable automorphism is *computable* by Thm. 9.2.3. Together with Thm. 9.1.16, we get the following corollary.

Corollary 9.2.4. *Tr is complete for solvable loops with real eigenvalues.*

Note that for solvable loops $(\varphi, \vec{u})$, instead of computing η using Thm. 9.2.3, the proof of Thm. 9.1.16 yields a more efficient way to compute a linear automorphism η such that $Tr_\eta(\varphi, \vec{u})$ is *twn*. For this, one computes the Jordan normal form of each A_i (see Def. 4.1.6), which is possible in polynomial time (see e.g., [Cai94; Gie95; RV96]).

Our second technique to find suitable automorphisms for our transformation is restricted to *linear* automorphisms. In this case, it is decidable whether a loop can be transformed into a *twn*-loop $(\varphi', \vec{u}')$ where the monomial for x_i has the coefficient 1 in each u_i'. The decision procedure checks whether a certain Jacobian matrix is *strongly nilpotent*, i.e., it is not based on a reduction to $\mathrm{Th}_\exists(\mathbb{R}_A)$.

Definition 9.2.5 (Strong Nilpotence)**.** Let $J \in \left(\mathbb{R}_A[\vec{x}]\right)^{d\times d}$ be a matrix of polynomials. For all $1 \leq i \leq d$, let $\vec{y}^{(i)}$ be a vector of fresh variables. J is *strongly nilpotent* if $\prod_{i=1}^{d} J[\vec{x}/\vec{y}^{(i)}] = 0^{d\times d}$, where $0^{d\times d} \in \left(\mathbb{R}_A[\vec{x}]\right)^{d\times d}$ is the zero matrix.

Our second technique is formulated in the following theorem which follows from an existing result in linear algebra [EH96, Thm. 1.6.].

Theorem 9.2.6 (*Tr* With Linear Automorphisms, [EH96])**.** *Let $(\varphi, \vec{u})$ be a loop. The Jacobian matrix $(\frac{\partial(u_i - x_i)}{\partial x_j})_{1\leq i,j\leq d} \in \left(\mathbb{R}_A[\vec{x}]\right)^{d\times d}$ is strongly nilpotent iff there exists a linear automorphism $\eta \in \mathrm{Aut}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$ with*

$$Tr_\eta(\varphi, \vec{u}) = (\varphi', (x_1 + p_1, \ldots, x_d + p_d)) \tag{9.2.1}$$

and $p_i \in \mathbb{R}_A[x_{i+1}, \ldots, x_d]$. Thus, $Tr_\eta(\varphi, \vec{u})$ is twn.

As strong nilpotence of the Jacobian matrix is clearly decidable, Thm. 9.2.6 gives rise to a decision procedure for the existence of a linear automorphism that transforms $(\varphi, \vec{u})$ to the form (9.2.1).

$$\textbf{while } \left(4 \cdot {x_2}^2 + x_1 + x_2 + x_3 > 0\right) \textbf{ do } \{\, (x_1, x_2, x_3) \longleftarrow (u_1, u_2, u_3) \,\}$$

$$\begin{aligned} \text{with} \quad u_1 &= x_1 + 8 \cdot x_1 \cdot {x_2}^2 + 16 \cdot {x_2}^3 + 16 \cdot {x_2}^2 \cdot x_3 \\ u_2 &= x_2 - {x_1}^2 - 4 \cdot x_1 \cdot x_2 - 4 \cdot x_1 \cdot x_3 - 4 \cdot {x_2}^2 - 8 \cdot x_2 \cdot x_3 - 4 \cdot {x_3}^2 \\ u_3 &= x_3 - 4 \cdot x_1 \cdot {x_2}^2 - 8 \cdot {x_2}^3 - 8 \cdot {x_2}^2 \cdot x_3 + {x_1}^2 + 4 \cdot x_1 \cdot x_2 + \\ &\quad 4 \cdot x_1 \cdot x_3 + 4 \cdot {x_2}^2 + 8 \cdot x_2 \cdot x_3 + 4 \cdot {x_3}^2 \end{aligned}$$

Figure 9.2: Loop $\mathbf{L}_{hrb}$

Example 9.2.7. The loop $\mathbf{L}_{hrb}$ in Fig. 9.2 on $\mathcal{S}^3$ shows how our results enlarge the class of loops where termination is reducible to $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathrm{A})$. This loop is clearly *not* in *twn*-form. To transform it, we use Thm. 9.2.6. The Jacobian matrix J of $(u_1 - x_1, u_2 - x_2, u_3 - x_3)$ is:

$$\begin{bmatrix} 8{\cdot}x_2^2 & 16{\cdot}x_1{\cdot}x_2{+}48{\cdot}x_2^2{+}32{\cdot}x_2{\cdot}x_3 & 16{\cdot}x_2^2 \\ -2{\cdot}x_1{-}4{\cdot}x_2{-}4{\cdot}x_3 & -4{\cdot}x_1{-}8{\cdot}x_2{-}8{\cdot}x_3 & -4{\cdot}x_1{-}8{\cdot}x_2{-}8{\cdot}x_3 \\ -4{\cdot}x_2^2{+}2{\cdot}x_1{+}4{\cdot}x_2{+}4{\cdot}x_3 & -8{\cdot}x_1{\cdot}x_2{-}24{\cdot}x_2^2{-}16{\cdot}x_2{\cdot}x_3{+}4{\cdot}x_1{+}8{\cdot}x_2{+}8{\cdot}x_3 & -8{\cdot}x_2^2{+}4{\cdot}x_1{+}8{\cdot}x_2{+}8{\cdot}x_3 \end{bmatrix}$$

One easily checks that J is strongly nilpotent. Thus, by Thm. 9.2.6 the loop can be transformed into *twn*-form by a linear automorphism. Indeed, consider the linear automorphism $\eta \in \mathrm{Aut}_{\mathbb{R}_\mathrm{A}}\left(\mathbb{R}_\mathrm{A}[\vec{x}]\right)$ induced by the matrix $M = \left[\begin{smallmatrix} 1 & 1 & 1 \\ 0 & 2 & 0 \\ 1 & 2 & 2 \end{smallmatrix}\right]$, i.e.,

$$x_1 \mapsto x_1 + x_2 + x_3, \quad x_2 \mapsto 2 \cdot x_2, \quad x_3 \mapsto x_1 + 2 \cdot x_2 + 2 \cdot x_3$$

with its inverse η^{-1}

$$x_1 \mapsto 2 \cdot x_1 - x_3, \quad x_2 \mapsto \tfrac{1}{2} \cdot x_2, \quad x_3 \mapsto -x_1 - \tfrac{1}{2} \cdot x_2 + x_3.$$

If we transform $\mathbf{L}_{hrb}$ with η, we obtain the *twn*-loop $\mathbf{L}_{ex}$ in Fig. 4.1(b). If $\mathcal{S} = \mathbb{R}_\mathrm{A}$, $\mathbf{L}_{ex}$ terminates on $\mathbb{R}_\mathrm{A}^3$ iff $\mathbf{L}_{hrb}$ terminates on $\mathbb{R}_\mathrm{A}^3$ by Thm. 9.1.10, i.e., as seen in Ex. 6.1.8, $\mathbf{L}_{hrb}$ does not terminate on $\mathbb{R}_\mathrm{A}^3$. Now assume $\mathcal{S} = \mathbb{Z}$, i.e., we are interested in termination of $\mathbf{L}_{hrb}$ on $\mathbb{Z}^3$ instead of $\mathbb{R}_\mathrm{A}^3$. Note that $\hat{\eta}$ maps $\mathbb{Z}^3$ to

the set of all $\mathbb{Z}$-linear combinations of columns of M, i.e.,

$$\hat{\eta}(\mathbb{Z}^3) = \{a \cdot (1,0,1) + b \cdot (1,2,2) + c \cdot (1,0,2) \mid a,b,c \in \mathbb{Z}\}.$$

By Cor. 9.1.17, $\mathbf{L}_{ex}$ terminates on $\hat{\eta}(\mathbb{Z}^3)$ iff $\mathbf{L}_{hrb}$ terminates on $\mathbb{Z}^3$. Moreover, $\hat{\eta}(\mathbb{Z}^3)$ is $\mathrm{Th}_\exists(\mathbb{Z}, \mathbb{R}_\mathrm{A})$-definable: We have $(x_1, x_2, x_3) \in \hat{\eta}(\mathbb{Z}^3)$ iff

$$\exists a,b,c \in \mathbb{Z}.\ x_1 = a \cdot 1 + b \cdot 1 + c \cdot 1 \wedge x_2 = b \cdot 2 \wedge x_3 = a \cdot 1 + b \cdot 2 + c \cdot 2.$$

We will discuss how to analyze termination of *tnn*-loops like $\mathbf{L}_{ex}$ on update invariant and $\mathrm{Th}_\exists(\mathbb{Z}, \mathbb{R}_\mathrm{A})$-definable sets like $\hat{\eta}(\mathbb{Z}^3)$ in Sect. 9.3.

To summarize, if a loop is *twn*-transformable, we can *always* find a suitable automorphism via Thm. 9.2.3. So whenever Thm. 9.2.6 is applicable, a suitable linear automorphism can also be obtained by using Thm. 9.2.3 for some fixed degree $\delta \geq 1$. Hence, our first technique from Thm. 9.2.3 subsumes our second one from Thm. 9.2.6. However, while Thm. 9.2.3 is *always* applicable, Thm. 9.2.6 is *easier* to apply. The reason is that for Thm. 9.2.3 one has to check validity of a possibly *non-linear* formula over the reals, where the degree of the occurring polynomials depends on δ and the update $\vec{u}$ of the loop, and the number of variables can be exponential in d, see Lemma 9.2.2. So even when searching for a linear automorphism, one may obtain a non-linear formula if the loop is non-linear. In contrast, Thm. 9.2.6 only requires linear algebra. So it is preferable to first check whether the loop can be transformed into a *twn*-loop $(\varphi', (x_1 + p_1, \ldots, x_d + p_d))$ with $x_i \notin \mathcal{V}(p_i)$ via a linear automorphism. This check is *decidable* due to Thm. 9.2.6.

Note that the proof of Thm. 9.2.3 is constructive. Moreover, the proof of [EH96, Thm. 1.6.] is also constructive: the idea is to use basic results from linear algebra to compute an invertible matrix $T \in \mathbb{R}_\mathrm{A}^{d \times d}$ such that $T \cdot J \cdot T^{-1}$ is a triangular matrix where J is the Jacobian matrix $(\frac{\partial(u_i - x_i)}{\partial x_j})_{1 \leq i,j \leq d}$. Then η with $\eta(\vec{x}) = T \cdot \vec{x}$ transforms the loop into the form (9.2.1). Hence, Thm. 9.2.6 is also constructive. Thus, we can not only check the existence of a suitable automorphism, but we can also compute it if its existence can be proven.

9.3 Analyzing *twn*-transformable loops

In this section, we generalize our results from Ch. 6 to 8 to *twn*-transformable loops. As explained before, a *twn*-transformable loop over some update-invariant and $\mathrm{Th}_\exists(S, \mathbb{R}_\mathrm{A})$-definable set is transformed into a *twn*-loop over some update-invariant

and $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$-definable set F. Thus, in this section we fix a *twn*-loop $(\varphi, \vec{u})$ and such a set F. Let $\psi_F \in \mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$ define F.

9.3.1 Deciding Termination

We begin with generalizing our results from Ch. 6, i.e., we show how to reduce termination of $(\varphi, \vec{u})$ on F to $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$. Note that here we indeed rely on the update-invariance of F: otherwise, eventual non-termination and non-termination of $(\varphi, \vec{u})$ on F would not be equivalent. In Ch. 6, this equivalence is crucial since we reduce non-termination via eventual non-termination to $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$.

Again, let $\vec{q}_{norm}$ be the normalized closed form of $\vec{u}$. Similar to (6.1.5), $(\varphi, \vec{u})$ is eventually non-terminating on F iff

$$\exists \vec{x} \in F,\ n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ \varphi(\vec{q}_{norm}). \tag{9.3.1}$$

In Thm. 6.1.9, we have seen that given a propositional formula ξ over the atoms $\{p \triangleright 0 \mid p \in \mathbb{NPE}[\vec{x}], \triangleright \in \{\geq, >\}\}$, one can reduce validity of $\exists \vec{x} \in \mathcal{S}^d, n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ \xi$ to validity of a formula $\exists \vec{x} \in \mathcal{S}^d.\, \mathrm{red}(\xi) \in \mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$ and the resulting formula can be computed in polynomial time from ξ. Thus, by considering the formula $\exists \vec{x} \in \mathbb{R}_\mathbb{A}^d.\, \mathrm{red}(\xi) \wedge \psi_F \in \mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$ we obtain the following corollary.

Corollary 9.3.1. *Given a propositional formula ξ over the atoms $\{p \triangleright 0 \mid p \in \mathbb{NPE}[\vec{x}], \triangleright \in \{\geq, >\}\}$, one can reduce validity of*

$$\exists \vec{x} \in F, n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ \xi$$

to validity of a formula in $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$ in polynomial time.

Combining (9.3.1) and Cor. 9.3.1, we get the following result which is a refined version of Thm. 6.1.10.

Corollary 9.3.2 (Reducing Termination)**.** *Termination of tnn-loops (resp. twn-loops) on F is reducible to $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$.*

Example 9.3.3. Reconsider Ex. 9.2.7, where we have seen that $\mathbf{L}_{hrb}$ (see Fig. 9.2) terminates on $\mathbb{Z}^3$ iff $\mathbf{L}_{ex}$ (see Fig. 4.1(b)) terminates on the update-invariant and $\mathrm{Th}_\exists(\mathbb{Z}, \mathbb{R})$-definable set $\hat{\eta}(\mathbb{Z}^3) = F$ defined by

$$\exists a, b, c \in \mathbb{Z}.\ x_1 = a \cdot 1 + b \cdot 1 + c \cdot 1 \wedge x_2 = b \cdot 2 \wedge x_3 = a \cdot 1 + b \cdot 2 + c \cdot 2.$$

In Ex. 6.1.8, we have seen that $(-4, 2, 1)$ witnesses eventual non-termination

of $\mathbf{L}_{ex}$. As $\hat{\eta}(-9,1,4) = (-4,2,1)$, we have $(-4,2,1) \in F$. Furthermore, $(-9,1,4)$ witnesses eventual non-termination of $\mathbf{L}_{hrb}$ on $\mathbb{Z}^3$ by Cor. 9.1.17. Hence, $\mathbf{L}_{hrb}$ does *not* terminate on $\mathbb{Z}^3$.

Additionally, we get the following refined version of Cor. 6.1.11.

Corollary 9.3.4 ((Semi-)Decidability of (Non-)Termination)**.** *Let $(\varphi, \vec{u})$ be a twn-loop and F update-invariant and $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$-definable.*

(i) The loop $(\varphi, \vec{u})$ terminates over $\mathbb{R}_\mathbb{A}$ iff it terminates over $\mathbb{R}$.

(ii) Termination of $(\varphi, \vec{u})$ on F is decidable if $\mathcal{S} = \mathbb{R}_\mathbb{A}$ or $\mathcal{S} = \mathbb{R}$.

(iii) Non-termination of $(\varphi, \vec{u})$ on F is semi-decidable if $\mathcal{S} = \mathbb{Z}$ or $\mathcal{S} = \mathbb{Q}$.

Of course, Lemma 6.1.12 also holds in this setting.

Corollary 9.3.5. *Let $\xi = \varphi(\vec{q}_{norm})$. Then $\vec{e} \in \mathbb{R}_\mathbb{A}$ witnesses eventual non-termination of $(\varphi, \vec{u})$ on F iff $\mathrm{red}(\xi)(\vec{e}) \wedge \psi_F(\vec{e})$.*

Finally, Algorithm 9.3.1 refines Algorithm 6.1.1 to check termination of *twn*-transformable-loops.

Algorithm 9.3.1: Checking Termination

Input: a *twn*-transformable-loop $(\varphi, \vec{u})$ and $\psi_F \in \mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathbb{A})$
Result: $\top$ resp. $\bot$ if (non-)termination of $(\varphi, \vec{u})$ on F is proven, ? otherwise
$(\varphi, \vec{u}) \leftarrow Tr_\eta(\varphi, \vec{u})$, $\psi_F \leftarrow \psi_{\hat{\eta}(F)}$, such that $(\varphi, \vec{u})$ becomes *twn*
if $(\varphi, \vec{u})$ *is not tnn* **then** $(\varphi, \vec{u}) \leftarrow (\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$
$\vec{q} \leftarrow$ closed form for $\vec{u}$
if *(in)validity of* $\psi_F \wedge \mathrm{red}(\varphi(\vec{q}_{norm}))$ *cannot be proven* **then return** ?
if $\psi_F \wedge \mathrm{red}(\varphi(\vec{q}_{norm}))$ *is valid* **then return** $\bot$ **else return** $\top$

9.3.2 Deciding the Halting Problem

Our results on the decidability of the halting problem can also be extended to *twn*-transformable loops in a straightforward way. In Thm. 9.1.10, we showed that the transformation Tr induces a bijection on the sets of non-terminating inputs. Thus, given a non-*twn* loop $(\varphi, \vec{u})$, an input $\vec{e} \in \mathbb{R}_\mathbb{A}^d$, and a polynomial automorphism η which transforms $(\varphi, \vec{u})$ into a *twn*-loop $(\varphi', \vec{u}')$, we can proceed as follows to decide whether $(\varphi, \vec{u})$ terminates on $\vec{e}$:

1. Transform the loop $(\varphi, \vec{u})$ into the *twn*-loop $(\varphi', \vec{u}\,')$ by using η.
2. Transform the input $\vec{e}$ into $\vec{e}\,' = \hat{\eta}(\vec{e})$.
3. Apply Algorithm 7.1.1 to $(\varphi', \vec{u}\,')$ and $\vec{e}\,'$.

So we can not only decide the halting problem for all *twn*-loops over $\mathbb{R}_{\mathbb{A}}$, but we can decide it for all *twn-transformable* loops which can be transformed into *twn*-form by an $\mathbb{R}_{\mathbb{A}}$-automorphism, i.e., we have derived the following corollary.

Corollary 9.3.6. *The halting problem for twn-transformable loops is decidable.*

Nevertheless, we have to keep in mind that *twn*-transformability itself is only semi-decidable (see Thm. 9.2.3).

> **Example 9.3.7.** Reconsider Ex. 7.1.9, where we showed that $(0, -2, 1)$ witnesses non-termination of $\mathbf{L}_{ex}$. By Ex. 9.3.3, we have $\hat{\eta}(-1, -1, 2) = (0, -2, 1)$. Thus, $(-1, -1, 2)$ witnesses non-termination of $\mathbf{L}_{hrb}$.

9.3.3 Computing Runtime Bounds

Our program transformation allows us to extend our computability results for runtime bounds to certain *twn*-transformable loops as well. But while the results of Ch. 6 and 7 immediately carry over to any *twn-transformable* loop, this is not true for the results of Ch. 8. The reason is that we assumed that the coefficients of the poly-exponential expressions in $\varphi(\vec{q})$ are elements of $\mathbb{Q}[\vec{x}]$ (resp. $\mathbb{Z}[\vec{x}]$, see Simplification 4). This is crucial for the proof of Lemma 8.1.8.

So when applying an automorphism to a non-*twn* loop over the integers, we have to ensure that this requirement is still valid after the transformation. But when applying arbitrary $\mathbb{R}_{\mathbb{A}}$-automorphisms, this cannot be ensured since the transformed program may contain real algebraic numbers. Thus, we have to restrict ourselves to $\mathbb{Q}$-automorphisms instead of $\mathbb{R}_{\mathbb{A}}$-automorphisms. Then the technique presented in the current section can indeed be applied to all loops that can be transformed into *twn*-form via $\mathbb{Q}$-automorphisms, e.g., solvable loops with integer eigenvalues. Note that the inequation (8.1.4) does not hold directly anymore as we now have $\vec{e} \in \hat{\eta}\left(\mathbb{Z}^d\right) \subseteq \mathbb{Q}^d$ instead of $\vec{e} \in \mathbb{Z}^d$ and thus, we may have $0 < |\alpha_\ell(\vec{e})| < 1$. However, this can easily be fixed. The reason is that for each $\vec{e} \in \hat{\eta}\left(\mathbb{Z}^d\right)$, the occurring denominators stem from the polynomials $\eta(x_1), \ldots, \eta(x_d) \in \mathbb{Q}[\vec{x}]$. Hence, the least common multiple k of the denominators that occur in $\eta(x_1), \ldots, \eta(x_d)$ is a multiple of the denominators that occur in $\vec{e}$, i.e., $\hat{\eta}\left(\mathbb{Z}^d\right) \subseteq \frac{1}{k} \cdot \mathbb{Z}^d$. So by multiplying the inequations $p \triangleright 0$ in $\varphi(\vec{q})$ by

k, one obtains a poly-exponential expression with coefficients α such that $\alpha(\vec{e}) \in \mathbb{Z}$ for all $\vec{e} \in \hat{\eta}\left(\mathbb{Z}^d\right)$. Here, however, the following has to be taken into account: When $(\varphi, \vec{u})$ is transformed into $(\varphi', \vec{u}\,')$ via a $\mathbb{Q}$-automorphism η, then the run of $(\varphi, \vec{u})$ on $\vec{e} \in \mathbb{Z}^d$ corresponds to the run of $(\varphi', \vec{u}\,')$ on $\hat{\eta}(\vec{e})$ (and vice versa, see Lemma 9.1.7). Thus, if $f(\|\vec{e}\,'\|)$ is a bound on $rt_{(\varphi', \vec{u}\,')}(\vec{e}\,')$ for all $\vec{e}\,' \in \hat{\eta}\left(\mathbb{Z}^d\right)$, we have for all $\vec{e} \in \mathbb{Z}^d$

$$f(\|\hat{\eta}(\vec{e})\|) \geq rt_{(\varphi', \vec{u}\,')}(\hat{\eta}(\vec{e})) \overset{\text{Lemma 9.1.7}}{=} rt_{(\varphi, \vec{u})}(\vec{e}).$$

So besides the loop under consideration, we also have to transform the resulting upper runtime bound.

Example 9.3.8. In Ex. 9.2.7, we have transformed the integer loop $\mathbf{L}_{hrb}$ into the loop $\mathbf{L}_{ex}$ via the automorphism $\eta \in \mathrm{Aut}_{\mathbb{Q}}\left(\mathbb{Q}[\vec{x}]\right)$, where $\eta(\vec{x}) = M \cdot \vec{x}$ and $M = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 0 \\ 1 & 2 & 2 \end{bmatrix}$. We have computed the runtime bound $f(\|(e_1, e_2, e_3)\|) = 24 \cdot \sum_{i=1}^{5}(|e_1| + |e_2| + |e_3|)^i + 27$ on the runtime of the loop $\mathbf{L}_{ex}$ on a terminating input $\vec{e} \in \mathbb{Z}^3$ in Ex. 8.1.9.

Now $\hat{\eta}(\mathbb{Z}^3) = M \cdot \mathbb{Z}^3 \subseteq \mathbb{Z}^3$, i.e., we do not need to do any rescaling as explained in the paragraph above. However, the above stated runtime bound cannot be transferred to $\mathbf{L}_{hrb}$ directly since we only know that the runtime of $\mathbf{L}_{hrb}$ on $\vec{e}$ is bounded from above by the runtime of $\mathbf{L}_{ex}$ on $\hat{\eta}(\vec{e})$. Thus, we have to relate the size of $\hat{\eta}(\vec{e})$ to the size of $\vec{e}$. In the special case of linear automorphisms, we can proceed as follows. Remember that we consider the 1-norm $\|\cdot\|$. W.l.o.g., we assume that $\vec{e} \neq \vec{0}$:

$$\|\eta(\vec{e})\| = \|M \cdot \vec{e}\| = \frac{\|M \cdot \vec{e}\|}{\|\vec{e}\|} \cdot \|\vec{e}\| \leq \sup_{\vec{y} \in \mathbb{R}^3 \setminus \{\vec{0}\}} \frac{\|M \cdot \vec{y}\|}{\|\vec{y}\|} \cdot \|\vec{e}\|$$

A standard result from mathematics (see, e.g., [DR06, Eq. 2.27]) is that $\sup_{\vec{y} \in \mathbb{R}^3 \setminus \{\vec{0}\}} \frac{\|M \cdot \vec{y}\|}{\|\vec{y}\|}$ is the maximal 1-norm of a column of M, i.e., $\sup_{\vec{y} \in \mathbb{R}^3 \setminus \{\vec{0}\}} \frac{\|M \cdot \vec{y}\|}{\|\vec{e}\|} = 5$. Thus,

$$\|\eta(\vec{e})\| \leq 5 \cdot \|\vec{e}\|.$$

Hence, the runtime of $\mathbf{L}_{hrb}$ on $\vec{e}$ is bounded by

$$f(5 \cdot \|\vec{e}\|) = 24 \cdot \sum_{i=1}^{5} 5^i \cdot (|e_1| + |e_2| + |e_3|)^i + 27.$$

Asymptotically, the relation between the sizes of $\vec{e}$ and $\hat{\eta}(\vec{e})$ is described by $\|\hat{\eta}(\vec{e})\| \in$

$\mathcal{O}(\|\vec{e}\|^{\deg(\eta)})$, e.g., if η is *linear*, those sizes are asymptotically equal, if η has degree 2, $\|\hat{\eta}(\vec{e})\|$ is asymptotically at most quadratic in $\|\vec{e}\|$ and so on.

Thus, we obtain the following corollary from Cor. 8.2.4 on an asymptotic runtime bound for a *twn*-transformable loop.

Corollary 9.3.9. *Let $(\varphi, \vec{u})$ be a loop over $\mathbb{Z}$ which can be transformed into a twn-loop $(\varphi', \vec{u}\,')$ by $\eta \in \mathrm{Aut}_{\mathbb{Q}}\left(\mathbb{Q}[\vec{x}]\right)$. Then the runtime of $(\varphi, \vec{u})$ on a terminating input is bounded from above by a polynomial with degree*

$$\deg(\eta) \cdot \max\left\{ \sum_{i=1}^{d} \deg_{x_i}(m) \cdot \mathrm{depdeg}_{\vec{u}\,'}(x_i) \;\middle|\; r \triangleright 0 \text{ atom in } \varphi',\ m \text{ monomial in } r \right\}.$$

Example 9.3.10. Using Cor. 9.3.9, we obtain that the runtime of $\mathbf{L}_{hrb}$ on a terminating input $\vec{e} \in \mathbb{Z}^d$ is asymptotically bounded by $\|\vec{e}\|^5$ since we transformed $\mathbf{L}_{hrb}$ into $\mathbf{L}_{ex}$ via a linear automorphism.

Chapter 10

Complexity of Deciding Termination

In this chapter, we study the complexity of deciding termination for certain classes of loops by using our results from Ch. 6 and Sect. 9.3.1. So in contrast to Ch. 8, we are not interested in the runtime of a specific loop $(\varphi, \vec{u})$ on a specific input $\vec{e}$ but in how many operations we need to decide whether a given loop terminates. We first regard *linear-update* loops in Sect. 10.1, i.e., where the update is of the form $\vec{x} \leftarrow A \cdot \vec{x} + \vec{b}$ with $A \in \mathcal{S}^{d \times d}$ and $\vec{b} \in \mathcal{S}^d$. More precisely, we show that termination of linear loops with real spectrum is Co-NP-complete if $\mathcal{S} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}_{\mathbb{A}}\}$ and that termination of linear-update loops with real spectrum is $\forall\mathbb{R}$-complete if $\mathcal{S} = \mathbb{R}_{\mathbb{A}}$. Since our proof is based on a reduction to $\mathrm{Th}_{\exists}(\mathcal{S}, \mathbb{R}_{\mathbb{A}})$, and $\mathbb{R}_{\mathbb{A}}$ and $\mathbb{R}$ are elementary equivalent (see, e.g., [BPR06]), our results also hold if the program variables range over $\mathbb{R}$.

In Sect. 10.2, we use these results to discuss the complexity of deciding termination of arbitrary *twn*-loops. Thereafter, we also consider the *parametric complexity* of deciding termination for a certain class of loops in Sect. 10.3 and 10.4.

For obtaining the results in Sect. 10.1, 10.3, and 10.4 we use our transformation *Tr* which we have discussed in Ch. 9. Our results in Sect. 10.2 rely on our discussion of linearizing *twn*-loops in Sect. 8.3.

Before we begin with presenting our results, we briefly discuss the encoding of real algebraic numbers which we use in this chapter. For our complexity results, we assume the usual dense encoding of univariate polynomials, i.e., a polynomial of degree k is represented as a list of $k + 1$ coefficients. As discussed in [Roc18], many problems which are considered to be efficiently solvable become intractable if polynomials are encoded sparsely (e.g., as lists of monomials where each monomial is

while (φ) **do** {
$\quad \vec{x} \leftarrow A \cdot \vec{x} + \vec{b}$
}

(a) Linear-Update Loop

while $(\varphi \wedge x_{\vec{b}})$ **do** {
$\quad \begin{bmatrix} \vec{x} \\ x_{\vec{b}} \end{bmatrix} \leftarrow \begin{bmatrix} A & \vec{b} \\ \vec{0}^T & 1 \end{bmatrix} \cdot \begin{bmatrix} \vec{x} \\ x_{\vec{b}} \end{bmatrix}$
}

(b) Reduced Linear-Update Loop

Figure 10.1: Reduction of Linear-Update Loops

a pair of its non-zero coefficient and its degree). With densely encoded polynomials, all common representations of algebraic numbers can be converted into each other in polynomial time [BM17].

10.1 Linear-Update Loops

When analyzing linear-update loops, w.l.o.g. we can assume $\vec{b} = \vec{0}$ since a loop of the form in Fig. 10.1(a) terminates iff the loop in Fig. 10.1(b) terminates, where $x_{\vec{b}}$ is a fresh variable (see [HOW19; OPW15]). Moreover, $\vec{e}$ witnesses (eventual) non-termination for the loop in Fig. 10.1(a) iff $\begin{bmatrix} \vec{e} \\ 1 \end{bmatrix}$ witnesses (eventual) non-termination for the loop in Fig. 10.1(b). Note that the only eigenvalue of $\begin{bmatrix} A & \vec{b} \\ \vec{0}^T & 1 \end{bmatrix}$ whose multiplicity increases in comparison to A is 1. Thus, to decide termination of linear-update loops with real spectrum, it suffices to decide termination of loops of the form $(\varphi, A \cdot \vec{x})$ where A has only real eigenvalues.

Such loops can *always* be transformed into *twn*-form using our transformation Tr from Ch. 9. To compute the required automorphism η, we compute the Jordan normal form Q of A together with the corresponding transformation matrix T, i.e., T is an invertible real matrix such that $A = T^{-1} \cdot Q \cdot T$. Then Q is a triangular real matrix whose diagonal consists of the eigenvalues $\lambda \in \mathbb{R}_{\mathbb{A}}$ of A. Now we define $\eta \in \mathrm{End}_{\mathbb{R}_{\mathbb{A}}}\left(\mathbb{R}_{\mathbb{A}}[\vec{x}]\right)$ by $\eta(\vec{x}) = T \cdot \vec{x}$. Then $\eta \in \mathrm{Aut}_{\mathbb{R}_{\mathbb{A}}}\left(\mathbb{R}_{\mathbb{A}}[\vec{x}]\right)$ has the inverse $\eta^{-1}(\vec{x}) = T^{-1} \cdot \vec{x}$. Thus, $Tr_{\eta}(\varphi, A \cdot \vec{x})$ is a *twn*-loop with the update

$$(\eta(\vec{x}))\,(A \cdot \vec{x})\,(\eta^{-1}(\vec{x})) = T \cdot A \cdot T^{-1} \cdot \vec{x} = Q \cdot \vec{x}.$$

Note that if we want to analyze termination of the loop on $\mathcal{S}^d$, we have to consider termination of the transformed loop on $F = \eta(\mathcal{S}^d) = T \cdot \mathcal{S}^d$ (see Cor. 9.1.17).

The Jordan normal form Q as well as the matrices T and T^{-1} can be computed in polynomial time [Gie95; RV96]. Hence, we can decide whether all eigenvalues are real numbers in polynomial time by checking the diagonal entries of Q. Thus, we obtain the following lemma.

Lemma 10.1.1. *Let $(\varphi, A \cdot \vec{x})$ be a linear-update loop.*

(i) It is decidable in polynomial time whether A has only real eigenvalues.

(ii) If A has only real eigenvalues, we can compute a linear $\eta \in \mathrm{Aut}_{\mathbb{R}_A}\left(\mathbb{R}_A[\vec{x}]\right)$ such that $Tr_\eta(\varphi, A \cdot \vec{x})$ is a linear-update twn-loop in polynomial time.

(iii) If $(\varphi, A \cdot \vec{x})$ is a linear loop, so is $Tr_\eta(\varphi, A \cdot \vec{x})$.

Hence, the transformation *Tr* from Ch. 9 is *complete* for linear(-update) loops with real spectrum, i.e., every such loop can be transformed into a linear(-update) *twn*-loop. Note that the first part of Lemma 10.1.1 yields an efficient check whether a given linear(-update) loop has real spectrum.

As chaining (Def. 5.1.1) can clearly be done in polynomial time, w.l.o.g. we may assume that $Tr_\eta(\varphi, A \cdot \vec{x}) = (\varphi', Q \cdot \vec{x})$ is *tnn*. Next, to analyze termination of a *tnn*-loop, our technique of Ch. 6 (resp. Sect. 9.3.1) uses a closed form for the n-fold application of the update. For *tnn*-loops of the form $(\varphi', Q \cdot \vec{x})$ where Q is a triangular matrix with non-negative diagonal entries, a suitable (i.e., poly-exponential) closed form can be computed in polynomial time analogously to [Kin+19, Prop. 5.2]. This closed form is linear in $\vec{x}$ (we will discuss this closed form in Sect. 10.3, see Lemma 10.3.6).

According to our approach in Sect. 9.3.1, we now proceed as in Algorithm 9.3.1 and compute $\mathrm{red}(\varphi(\vec{q}_{norm})) \wedge \psi_F \in \mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_A)$. The construction of this formula can be done in polynomial time due to Cor. 9.3.1. Hence, we get the following lemma.

Lemma 10.1.2. *Let $(\varphi, A \cdot \vec{x})$ be a linear-update loop with real spectrum. Then we can compute a closed formula $\psi \in \mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_A)$ in polynomial time such that ψ is valid iff the loop is non-terminating. If φ is linear, so is ψ.*

Note that ψ is existentially quantified. Hence, if φ (and thus also ψ) is linear and $\mathcal{S} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}_A, \mathbb{R}\}$, invalidity of ψ is in Co-NP as validity of such formulas is in NP [PDM17]. Thus, we obtain the first main result of this section.

Theorem 10.1.3 (Co-NP-Completeness). *Termination of linear loops $(\varphi, A \cdot \vec{x} + \vec{b})$ with real spectrum over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}_A$, and $\mathbb{R}$ is* Co-NP*-complete.*

For Co-NP-hardness, let ξ be a propositional formula over the variables $\vec{x}$. Then $(\xi[x_i/(x_i > 0) \mid 1 \leq i \leq d], \vec{x})$ terminates iff ξ is unsatisfiable. So Co-NP-hardness follows from Co-NP-hardness of unsatisfiability of propositional formulas.

We now consider linear-update loops with real spectrum (and possibly non-linear conditions) on $\mathbb{R}_A^d$ and $\mathbb{R}^d$. Here, non-termination is $\exists\mathbb{R}$-complete.

Definition 10.1.4 ($\exists\mathbb{R}$ [Sch09; SchSt17]). Let $\mathrm{Th}_\exists(\mathbb{R})_\top = \{\psi \in \mathrm{Th}_\exists(\mathbb{R}) \mid \psi \text{ is satisfiable}\}$. The complexity class $\exists\mathbb{R}$ is the closure of $\mathrm{Th}_\exists(\mathbb{R})_\top$ under poly-time-reductions.

We have NP $\subseteq \exists\mathbb{R} \subseteq$ PSPACE (see [Can88]). By Lemma 10.1.2, non-termination of linear-update loops with real spectrum is in $\exists\mathbb{R}$. It is also $\exists\mathbb{R}$-hard since $(\varphi, \vec{x})$ is non-terminating iff φ is satisfiable. So non-termination is $\exists\mathbb{R}$-complete, i.e., termination is Co-$\exists\mathbb{R}$-complete (where Co-$\exists\mathbb{R}$ = $\forall\mathbb{R}$ [SchSt17]).

Theorem 10.1.5 ($\forall\mathbb{R}$-Completeness). *Termination of linear-update loops with real spectrum over $\mathbb{R}_A$ and $\mathbb{R}$ is $\forall\mathbb{R}$-complete.*

10.2 Arbitrary *twn*-Loops over $\mathcal{S} = \mathbb{R}_A, \mathbb{R}$

Now we discuss the complexity of deciding termination for arbitrary *twn*-loops. First of all, termination of arbitrary *twn*-loops over $\mathbb{R}_A$ and $\mathbb{R}$ is $\forall\mathbb{R}$-hard since any linear-update loop with real spectrum can be transformed into a *twn*-loop in polynomial time. Thus, we have a lower bound on the complexity of deciding termination for arbitrary *twn*-loops. Moreover, w.l.o.g. we restrict ourselves to *tnn*-loops as any *twn*-loop can be transformed into a *tnn*-loop by chaining (Def. 5.1.1) in polynomial time.

In Sect. 8.3, we have discussed the technique from [OBP16a] to linearize *tnn*-loops, i.e., given a *tnn*-loop $(\varphi, \vec{u})$ we can compute a linear-update *tnn*-loop $(\varphi', \vec{u}\,')$ such that $(\varphi, \vec{u})$ terminates if and only if $(\varphi', \vec{u}\,')$ does (Lemma 8.3.6). Clearly, linear-update *tnn*-loops have also real spectrum as the eigenvalues of a triangular matrix are its diagonal entries. So the question arises whether we can give tight complexity bounds for deciding termination of *tnn*-loops using this linearization. However, the loop $\mathbf{L}_{np}$ in Fig. 6.1 showed that in general, linearizing a *tnn*-loop is not in PSPACE. Thus, our technique is not in PSPACE so in particular not in $\forall\mathbb{R}$. Still, we can present an upper bound on the complexity of deciding termination for arbitrary *tnn*-loops.

In Thm. 8.3.12, we have seen that to linearize a loop $(\varphi, \vec{u})$, we have to introduce at most monomials of degree $mdepdeg = \max_{1\leq i\leq d} \mathrm{depdeg}_{\vec{u}}(x_i)$. It is a standard result from mathematics that there are $\binom{d+mdepdeg}{mdepdeg} = \binom{d+mdepdeg}{d} \in \mathcal{O}((mdepdeg +$

$d)^{mdepdeg}) \cap \mathcal{O}((mdepdeg + d)^d)$ many monomials in the variables $\vec{x}$ of degree at most $mdepdeg$. Thus, Algorithm 8.3.1 computes at most $\binom{d+mdepdeg}{mdepdeg}$ new updates.

The value that $mdepdeg$ takes can be bounded in terms of the number of variables d and the highest degree m occurring in a polynomial in $\vec{u}$. By definition, we have $mdepdeg \leq m^{d-1}$ (since at least one variable has dependency degree at most 1). For example, $\mathbf{L}_{np}$, where $m = d$, yields $mdepdeg = d^{d-1}$. Hence,

$$\binom{d + mdepdeg}{mdepdeg} \in \mathcal{O}((m^{d-1} + d)^d) \cap \mathcal{O}((m^{d-1} + d)^{m^{d-1}}) \subseteq \mathcal{O}((m^{d-1} + d)^d)$$

Now let us consider the expression $(m^{d-1} + d)^d$. Here, we denote by ld the logarithm with basis 2.

$$\begin{aligned}
& (m^{d-1} + d)^d \\
= & \sum_{k=0}^{d} \binom{d}{k} \cdot \left(m^{d-1}\right)^k \cdot d^{d-k} \\
\leq & \sum_{k=0}^{d} \binom{d}{k} \cdot \max\left(m^{d-1}, d\right)^d \\
\leq & \sum_{k=0}^{d} \binom{d}{k} \cdot \max\left(\left(m^{d-1}\right)^d, d^d\right) \\
\leq & \sum_{k=0}^{d} \binom{d}{k} \cdot \left(\left(m^{d-1}\right)^d + d^d\right) \\
= & \, 2^d \cdot \left(\left(m^{d-1}\right)^d + d^d\right) \\
= & \, 2^{d+\mathrm{ld}(m)\cdot(d-1)\cdot d} + 2^{d+\mathrm{ld}(d)\cdot d} \\
\leq & \, 2 \cdot 2^{d+\mathrm{ld}(m)\cdot(d-1)\cdot d}
\end{aligned}$$

Thus, $(m^{d-1} + d)^d$ is *exponential* in m and d, i.e., the number of iterations of Algorithm 8.3.1 is at most exponential in m and d. Here, we only take the iterations of the innermost loop into account, see Rem. 8.3.10. Any iteration of Algorithm 8.3.1 takes polynomial time in the size of the coefficients at the beginning of this iteration, i.e., the coefficients of the already computed updates. Moreover, the new coefficients which are computed in this iteration have a size which is polynomial in the size of the coefficients at the beginning of this iteration. Combining this insight and the exponential bound on the number of iterations, we have proven Lemma 10.2.1 (i).

However, if the number of variables d is bounded by a constant D, linearization is even in EXPTIME since Algorithm 8.3.1 does $\binom{d+mdepdeg}{mdepdeg} \in \mathcal{O}((D + m^D)^D) \subseteq \mathcal{O}(m^{D^2})$ many iterations, i.e., polynomially many in m. This proves Lemma 10.2.1 (ii).

Lemma 10.2.1. *Let $D \in \mathbb{N}$ be fixed. Linearizing a tnn-loop*

(i) is in 2-EXPTIME.

(ii) is in EXPTIME *if the number of variables is at most D.*

Combining Lemma 10.2.1 (i) with Def. 5.1.1 and Thm. 10.1.5 yields Thm. 10.2.2 (i): here, we use $\forall\mathbb{R} \subseteq$ PSPACE $\subseteq$ EXPTIME. Moreover, if the number of variables is bounded, checking validity of an existential formula over the reals is in P (see [BPR06]). So in this case, combining Lemma 10.2.1 (ii) with Def. 5.1.1 and Lemma 10.1.2 yields Thm. 10.2.2 (ii).

Theorem 10.2.2 (Membership in 3-EXPTIME)**.** *Let $D \in \mathbb{N}$ be fixed. Termination of twn-loops over $\mathbb{R}_\mathbb{A}$ and $\mathbb{R}$*

(i) is in 3-EXPTIME.

(ii) is in EXPTIME *if the number of variables is at most D.*

By Thm. 10.2.2 (i) , we have an upper bound on the complexity of deciding termination for *twn*-loops. So the complexity of deciding termination for *twn*-loops is between $\forall\mathbb{R}$ and 3-EXPTIME.

Finally, let us briefly discuss so-called solvable loops with periodic real eigenvalues, which are studied in [Kin+19]. These loops subsume *twn*-loops, i.e., deciding termination over $\mathbb{R}_\mathbb{A}$ or $\mathbb{R}$ for these loops is $\forall\mathbb{R}$-hard.

An algebraic number λ is *periodic real* if $\lambda^m \in \mathbb{R}$ for some $m \in \mathbb{N}_{\geq 1}$. For example, i is periodic real since $i^2 = -1 \in \mathbb{R}$. By a generalized version of chaining such a solvable loop with periodic real eigenvalues, we obtain a solvable loop with only real eigenvalues. By Thm. 9.1.16, solvable loops with real eigenvalues can be transformed into *twn*-loops in polynomial time (see the discussion after Cor. 9.2.4). Nevertheless, this generalized chaining is in EXPSPACE (it is in P if the number of variables is bounded by a constant), see e.g., [Eve+03; Kin+19; OPW15].

Thus, we obtain the following corollary from Thm. 10.2.2:

Corollary 10.2.3. *Let $D \in \mathbb{N}$ be fixed. Termination of solvable loops with periodic real eigenvalues over $\mathbb{R}_\mathbb{A}$ and $\mathbb{R}$*

(i) is in 4-EXPTIME.

(ii) is in EXPTIME *if the number of variables is at most D.*

Since EXPSPACE is a strict superset of Co-NP and ∀ℝ (see, e.g., [AB09]), this insight cannot be used to lift our completeness results from Sect. 10.1 to linear(-update) loops with only periodic real eigenvalues.

10.3 Uniform Loops

In Sect. 10.1, we showed that termination of general linear loops with real spectrum is Co-NP-complete. However, for proving Co-NP-hardness of termination for linear loops with real spectrum, we used the trivial update $\vec{x} \leftarrow \vec{x}$, i.e., the update induced by the identity matrix. Therefore, the question arises whether adding further non-trivial restrictions to the update matrix, i.e., which exclude the identity matrix, leads to a "more efficient" decision procedure for termination (when assuming P ≠ NP). We now analyze a special case of linear loops (so-called *uniform* loops) and show that for these loops deciding termination is *polynomial*, provided that one fixes the number of eigenvalues of the update matrix.

In Sect. 10.3.1, we introduce the notions of uniform loops and parameterized decision problems, and state our main result on the complexity of deciding termination for uniform loops (Thm. 10.3.4). To prove this result, we first connect the closed forms for uniform loops to so-called *hierarchical expressions* (Sect. 10.3.2). However, the closed forms have an even more specific structure, which leads to the concept of *hierarchical partitions* (Sect. 10.3.3). Combining this concept with our decision procedure from Ch. 6 gives rise to so-called *sign-conditions* (Sect. 10.3.4). Finally, we present our algorithm to decide termination for uniform loops in Sect. 10.3.5. This then concludes the proof of Thm. 10.3.4.

10.3.1 Problem Statement

We begin with a formal definition of uniform loops.

Definition 10.3.1 (Uniform Loop)**.** A linear loop $(\varphi, A \cdot \vec{x})$ is *uniform* if all eigenvalues of A are *non-negative* real algebraic numbers and each eigenspace of A is one-dimensional, i.e., each eigenvalue has *geometric multiplicity* 1.

The latter property is equivalent to requiring that there is one and only one Jordan block for each eigenvalue in A's Jordan normal form.

In particular, this means that if A is triangular and all variables share the same factor λ in the update (i.e., if we have $x_i \leftarrow \lambda \cdot x_i + p_i$ for all $1 \leq i \leq d$), the relation

$>_{\vec{u}}$ must be total. So in particular, there must not be more than one update of the form $x_i \longleftarrow x_i$. However, the condition can still be an arbitrary Boolean formula over linear inequations. Thus, our complexity result is quite surprising since it shows that termination is easier to decide than satisfiability of the condition for this class of loops (e.g., unsatisfiability of linear arithmetic formulas over $\mathbb{R}_{\mathbb{A}}$ is Co-NP-complete). Intuitively, the reason is that our class of loops *prohibits* multiple updates like $x_i \longleftarrow x_i$ where variables "stabilize" and termination is essentially equivalent to satisfiability of the condition.

More precisely, we show that deciding termination of uniform loops is in XP (i.e., it is *slicewise polynomial*). XP is a *parameterized complexity class*, consisting of *parameterized decision problems*. These are decision problems that depend on *several* components and parameterized complexity regards their complexity as a function of these components.

Definition 10.3.2 (Parameterized Decision Problem, XP [DF99])**.** A *parameterized decision problem* is a language $\mathcal{L} \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. The second component (from $\mathbb{N}$) is called the *parameter* of the problem.

A parameterized problem $\mathcal{L}$ is *slicewise polynomial* if the time needed for deciding the question "$(x, k) \in \mathcal{L}$?" is in $\mathcal{O}\left(|x|^{f(k)}\right)$ where f is a computable function depending only on k. The corresponding complexity class is called XP.

In the following, we show that deciding termination of uniform loops is in XP, where the parameter is the number of eigenvalues. Here, we only consider linear loops $(\varphi, A \cdot \vec{x})$ as termination of a loop of the form in Fig. 10.1(a) is equivalent to termination of the loop in Fig. 10.1(b). For such loops and an underlying ring $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathbb{A}}$, we are interested in *k-termination*.

Definition 10.3.3 (k-Termination)**.** We define the parameterized decision problem *k-termination* as follows: We have $((\varphi, A \cdot \vec{x}), k, \mathcal{S}) \in \mathcal{L}_{k\text{-termination}}$ iff $(\varphi, A \cdot \vec{x})$ terminates over $\mathcal{S}$ and A has k eigenvalues.

In the remainder of this section, we prove the following theorem.

Theorem 10.3.4 (Parameterized Complexity of k-Termination)**.** *For uniform loops over $\mathcal{S} \in \{\mathbb{R}_{\mathbb{A}}, \mathbb{R}\}$, k-termination is in* XP.

So in other words, for arbitrary but fixed $k \in \mathbb{N}$, termination over the reals of uniform loops with k eigenvalues is decidable in polynomial time.

while $(x_1 \geq -x_3)$ **do** {
$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 + x_2 \\ x_2 + x_3 \\ x_3 - 1 \end{bmatrix}$
}

(a) Update via Polynomials

while $(x_1 \geq -x_3 \wedge x_4 = 1)$ **do** {
$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$
}

(b) Update via Matrix Multiplication

Figure 10.2: Linear Loops from [BDG19]

while (φ) **do** {
$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} \lambda \cdot x_1 + c_1 \\ \lambda \cdot x_2 + c_2 \end{bmatrix}$
}

(a) Independent Updates

while (φ) **do** {
$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leftarrow \begin{bmatrix} x_1 + x_2 + x_3 \\ 2 \cdot x_2 + x_3 \\ x_3 + x_4 \\ 2 \cdot x_4 \end{bmatrix}$
}

(b) Update Inducing a Total Order

Figure 10.3: Non-Uniform Loops

To grasp the class of uniform loops intuitively, let us look at triangular linear loops. A triangular linear loop where $\succ_{\vec{u}}$ is a total ordering (or equivalently, the variables can be re-ordered such that the super-diagonal of A does not contain zeros) and all diagonal entries of A are non-negative and equal, is uniform. Thus, loops like the leading example from [BDG19] given in Fig. 10.2(a) which is equivalent to the loop in Fig. 10.2(b) are uniform.

Lemma 10.3.5. *A triangular linear loop where all diagonal entries are identical and non-negative is uniform iff $\succ_{\vec{u}}$ is a total ordering .*

Indeed, triangular loops of the form in Fig. 10.3(a) where $c_1, c_2 \in \mathcal{S}$ are constants and x_1 and x_2 are multiplied by the same factor λ in each iteration are not uniform. The reason is that x_2 does not occur in the update of x_1 and x_1 does not occur in the update of x_2. Hence, we have $x_1 \not\succ_{\vec{u}} x_2$ and $x_2 \not\succ_{\vec{u}} x_1$.

Unfortunately, the intuition of Lemma 10.3.5 cannot be used if the update matrix has multiple eigenvalues. Consider the triangular loop in Fig. 10.3(b). Here, $x_1 \succ_{\vec{u}} x_2 \succ_{\vec{u}} x_3 \succ_{\vec{u}} x_4$, i.e., the ordering induced by the update is total. However, the matrix

$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 2 \end{bmatrix}$ has a two-dimensional eigenspace w.r.t. its eigenvalue 1, i.e., the loop is not uniform.

To give an intuition on how hard the restriction to uniform loops is, note that in *practice* one is mainly interested in deciding termination of triangular loops where the eigenvalues are from $\{0, 1\}$. The reason is that after the transformation into *tnn*-form there are no negative eigenvalues anymore, and eigenvalues that are greater than 1 lead to exponential growth of the variable. In that case, if the loop terminates, termination is usually reached after very few steps (e.g., consider a loop like $(x \leq c, 2 \cdot x)$ for any constant c). By the same considerations, there are usually several program variables that belong to the same eigenvalue 1 or 0. Hence, the number of eigenvalues k is usually smaller than the number d of program variables. Thus, then the complexity result of this section improves over the result of Ch. 10 as it shows that the complexity for deciding termination is only exponential in k but not in d.

Nevertheless, we should note that –although they occur frequently in practice– from a *theoretical* point of view uniform loops where $k < d$ are "rare". Intuitively, this means that when generating a matrix with real algebraic entries randomly, the probability that it is uniform is very small. For example, it is well known that in the set of *complex* $d \times d$ matrices the matrices with d distinct eigenvalues form a dense subset, i.e., the probability that a randomly generated complex $d \times d$ matrix has d distinct eigenvalues is 1. See, e.g., [Ede97; EKS94; HLM07] for further reading on this topic. However, this is in line with our results: as we have seen that deciding termination for linear loops with real spectrum is Co-NP-complete, a (non-trivial) class of loops where termination can be decided efficiently has to be very restricted.

10.3.2 Hierarchical Expressions

We now elaborate on the closed forms arising from uniform loops which give rise to *hierarchical expressions* (Def. 10.3.11). To this end, we fix a uniform loop $(\varphi, \vec{u})$. Let $\text{spec}(A) = \{\lambda_1, \ldots, \lambda_k\}$ be A's eigenvalues where $0 \leq \lambda_1 < \ldots < \lambda_k$, let Q be A's Jordan normal form, and let T be the corresponding transformation matrix, i.e., $T^{-1} \cdot A \cdot T = Q$. Moreover, let η be the automorphism defined by $\eta(\vec{x}) = T \cdot \vec{x}$ and let $Tr_\eta(\varphi, A \cdot \vec{x}) = (\eta^{-1}(\varphi), Q \cdot \vec{x}) = (\varphi', Q \cdot \vec{x})$ as in Sect. 10.1. By uniformity, $Q = \text{diag}(B_{\lambda_1}, \ldots, B_{\lambda_k})$ consists of k Jordan blocks $B_{\lambda_1}, \ldots, B_{\lambda_k}$ where

$$B_\lambda = \begin{bmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda & 1 \\ 0 & 0 & \cdots & 0 & \lambda \end{bmatrix}.$$

For each eigenvalue λ, let $\nu(\lambda)$ be the dimension of B_λ. Since each eigenvalue has geometric multiplicity 1, this means that $\nu(\lambda)$ is the *algebraic multiplicity* of λ, i.e., the multiplicity as a root of the characteristic polynomial of A. Let the *index* $\mathrm{idx}(\lambda)$ be the sum of the algebraic multiplicities of all smaller eigenvalues than λ, i.e.,

$$\mathrm{idx}(\lambda) = \sum_{\substack{\lambda' \in \mathrm{spec}(A), \\ \lambda' < \lambda}} \nu(\lambda').$$

Let $B = B_\lambda$, $\nu = \nu(\lambda)$, and $\mathrm{idx} = \mathrm{idx}(\lambda)$ for some $\lambda \in \mathrm{spec}(A)$. As in [Kin+19; OPW15], we have

$$B^n = \begin{bmatrix} \binom{n}{0} \cdot \lambda^n & \binom{n}{1} \cdot \lambda^{n-1} & \binom{n}{2} \cdot \lambda^{n-2} & \cdots & \binom{n}{\nu-1} \cdot \lambda^{n-\nu+1} \\ 0 & \binom{n}{0} \cdot \lambda^n & \binom{n}{1} \cdot \lambda^{n-1} & \cdots & \binom{n}{\nu-2} \cdot \lambda^{n-\nu+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \binom{n}{1} \cdot \lambda^{n-1} \\ 0 & 0 & 0 & \cdots & \binom{n}{0} \cdot \lambda^n \end{bmatrix}$$

where $\binom{n}{\gamma} = 0$ if $n < \gamma$. This directly yields a closed form $\vec{q}$ for the n-fold application of the update $Q \cdot \vec{x}$ of the transformed loop.

Lemma 10.3.6. *For all $\lambda \in \mathrm{spec}(A)$ and $1 \leq r \leq \nu(\lambda)$ the $(\mathrm{idx}(\lambda) + r)$-th element of $\vec{q} = Q^n \cdot \vec{x}$ is given by*

$$\sum_{s=r}^{\nu(\lambda)} \binom{n}{s-r} \cdot \lambda^{n-s+r} \cdot x_{\mathrm{idx}(\lambda)+s}.$$

In general, the terms $\binom{n}{s-r}$ are not polynomials in n, e.g., $\binom{n}{2}$ is not a polynomial in n for $n \in \mathbb{N}$ as one would have to perform a case analysis to distinguish the case $\binom{n}{2} = 0$ for $n < 2$ from the case where $n \geq 2$. Here, n is a distinguished variable and $s - r \in \mathbb{N}$. Nevertheless, $\binom{n}{s-r}$ is a polynomial in n whenever $n \geq s - r$. Since our approach from Ch. 6 (resp. Sect. 9.3.1) uses the normalized closed form, we may assume that $n \geq s - r$ for all such occurring values of $s - r$. Under this assumption, $\binom{n}{s-r}$ is indeed a polynomial from $\mathbb{R}_{\mathbb{A}}[n]$ of degree $s - r$, i.e., there are coefficients $c_{s-r,m} \in \mathbb{R}_{\mathbb{A}}$ such that for all $n \geq s - r$ we have

$$\binom{n}{s-r} = \sum_{m=0}^{s-r} c_{s-r,m} \cdot n^m.$$

Let us elaborate on the coefficients $c_{s-r,m}$ first. It turns out that $c_{s-r,m} = \frac{\mathrm{stir}(s-r,m)}{(s-r)!}$ where stir denotes the *signed Stirling number of the first kind* (see [GKP94, Ch. 6]). While the formal definition of stir is not of interest for us, we use the fact $\mathrm{stir}(s - r, s - r) = 1 \neq 0$. We obtain the following corollary from Lemma 10.3.6.

Lemma 10.3.7. *For all* $\lambda \in \operatorname{spec}(A)$ *and* $1 \leq r \leq \nu(\lambda)$ *the* $(\operatorname{idx}(\lambda) + r)$*-th element of* $\vec{q}_{norm}$ *is given by*

$$\sum_{s=r}^{\nu(\lambda)} \left(\sum_{m=0}^{s-r} c_{s-r,m} \cdot x_{\operatorname{idx}(\lambda)+s} \cdot \lambda^{r-s} \cdot n^m \cdot \lambda^n \right) \in \mathbb{NPE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}].$$

Example 10.3.8. Consider the matrix in Jordan normal form

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

with the $k = 2$ eigenvalues $\lambda_1 = 1$ of algebraic multiplicity $\nu_1 = 2$ and $\lambda_2 = 2$ of algebraic multiplicity $\nu_2 = 3$, both of which have geometric multiplicity one. Moreover, $\operatorname{idx}(\lambda_1) = 0$ and $\operatorname{idx}(\lambda_2) = 0 + \nu_1 = 2$. Here, we have $d = 5$.

Regarding Lemma 10.3.7, the normalized closed form $\vec{q}_{norm}$ is given by

$$\vec{q}_{norm} = \begin{bmatrix} x_1 + n \cdot x_2 \\ x_2 \\ x_3 \cdot 2^n + \left(\frac{x_4}{2} - \frac{x_5}{8}\right) \cdot n \cdot 2^n + \frac{x_5}{8} \cdot n^2 \cdot 2^n \\ x_4 \cdot 2^n + \frac{x_5}{2} \cdot n \cdot 2^n \\ x_5 \cdot 2^n \end{bmatrix}$$

If $\lambda_1 = 0$, all entries $q_1, \cdots, q_{\operatorname{idx}(\lambda_2)-1}$ of $\vec{q}_{norm}$ vanish, i.e., these closed forms are "irrelevant" for the technique from Ch. 6: when inserting the closed form into the condition φ, the corresponding variables vanish. By replacing the variables $x_1, \ldots, x_{\operatorname{idx}(\lambda_2)-1}$ by zero, i.e., by deleting these variables (remember that φ is linear), we can assume that

0 is not an eigenvalue of A (Simplification 1)

Remark 10.3.9. Let us briefly discuss Simplification 1. Since we just ignore the variables which belong to the eigenvalue zero, the results of this section can also be generalized to matrices where all positive eigenvalues have geometric multiplicity 1. The only remaining eigenvalue 0 may have a geometric multiplicity > 1.

First, let us establish a shorthand notation for the set of linear polynomials.

Definition 10.3.10 (Linear Polynomials)**.** The set $\mathbb{R}_{\mathbb{A}}[\vec{x}]_{\text{lin}}$ is the set of *linear*

polynomials from $\mathbb{R}_{\mathbb{A}}[\vec{x}]$, i.e., the polynomials in $\mathbb{R}_{\mathbb{A}}[\vec{x}]$ of degree at most 1.

Let us elaborate on the closed form in Lemma 10.3.7. When re-arranging the order of summation we obtain

$$\begin{aligned}&\sum_{s=r}^{\nu(\lambda)}\left(\sum_{m=0}^{s-r} c_{s-r,m}\cdot x_{\mathrm{idx}(\lambda)+s}\cdot\lambda^{r-s}\cdot n^m\cdot\lambda^n\right)\\ =&\sum_{s=r}^{\nu(\lambda)}\underbrace{\left(\sum_{j=0}^{\nu(\lambda)-s} c_{s-r+j,s-r}\cdot x_{\mathrm{idx}(\lambda)+s+j}\cdot\lambda^{r-s-j}\right)}_{=\alpha_s}\cdot n^{s-r}\cdot\lambda^n.\end{aligned}\tag{10.3.1}$$

Here, α_s is a linear polynomial in the variables $x_{\mathrm{idx}(\lambda)+s},\dots,x_{\mathrm{idx}(\lambda)+\nu(\lambda)}$. Note that the coefficient of $x_{\mathrm{idx}(\lambda)+s}$ in α_s is $c_{s-r,s-r} = \frac{\mathrm{stir}(s-r,s-r)}{(s-r)!} = \frac{1}{(s-r)!} \neq 0$. Thus, the normalized poly-exponential expressions in $\vec{q}_{norm}$ have a special structure. We call them *hierarchical expressions*.

Definition 10.3.11 (Hierarchical Expression)**.** Let $h \in \mathbb{NPE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}]$. We call h a *hierarchical expression* over the indices $1 \leq i_1 < \dots < i_\nu \leq d$ if there exist $1 \leq r \leq \nu$ and $\lambda \in (\mathbb{R}_{\mathbb{A}})_{>0}$ such that

$$h = \sum_{s=r}^{\nu} \alpha_s(x_{i_s},\dots,x_{i_\nu})\cdot n^{s-r}\cdot\lambda^n,$$

where $\alpha_s(x_{i_s},\dots,x_{i_\nu}) \in \mathbb{R}_{\mathbb{A}}[x_{i_s},\dots,x_{i_\nu}]_{\mathrm{lin}}$, $\alpha_s(0,\dots,0) = 0$, and $\deg_{x_{i_s}}(\alpha_s) = 1$. We call $\mathrm{off}(h) = r$ the *offset*, ν the *order*, $\mathrm{base}(h) = \lambda$ the *base*, and $\mathrm{maxVar}(h) = x_{i_\nu}$ the *maximal variable* of h. The *dominant term* of h is $\mathrm{domTerm}(h) = \alpha_\nu(x_{i_\nu})\cdot n^{\nu-r}\cdot\lambda^n$. By $\vec{x}_h$ we denote the vector of variables $(x_{i_r},\dots,x_{i_\nu})$.

Note that in Def. 10.3.11, we implicitly require the underlying poly-exponential expression to only have *linear homogeneous* coefficients. Moreover, a hierarchical expression can have different offsets depending on the chosen indices. However, we later *fix* these indices and consider multiple hierarchical expressions over these indices (see Sect. 10.3.3).

Example 10.3.12. Reconsider Ex. 10.3.8. The poly-exponential expression

$$h = q_4 = x_4 \cdot 2^n + \frac{x_5}{2} \cdot n \cdot 2^n = x_{i_1} \cdot 2^n + \frac{x_{i_2}}{2} \cdot n \cdot 2^n$$

is a hierarchical expression over the indices $i_1 = 3$, $i_2 = 4$, and $i_3 = 5$ with offset 2, order 2, base 2, and maximal variable x_5. It is also a hierarchical expression over the indices $i_1 = 4$ and $i_2 = 5$ with offset 1.

In (10.3.1), we have motivated the definition of hierarchical expressions. Thus, we obtain the following corollary from (10.3.1).

Corollary 10.3.13. *For all $\lambda \in \operatorname{spec}(A)$ and $1 \leq r \leq \nu(\lambda)$ the $(\operatorname{idx}(\lambda) + r)$-th element of $\vec{q}_{norm}$ is a hierarchical expression over the indices $\operatorname{idx}(\lambda) + 1, \operatorname{idx}(\lambda) + 2, \dots, \operatorname{idx}(\lambda) + \nu$ with offset r, order $\nu(\lambda)$, base λ, and maximal variable $x_{\operatorname{idx}(\lambda)+\nu}$.*

Note that here we indeed need $\operatorname{stir}(s - r, s - r) = 1 \neq 0$, i.e., $\deg_{x_{\operatorname{idx}(\lambda)+s}}(\alpha_s) = 1$. Moreover, we fixed the indices. This will be used when introducing hierarchical partitions in Sect. 10.3.3 (see Cor. 10.3.21).

By Def. 10.3.11, we observe that for a hierarchical expression h with offset r, order ν, and base λ, the dominant term is in fact

$$\operatorname{domTerm}(h) = c_{\nu-r} \cdot x_{i_\nu} \cdot \lambda^\nu \cdot n^{\nu-r} \cdot \lambda^n,$$

for some $0 \neq c_{\nu-r} \in \mathbb{R}_{\mathbb{A}}$, i.e., the *coefficient* in the dominant term is $c_{\nu-r} \cdot x_{i_\nu}$. So when instantiating the variables $\vec{x}_h$ in h by $\vec{v} = (v_r, \dots, v_\nu) \in \mathbb{R}_{\mathbb{A}}^{\nu-r+1}$, by Def. 10.3.11 we have $\operatorname{domTerm}(h[\vec{x}_h/\vec{v}]) = c_{\nu-r} \cdot v_\nu \cdot n^{\nu-r} \cdot \lambda^n$ whenever x_{i_ν} is instantiated with some value $v_\nu \neq 0$. Here, the sign of $c_{\nu-r} \cdot v_\nu$ is determined uniquely by v_ν. However, if $v_\nu = 0$ and $v_{\nu-1} \neq 0$, by Def. 10.3.11 we have $\operatorname{domTerm}(h[\vec{x}_h/\vec{v}]) = c_{\nu-r-1} \cdot v_{\nu-1} \cdot n^{\nu-r-1} \cdot \lambda^n$ and again, the sign of $c_{\nu-r-1} \cdot v_{\nu-1}$ is determined uniquely by the sign of $v_{\nu-1}$. Thus, we obtain the following corollary.

Corollary 10.3.14. *Let $h = \sum_{s=r}^{\nu} \alpha_s(x_{i_s}, \dots, x_{i_\nu}) \cdot n^{s-r} \cdot \lambda^n$ be a hierarchical expression for the indices $i_1 < \cdots < i_\nu$ with offset r, order ν, and base λ. Let $\vec{v} = (v_r, \dots, v_\nu) \in \mathbb{R}_{\mathbb{A}}^{\nu-r+1}$. If for some $r \leq m \leq \nu$ we have $v_m \neq 0$ and $v_{m+1} = 0, \dots, v_\nu = 0$, the dominant term in $h[\vec{x}_h/\vec{v}]$ is given by $c_m \cdot v_m \cdot n^{m-r} \cdot \lambda^n$ for some $0 \neq c_m \in \mathbb{R}_{\mathbb{A}}$. If $\vec{v} = \vec{0}$, $h[\vec{x}_h/\vec{v}] = 0$.*

Proof. If $\vec{v} = \vec{0}$, $h[\vec{x}_h/\vec{v}] = 0$ since all coefficients are homogeneous. By Def. 10.3.11, we have

$$h[\vec{x}_h/\vec{v}] = \sum_{s=r}^{\nu} \alpha_s(v_s, \dots, v_\nu) \cdot n^{s-r} \cdot \lambda^n = \sum_{s=r}^{m} \alpha_s(v_s, \dots, v_\nu) \cdot n^{s-r} \cdot \lambda^n.$$

So $\mathrm{domTerm}(h[\vec{x}_h/\vec{v}]) = \alpha_m(v_m, 0, \dots, 0) \cdot n^{m-r} \cdot \lambda^n$. By Def. 10.3.11, we have $\alpha_m(v_m, 0, \dots, 0) = c_m \cdot v_m$ for some $0 \neq c_m \in \mathbb{R}_{\mathbb{A}}$. □

Example 10.3.15. Reconsider the setting from Ex. 10.3.12 for the indices $3, 4, 5$ and $\vec{v} = (v_2, v_3) \in \mathbb{R}^2_{\mathbb{A}}$. Here, the dominant term is given by $\mathrm{domTerm}(h[\vec{x}_h/\vec{v}]) = \frac{v_3}{2} \cdot n \cdot 2^n$, whenever $v_3 \neq 0$. So this is the case for $m = 3$. However, if $v_3 = 0$, i.e., $m = 2$, the dominant term is given by $\mathrm{domTerm}(h[\vec{x}_h/\vec{v}]) = v_2 \cdot 2^n$ if $v_2 \neq 0$.

Before we continue, let us first define the active variables of a linear polynomial.

Definition 10.3.16 (Active Variables)**.** Let $f(\vec{x}) = z_0 + \sum_{m=1}^{d} z_m \cdot x_m \in \mathbb{R}_{\mathbb{A}}[\vec{x}]_{\mathrm{lin}}$. We define $\mathrm{actVar}(f) = \{x_m \mid z_m \neq 0\}$. If $x_m \in \mathrm{actVar}(f)$, then $\mathrm{coeff}(f, x_m) = z_m$.

Example 10.3.17. Consider $f(\vec{x}) = -x_1 + 3 \cdot x_3 + 4 \in \mathbb{R}_{\mathbb{A}}[x_1, \dots, x_5]_{\mathrm{lin}}$. Then $\mathrm{actVar}(f) = \{x_1, x_3\}$, $\mathrm{coeff}(f, x_1) = -1$, and $\mathrm{coeff}(f, x_3) = 3$.

10.3.3 Hierarchical Partitions

Still, our goal remains to analyze termination of linear uniform loops by deciding eventual non-termination as in Ch. 6 (resp. Sect. 9.3.1). Thus, we have to be able to analyze poly-exponential expressions which result from replacing the variables $x_1, \dots, x_d$ by their normalized closed form in some linear polynomial. As we have already seen, these closed forms are all hierarchical expressions.

We now define so-called *hierarchical partitions*, which give us even more structure for proving Thm. 10.3.4. However, we first need another technical definition.

Definition 10.3.18 (Blocks)**.** Let $\nu_1, \dots, \nu_k \in \mathbb{N}$ form a *k-partition* of d, i.e., $\nu_1 + \cdots + \nu_k = d$ and all $\nu_i > 0$. The *blocks* associated to the partition $\nu_1, \dots, \nu_k$ are

$$B_1 = \{1, \dots, \nu_1\}, B_2 = \{\nu_1 + 1, \dots, \nu_1 + \nu_2\}, \dots, B_k = \{\nu_1 + \cdots + \nu_{k-1} + 1, \dots, d\}.$$

Definition 10.3.19 (Hierarchical Partition)**.** Hierarchical expressions $h_1, \dots, h_d$ are a *hierarchical k-partition* of d via $\nu_1, \dots, \nu_k$ if $\nu_1, \dots, \nu_k$ form a k-partition of d with associated blocks $B_1, \dots, B_k$ and $0 < \lambda_1 < \cdots < \lambda_k$ are algebraic numbers such that for all $1 \leq i \leq k$

(i) h_j is a hierarchical expression over the indices B_i for all $j \in B_i$,

(ii) base $\left(h_j\right) = \lambda_i$ for all $j \in B_i$,

(iii) h_j has order v_i for all $j \in B_i$,

(iv) $\mathrm{off}(h_{\min(B_i)}) = 1$,

(v) if $j, j+1 \in B_i$, then $\mathrm{off}(h_j) + 1 = \mathrm{off}(h_{j+1})$.

We call $\lambda_1 < \cdots < \lambda_k$ the bases of the partition.

So here we fix the indices for the hierarchical expressions h_j with $j \in B_i$ to be B_i.

Example 10.3.20. Reconsider Ex. 10.3.8. Then $\vec{h} = \vec{q}_{norm}$ is a hierarchical k-partition of 5 via $v_1 = 2$ and $v_2 = 3$, blocks $B_1 = \{1, 2\}$ and $B_2 = \{3, 4, 5\}$ and $\lambda_1 = 1 < 2 = \lambda_2$. Moreover, $\mathrm{off}(h_1) = 1$ and $\mathrm{off}(h_2) = 2$, while $\mathrm{off}(h_3) = 1$, $\mathrm{off}(h_4) = 2$, and $\mathrm{off}(h_5) = 3$. This is the reason why we considered the indices 3, 4, 5 in Ex. 10.3.12 and 10.3.15.

The last example already illustrates the connection of Lemma 10.3.7 and Def. 10.3.19.

Corollary 10.3.21. *Let A be the update matrix of a uniform loop with eigenvalues $0 < \lambda_1 < \cdots < \lambda_k$ and algebraic multiplicities $v_1, \ldots, v_k$, and Q its Jordan normal form. The normalized closed form $\vec{q}_{norm}$ of Q is a* hierarchical k-partition *of d via $v_1, \ldots, v_k$ and bases $\lambda_1 < \cdots < \lambda_k$. Here the i-th block is $B_i = \{\mathrm{idx}(\lambda_i) + 1, \ldots, \mathrm{idx}(\lambda_i) + v_i\}$.*

Now we use the just defined concept to extend Cor. 10.3.14 to poly-exponential expressions which are built from substituting a hierarchical partition into a linear polynomial. However, there is an obstacle: in the case that one of the bases of the hierarchical partition is 1, it interacts with the constant term in the polynomial.

Lemma 10.3.22. *Let $h_1, \ldots, h_d$ be a hierarchical k-partition of d via $v_1, \ldots, v_k$ with bases $0 < \lambda_1 < \cdots < \lambda_k$, $f \in \mathbb{R}_{\mathbb{A}}[\vec{x}]_{\mathrm{lin}}$, $p = f(h_1, \ldots, h_d) \in \mathbb{NPE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}]$, and $\vec{v} \in \mathbb{R}_{\mathbb{A}}^d$. For $1 \le i \le k$ let $I(i, f, \vec{h}) = \{j \in B_i \mid x_j \in \mathrm{actVar}(f)\}$ and $V(i, \vec{h}) = \{j \in B_i \mid v_j \neq 0\}$. Moreover, let $j_{\min}(i) = \min I(i, f, \vec{h})$ and $j_{\max}(i) = \max V(i, \vec{h})$, where we use $\max \emptyset = 0$ and $\min \emptyset = \omega$*[1].

[1] Here, ω denotes the "supremum" of $\mathbb{N}$, i.e., $\omega > n$ for all $n \in \mathbb{N}$.

(i) If $j_{\max}(i) < j_{\min}(i)$ *for all* $1 \leq i \leq k$*, then* $p[\vec{x}/\vec{v}] = \mathrm{domTerm}(p[\vec{x}/\vec{v}]) = f(0,\dots,0)$.

(ii) Otherwise let $i_{\max} = \max_{1\leq i\leq k}(j_{\max}(i) \geq j_{\min}(i))$*. We have three different cases:*

a) We have $f(0,\dots,0) \neq 0$ *and* $\lambda_{i_{\max}} < 1$*. Then*

$$\mathrm{domTerm}\left(p[\vec{x}/\vec{v}]\right) = f(0,\dots,0).$$

b) We have $\lambda_{i_{\max}} = 1$ *and* $j_{\min}(i_{\max}) = j_{\max}(i_{\max}) = j$*. Then* $\mathrm{domTerm}(h_j)[\vec{x}/\vec{v}] = c_j \cdot v_j$ *for some* $0 \neq c_j \in \mathbb{R}_{\mathbb{A}}$ *and*

$$\mathrm{domTerm}\left(p[\vec{x}/\vec{v}]\right) = \mathrm{coeff}\left(f, x_j\right) \cdot c_j \cdot v_j + f(0,\dots,0),$$

whenever the right-hand side is non-zero.

c) Otherwise $\mathrm{domTerm}(h_{j_{\min}(i_{\max})})[\vec{x}/\vec{v}] = c_{j_{\max}(i_{\max})} \cdot v_{j_{\max}(i_{\max})} \cdot n^a \cdot \lambda_{i_{\max}}^n$ *for some* $c_{j_{\max}(i_{\max})} \neq 0$ *and*

$$\mathrm{domTerm}\left(p[\vec{x}/\vec{v}]\right) = \mathrm{coeff}\left(f, x_{j_{\min}(i_{\max})}\right) \cdot c_{j_{\max}(i_{\max})} \cdot v_{j_{\max}(i_{\max})} \cdot n^a \cdot \lambda_{i_{\max}}^n.$$

Here, $a = j_{\max}(i_{\max}) - j_{\min}(i_{\max})$*. Thus, the sign of* $\mathrm{domTerm}\left(p[\vec{x}/\vec{v}]\right)$ *is uniquely determined by the sign of* $v_{j_{\max}(i_{\max})}$.

Proof. We prove this lemma by a case analysis.

(i) This is a direct consequence of Cor. 10.3.14: if x_s occurs in f, all variables occurring in h_s are instantiated with zero by $\vec{v}$. Hence, $p[\vec{x}/\vec{v}]$ only consists of the constant term of f, i.e., $f(0,\dots,0)$.

(ii) We prove this case by a case analysis, too.

a) In this case, the greatest base of a poly-exponential expression which occurs in $p[\vec{x}/\vec{v}]$ is 1 and its coefficient is $f(0,\dots,0) \neq 0$.

b) In this case, the greatest base of a poly-exponential expression which occurs in $p[\vec{x}/\vec{v}]$ is 1, too. The only variable that occurs with a non-zero coefficient in f and in h_j and is assigned a non-zero value by $\vec{v}$ is x_j, i.e., $\mathrm{domTerm}(h_j[\vec{x}/\vec{v}]) = c_j \cdot v_j$ for some $0 \neq c_j \in \mathbb{R}_{\mathbb{A}}$ (see Cor. 10.3.14). This means that $\mathrm{domTerm}\left(p[\vec{x}/\vec{v}]\right) = \mathrm{coeff}\left(f, x_j\right) \cdot c_j \cdot v_j + f(0,\dots,0)$, whenever the right-hand side is non-zero. This proves the claim.

c) Let $i > i_{\max}$ and $j \in B_i$ such that $x_j \in \operatorname{actVar}(f)$. Since $j_{\min}(i) > j_{\max}(i)$, we have $h_j[\vec{x}/\vec{v}] = 0$ by Cor. 10.3.14. Thus, by linearity of f and Def. 10.3.19 we have

$$\operatorname{domTerm}(p[\vec{x}/\vec{v}]) = \operatorname{coeff}(x_{j_{\min}}, f) \cdot \operatorname{domTerm}(h_{j_{\min}}[\vec{x}/\vec{v}]).$$

By Cor. 10.3.14, we again have

$$\operatorname{domTerm}(h_{j_{\min}}[\vec{x}/\vec{v}]) = c_{j_{\max}(i_{\max})} \cdot v_{j_{\max}(i_{\max})} \cdot n^a \cdot \lambda_{i_{\max}}^n,$$

for some $c_{j_{\max}(i_{\max})} \neq 0$. By Def. 10.3.19, the offset of $h_{j_{\min}(i_{\max})}$ is r_1, if $j_{\min}(i_{\max})$ is the r_1-th element when sorting the elements of $B_{i_{\max}}$ w.r.t. the standard ordering on the natural numbers. Let $r_2 \geq r_1$ be the position of $j_{\max}(i_{\max})$ in this ordering. In the setting of Cor. 10.3.14, we then have $m = r_2$. Thus, by Cor. 10.3.14 we have

$$a = r_2 - r_1 = j_{\max}(i_{\max}) - j_{\min}(i_{\max}).$$

□

Example 10.3.23. Let us continue Ex. 10.3.15 with f from Ex. 10.3.17 and $\vec{h} = \vec{q}_{norm}$. We then have

$$p = f(h_1, \dots, h_5) = (-x_1 + 4) - x_2 \cdot n + 3 \cdot x_3 \cdot 2^n + \left(\tfrac{3 \cdot x_4}{2} - \tfrac{3 \cdot x_5}{8}\right) \cdot n \cdot 2^n + \tfrac{3 \cdot x_5}{8} \cdot n^2 \cdot 2^n.$$

In the notation of Lemma 10.3.22, we have $I(1, f, \vec{h}) = \{1\}$ and $I(2, f, \vec{h}) = \{3\}$.

Consider $\vec{v} = (0, 1, -1, 0, 1)$. We then have

$$p[\vec{x}/\vec{v}] = 4 - 1 \cdot n - 3 \cdot 2^n - \tfrac{3}{8} \cdot n \cdot 2^n + \tfrac{3}{8} \cdot n^2 \cdot 2^n.$$

Thus, $V(1, \vec{h}) = \{2\}$, and $V(2, \vec{h}) = \{3, 5\}$ i.e., $i_{\max} = 2$, $j_{\min}(2) = 3$, and $j_{\max}(2) = 5$. Indeed, we have

$$\operatorname{domTerm}(p[\vec{x}/\vec{v}]) = \tfrac{3}{8} \cdot n^2 \cdot 2^n = \operatorname{coeff}(f, x_3) \cdot \operatorname{domTerm}(h_3[\vec{x}/\vec{v}]),$$

with $\operatorname{domTerm}(h_3[\vec{x}/\vec{v}]) = \frac{1}{8} \cdot n^2 \cdot 2^n = \frac{1}{8} \cdot v_5 \cdot n^2 \cdot 2^n$. So, the sign of $\operatorname{domTerm}(p[\vec{x}/\vec{v}])$ is uniquely determined by v_5. Moreover, $a = 2 = 5 - 3 = j_{\max}(2) - j_{\min}(2)$.

Now consider $\vec{v}' = (1, 0, 0, 0, 0)$. We then have

$$p[\vec{x}/\vec{v'}] = 3.$$

In this case, $V(1, \vec{h}) = \{1\}$ and $V(2, \vec{h}) = \emptyset$, i.e., $i_{\max} = 1$ and $j_{\max}(1) = j_{\min}(1) = 1$. Furthermore, $\text{domTerm}(h_1)[\vec{x}/\vec{v}] = 1 = 1 \cdot v_1$, i.e., $c_1 = 1$. Since $\lambda_1 = 1$, we have

$$\text{domTerm}(p[\vec{x}/\vec{v'}]) = 3 = (-1 + 4) = \text{coeff}(f, x_1) \cdot c_1 \cdot v_1 + f(0, \dots, 0).$$

10.3.4 Sign Conditions

We now prove that in the setting of a hierarchical k-partition the literals in $\text{red}(p \triangleright 0)$ (see Lemma 6.1.7), where p is one of the poly-exponential expressions obtained from substituting the variables by their closed forms in the polynomials of the condition (see the procedure presented in Ch. 6 and Sect. 9.3.1, respectively), are equivalent to so-called *sign-conditions*, i.e., they require a certain variable to have a certain sign and force other variables to be zero. Unfortunately, we also have a special case arising from the constant term. Here, we have to deal with so-called *interval conditions*, where one variable is forced to be contained in a certain (possibly unbounded) interval and other variables are forced to be zero, and *quasi sign conditions* where one variable is required to have a certain non-zero value.

We show that $\text{red}(p \triangleright 0)$ is equivalent to a combination of these special formulas. This then enables us to check validity of $\exists \vec{x} \in S^d.\ \text{red}(p \triangleright 0)$ in a simpler way than with the standard techniques for deciding satisfiability of linear arithmetic formulas.

Definition 10.3.24 ((Quasi) Sign Conditions, Interval Conditions)**.** Let $1 \leq i \leq d$, $I \subseteq \{1, \dots, d\} \setminus \{i\}$, $\text{sign} \in \{-1, 1\}$, and $0 \neq c \in \mathbb{R}_{\mathbb{A}}$.

(i) The formula

$$\psi = \text{sign} \cdot x_i > 0 \wedge \bigwedge_{j \in I} (x_j = 0)$$

is called a *sign condition*. We call the formula $\psi \wedge x_{i'} = c$ a *quasi* sign condition if $i \neq i' \in \{1, \dots, d\} \setminus I$.

(ii) The formula

$$\text{sign} \cdot x_i + c > 0 \wedge \bigwedge_{j \in I} (x_j = 0)$$

is called an *interval condition*.

Example 10.3.25. The formula

$$x_3 > 0 \wedge x_4 = 0 \wedge x_5 = 0$$

is a sign condition since it requires $\text{sign}(x_3) = 1$. For example, the formula

$$-x_1 + 4 > 0 \wedge x_2 = 0 \wedge x_3 = 0 \wedge x_4 = 0 \wedge x_5 = 0$$

is an interval condition since it requires x_1 to be contained in the unbounded open interval $(-\infty, 4)$.

To ease notation in the following reasoning, we define formulas which are true iff all coefficients of terms with basis λ_i vanish by requiring that the variables related to the i-th block vanish. Again, we also have the special case that a basis is 1.

Definition 10.3.26 (Formulas for Vanishing of Blocks)**.** Let $f \in \mathbb{R}_{\mathbb{A}}[\vec{x}]_{\text{lin}}$, $\vec{h} = (h_1, \dots, h_d)$ a hierarchical k-partition via $v_1, \dots, v_k$ with bases $\lambda_1 < \cdots < \lambda_k$ and let $1 \le i \le k$. Using $I(i, f, \vec{h}) = \{j \in B_i \mid x_j \in \text{actVar}(f)\}$ from Lemma 10.3.22 and $j_{\min} = \min\left(I(1, f, \vec{h})\right)$ we define $M = \frac{-f(0,\dots,0)}{\text{coeff}(f, x_{j_{\min}}) \cdot c_{j_{\min}}}$ if $I(1, f, \vec{h}) \neq \emptyset$, where $c_{j_{\min}}$ is the unique coefficient of $x_{j_{\min}}$ in $h_{j_{\min}}$. We further define

$$\text{blockZero}(i, f, \vec{h}) = \begin{cases} f(0, \dots, 0) \neq 0, & \lambda_i = 1, I(i, f, \vec{h}) = \emptyset \\ \text{true}, & \lambda_i \neq 1, I(i, f, \vec{h}) = \emptyset \\ \bigwedge_{s \in B_i,\ s > j_{\min}} (x_s = 0) \wedge x_{j_{\min}} = M, & \lambda_i = 1, I(i, f, \vec{h}) \neq \emptyset \\ \bigwedge_{s \in B_i,\ s \ge j_{\min}} (x_s = 0), & \lambda_i \neq 1, I(i, f, \vec{h}) \neq \emptyset. \end{cases}$$

Note that by Def. 10.3.11 and 10.3.19, $x_{j_{\min}}$ occurs only in a unique (linear) coefficient of $h_{j_{\min}}$ and its coefficient in this linear polynomial is non-zero.

By Lemma 10.3.22, we obtain the following corollary.

Corollary 10.3.27. *Let $f \in \mathbb{R}_{\mathbb{A}}[\vec{x}]_{\text{lin}}$, $\vec{h} = (h_1, \dots, h_d)$ a hierarchical k-partition via $v_1, \dots, v_k$ with bases $\lambda_1 < \cdots < \lambda_k$ and let $1 \le i \le k$. Then $\vec{v} \in \mathbb{R}_{\mathbb{A}}^d$ satisfies* $\text{blockZero}(i, f, \vec{h})$ *iff all terms with basis λ_i vanish in $f(h_1, \dots, h_d)[\vec{x}/\vec{v}]$.*

Example 10.3.28. Let us continue Ex. 10.3.23, where $I(1,f,\vec{h}) = \{1\}$ and $I(2,f,\vec{h}) = \{3\}$, i.e., $j_{\min} = 1$, $c_{j_{\min}} = 1$, $\operatorname{coeff}(f, x_{j_{\min}}) = -1$ and thus $M = \frac{-4}{-1} = 4$. Thus,

$$\operatorname{blockZero}(1,f,\vec{h}) = (x_1 = 4) \wedge (x_2 = 0)$$

and

$$\operatorname{blockZero}(2,f,\vec{h}) = (x_3 = 0) \wedge (x_4 = 0) \wedge (x_5 = 0).$$

So $\vec{v} \in \mathbb{R}_{\mathbb{A}}^5$ satisfies $\operatorname{blockZero}(1,f,\vec{h})$ iff all terms with basis 1 vanish in $f(h_1,\dots,h_5)[\vec{x}/\vec{v}]$, and $\vec{v} \in \mathbb{R}_{\mathbb{A}}^5$ satisfies $\operatorname{blockZero}(2,f,\vec{h})$ iff all terms with basis 2 vanish in $f(h_1,\dots,h_5)[\vec{x}/\vec{v}]$.

For some hierarchical k-partition $\vec{h}\,'$ via $v_1 = 2$ and $v_2 = 3$ of 5 where none of the bases is 1 and $I(1,f,\vec{h}\,') = \{1\}$ we have

$$\operatorname{blockZero}(1,f,\vec{h}\,') = (x_1 = 0) \wedge (x_2 = 0).$$

If we consider $f' = -7 \cdot x_5 + 1$ instead, we obtain $I(1,f',\vec{h}) = \emptyset$, and $I(2,f',\vec{h}) = \{5\}$, i.e.,

$$\operatorname{blockZero}(1,f',\vec{h}) = (1 \neq 0) = \text{false}$$

and

$$\operatorname{blockZero}(2,f',\vec{h}) = (x_5 = 0).$$

Again, $\vec{v} \in \mathbb{R}_{\mathbb{A}}^5$ satisfies $\operatorname{blockZero}(2,f',\vec{h})$ iff all terms with basis 2 vanish in $f'(h_1,\dots,h_5)[\vec{x}/\vec{v}]$.

After simplifying the notation, we can now prove our most important result which is based on Def. 6.1.5, Lemma 6.1.7, and Thm. 6.1.9.

Lemma 10.3.29. *Let $h_1,\dots,h_d$ be a hierarchical k-partition of d via $v_1,\dots,v_k$ with bases $0 < \lambda_1 < \cdots < \lambda_k$, $0 \neq f \in \mathbb{R}_{\mathbb{A}}[\vec{x}]_{\text{lin}}$, $p = f(h_1,\dots,h_d) \in \mathbb{NPE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}]$, and $\operatorname{coefs}(p) = \{\alpha_1,\dots,\alpha_\ell\}$ (see Def. 6.1.5). Fix $1 \le m \le \ell$.*

(i) If $\alpha_m = \alpha_m^{(0,1)}$, $\lambda_{i_0} = 1$ for some $1 \le i_0 \le k$, and $I(i_0,f,\vec{h}) \neq \emptyset$, let $j_{\min} = \min I(i_0,f,\vec{h})$. Then there is a $C \in \mathbb{R}_{\mathbb{A}}$ such that

$$\alpha_m > 0 \wedge \bigwedge_{s=m+1}^{\ell} (\alpha_s = 0)$$

$$\iff \operatorname{sign}(C) \cdot x_{j_{\min}} + \frac{f(0,\dots,0)}{|C|} > 0 \wedge \bigwedge_{\substack{j \in B_{i_0}, \\ j_{\min} < j}} (x_j = 0) \wedge \bigwedge_{i=i_0+1}^{k} \operatorname{blockZero}(i_0, f, \vec{h}).$$

(ii) If $\alpha_m = \alpha_m^{(0,1)}$, $f(0,\dots,0) \neq 0$, and either $\lambda_i \neq 1$ for all i, or $\lambda_{i_0} = 1$ and $I(i_0, f, \vec{h}) = \emptyset$ for some $1 \leq i_0 \leq k$, then

$$\alpha_m > 0 \wedge \bigwedge_{s=m+1}^{\ell} (\alpha_s = 0) \quad \iff \quad f(0,\dots,0) > 0 \wedge \bigwedge_{\substack{i=1, \\ \lambda_i > 1}}^{k} \operatorname{blockZero}(i, f, \vec{h}).$$

(iii) If $\alpha_m = \alpha_m^{(a,\lambda)}$ for some $\lambda < 1$, $f(0,\dots,0) \neq 0$, and $\lambda_i \neq 1$ for all i, then

$$\alpha_m > 0 \wedge \bigwedge_{s=m+1}^{\ell} (\alpha_s = 0) \quad \iff \quad \text{false}$$

(iv) Otherwise there exist unique indices $1 \leq i_{\max} \leq k$ and $j_{\max} \in B_{i_{\max}}$ such that

$$\begin{aligned} &\alpha_m > 0 \wedge \bigwedge_{s=m+1}^{\ell} (\alpha_s = 0) \\ \iff\ &\operatorname{sign} \cdot x_{j_{\max}} > 0 \wedge \bigwedge_{\substack{s \in B_{i_{\max}}, \\ s > j_{\max}}} (x_s = 0) \wedge \bigwedge_{i=i_{\max}+1}^{k} \operatorname{blockZero}(i, f, \vec{h}), \end{aligned}$$

for some $\operatorname{sign} \in \{-1, 1\}$.

We denote the formulas presented here that are equivalent to $\alpha_m > 0 \wedge \bigwedge_{s=m+1}^{\ell} (\alpha_s = 0)$ by $\rho_{f,m}$.

Proof. Let $\vec{v} \in \mathbb{R}_{\mathrm{A}}^d$. Then $\alpha_m[\vec{x}/\vec{v}]$ is the coefficient of the dominant term of $p[\vec{x}/\vec{v}]$ if and only if $\alpha_m[\vec{x}/\vec{v}] \neq 0$ and $\bigwedge_{s=m+1}^{\ell} \left(\alpha_s[\vec{x}/\vec{v}] = 0\right)$.

(i) In this case, $\vec{v}$ makes $\alpha_m[\vec{x}/\vec{v}]$ the coefficient of the dominant term of $p[\vec{x}/\vec{v}]$ if and only if in addition to $\alpha_m[\vec{x}/\vec{v}] \neq 0$ we have

$$\bigwedge_{\substack{j \in B_{i_0}, \\ j_{\min} < j}} (v_j = 0) \wedge \bigwedge_{i=i_0+1}^{k} \operatorname{blockZero}(i, f, \vec{h})[\vec{x}/\vec{v}]$$

by Lemma 10.3.22 and Cor. 10.3.27. Again, by Lemma 10.3.22 we then have $\alpha_m[\vec{x}/\vec{v}] = \text{coeff}(f, x_{j_{\min}}) \cdot c_{j_{\min}} \cdot v_{j_{\min}} + f(0, \dots, 0)$, where $\text{domTerm}(h_{j_{\min}}[\vec{x}/\vec{v}]) = c_{j_{\min}} \cdot v_{j_{\min}}$. So, in this case $\alpha_m > 0 \wedge \bigwedge_{s=m+1}^{\ell} (\alpha_s = 0)$ is equivalent to

$$\text{coeff}(f, x_{j_{\min}}) \cdot c_{j_{\min}} \cdot x_{j_{\min}} + f(0, \dots, 0) > 0 \wedge \bigwedge_{\substack{j \in B_{i_0}, \\ j_{\min} < j}} (x_j = 0) \wedge \bigwedge_{i=i_0+1}^{k} \text{blockZero}(i, f, \vec{h}),$$

which directly implies the desired result when dividing by the positive value $\left|c_{j_{\min}} \cdot \text{coeff}(f, x_{j_{\min}})\right|$ and abbreviating $C = c_{j_{\min}} \cdot \text{coeff}(f, x_{j_{\min}})$.

(ii) By Lemma 10.3.22 and Cor. 10.3.27, $\vec{v} \in \mathbb{R}_{\mathbb{A}}^d$ makes $\alpha_m[\vec{x}/\vec{v}]$ the coefficient of the dominant term of $p[\vec{x}/\vec{v}]$ if and only if $\vec{v}$ satisfies

$$\bigwedge_{\substack{i=1, \\ \lambda_i > 1}}^{k} \text{blockZero}(i, f, \vec{h}).$$

Moreover, whenever $\vec{v} \in \mathbb{R}_{\mathbb{A}}^d$ makes $\alpha_m[\vec{x}/\vec{v}]$ the coefficient of the dominant term of $p[\vec{x}/\vec{v}]$, we have $\alpha_m[\vec{x}/\vec{v}] = f(0, \dots, 0)$ by Lemma 10.3.22 (since for these $\vec{v}$, $\lambda_{i_{\max}} < 1$ or $p[\vec{x}/\vec{v}] = f(0, \dots, 0)$). This proves the claim.

(iii) In this case, the coefficient $\alpha^{(0,1)}[\vec{x}/\vec{v}]$ of $p[\vec{x}/\vec{v}]$ is the constant non-zero value $f(0, \dots, 0)$ for any $\vec{v} \in \mathbb{R}_{\mathbb{A}}^d$. Thus, $\alpha_m[\vec{x}/\vec{v}]$ cannot become the leading term of $p[\vec{x}/\vec{v}]$ for any $\vec{v} \in \mathbb{R}_{\mathbb{A}}^d$.

(iv) Let $\vec{v} \in \mathbb{R}_{\mathbb{A}}^d$ such that the leading term of $p[\vec{x}/\vec{v}]$ has the coefficient $\alpha_m[\vec{x}/\vec{v}]$. When applying Lemma 10.3.22 to p and $\vec{v}$, this means that there exist $1 \leq i_{\max} \leq k$ and $j_{\min} \leq j_{\max} \in B_{i_{\max}}$ such that

$$\begin{aligned} & \text{domTerm}(p[\vec{x}/\vec{v}]) \\ = \; & \text{coeff}(f, x_{j_{\min}}) \cdot \text{domTerm}(h_{j_{\min}}[\vec{x}/\vec{v}]) \\ = \; & \alpha_m[\vec{x}/\vec{v}] \cdot n^{j_{\max} - j_{\min}} \cdot \lambda_{i_{\max}}^n \end{aligned}$$

and the sign of $\alpha_m[\vec{x}/\vec{v}]$ is uniquely determined by the sign of $v_{j_{\max}}$. We observe that $i_{\max}$, $j_{\min}$, and $j_{\max}$ are uniquely determined by α_m due to Lemma 10.3.22.

Now, $\vec{v}$ makes $\alpha_m[\vec{x}/\vec{v}]$ the coefficient of the leading term of $p[\vec{x}/\vec{v}]$, i.e., $\vec{v}$ gives exactly rise to the above used values for $i_{\max}$, $j_{\min}$, and $j_{\max}$ if and only if the following conditions hold:

a) If $i > i_{\max}$ and $x_j \in \mathrm{actVar}(f)$ for some $j \in B_i$, then $v_{j'} = 0$ for all $j \leq j' \in B_i$, i.e., $\vec{v}$ satisfies $\mathrm{blockZero}(i, f, \vec{h})$ for all $i > i_{\max}$.

b) If $j_{\max} < j \in B_{i_{\max}}$, then $v_j = 0$.

Combining these conditions with the requirement $\alpha_m[\vec{x}/\vec{v}] > 0$ yields

$$\mathrm{sign} \cdot x_{j_{\max}} > 0 \wedge \bigwedge_{\substack{s \in B_{i_{\max}}, \\ s > j_{\max}}} (x_s = 0) \wedge \bigwedge_{i=i_{\max}+1}^{k} \mathrm{blockZero}(i, f, \vec{h})$$

for some $\mathrm{sign} \in \{1, -1\}$ which is uniquely determined by α_m. On the other hand, we have seen that any $\vec{v} \in \mathbb{R}_{\mathrm{A}}^d$ which satisfies this formula ensures that $\mathrm{domTerm}(p[\vec{x}/\vec{v}]) = \alpha_m[\vec{x}/\vec{v}] \cdot n^{j_{\max} - j_{\min}} \cdot \lambda_{i_{\max}}^n$, and in addition $\alpha_m[\vec{x}/\vec{v}] > 0$. This proves the statement.

□

Note that in all of the cases the equivalent formula can be computed in polynomial time from $h_1, \ldots, h_d$ and f.

Example 10.3.30. Reconsider Ex. 10.3.23, where

$$p = f(h_1, \ldots, h_5) = (-x_1 + 4) - x_2 \cdot n + 3 \cdot x_3 \cdot 2^n + \left(\tfrac{3 \cdot x_4}{2} - \tfrac{3 \cdot x_5}{8}\right) \cdot n \cdot 2^n + \tfrac{3 \cdot x_5}{8} \cdot n^2 \cdot 2^n.$$

Here, $\mathrm{coefs}(p)$ is $\{\alpha_1^{(1,0)}, \alpha_2^{(1,1)}, \alpha_3^{(2,0)}, \alpha_4^{(2,1)}, \alpha_5^{(2,2)}\}$ where

$$\alpha_1 = -x_1 + 4, \quad \alpha_2 = -x_2, \quad \alpha_3 = 3 \cdot x_3, \quad \alpha_4 = \tfrac{3 \cdot x_4}{2} - \tfrac{3 \cdot x_5}{8}, \quad \alpha_5 = \tfrac{3 \cdot x_5}{8}.$$

In terms of Lemma 10.3.29, we then have for $\alpha_4 = \alpha_4^{(2,1)}$

$$\alpha_4 > 0 \wedge \alpha_5 = 0 \quad \Longleftrightarrow \quad x_4 > 0 \wedge x_5 = 0.$$

Here, $i_{\max} = 2$, $j_{\min} = \min I(2, f, \vec{h}) = 3$ and thus $j_{\max} = 3 + 1 = 4$. Since $\lambda_1 = 1$ and $I(1, f, \vec{h}) = \{1\} \neq \emptyset$, we have by Lemma 10.3.29 with $j_{\min} = \min I(1, f, \vec{h}) = 1$

$$\alpha_1 > 0 \wedge \bigwedge_{s=2}^{5} (\alpha_s = 0) \quad \Longleftrightarrow \quad -x_1 + 4 > 0 \wedge x_2 = 0 \wedge \mathrm{blockZero}(2, f, \vec{h}),$$

where $\text{blockZero}(2, f, \vec{h}) = (x_3 = 0) \wedge (x_4 = 0) \wedge (x_5 = 0)$ by Ex. 10.3.28. Note that here $C = -1$ and $f(0, \dots, 0) = 4$.

So far, we have not dealt with the case that for some p, all coefficients vanish. This can be directly obtained from Lemma 10.3.29 and Cor. 10.3.27.

Corollary 10.3.31. *Let $h_1, \dots, h_d$ be a hierarchical k-partition of d via $v_1, \dots, v_k$ with bases $\lambda_1 < \dots < \lambda_k$, $0 \neq f \in \mathbb{R}_{\mathbb{A}}[\vec{x}]_{\text{lin}}$, $p = f(h_1, \dots, h_d) \in \mathbb{NPE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}]$, and $\text{coefs}(p) = \{\alpha_1, \dots, \alpha_\ell\}$.*

(i) If $f(0, \dots, 0) \neq 0$, and either $\lambda_i \neq 1$ for all $1 \leq i \leq k$, or $\lambda_{i_0} = 1$ and $I(i_0, f, \vec{h}) = \emptyset$ for some $1 \leq i_0 \leq k$, then

$$\bigwedge_{s=1}^{\ell} (\alpha_s = 0) \iff \text{false}.$$

(ii) Otherwise $\bigwedge_{s=1}^{\ell} (\alpha_s = 0)$ is equivalent to the formula

$$\bigwedge_{i=1}^{k} \text{blockZero}(i, f, \vec{h}).$$

We denote the formula presented here that is equivalent to $\bigwedge_{s=1}^{\ell} (\alpha_s = 0)$ by $\rho_{f,0}$.

Here, we indeed need to distinguish two cases since Cor. 10.3.27 does not apply to the case where $\lambda_i \neq 1$ for all $1 \leq i \leq k$.

Example 10.3.32. Reconsider Ex. 10.3.30. Here, $\lambda_1 = 1$, i.e., $i_0 = 1$, and $I(1, f, \vec{h}) \neq \emptyset$. By Cor. 10.3.31,

$$\bigwedge_{s=1}^{5} (\alpha_s = 0) \iff \text{blockZero}(1, f, \vec{h}) \wedge \text{blockZero}(2, f, \vec{h}),$$

where $\text{blockZero}(1, f, \vec{h}) = (x_1 = 4) \wedge (x_2 = 0)$ and $\text{blockZero}(2, f, \vec{h}) = (x_3 = 0) \wedge (x_4 = 0) \wedge (x_5 = 0)$ by Ex. 10.3.28.

We can now combine Lemma 10.3.29 and Cor. 10.3.31 with Thm. 6.1.9 to obtain the following result.

Corollary 10.3.33. *Let $h_1, \dots, h_d$ be a hierarchical k-partition of d via $v_1, \dots, v_k$ with bases $0 < \lambda_1 < \cdots < \lambda_k$, $0 \neq f \in \mathbb{R}_{\mathbb{A}}[\vec{x}]_{\text{lin}}$, $p = f(h_1, \dots, h_d) \in \mathbb{NPE}_{\mathbb{R}_{\mathbb{A}}}[\vec{x}]$, and $|\text{coefs}(p)| = \ell$. Moreover, let $\rho_{f,0}, \rho_{f,1}, \dots, \rho_{f,\ell}$ be the formulas from Lemma 10.3.29 and Cor. 10.3.31. Then*

$$\text{red}(p > 0) \text{ is equivalent to } \text{sic}(p > 0) = \bigvee_{m=1}^{\ell} \rho_{f,m}$$
$$\text{and} \quad \text{red}(p \geq 0) \text{ is equivalent to } \text{sic}(p \geq 0) = \text{sic}(p > 0) \vee \rho_{f,0},$$

where sic *stands for sign and interval conditions.*

Example 10.3.34. Reconsider Ex. 10.3.30. By Cor. 10.3.33, $\text{red}(p > 0)$ is equivalent to

$$\begin{aligned} &\text{sic}(p > 0) \\ = \ &\left(-x_1 + 4 > 0 \wedge x_2 = 0 \wedge \text{blockZero}(2, f, \vec{h})\right) \vee \left(-x_2 > 0 \wedge \text{blockZero}(2, f, \vec{h})\right) \\ &\vee (x_3 > 0 \wedge x_4 = 0 \wedge x_5 = 0) \vee (x_4 > 0 \wedge x_5 = 0) \vee x_5 > 0. \end{aligned}$$

By Cor. 10.3.33 and Ex. 10.3.32, $\text{red}(p \geq 0)$ is equivalent to

$$\text{sic}(p \geq 0) = \text{sic}(p > 0) \vee \left(\text{blockZero}(1, f, \vec{h}) \wedge \text{blockZero}(2, f, \vec{h})\right).$$

It is straightforward to give a satisfying assignment for a (quasi) sign condition: x_i is set to some non-zero value with the required sign and x_j is set to 0 (or the required constant) for all $j \in I$. The same holds for interval conditions: x_i is set to an element in the interval and x_j is set to 0 for all $j \in I$. However, there are no restrictions on the other variables, so there are of course infinitely many satisfying assignments for sign conditions. Nevertheless, in our special situation we show that we only need to consider some kind of "abstract" assignments: In such an assignment, we can not only assign a variable a special value but we can also assign variables the abstract value $\star$ which stands for "some non-zero value", i.e., all these variables are assigned the same value which is not yet specified. However, later on we only deal with abstract assignments where only *one* variable is assigned the abstract value $\star$.

Definition 10.3.35 (Evaluation)**.** Let ρ be a Boolean combination of formulas of the form $\text{sign} \cdot x_j > 0$, $\text{sign} \cdot x_j + c > 0$, and $\text{sign} \cdot x_j = c$ in the variables $x_1, \dots, x_d$

where $c \in \mathbb{R}_A$, and let $\vec{v} \in \{0, 1, -1, \star\}^d$. We define the *evaluation of ρ w.r.t. $\vec{v}$* (written $\rho[\vec{x}/\vec{v}]\downarrow$) inductively as follows.

$$(\rho_1 \wedge \rho_2)[\vec{x}/\vec{v}]\downarrow = \begin{cases} \text{false} & \text{if } \rho_1[\vec{x}/\vec{v}]\downarrow = \text{false} \\ & \text{or } \rho_2[\vec{x}/\vec{v}]\downarrow = \text{false} \\ \rho_2[\vec{x}/\vec{v}]\downarrow & \text{if } \rho_1[\vec{x}/\vec{v}]\downarrow = \text{true} \\ \rho_1[\vec{x}/\vec{v}]\downarrow & \text{if } \rho_2[\vec{x}/\vec{v}]\downarrow = \text{true} \\ \rho_1[\vec{x}/\vec{v}]\downarrow \wedge \rho_2[\vec{x}/\vec{v}]\downarrow & \text{otherwise} \end{cases}$$

$$(\rho_1 \vee \rho_2)[\vec{x}/\vec{v}]\downarrow = \begin{cases} \text{true} & \text{if } \rho_1[\vec{x}/\vec{v}]\downarrow = \text{true} \\ & \text{or } \rho_2[\vec{x}/\vec{v}]\downarrow = \text{true} \\ \rho_2[\vec{x}/\vec{v}]\downarrow & \text{if } \rho_1[\vec{x}/\vec{v}]\downarrow = \text{false} \\ \rho_1[\vec{x}/\vec{v}]\downarrow & \text{if } \rho_2[\vec{x}/\vec{v}]\downarrow = \text{false} \\ \rho_1[\vec{x}/\vec{v}]\downarrow \vee \rho_2[\vec{x}/\vec{v}]\downarrow & \text{otherwise} \end{cases}$$

$$(x_j = 0)[\vec{x}/\vec{v}]\downarrow = \begin{cases} \text{true} & \text{if } v_j = 0 \text{ holds} \\ \text{false} & \text{otherwise} \end{cases}$$

$$(\text{sign} \cdot x_j + c > 0)[\vec{x}/\vec{v}]\downarrow = \text{sign} \cdot v_j + c > 0$$

$$(\text{sign} \cdot x_j = c)[\vec{x}/\vec{v}]\downarrow = (\text{sign} \cdot v_j = c)$$

Note that in the last two cases, we evaluate the formula if $v_j \in \{0, 1, -1\}$ directly to true or false. On the other hand, $v_j = \star$ is handled as a fresh variable, e.g., $\text{sign} \cdot \star + c > 0$ cannot be simplified. If $\rho[\vec{x}/\vec{v}]\downarrow \neq \text{false}$, we write $v \models^? \rho$.

Here, $v \models^? \rho$ means that there *could* be a value for $\star$ such that $\rho[\vec{x}/\vec{v}]\downarrow = \text{true}$.

With this definition at hand, we can define a set of abstract candidate assignments for the formulas $\rho_{f,m}$ in Lemma 10.3.29 and Cor. 10.3.31. Here, $\exists^{\leq 1}$ means that there exists *at most* one element in the specified domain.

Definition 10.3.36 (Sets of Candidate Assignments)**.** Let $h_1, \dots, h_d$ be a hierarchical k-partition of d, $f \in \mathbb{R}_A[\vec{x}]_{\text{lin}}$, $p = f(h_1, \dots, h_d)$, and $|\text{coefs}(p)| = \ell$. We then define for each $0 \leq m \leq \ell$

$$\text{candAss}(\rho_{f,m}) = \left\{ \vec{v} \in \{0, 1, -1, \star\}^d \mid \vec{v} \models^? \rho_{f,m}, \right.$$

$$\forall\, 1 \le i \le k\ \exists^{\le 1} j \in B_i \text{ with } v_j \neq 0,$$
$$v_j = \star \implies j \in B_{i_0} \text{ where } \lambda_{i_0} = 1.\Big\},$$

where $\rho_{f,m}$ is the formula constructed in Lemma 10.3.29 (resp. Cor. 10.3.31).

So, we only allow $\star$ if 1 occurs as a basis in $\vec{h}$. Otherwise $\star$ does not occur at all.

Intuitively, $\mathrm{candAss}(\rho_{f,m})$ contains all assignments $\vec{v} \in \{0, 1, -1, \star\}^d$ which *may* satisfy the formula $\rho_{f,m}$ (if a suitable value for $\star$ is found) and in addition allow for each i *at most one* variable x_j with $j \in B_i$ to be assigned the value 1 or -1 (or $\star$ under certain conditions) in $\vec{v}$, i.e., a non-zero value.

Example 10.3.37. Let us continue Ex. 10.3.30 and 10.3.32. We have

$$\mathrm{candAss}(\rho_{f,4}) = \left\{ \begin{bmatrix} 1\\0\\0\\1\\0 \end{bmatrix}, \begin{bmatrix} \star\\0\\0\\1\\0 \end{bmatrix}, \begin{bmatrix} -1\\0\\0\\1\\0 \end{bmatrix}, \begin{bmatrix} 0\\1\\0\\1\\0 \end{bmatrix}, \begin{bmatrix} 0\\\star\\0\\1\\0 \end{bmatrix}, \begin{bmatrix} 0\\-1\\0\\1\\0 \end{bmatrix}, \begin{bmatrix} 0\\0\\0\\1\\0 \end{bmatrix} \right\}$$

and

$$\mathrm{candAss}(\rho_{f,0}) = \left\{ \begin{bmatrix} \star\\0\\0\\0\\0 \end{bmatrix} \right\}.$$

Lemma 10.3.38. *Let $h_1, \dots, h_d$ be a hierarchical k-partition of d, let $f \in \mathbb{R}_{\mathbb{A}}[\vec{x}]_{\mathrm{lin}}$, $p = f(h_1, \dots, h_d)$, and $|\mathrm{coefs}(p)| = \ell$. Then for each $0 \le m \le \ell$*

$$\left|\mathrm{candAss}(\rho_{f,m})\right| \le \left(3 \cdot \max_{1 \le i \le k} v_i + 1\right)^k.$$

Proof. For any $0 \le m \le \ell$ we have $\mathrm{candAss}(\rho_{f,m}) \subseteq S$ where

$$S = \left\{ v \in \{0, 1, -1, \star\}^d \;\middle|\; \forall\, 1 \le i \le k\ \exists^{\le 1} j \in B_i \text{ with } v_j \neq 0 \right\}.$$

Now let us calculate the cardinality of S. If $\vec{v} \in S$, for any $1 \le i \le k$ we have $v_j \neq 0$ for at most one $j \in B_i$. But this means we have $3 \cdot |B_i| + 1$ possible combinations for the values of $v_j, j \in B_i$: either exactly one is 1, -1, or $\star$, or $v_j = 0$ for all $j \in B_i$. Now combining this result for all $1 \le i \le k$ by multiplication, we obtain

$$|S| = \prod_{i=1}^{k} (3 \cdot |B_i| + 1) = \prod_{i=1}^{k} (3 \cdot v_i + 1) \le \left(3 \cdot \max_{1 \le i \le k} v_i + 1\right)^k.$$

Since $\left|\text{candAss}(\rho_{f,m})\right| \leq |S|$, this proves the claim. □

The result of Lemma 10.3.38 is crucial for our algorithm to decide k-termination for uniform loops: we have just proven that the cardinality of $\text{candAss}(\rho_{f,m})$ is polynomial in d if k is assumed to be a parameter. This is due to the fact that the v_i form a k-partition of d, i.e., $v_i \leq d$ for $1 \leq i \leq k$. Hence, computing the set $\text{candAss}(\rho_{f,m})$ is slicewise polynomial, i.e., in XP.

Example 10.3.39. In Ex. 10.3.37, we have $\text{candAss}(\rho_{f,4}) = 7 \leq 100 = 10^2 = (3 \cdot 3 + 1)^2$. Note that here, $v_1 = 2$ and $v_2 = 3$.

This example already shows that the bound obtained in Lemma 10.3.38 is in general a coarse bound. Still, it suffices for our analysis.

We now combine Cor. 10.3.33 and Def. 10.3.36.

Definition 10.3.40 (Candidate Assignments for Inequation)**.** Let $h_1, \dots, h_d$ be a hierarchical k-partition of d, let $f \in \mathbb{R}_{\mathbb{A}}[\vec{x}]_{\text{lin}}$, $p = f(h_1, \dots, h_d)$, and $|\text{coefs}(p)| = \ell$. We then define

$$\begin{aligned} & \text{candAss}(\text{sic}(p > 0)) = \bigcup_{m=1}^{\ell} \text{candAss}(\rho_{f,m}) \\ \text{and} \quad & \text{candAss}(\text{sic}(p \geq 0)) = \text{candAss}(\text{sic}(p > 0)) \cup \text{candAss}(\rho_{f,0}). \end{aligned}$$

Corollary 10.3.41. *Let $h_1, \dots, h_d$ be a hierarchical k-partition of d via $v_1, \dots, v_k$, let $f \in \mathbb{R}_{\mathbb{A}}[\vec{x}]_{\text{lin}}$, $p = f(h_1, \dots, h_d)$, and $\triangleright \in \{>, \geq\}$.*

$$|\text{candAss}(\text{sic}(p \triangleright 0))| \leq (d+1) \cdot \left(3 \cdot \max_{1 \leq i \leq k} v_i + 1\right)^k$$

Proof. We show that $\ell \leq d + 1$. Then the result follows from Lemma 10.3.38 and Def. 10.3.40. The number of coefficients of p is determined by the number of terms of the form $n^a \cdot b^n$ occurring in p. But these terms are connected to the bases of the hierarchical k-partition: either $b = \lambda_i$ for some $1 \leq i \leq k$ and $0 \leq a \leq v_i - 1$ *or* $a = 0$ and $b = 1$. So we have at most $1 + \sum_{i=1}^{k} v_i = 1 + d$ such terms in p, i.e., $\ell \leq d + 1$. □

10.3.5 The Algorithm

Now we can present our algorithm for deciding termination of uniform loops. To this end, assume $f_1, \dots, f_e \in \mathbb{R}_{\mathbb{A}}[\vec{x}]_{\text{lin}}$ and let us assume that the normalized closed form of

the condition of the loop is a Boolean combination of the atoms $p_1 = f_1(h_1, \dots, h_d) \rhd_1 0, \dots, p_e = f_e(h_1, \dots, h_d) \rhd_e 0$, where $\rhd_1, \dots, \rhd_e \in \{>, \geq\}$. Let $\ell_1, \dots \ell_e$ be the number of coefficients of $p_1, \dots, p_e$, i.e., $|\text{coefs}(p_s)| = \ell_s$ for $1 \leq s \leq e$. The basic idea of our algorithm is to compute the sets $\text{candAss}(\rho_{f_s,m})$ for each $1 \leq s \leq e$ and each $1 \leq m \leq \ell_s$ (resp. $0 \leq m \leq \ell_s$ if $\rhd_s$ is $\geq$) and to then check for each element in the union of these sets whether it gives rise to a witness for eventual non-termination. We wrap up this idea in Algorithm 10.3.1. To do so, we first introduce the notion of a linear formula.

Definition 10.3.42 (Linear Formula)**.** A *linear formula* ξ is a propositional formula built from the Boolean operators $\wedge$ and $\vee$ over the atoms $\{f \rhd 0 \mid f \in \mathbb{R}_{\mathbb{A}}[\vec{x}]_{\text{lin}}, \rhd \in \{>, \geq\}\}$.

If $h_1, \dots, h_d$ is a hierarchical k-partition of d, the formula $\text{sic}(\xi[\vec{x}/\vec{h}])$ results from replacing each atom $f(h_1, \dots, h_d) \rhd 0$ in $\xi[\vec{x}/\vec{h}]$ by $\text{sic}(f(h_1, \dots, h_d) \rhd 0)$.

By $\text{candAss}(\text{sic}(\xi[\vec{x}/\vec{h}]))$ we denote the set

$$\bigcup_{f(h_1,\dots,h_d)\rhd 0 \text{ atom in } \xi[\vec{x}/\vec{h}]} \text{candAss}(\text{sic}(f(h_1, \dots, h_d) \rhd 0)).$$

We now present an algorithm to decide whether for a hierarchical k-partition $\vec{h} = (h_1, \dots, h_d)$ and linear formula ξ the formula

$$\exists \vec{x} \in \mathcal{S}^d, n_0 \in \mathbb{N}. \forall n \in \mathbb{N}_{>n_0}. \xi[\vec{x}/\vec{h}] \qquad \text{(see (6.1.10))}$$

is valid. Our algorithm calls the method SMT: $\text{SMT}(\psi, \mathcal{V}, \mathcal{S})$ denotes an algorithm checking whether the linear arithmetic formula ψ in the variables $\mathcal{V}$ over the ring $\mathcal{S} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}_{\mathbb{A}}\}$ is satisfiable, i.e., the variables $\mathcal{V}$ range over $\mathcal{S}$ and the coefficients of linear polynomials are also from $\mathcal{S}$. Thus, we write $\text{SMT}(\psi, \mathcal{V}, \mathcal{S}) \in \{\text{true}, \text{false}\}$. In our case, $|\mathcal{V}| = 1$. With these restrictions, the algorithm SMT has *polynomial* runtime (see [BPR06], [Len83b]).

First of all, note that in Algorithm 10.3.1 the method SMT is called to determine whether $\star$ can be assigned a non-zero value such that $\psi[\vec{x}/\vec{v}]\downarrow$ is satisfiable. Here, $\star$ denotes *one* undefined value in $\vec{v}$ and thus, we have to assign *all* occurrences of $\star$ in the formula $\psi[\vec{x}/\vec{v}]\downarrow$ the same value.

Let us discuss the complexity of Algorithm 10.3.1. The formula $\text{sic}(\xi[\vec{x}/\vec{h}])$ can be computed in polynomial time from ξ. By Cor. 10.3.41, the set $\text{candAss}(\text{sic}(\xi[\vec{x}/\vec{h}]))$ has at most $|\xi| \cdot (d+1) \cdot (3 \cdot \max_{1 \leq i \leq k} \nu_i + 1)^k$ elements, where $|\xi|$ is the number of atoms in ξ, each of which can be computed in polynomial time, and $\max_{1 \leq i \leq k} \nu_i \leq d$.

Algorithm 10.3.1: Checking Sign and Interval Conditions

Input: a linear formula ξ over the atoms $\{f \triangleright 0 \mid f \in \mathbb{R}_{\mathbb{A}}[\vec{x}], \triangleright \in \{>, \geq\}\}$, a hierarchical k-partition $h_1, \dots, h_d$ of d, $\mathcal{S}$
Result: $\top$ if $\exists \vec{x} \in \mathcal{S}^d.\ \mathrm{sic}(\xi[\vec{x}/\vec{h}])$ is valid, $\bot$ otherwise
$\psi \longleftarrow \mathrm{sic}(\xi[\vec{x}/\vec{h}])$
for $\vec{v} \in \mathrm{candAss}(\psi)$ **do**
  $\psi' \longleftarrow \psi[\vec{x}/\vec{v}]\downarrow$
  if *SMT*$(\psi' \wedge \star \neq 0), \{\star\}, \mathcal{S})$ **then**
    return $\top$
return $\bot$

Thus, when considering k to be a parameter, $\mathrm{candAss}(\mathrm{sic}(\xi[\vec{x}/\vec{h}]))$ can be computed in polynomial time. Moreover, evaluating a formula w.r.t. Def. 10.3.35 is possible in polynomial time, too. Finally, SMT has polynomial runtime as discussed before. So the runtime of the algorithm is polynomial when regarding k as a parameter.

We now prove that Algorithm 10.3.1 is sound and complete.

Theorem 10.3.43 (Soundness and Completeness of Algorithm 10.3.1)**.** *Algorithm 10.3.1 is sound and complete, i.e.,*

$$\mathrm{Sat}(\xi) = \top \quad \textit{iff} \quad \exists \vec{x} \in \mathcal{S}^d.\ \mathrm{sic}(\xi[\vec{x}/\vec{h}]) \textit{ is valid.}$$

Proof. We first prove soundness and then completeness.

Soundness: If Algorithm 10.3.1 returns $\top$, there is some $\vec{v} \in \mathrm{candAss}(\mathrm{sic}(\xi[\vec{x}/\vec{h}]))$ such that $\mathrm{sic}(\xi[\vec{x}/\vec{h}])[\vec{x}/\vec{v}]$ is satisfiable, i.e., there is a value $w \in \mathcal{S}$ such that $\vec{v}[\star/w]$ is a model of $\mathrm{sic}(\xi[\vec{x}/\vec{h}])$. But then $\exists \vec{x} \in \mathcal{S}^d.\ \mathrm{sic}(\xi[\vec{x}/\vec{h}])$ is valid.

Completeness: Let $\psi = \mathrm{sic}(\xi[\vec{x}/\vec{h}])$ and let $\exists \vec{x} \in \mathcal{S}^d.\ \psi$ be valid, i.e., there is some $\vec{v} \in \mathcal{S}^d$ such that $\psi[\vec{x}/\vec{v}]$ holds. We have to prove that there is some $\vec{v}' \in \mathrm{candAss}(\mathrm{sic}(\xi[\vec{x}/\vec{h}]))$ such that $\psi[\vec{x}/\vec{v}']\downarrow$ is satisfiable. Then the claim follows since the algorithm calls SMT on $\psi[\vec{x}/\vec{v}']\downarrow$ and therefore returns $\top$.

Note that ξ is a propositional formula only built from the operators $\wedge$ and $\vee$, i.e., it does not contain $\neg$. By construction, $\mathrm{sic}(\xi[\vec{x}/\vec{h}])$ then results from ξ by replacing each atom $f \triangleright 0$ by $\mathrm{sic}(f(h_1, \dots, h_d) \triangleright 0)$. So we can assume that $\vec{v}$ satisfies the formulas $\mathrm{sic}(f_1(h_1, \dots, h_d) \triangleright_1 0), \dots, \mathrm{sic}(f_e(h_1, \dots, h_d) \triangleright_e 0)$ and that satisfying these formulas is sufficient for satisfying ψ.

Let $\ell_1, \dots, \ell_e$ be the number of coefficients in the poly-exponential expressions $f_1(h_1, \dots, h_d), \dots, f_e(h_1, \dots, h_d)$. By construction, for each $1 \leq s \leq e$ the formula $\mathrm{sic}(f_s(h_1, \dots, h_d) \rhd_s 0)$ is true when assigning the variables $\vec{x}$ by $\vec{v}$, if and only if a *unique* formula ρ_{f_s,m_s} is true when assigning the variables $\vec{x}$ by $\vec{v}$ (note that the formulas $\rho_{f_s,0}, \dots \rho_{f_s,\ell_s}$ are pairwise exclusive). Here, $0 \leq m_s \leq \ell_s$.

By construction, all of the formulas ρ_{f_s,m_s} are non-exclusive, i.e., if a variable is required to have value zero by one of these formulas, none of the other formulas requires it to be non-zero and vice versa. Otherwise $\vec{v}$ could not satisfy all of the $\rho_{f_1,m_1}, \dots, \rho_{f_e,m_e}$ simultaneously. Moreover, each block of variables which is affected by one of the formulas ρ_{f_s,m_s}, i.e., one of its variables occurs in the formula, can be assumed to have at most one variable which is assigned a non-zero value by construction. Note that this assumption is only possible since we consider satisfiability over $\mathcal{S}^d$. So, w.l.o.g., we can assume that all variables which do not occur in any of the formulas ρ_{f_s,m_s} are assigned the value 0 by $\vec{v}$. Hence, $\vec{v}$ assigns at most one variable in each block of variables to a non-zero value.

(i) Case $\lambda_{i_0} = 1$ for some $1 \leq i_0 \leq k$: In this case, for the non-zero entries in $\vec{v}$ belonging to indices in B_i, $i \neq i_0$, where the B_j are the blocks from the underlying k-partition of d, only their sign is important since the formulas requiring them to be non-zero are sign conditions.

If there is a (unique) non-zero value v_j with $j \in B_{i_0}$, its actual value is indeed important: the formulas requiring this value to be non-zero can be interval conditions or it can be required to have a certain non-zero value by another quasi sign condition. Thus, we define

$$v'_j = \begin{cases} 0, & v_j = 0, \\ \star, & v_j \neq 0 \wedge j \in B_{i_0}, \\ 1, & v_j > 0 \wedge j \notin B_{i_0}, \\ -1, & v_j < 0 \wedge j \notin B_{i_0}. \end{cases}$$

As discussed before, $\psi[\vec{x}/\vec{v}']\downarrow$ is satisfiable since $\rho_{f_1,m_1}[\vec{x}/\vec{v}']\downarrow, \dots, \rho_{f_e,m_e}[\vec{x}/\vec{v}']\downarrow$ are satisfiable by construction: assigning $\star$ the unique non-zero value $v_j, j \in B_{i_0}$ is a satisfying assignment.

(ii) Case $\lambda_{i_0} \neq 1$ for all $1 \leq i_0 \leq k$: In this case, $\rho_{f_1,m_1}, \dots, \rho_{f_e,m_e}$ are sign conditions or only require variables to be zero. Thus, only the sign of the non-zero values

in $\vec{v}$ is important to satisfy these formulas. Hence we define

$$v'_j = \begin{cases} 0, & v_j = 0, \\ 1, & v_j > 0, \\ -1, & v_j < 0. \end{cases}$$

Thus, $\psi[\vec{x}/\vec{v}']\downarrow$ is satisfiable since $\rho_{f_1,m_1}[\vec{x}/\vec{v}']\downarrow, \dots, \rho_{f_e,m_e}[\vec{x}/\vec{v}']\downarrow$ are true by construction.

So in both cases

$$\vec{v}' \in \bigcup_{1 \le s \le e} \text{candAss}\left(\rho_{f_s,m_s}\right) \subseteq \text{candAss}(\psi),$$

i.e., Algorithm 10.3.1 outputs $\top$. Thus, completeness is shown. □

Example 10.3.44. Let us continue Ex. 10.3.37. Let $\xi = -x_1 + 3 \cdot x_3 + 4 > 0 \wedge 2 \cdot x_1 - 5 > 0$. Here, $\psi = \text{sic}(\xi[\vec{x}/\vec{h}])$ is equivalent to $\text{sic}(p > 0) \wedge x_1 - \frac{5}{2} > 0$, where $\text{sic}(p > 0)$ is stated in Ex. 10.3.34. Now consider $\vec{v} = (\star, 0, 0, 1, 0)$. Then

$$\psi[\vec{x}/\vec{v}]\downarrow = -\star + 4 > 0 \wedge \star - \tfrac{5}{2} > 0$$

which is satisfiable with the model $\star = 3$. Thus Algorithm 10.3.1 proves validity of $\exists \vec{x} \in \mathcal{S}^d.\ \text{sic}(\xi[\vec{x}/\vec{h}])$.

Lemma 10.3.45. *k-Termination of uniform loops over $\mathcal{S} \in \{\mathbb{Q}, \mathbb{R}_{\mathbb{A}}\}$ where the update matrix has only eigenvalues in $\mathcal{S}$ is in XP.*

Proof. We first transform the uniform loop with a matrix with entries in $\mathcal{S}$ in such a way that the update matrix is in Jordan normal form and then compute the normalized closed form of the transformed loop as in Lemma 10.3.6 in polynomial time. This closed form is a hierarchical partition by Cor. 10.3.21. Note that we only deal with linear loops and thus, the condition is a linear formula. By combining Cor. 10.3.33 and Thm. 10.3.43, we can then use Algorithm 10.3.1 to decide validity of the formula from Thm. 6.1.9, i.e., termination of the transformed loop. But this decides termination of the loop over $\mathcal{S}^d$ (see Cor. 9.1.17).

As the computation of the equivalent sign and interval conditions in Cor. 10.3.33 clearly works in polynomial time and we have discussed that Algorithm 10.3.1 runs in polynomial time when k is assumed to be a parameter, this proves the statement. □

We can finally prove Thm. 10.3.4.

Proof of Thm. 10.3.4. By Lemma 10.3.45, k-Termination of uniform loops over $\mathbb{R}_{\mathrm{A}}$ is in XP. These loops terminate over $\mathbb{R}_{\mathrm{A}}$ iff they terminate over $\mathbb{R}$ by Cor. 6.1.11. □

So we have seen that in case of a uniform loop $(\varphi, \vec{u})$ over $\mathbb{R}_{\mathrm{A}}$, the formula $\varphi[\vec{x}/\vec{q}_{norm}]$ is equivalent to a formula only containing so-called (quasi) sign conditions and interval conditions which can be computed efficiently. Using this insight, we reduced the search space for proving validity drastically. Hence, we obtained an algorithm which is more efficient than the decidability procedure presented in Ch. 6 in case $k < d$.

However, in Lemma 10.3.45 we have only dealt with uniform loops over $\mathbb{Q}$ or $\mathbb{R}_{\mathrm{A}}$ (resp. $\mathbb{R}$) with only eigenvalues in $\mathbb{Q}$ or in $\mathbb{R}_{\mathrm{A}}$. We have not yet dealt with uniform loops over $\mathbb{Q}$ with irrational eigenvalues or uniform loops over $\mathbb{Z}$. Here, the difficulty is that in addition to the formula $\mathrm{red}(\xi)$ we also have an update-invariant and $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_{\mathrm{A}})$-definable subset F defined by a formula ψ_F which stems from the transformation into Jordan normal form (see Ch. 9). For example, it is in general not possible to transform an integer matrix with only integer eigenvalues into Jordan normal form by an invertible integer matrix (see e.g., [Sid19]). Thus, to decide termination we have to decide validity of the formula $\exists \vec{x} \in \mathbb{R}_{\mathrm{A}}^d.\ \mathrm{red}(\xi) \wedge \psi_F$. We will discuss this issue in detail in the next section for loops over the integers. Nevertheless, our insights in the next section also hold over $\mathbb{Q}$.

Finally, it is worth noting that for proving Thm. 10.3.4 and Lemma 10.3.45 it was crucial that we could *transform* the loop in such a way that the update matrix is in Jordan normal form: here we relied on a special closed form coming from the Jordan normal form, while in Sect. 10.1 we only made use of the transformation to argue why the closed form is computable in polynomial time. Thus, in this section we have seen that the transformation from Ch. 9 does not only generalize our results from Ch. 6 to 8 to a wider class of loops but also gives rise to further results which would have been more difficult to prove without using it.

10.4 Termination of Uniform Loops Over the Integers

So far, we have only proven that k-termination of uniform loops over the real (algebraic) numbers is in XP (Thm. 10.3.4) and that k-termination of uniform loops over $\mathbb{Q}$ with only rational eigenvalues is in XP (Lemma 10.3.45). However, in "real-world" programming, termination over the integers is in general more interesting than termination over the reals (or rationals). While Algorithm 10.3.1 works over any ring $\mathcal{S}$, this is only of use for uniform loops $(\varphi, \vec{u})$ over $\mathbb{Z}$ where the update matrix can be transformed into Jordan normal form over the integers. If this is not the case, we have to transform $\mathbb{Z}^d$ according to Ch. 9 and obtain a $\mathrm{Th}_\exists(\mathbb{Z}, \mathbb{R}_{\mathrm{A}})$-definable and update

invariant set $F \neq \mathbb{Z}^d$. However, the satisfying assignment which Algorithm 10.3.1 might compute is not necessarily in F.

In this section, we discuss this problem and its intricacies. Unfortunately, we cannot present a clear result for k-termination of uniform loops over $\mathbb{Z}$. Nevertheless, we present some insights and methodology related to this problem. Still, we show that k-termination of uniform loops over $\mathbb{Z}$ with only integer eigenvalues is in XP.

A fundamental notion in this section is that of a *lattice*.

Definition 10.4.1 (Lattice). Let $\vec{w}_1, \dots, \vec{w}_m \in \mathbb{R}^d$ be linearly independent. The set $\left\{ \sum_{j=1}^{m} a_j \cdot \vec{w}_j \mid a_1, \dots, a_m \in \mathbb{Z} \right\}$ is called a *lattice* of rank m. The vectors $\vec{w}_1, \dots, \vec{w}_m$ are called a basis of the lattice.

There is a vast amount of research on lattice theory. However, let us briefly discuss why we are interested in lattices. Again, we fix a uniform loop (φ, A) over the *integers*, i.e., $A \in \mathbb{Z}^{d \times d}$ is an integer matrix with k eigenvalues $0 < \lambda_1 < \cdots < \lambda_k \in \mathbb{R}_{\mathbb{A}}$ each of geometric multiplicity one. Again, let $\nu_1, \dots, \nu_k \in \mathbb{N}$ be the algebraic multiplicities of $\lambda_1, \dots, \lambda_k$. Then there is some matrix $T \in \mathbb{R}_{\mathbb{A}}^{d \times d}$ such that $T^{-1} \cdot A \cdot T = Q$ is in Jordan normal form, where the size of the unique Jordan block of λ_i is ν_i.

Let $\eta(\vec{x}) = T \cdot \vec{x}$ be the induced automorphism. Then termination of (φ, A) on $\mathbb{Z}^d$ is equivalent to termination of $(\eta^{-1}(\varphi), Q)$ on $\eta(\mathbb{Z}^d) = T \cdot \mathbb{Z}^d$ by Cor. 9.1.17. But $T \cdot \mathbb{Z}^d$ is the set of all *integer* linear combinations of T's columns, i.e., the *lattice* generated by the columns of T which we denote by $\mathcal{L}_T$.

Example 10.4.2. In this section, we consider a loop with update induced by the matrix $A = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 6 \\ 0 & 1 & 0 & -11 \\ 0 & 0 & 1 & 6 \end{bmatrix} \in \mathbb{Z}^{4\times 4}$, i.e., $d = 4$. This matrix has the $k = 2$ positive eigenvalues $\frac{3 \pm \sqrt{5}}{2}$. The Jordan normal form of A is $Q = \begin{bmatrix} \frac{3-\sqrt{5}}{2} & 1 & 0 & 0 \\ 0 & \frac{3-\sqrt{5}}{2} & 0 & 0 \\ 0 & 0 & \frac{3+\sqrt{5}}{2} & 1 \\ 0 & 0 & 0 & \frac{3+\sqrt{5}}{2} \end{bmatrix}$ in which A is transformed by the transformation matrix

$$T = \begin{bmatrix} -\frac{3}{10} - \frac{1}{10} \cdot \sqrt{5} & \frac{1}{2} + \frac{9}{50} \cdot \sqrt{5} & -\frac{3}{10} + \frac{1}{10} \cdot \sqrt{5} & \frac{1}{2} - \frac{9}{50} \cdot \sqrt{5} \\ \frac{11}{10} + \frac{3}{10} \cdot \sqrt{5} & \frac{6}{25} \cdot \sqrt{5} & \frac{11}{10} - \frac{3}{10} \cdot \sqrt{5} & -\frac{6}{25} \cdot \sqrt{5} \\ -\frac{9}{10} - \frac{1}{10} \cdot \sqrt{5} & -\frac{9}{25} \cdot \sqrt{5} & -\frac{9}{10} + \frac{1}{10} \cdot \sqrt{5} & \frac{9}{25} \cdot \sqrt{5} \\ \frac{1}{5} & \frac{2}{25} \cdot \sqrt{5} & \frac{1}{5} & -\frac{2}{25} \cdot \sqrt{5} \end{bmatrix}.$$

Then,

$$\mathcal{L}_T = \left\{ a_1 \cdot \begin{bmatrix} -\frac{3}{10} - \frac{1}{10}\cdot\sqrt{5} \\ \frac{11}{10} + \frac{3}{10}\cdot\sqrt{5} \\ -\frac{9}{10} - \frac{1}{10}\cdot\sqrt{5} \\ \frac{1}{5} \end{bmatrix} + a_2 \cdot \begin{bmatrix} \frac{1}{2} + \frac{9}{50}\cdot\sqrt{5} \\ \frac{6}{25}\cdot\sqrt{5} \\ -\frac{9}{25}\cdot\sqrt{5} \\ \frac{2}{25}\cdot\sqrt{5} \end{bmatrix} + a_3 \cdot \begin{bmatrix} -\frac{3}{10} + \frac{1}{10}\cdot\sqrt{5} \\ \frac{11}{10} - \frac{3}{10}\cdot\sqrt{5} \\ -\frac{9}{10} + \frac{1}{10}\cdot\sqrt{5} \\ \frac{1}{5} \end{bmatrix} + a_4 \cdot \begin{bmatrix} \frac{1}{2} - \frac{9}{50}\cdot\sqrt{5} \\ -\frac{6}{25}\cdot\sqrt{5} \\ \frac{9}{25}\cdot\sqrt{5} \\ -\frac{2}{25}\cdot\sqrt{5} \end{bmatrix} \;\middle|\; a_i \in \mathbb{Z} \right\}$$

So, we would like to adapt Algorithm 10.3.1 in such a way, that we can also decide whether a satisfying assignment exists in $\mathcal{L}_T$. Thus, let $h_1, \dots, h_d$ be the hierarchical k-partition of d induced by A. Moreover, let us assume that Algorithm 10.3.1 outputs $\top$ in the iteration related to the assignment $\vec{v}$ for the input $(\varphi, \vec{h}, \mathbb{R}_{\mathbb{A}})$. Thus, there are atoms $f_1(h_1, \dots, h_d) \triangleright_1 0, \dots, f_e(h_1, \dots, h_d) \triangleright_e 0$ in $\varphi[\vec{x}/\vec{h}]$ such that $\vec{v}$ satisfies $\mathrm{sic}(f_s(h_1, \dots, h_d) \triangleright_s 0)$ for each $1 \leq s \leq e$ when assigning a suitable value from $\mathbb{R}_{\mathbb{A}}$ to $\star$.

By construction, for each $1 \leq s \leq e$ the formula $\mathrm{sic}(f_s(h_1, \dots, h_d) \triangleright_s 0)$ is true when instantiating the variables $\vec{x}$ by $\vec{v}$, if and only if a *unique* formula ρ_{f_s, m_s} is true when instantiating the variables $\vec{x}$ by $\vec{v}$ (note that the formulas $\rho_{f_s,0}, \dots \rho_{f_s,\ell_s}$ are pairwise exclusive, where $\ell_s = |\mathrm{coefs}(f_s(h_1, \dots, h_d))|$, see the proof of Thm. 10.3.43). Here, $0 \leq m_s \leq \ell_s$. By Lemma 10.3.29 and Cor. 10.3.31, all of the formulas are either (quasi) sign conditions or interval conditions. Thus, for each entry v_j of $\vec{v}$ we can find out whether $x_j = v_j$ is required by some $\rho_{f_s,0}$, whether $\mathrm{sign}(x_j) = \mathrm{sign}(v_j)$ or $v_j \in I$ is crucial to satisfy all of the $\rho_{f_s,0}$, where I is a disjoint union of intervals, or whether $v_j = 0$ is just due to setting variables to zero by default, i.e., the formula would still hold when assigning an arbitrary value from $\mathbb{R}_{\mathbb{A}}$ to v_j.

We call this an *abstract assignment*: for each entry of $\vec{v}$ we know whether its *actual value*, its *sign*, or its containment in a certain union of intervals I is necessary to be a satisfying assignment or not.

Definition 10.4.3 (Abstract Assignment)**.** An *abstract assignment* is an element of $\{\text{“}= 0\text{”}, \text{“}\in I_1\text{”}, \dots, \text{“}\in I_m\text{”}\}^d$, where $I_1, \dots, I_m$ are unions of real intervals and “$= 0$”, “$\in I_1$”,…,“$\in I_m$” are symbols.

Here, an interval can be a singleton to indicate a specific value, e.g., the interval $[3,3] = \{3\}$, the interval $(-\infty, \infty) = \mathbb{R}$ to indicate an arbitrary value, or one of the intervals $(-\infty, 0)$, $(0, \infty)$ to indicate the sign of the value.

Example 10.4.4 (Abstract Assignment)**.** Let us assume that Algorithm 10.3.1 yields the assignment $\vec{v} = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix}$ for a loop with the update matrix A from Ex. 10.4.2 and some condition. Moreover, let us assume that we need $v_1 > 0$, $v_2 = 0$, $v_3 < 0$, and $v_4 = 0$. Therefore, the abstract assignment is $\begin{bmatrix} \text{“} \in (0,\infty)\text{”} \\ \text{“} = 0\text{”} \\ \text{“} < (-\infty, 0)\text{”} \\ \text{“} = 0\text{”} \end{bmatrix}$.

In case that 1 is an eigenvalue of the matrix, we could also have an abstract assignment of the form $\begin{bmatrix} \text{“} \in I\text{”} \\ \text{“} = 0\text{”} \\ \text{“} = 0\text{”} \\ \text{“} = 0\text{”} \end{bmatrix}$, where $I = (-3, 3) \cup (5, \infty)$.

Moreover, note that the unions of intervals are at most the union of $e \leq |\varphi|$ many disjoint intervals. We now need to test whether there is a point in the lattice $\mathcal{L}_T$ satisfying these requirements. We discuss two steps for a given abstract assignment resulting from $\vec{v}$ to find an element in $F = \mathcal{L}_T$ meeting the requirements from the abstract assignment.

(i) Meet the conditions requiring certain entries to be zero (see Sect. 10.4.1).

(ii) Decide whether the remaining conditions in $\vec{v}$ can be satisfied (see Sect. 10.4.2).

Before we go into the details, note that the abstract assignment can only enforce at most *one* value to be non-zero in each block of the blocks of variables w.r.t. the partition $\nu_1, \dots, \nu_k$ due to Lemma 10.3.29 and Cor. 10.3.31. Thus, in (ii) we have to deal with at most $|\varphi| \cdot k$ many conditions.

10.4.1 Removing the Zero-Constraints

In this section, we assume that a satisfying abstract assignments $\vec{v}$ for the formula $\varphi[\vec{x}/\vec{h}]$ is given. Let $N \subseteq \{1, \dots, d\}$ be the indices such that $v_j = 0$ is necessary to satisfy all of the formulas ρ_{f_s, m_s} iff $j \in N$. Then,

$$\mathcal{L}_N = \{\vec{w} \in \mathcal{L}_T \mid w_j = 0 \text{ for } j \in N\}$$

is also a lattice, i.e., it is a *sublattice* of $\mathcal{L}_T$.

Example 10.4.5. Let us continue Ex. 10.4.4 where $N = \{2, 4\}$. Hence, $\mathcal{L}_N = \left\{ \vec{w} \in \mathcal{L}_T \;\middle|\; w_2 = 0 \text{ and } w_4 = 0 \right\}$.

We want to compute a basis of $\mathcal{L}_N$ in terms of the columns of the matrix T, i.e., we would like to solve the system of linear equations $\bigwedge_{j \in N} (T \cdot \vec{x})_j = 0$ where $\vec{x} \in \mathbb{Z}^d$. However, this has a drawback: While there exist efficient algorithms (i.e., they run in polynomial time) to solve this problem, they only work if the matrix T consists of only *rational* entries (see, e.g., [Coh93; LLL82; SL96]). In contrast, we have $T \in \mathbb{Q}(\lambda_1, \dots, \lambda_k)^{d \times d}$, i.e., T's entries are (possibly irrational) algebraic numbers from the smallest subfield of $\mathbb{R}_{\mathbb{A}}$ containing all eigenvalues $\lambda_1, \dots, \lambda_k$ of the matrix A as we have seen in Ex. 10.4.2. So, these algorithms cannot be applied directly to our situation. Nevertheless, we present a "work-around". Unfortunately, this has only polynomial runtime when regarding k as a parameter. The reason is that when regarding $\mathbb{Q}(\lambda_1, \dots, \lambda_k)$ as a finite-dimensional vector space over $\mathbb{Q}$, its dimension is in $\mathcal{O}(d^k)$.

Lemma 10.4.6. *In the situation above, the $\mathbb{Q}$-dimension of $\mathbb{Q}(\lambda_1, \dots, \lambda_k)$ is in $\mathcal{O}(d^k)$.*

Example 10.4.7. Let us reconsider Ex. 10.4.2. Here, $\mathbb{Q}\left(\frac{3+\sqrt{5}}{2}, \frac{3-\sqrt{5}}{2}\right) = \mathbb{Q}(\sqrt{5})$ has dimension 2 over $\mathbb{Q}$ as 1 and $\sqrt{5}$ form a basis of $\mathbb{Q}(\sqrt{5})$. Of course, $2 \leq 16 = 4^2 = d^k$.

From now on, let $D = \dim_{\mathbb{Q}}(\mathbb{Q}(\lambda_1, \dots, \lambda_k))$. Since the characteristic polynomial of A is a monic polynomial with integer coefficients, we can compute a basis $b_1, \dots, b_D$ for the field $\mathbb{Q}(\lambda_1, \dots, \lambda_k)$ in computation time which is a polynomial in the $\mathbb{Q}$-dimension $D \in \mathcal{O}(d^k)$, i.e., the computation requires only polynomial time when regarding k as a parameter (see e.g., [Coh93; Len83a; Len92; ORY09]).

Now let

$$\kappa_B : \mathbb{Q}(\lambda_1, \dots, \lambda_k) \to \mathbb{Q}^D, \sum_{j=1}^{D} r_j \cdot b_j \mapsto \begin{bmatrix} r_1 \\ \vdots \\ r_D \end{bmatrix}$$

be the natural isomorphism of $\mathbb{Q}$-vector spaces induced by the basis $b_1, \dots, b_D$. Computing this isomorphism, i.e., getting the basis representation for some element of $\mathbb{Q}(\lambda_1, \dots, \lambda_k)$, is possible in polynomial time in D (see e.g., [Coh93; Len83a; Len92; ORY09]). Therefore, it is possible in polynomial time when regarding k as a parameter.

Let $T_1, \dots, T_d$ be the columns of the matrix and let $\vec{y} = (y_1, \dots, y_d)$ be fresh *integer* variables. To remove the zero constraints, we have to solve the following system of linear equations, where N is the set from the beginning of this paragraph.

$$\left(T \cdot \vec{y}\right)_j = 0, j \in N. \tag{10.4.1}$$

We claim that this is equivalent to solving the following system of linear equations:

$$\left(\kappa_B(T) \cdot \vec{y}\right)_{(j-1)\cdot D+s} = 0, j \in N, s \in \{1, \dots, D\}. \tag{10.4.2}$$

Here, $\kappa_B(T) \in \mathbb{Q}^{(d\cdot D)\times d}$ is the $(d \cdot D) \times d$ matrix obtained from T when applying the automorphism κ_B componentwise.

The advantage of the system (10.4.2) is that it is a system of linear equations over integer variables with *rational* coefficients, whereas (10.4.1) might contain *irrational* coefficients. As proven in [Fru77; KB79], a solution of the system (10.4.2) can be computed in polynomial time in the size of the coefficients in this system. Thus, it can be solved in polynomial time when regarding k as a parameter due to the application of κ_B. We prove the correctness of this step:

Lemma 10.4.8. *A vector $\vec{y} \in \mathbb{Z}^d$ is a solution to (10.4.1) iff it is a solution to (10.4.2).*

Proof. We prove both directions separately.

"$\Longrightarrow$" Let $\left(T \cdot \vec{y}\right)_j = 0$ for all $j \in N$. Then, by denoting with T_m the m-th column of T, we have $\left(\sum_{m=1}^{d} y_m \cdot T_m\right)_j = 0$ for all $j \in N$. When applying κ_B componentwise then since κ_B is linear we have $\left(\sum_{m=1}^{d} y_m \cdot \kappa_B(T_m)\right)_{(j-1)\cdot D+s} = 0$ for all $j \in N$ and $s \in \{1, \dots, D\}$, i.e., $\vec{y}$ solves the system (10.4.2).

"$\Longleftarrow$" Let $\left(\kappa_B(T) \cdot \vec{y}\right)_{(j-1)\cdot D+s} = 0$ for all $j \in N$ and $s \in \{1, \dots, D\}$. Then by applying the inverse of κ_B to groups of D-entries in a standard way, i.e., we start with the first D elements of the column, then the second D elements and so on, we obtain for each $j \in N$

$$\left(\sum_{m=1}^{d} y_m \cdot T_m\right)_j = \kappa_B^{-1}\left(\left(\left(\kappa_B(T) \cdot \vec{y}\right)_{(j-1)\cdot D+s}\right)_{s=1,\dots,D}\right) = \kappa_B^{-1}\left((0)_{s=1,\dots,D}\right) = 0.$$

□

Example 10.4.9. Let us reconsider Ex. 10.4.2 and 10.4.7. Here, $B = (1, \sqrt{5})$ is a $\mathbb{Q}$-basis of $\mathbb{Q}\left(\frac{3+\sqrt{5}}{2}, \frac{3-\sqrt{5}}{2}\right) = \mathbb{Q}(\sqrt{5})$ and $\kappa_B : \mathbb{Q}(\sqrt{5}) \longrightarrow \mathbb{Q}^2, a + b\sqrt{5} \mapsto \begin{bmatrix} a \\ b \end{bmatrix}$.

Thus, we have

$$\kappa_B(T) = \begin{bmatrix} -\frac{3}{10} & \frac{1}{2} & -\frac{3}{10} & \frac{1}{2} \\ -\frac{1}{10} & \frac{9}{50} & \frac{1}{10} & -\frac{9}{50} \\ \frac{11}{10} & 0 & \frac{11}{10} & 0 \\ \frac{3}{10} & \frac{6}{25} & -\frac{3}{10} & -\frac{6}{25} \\ -\frac{9}{10} & 0 & -\frac{9}{10} & 0 \\ -\frac{1}{10} & -\frac{9}{25} & \frac{1}{10} & \frac{9}{25} \\ \frac{1}{5} & 0 & \frac{1}{5} & 0 \\ 0 & \frac{2}{25} & 0 & -\frac{2}{25} \end{bmatrix} \in \mathbb{Q}^{8\times 4} = \mathbb{Q}^{2\cdot 4\times 4}.$$

Remember that we have $N = \{2, 4\}$. Then for $\vec{y} \in \mathbb{Z}^4$ we have

$$\begin{aligned} & \left(T \cdot \vec{y}\right)_{j\in\{2,4\}} = 0 \\ \iff & \begin{bmatrix} \frac{11}{10} + \frac{3}{10}\cdot\sqrt{5} & \frac{6}{25}\cdot\sqrt{5} & \frac{11}{10} - \frac{3}{10}\cdot\sqrt{5} & -\frac{6}{25}\cdot\sqrt{5} \\ \frac{1}{5} & \frac{2}{25}\cdot\sqrt{5} & \frac{1}{5} & -\frac{2}{25}\cdot\sqrt{5} \end{bmatrix} \cdot \vec{y} = 0 \\ \iff & \left| \begin{array}{cccccccc} \left(\frac{11}{10} + \frac{3}{10}\cdot\sqrt{5}\right)\cdot y_1 & + & \frac{6}{25}\cdot\sqrt{5}\cdot y_2 & + & \left(\frac{11}{10} - \frac{3}{10}\cdot\sqrt{5}\right)\cdot y_3 & + & \left(-\frac{6}{25}\cdot\sqrt{5}\right)\cdot y_4 = 0 \\ \frac{1}{5}\cdot y_1 & + & \frac{2}{25}\cdot\sqrt{5}\cdot y_2 & + & \frac{1}{5}\cdot y_3 & + & \left(-\frac{2}{25}\cdot\sqrt{5}\right)\cdot y_4 = 0 \end{array} \right| \end{aligned}$$

By Lemma 10.4.8, this is equivalent to

$$\begin{aligned} & \left(\kappa_B(T) \cdot \vec{y}\right)_{j\in\{3,4,7,8\}} = 0 \\ \iff & \begin{bmatrix} \frac{11}{10} & 0 & \frac{11}{10} & 0 \\ \frac{3}{10} & \frac{6}{25} & -\frac{3}{10} & -\frac{6}{25} \\ \frac{1}{5} & 0 & \frac{1}{5} & 0 \\ 0 & \frac{2}{25} & 0 & -\frac{2}{25} \end{bmatrix} \cdot \vec{y} = 0 \\ \iff & \left| \begin{array}{cccccccc} \frac{11}{10}\cdot y_1 & + & 0\cdot y_2 & + & \frac{11}{10}\cdot y_3 & + & 0\cdot y_4 = 0 \\ \frac{3}{10}\cdot y_1 & + & \frac{6}{25}\cdot y_2 & + & \left(-\frac{3}{10}\right)\cdot y_3 & + & \left(-\frac{6}{25}\right)\cdot y_4 = 0 \\ \frac{1}{5}\cdot y_1 & + & 0\cdot y_2 & + & \frac{1}{5}\cdot y_3 & + & 0\cdot y_4 = 0 \\ 0\cdot y_1 & + & \frac{2}{25}\cdot y_2 & + & 0\cdot y_3 & + & \left(-\frac{2}{25}\right)\cdot y_4 = 0 \end{array} \right| \end{aligned}$$

For the latter system, we compute the set of solutions $\left\{ \begin{bmatrix} 0 \\ a \\ 0 \\ a \end{bmatrix} \;\middle|\; a \in \mathbb{Z} \right\}$. Thus, we

$$\text{get } \mathcal{L}_N = \left\{ T \cdot \begin{bmatrix} 0 \\ a \\ 0 \\ a \end{bmatrix} \;\middle|\; a \in \mathbb{Z} \right\} = \left\{ \begin{bmatrix} a \\ 0 \\ 0 \\ 0 \end{bmatrix} \;\middle|\; a \in \mathbb{Z} \right\}.$$

Hence, we can get rid of the abstract conditions requiring the entries in N to be zero in polynomial time when regarding k as a parameter, i.e., in XP. Note that in general the system (10.4.1) contains a lot more variables than it contains equations. So, the solutions yield a certain linear dependence between the variables. This dependency can then be used to reduce the number of the variables in the system, i.e., it gives rise to a basis of $\mathcal{L}_N$, where each basis vector is represented by a $\mathbb{Z}$-linear combination of the columns of T. This concludes (i).

It is worth noting that the same method also works over the *rational* numbers, i.e., if A is a rational matrix, we can still compute a basis of the (in this case) vector space over $\mathbb{Q}$ given by $\{T \cdot \vec{x} \mid \vec{x} \in \mathbb{Q}^d, (T \cdot \vec{x})_j = 0 \text{ for all } j \in N\}$.

10.4.2 Handling the Inequations

After applying the technique from Sect. 10.4.1, we are left with a basis of $\mathcal{L}_N$. Let r be the rank of $\mathcal{L}_N$. Then we have a matrix $P \in \mathbb{R}_{\mathbb{A}}^{d \times r}$ whose columns form a basis of $\mathcal{L}_N$.

Let $Constr$ be the indices of the abstract assignment $\vec{v}$ which enforce containment in a disjoint union I of intervals. By decomposing disjoint unions and repeating the procedure if necessary, we can thus assume that for every index $j \in Constr$ there is an attached interval I_j such that $\vec{v}$ requires $v_j \in I_j$. Note that containment in such an interval can be described by at most 2 possibly strict inequations. To ease notation, we define $I_j = \mathbb{R}$, if $j \notin Constr$. All the sets I_j are *convex*, i.e., $K = \prod_{j=1}^{d} I_j \subseteq \mathbb{R}^d$ is also a convex set.

Definition 10.4.10 (Convex Set)**.** Let $S \subseteq \mathbb{R}^N$. S is *convex* if for every two elements $\vec{w}_1, \vec{w}_2$ and any $\lambda \in [0, 1]$ we have $\lambda \cdot \vec{w}_1 + (1 - \lambda) \cdot \vec{w}_2 \in S$.

Thus, a convex set S contains the line segment connecting any two elements of S.

To finally decide uniform termination over the integers, we have to decide whether there exists some $X \in \mathbb{Z}^r$ such that

$$\begin{aligned} & P \cdot X \in K \\ \iff & X \in P^{-1}(K) \\ \iff & X \in P^{-1}(K) \cap \mathbb{Z}^r, \end{aligned} \tag{10.4.3}$$

where $P^{-1}(K)$ is the preimage of K under the linear mapping P. By simple mathematical transformations, one obtains that $P^{-1}(K)$ is also convex.

Example 10.4.11. Continuing Ex. 10.4.9, $P = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, i.e., $r = 1$. Moreover, $Constr = \{1, 3\}$, where $I_1 = (0, \infty)$ and $I_3 = (-\infty, 0)$. Now we have to find an element $a \in \mathbb{Z}$ such that $P \cdot a = a \cdot P \in I_1 \times \mathbb{R} \times I_3 \times \mathbb{R}$, which is of course impossible since $(a \cdot P)_3 = 0 \notin I_3$. Hence, we conclude that there is no element in $\mathcal{L}_T$ satisfying the abstract assignment from Ex. 10.4.4.

Unfortunately, there is no known result which shows that deciding $P^{-1}(K) \cap \mathbb{Z}^r \neq \emptyset$ is possible in polynomial time when regarding k as a fixed parameter. Still, we can discuss existing results and explain why we have reduced the question to (10.4.3).

First of all, $P^{-1}(K)$ can be described by at most $2 \cdot k$ linear inequations where the coefficients of the occurring linear polynomials arise from P and are real algebraic numbers. Thus, $P^{-1}(K)$ is a *semi-algebraic set*.

Definition 10.4.12 (Semi-Algebraic Set)**.** A set is *semi-algebraic* if it is $\mathrm{Th}_\exists(\mathbb{R}_\mathrm{A})$-definable.

In [KP97], the authors show that if a convex semi-algebraic set $S \subseteq \mathbb{R}^L$ contains an integer point, it contains an integer point of size $M^{\mathcal{O}(L^4)}$, where M is the maximal degree of the polynomials describing S. This then shows that if the number of variables L is bounded by a constant, this problem can be solved in polynomial time. This result is one of the crucial ingredients for proving that termination of linear conjunctive loops over the integers is decidable [HOW19; OPW15]. Unfortunately, we cannot use this result: In [KP97], the polynomials are only allowed to have *rational* coefficients. In our case, $P^{-1}(K)$ is encoded by linear polynomials with *algebraic* coefficients. When encoding these algebraic numbers in this system as zeros of rational polynomials (i.e., using the standard encoding for algebraic numbers) the maximal degree occurring in the polynomials is $D \in \mathcal{O}(d^k)$ (Lemma 10.4.6). Thus, as $L = r$ and $r \in \Theta(d - k)$ we only obtain the bound

$$\left(d^k\right)^{\mathcal{O}((d-k)^4)}$$

which is of limited use. A similar bound holds when considering rationals instead of integers (i.e., $P^{-1}(K) \cap \mathbb{Q}^r$). Here we obtain the bound $\left(d^k\right)^{\mathcal{O}((d-k)^3)}$ (see [DZ10]).

However, linear integer programming with a fixed number of variables *or* a fixed number of constraints is possible in polynomial time (see [Len83b]). Again, only rational coefficients have been considered. The case for a fixed number of variables and coefficients from the real algebraic numbers also holds as was shown in [KP97]. This might give hope that deciding (10.4.3) is possible in polynomial time when fixing k as a parameter.

Nonetheless, when considering the decidability of the existential theory of the reals, i.e., $\mathrm{Th}_\exists(\mathbb{R})$, then the best known complexity bound is *exponential* in the number of atoms in a formula [BPR06], i.e., when fixing the number of atoms in a formula, i.e., the size of a formula, then deciding $\mathrm{Th}_\exists(\mathbb{R})$ is still only known to be decidable in exponential time. This might reduce confidence that termination of uniform loops over the integers is in general in XP.

In this section, we have seen how the question whether a uniform loop over the integers terminates can be reduced to checking whether a semi-algebraic set described by a formula of length $\mathcal{O}(k)$ contains an integer point. Additionally, we have discussed reasons why this could be possible in XP and why this might also be impossible. Moreover, we have mentioned that our discussion also holds when considering uniform loops over the rationals with irrational eigenvalues.

However, due to the result by [Len83b], if we have only rational eigenvalues, we get a linear integer program with a fixed number of constraints (in $\mathcal{O}(k)$): Since the variables range over the integers, we can turn all strict inequations into weak ones:

> **Example 10.4.13.** Let us assume that we would like to find an integer solution for $\frac{2}{5} \cdot x_1 - \frac{2}{3} \cdot x_2 < \frac{1}{3}$. This is equivalent to finding an integer solution for $6 \cdot x_1 - 10 \cdot x_2 < 5$. Since x_1 and x_2 are integers, this is equivalent to $6 \cdot x_1 - 10 \cdot x_2 \leq 4$.

This can indeed be solved in polynomial time, if k is considered to be a parameter. Moreover, note that a rational eigenvalue of an integer matrix is already an integer. Thus, by using our results from Sect. 10.4.1 we obtain the following result at the end of this section.

Theorem 10.4.14 (Parameterized Complexity of k-Termination Over $\mathbb{Z}$)**.** *For uniform loops $(\varphi, \vec{u})$ over $\mathbb{Z}$ with only integer eigenvalues k-termination is in XP.*

Proof. Given such a loop, it can be transformed into Jordan normal form with a matrix $T \in \mathbb{Q}^{d \times d}$. Thus, we can compute all elements of abstract assignments for witnesses for non-termination by Algorithm 10.3.1 in polynomial time when considering k to be a parameter. By the methods described here, we can check in XP whether $F = \mathcal{L}_T$

contains an element satisfying an abstract assignment. If yes, the loop does not terminate. If for none of the satisfying abstract assignments computed by Algorithm 10.3.1 F contains an element satisfying such an assignment, by Lemma 6.1.12, Cor. 10.3.33, and Thm. 10.3.43, there cannot be a witness for eventual non-termination in F. Thus, the loop terminates on $\mathbb{Z}^d$ by Cor. 9.1.17. □

Chapter 11

Related Work

In this chapter, we give an overview of the literature related to the results which we presented in this part. We discuss work related to Ch. 6 and 9 in Sect. 11.1. Afterwards, we review results related to Ch. 7 and 8 in Sect. 11.2. Finally, we discuss work related to Ch. 10 (Sect. 11.3) and suggest further reading (Sect. 11.4).

11.1 Work Related to Ch. 6 and 9

In contrast to *automated* termination analysis (see e.g., [Bab+13; BCF13; BDG19; BG17; BMS05a; BMS05b; BMS05c; BMS05d; Che+14; FG19b; Gie+17; Gie+19b; Gup+08; Lar+13; Lar+14; LH15; LH18; PR04]), we investigated *decidability* of termination for certain classes of loops in Ch. 6. It is well known that termination of general programs on a specific input is undecidable (i.e., the halting problem), which was first shown in Turing's seminal work [Tur37] and formalized in terms of modern computer science by Kleene [Kle52] and Davis [Dav58]. Herman [Her71] then showed that the universal halting problem, i.e., termination on *all* inputs, is not only undecidable but not even semi-decidable. Thus, decidability results can only be obtained for very restricted classes of programs.

Nevertheless, many techniques used in automated tools for termination analysis (for example, the application of ranking functions, see e.g., [Ali+10; BDG19; BG14; BG17; BMS05a; BMS05b; BMS05d; PR04]) focus on similar classes of loops since such loops occur as sub-programs in (abstractions of) real programs. Tools based on these techniques have turned out to be very successful, also for larger classes of programs. Thus, these tools could benefit from integrating our (semi-)decision procedures and

applying them instead of incomplete techniques for any sub-program that can be transformed into a *twn*-loop.

Related work on decidability of termination also considers related (and often more restricted) classes of loops. For linear conjunctive loops, termination over $\mathbb{R}$ [Li14; Li17b; Tiw04; Xia+11] and $\mathbb{Q}$ [Bra06] is decidable. Decidability of termination of linear conjunctive loops over $\mathbb{Z}$ was conjectured to be decidable in [Tiw04]. After several partial results [BIK14; FG19a; OPW15] this conjecture was confirmed recently in [HOW19]. In addition to these works, [BGM12] shows that termination of slight generalizations of linear conjunctive loops over $\mathbb{Z}$, where either a non-deterministic update or a single conditional branching in the loop body are allowed, is undecidable. Tiwari [Tiw04] uses the special case of our *twn*-transformation from Ch. 9 where the loop and the automorphism are linear. In contrast to these results, our approach applies to *non-linear* loops with *arbitrary* conditions over *various rings*.

As discussed in Sect. 8.3, *linearization* is an alternative attempt to handle non-linearity. While the *update* of solvable loops can be linearized [OBP16a; RK07], the *condition* cannot. Otherwise, one could linearize any loop $(p = 0, \vec{x})$, which terminates over $\mathbb{Z}$ iff p has no integer root. With [HOW19], this would imply decidability of Hilbert's Tenth Problem.

In the non-linear case, [Li16] proves decidability of termination for conjunctive loops on $\mathbb{R}^d$ for the case that the condition defines a compact and connected subset of $\mathbb{R}^d$. In [XZ10], decidability of termination of conjunctive linear-update loops on $\mathbb{R}^d$ with the *non-zero minimum property* is shown, which covers conjunctive linear-update loops with real spectrum. For general conjunctive linear-update loops on $\mathbb{R}^d$ undecidability is conjectured. Moreover, in [Wu+10] it is shown that termination of conjunctive linear-update loops where the update matrix has only periodic real eigenvalues is decidable, which also cover conjunctive linear-update loops with real spectrum. Here, a special case of our *twn*-transformation from Ch. 9 is used where the automorphism is linear. In combination with [OBP16a], [Wu+10; XZ10] both yield a decision procedure for termination of conjunctive *twn*-loops over $\mathbb{R}$. Furthermore, [NOW20] proves that termination of (not necessarily conjunctive) linear-update loops is decidable if the condition describes a compact set. Finally, [XL13] gives sufficient criteria for (non-)termination of solvable loops and [Li17a] introduces sufficient conditions under which termination of non-deterministic non-linear loops on $\mathbb{R}^d$ can be reduced to satisfiability of a semi-algebraic system.

For linear-update loops with real spectrum over $\mathbb{R}$, we prove $\forall\mathbb{R}$-completeness of termination, whereas [Wu+10; XZ10] do not give tight complexity results. The approach from [XL13] is incomplete, whereas we present a complete reduction from termination to the respective existential fragment of the first-order theory. The work in [Li17a] is orthogonal to ours as it only applies to loops that satisfy certain non-

trivial conditions. Moreover, we consider loops with arbitrary conditions over various rings, while [Li16; Li17a; Wu+10; XZ10] only consider conjunctive loops over $\mathbb{R}$ and [NOW20] only considers loops over $\mathbb{R}$ where the condition defines a compact set.

11.2 Work Related to Ch. 7 and 8

In addition to termination of a polynomial loop on *all* inputs, we also considered a decision procedure for termination on a *specific* input in Ch. 7. Moreover, the decision procedures for termination presented in [BIK14; Bra06; FG19a; HOW19; Li14; OPW15; Tiw04] and ours from Ch. 6 rely on the notion of *eventual* non-termination and thus, they do not yield witnesses for non-termination.

For linear loops $(\varphi, \vec{u})$ where φ is a conjunction of atoms of the form $pol > 0$, *subsets* of $NT_{(\varphi,\vec{u})}$ can be computed via the techniques presented in [Li14; Li17b]. Moreover, for the same class of loops, a complete characterization of $NT_{(\varphi,\vec{u})}$ in the case of two variables is given in [DX12]. Here, it is also proven that in the case of more than two variables $NT_{(\varphi,\vec{u})}$ cannot be described by a propositional formula over polynomial inequations. Furthermore, there are certain special cases of linear loops over $\mathbb{Z}$ with conjunctive conditions where $\vec{u}^n$ and thus also $NT_{(\varphi,\vec{u})}$ is known to be Presburger-definable [BIK14]. For non-linear loops, [Li16] shows how to compute witnesses for non-termination under certain prerequisites. In particular, the condition needs to characterize a closed, bounded, and connected set. When their sufficient criterion for non-termination of solvable loops succeeds, [XL13] show how to compute a witness-for non-termination. In contrast, our result on the decidability of the halting problem enables us to enumerate *all* non-terminating inputs, and it allows for non-linearity and more complex conditions.

Obviously, decidability of the halting problem does not imply decidability of (*universal*) termination: For example, termination is undecidable for *twn*-loops over $\mathbb{Z}$ with non-linear integer arithmetic (due to undecidability of Hilbert's 10-th problem, see Sect. 11.1). Nonetheless, we showed that the halting problem is decidable for *twn*-loops over $\mathbb{Z}$ (and over any other ring $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_{\mathrm{A}}$).

For the converse direction, note that in contrast to Turing-complete languages, in the restricted formalism of polynomial loops, decidability of termination does not trivially imply decidability of the halting problem. The reason is that one cannot encode a fixed initial value of the program variables in a loop of the form in Fig. 4.1(a).

For example, while termination of *linear* integer loops with conjunctive conditions is decidable [HOW19], the halting problem for these loops is equivalent to the so-called positivity problem (see [OW15]), whose decidability is open. The positivity problem asks whether a given linear integer recurrence sequence with *fixed initial*

values only takes positive values. Moreover, the closely related Skolem problem asks whether a given linear integer recurrence sequence with fixed initial values ever takes the value 0. While decidability of both the positivity and the Skolem problem is still open in the general case, there exist several partial solutions, see e.g., [ABV17; Ken+20; OW14a; OW14b; OW14c]. For example, in [OW14a] it is shown that the positivity problem is decidable for recurrences of order 5 or less, which implies decidability of the halting problem for *linear* integer loops with at most 4 variables and conjunctive conditions: here, the first step of the transformation from a loop $(\varphi, A \cdot \vec{x} + \vec{b})$ into a recurrence relation is to eliminate the vector $\vec{b}$ by introducing an additional variable as in the equivalence of termination of a loop of the form in Fig. 10.1(a) and the loop in Fig. 10.1(b). It is considered folklore that the Skolem and positivity problem are decidable if the characteristic polynomial of the recurrence sequence has real roots only [ABV17]. For the Skolem problem, a proof of (a generalization of) this claim can be found in [Hal+05]. Hence, the halting problem is decidable for linear integer loops with conjunctive conditions where the update matrix has real eigenvalues only. Moreover, decidability of the halting problem for linear loops where the update matrix has only periodic rational eigenvalues was shown in [Kin+19]. Additionally, it is shown in [BGM12] that for the same generalizations of linear loops as mentioned in Sect. 11.1, i.e., where either non-determinism in the update or a single conditional branching in the loop body are allowed, not only termination on all inputs but also the halting problem is undecidable.

In contrast to these results, we showed decidability of the halting problem for *twn*-loops with *non-linear arithmetic* where the condition is an *arbitrary propositional formula* over polynomial inequations, which is orthogonal to the positivity problem.

Furthermore, there is a large body of work on *automated* complexity analysis (see e.g., [Alb+11; Alb+19a; Bro+16; Car+17; Flo16; Fro+20; HAH12; HDW17; HS15; SSH17; SZV17]). This line of work aims at inferring bounds on the runtime of programs with complex control flow by using incomplete techniques. The most popular approaches to infer runtime bounds automatically synthesize *lexicographic combinations of linear ranking functions*. To see the necessity of complete approaches, consider the *twn*-loop in Fig. 11.1 which terminates, but is beyond the capabilities of such ranking functions [BDG19; LH15]. However, we compute the bound $2 \cdot (|x_1| + |x_2|) + 3$ on its runtime with our approach from Ch. 8.

There are also several works on runtime analysis for restricted program models such as vector addition systems with states (see e.g., [Brá+18; Zul19]), bounded polynomial loops where the number of iterations of the loop is fixed in advance and does not depend on the loop body (see e.g., [BH20; BJK08; BK12; BP16]), and max-plus automata (see e.g., [CDZ14]). These models are orthogonal to the (unbounded)

$$\textbf{while } (x_1 \geq x_2 \wedge x_2 \geq 1) \textbf{ do } \{ \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leftarrow \begin{bmatrix} 2 \cdot x_1 \\ 3 \cdot x_2 \end{bmatrix} \quad \}$$

Figure 11.1: Linear *twn*-Loop not Admitting a (Lexicographic) Linear Ranking Function

polynomial loops considered in this chapter.

11.3 Work Related to Ch. 10

Regarding complexity, [OPW15] proves that termination of conjunctive linear loops over $\mathbb{Z}$ with update $\vec{x} \leftarrow A \cdot \vec{x} + \vec{b}$ is in EXPSPACE if A is diagonalizable resp. in PSPACE if $|\vec{x}| \leq 4$: here its crucial ingredient is the method from [KP97], which we have discussed briefly at the end of Ch. 10. Additionally, the crucial step in [HOW19] also relies on [KP97]. Thus, this technique is at least in EXPSPACE. Moreover, as stated in [OPW15; Xia+11], the techniques from [Bra06; Tiw04] run in polynomial time since decidability of linear formulas from $\mathrm{Th}_\exists(\mathbb{R}_\mathbb{A})$ and $\mathrm{Th}_\exists(\mathbb{Q})$ only containing conjunctions is in P (see e.g., [BM07; Kar84; Kha80]). So termination of conjunctive linear loops over $\mathbb{Q}$ and $\mathbb{R}$ is in P. Furthermore, in [BG14] it is shown that existence of a linear (lexicographic) ranking function for linear conjunctive loops over $\mathbb{Q}$ or $\mathbb{Z}$ is Co-NP-complete.

Our Co-NP-completeness result is orthogonal to those results as we allow disjunctions in the condition. Moreover, Co-NP-completeness also holds for termination over $\mathbb{Z}$, while [Bra06; Tiw04] only consider termination over $\mathbb{Q}$ resp. $\mathbb{R}$. Additionally, we showed that k-termination of uniform loops over $\mathbb{R}_\mathbb{A}$ is in XP, where the parameter k is the number of eigenvalues. This result is also orthogonal to [Bra06; Tiw04] since we again allow disjunctions in the condition. Furthermore, existence of a linear (lexicographic) ranking function is not necessary for termination of linear loops as discussed above for the loop in Fig. 11.1.

11.4 Further Reading

Several works exploit the existence of closed forms for solvable (or similar classes of) loops to, e.g., analyze safety or reason about invariants (see e.g., [HBK20; HJK18; HK21; Kin+18; Kin+19; Kov08; OBP16a; OBP17; RK04; RK07]). While our approach covers solvable loops with real eigenvalues (by Cor. 9.2.4), it also applies to loops

which are not solvable, see Ex. 9.2.7. Note that our transformation from Ch. 9 may also be of interest for other techniques for solvable or other sub-classes of polynomial loops as it may be used to extend the applicability of such approaches.

Chapter 12

Conclusion and Future Work

In this part, we studied termination of *twn*-loops, i.e., loops of the form in Fig. 4.1(a) with polynomial arithmetic where the update $\vec{x} \leftarrow \vec{u}$ is a triangular system of polynomial equations and the use of non-linearity in $\vec{u}$ is mildly restricted.

First of all, we presented a reduction from termination of *twn*-loops to $\mathrm{Th}_\exists(\mathcal{S}, \mathbb{R}_\mathrm{A})$ in Ch. 5 and 6. This implies decidability of termination over $\mathcal{S} = \mathbb{R}_\mathrm{A}$ and $\mathcal{S} = \mathbb{R}$ and semi-decidability of non-termination over $\mathcal{S} = \mathbb{Z}$ and $\mathcal{S} = \mathbb{Q}$.

Additionally, we proved decidability of the halting problem for *twn*-loops, i.e., termination on a specific input. While we considered loops over $\mathbb{R}_\mathrm{A}$ in Ch. 7, decidability of the halting problem for loops over other rings $\mathbb{Z} \leq \mathcal{S} \leq \mathbb{R}_\mathrm{A}$ follows immediately. The key idea is to approximate the so-called *stabilization threshold* $sth_{(\varphi,\vec{u})}(\vec{e})$, i.e., the number of iterations after which the condition of the loop φ stops changing its truth value, for a *given* input $\vec{e}$. In this way, we also showed implicitly that the set of witnesses for non-termination of a *twn*-loop is recursively enumerable.

Furthermore, we generalized this idea to approximate the stabilization threshold $sth_{(\varphi,\vec{u})}(\vec{e})$ for *arbitrary* inputs $\vec{e}$ of *twn*-loops over the integers in Ch. 8. In this way, we showed that for all terminating inputs $\vec{e}$, the runtime complexity of a *twn*-loop over the integers is bounded by a polynomial in the 1-norm $\|\vec{e}\|$ of the input. Moreover, we presented how to *compute* such a polynomial. Additionally, we showed how to read off the degree of this polynomial directly from the loop. As a corollary, it follows that the runtime of triangular *linear* loops is bounded by a linear polynomial in $\|\vec{e}\|$.

In addition, we showed how to transform certain non-*twn*-loops into *twn*-form in Ch. 9, and discussed how this generalizes our results to a wider class of loops, including *solvable loops with real eigenvalues*. We also showed that *twn*-transformability is semi-decidable. Finally, we used our transformation and our decision procedure to prove Co-

NP-completeness (resp. $\forall\mathbb{R}$-completeness) of termination of linear (resp. linear-update) loops with real spectrum, and to prove that k-termination of uniform loops over $\mathcal{S} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}_\mathbb{A}, \mathbb{R}\}$ with only eigenvalues in $\mathcal{S}$ is in XP in Ch. 10. Additionally, note that for our results in Sect. 10.3 and 10.4 on deciding k-termination this transformation was crucial: Here, we relied on a very special form of the closed forms for the update of the loop which could only be achieved via transforming the loop in advance such that the update is in Jordan normal form.

Note that our results from Ch. 6 to 8 also apply to linear(-update) loops where all eigenvalues of the update matrix are periodic real. By Thm. 5.1.2 and Lemma 8.1.5, we can chain the loop until we obtain a matrix with only real eigenvalues before applying these techniques. However, as discussed in Sect. 10.2, this can take up to exponential space in the dimension of the matrix d. Thus, while this might increase the applicability of our techniques, it does not generalize our tight complexity results.

There are multiple directions for future work. First of all, while we showed decidability of termination of *twn*-transformable loops over $\mathbb{R}_\mathbb{A}$ and $\mathbb{R}$, we have only shown that *twn*-transformability is semi-decidable. Thus, a clear result on the (un)decidability of *twn*-transformability would be desirable. Moreover, our results only enabled us to show Co-NP-completeness ($\forall\mathbb{R}$-completeness) of deciding termination for linear(-update) loops since we cannot handle complex eigenvalues. Unfortunately, termination is only known to be in EXPSPACE for linear conjunctive loops over the integers so far. Furthermore, it is still open whether termination for arbitrary linear-update loops over $\mathbb{R}_\mathbb{A}$ is decidable. In future work, this gap might be closed. Additionally, any characterization of a larger class of loops where termination is decidable increases the applicability of complete techniques for verifying termination.

Moreover, we only proposed upper bounds on the (asymptotic) runtime of *twn*-loops in Ch. 8. An evaluation of the tightness of these bounds, e.g., by giving a complete technique for computing a lower bound, is left to future work.

Finally, the techniques presented in Ch. 6 to 9 should be automated and embedded into existing automatic termination and runtime complexity analyzers such as AProVE [Gie+17] or KoAT [Bro+16; KoAT]. In this way, our results may improve the automated termination and runtime complexity analysis of larger programs.

Part III

Probabilistic Programs

Chapter 13

Introduction

13.1 Motivation

In this part, we study probabilistic programs featuring discrete probabilistic choices, randomized assignments via discrete probability distributions, non-determinism, as well as *unbounded loops*. Randomized algorithms are the classical application of such programs. Moreover, applications in *biology*, *quantum computing*, *cyber security*, *machine learning*, and *artificial intelligence* recently led to rapidly growing interest in probabilistic programming [Gor+14].

Formal verification of probabilistic programs is strictly harder than for non-probabilistic programs [KKM19]. Given a random variable f, a key verification task is to reason about the *expected value of* f after termination of a program $\mathbf{P}$ on input s. If f is the indicator function of an event A, this expected value is the probability that A has occurred at termination of $\mathbf{P}$.

For verifying probabilistic loops, i.e., loops whose body features probabilistic behavior, most approaches share a common, *conceptually simple*, technique: an *induction rule* for verifying *upper bounds* on expected values, which are characterized as least fixed points (lfp) of a suitable function Φ. This rule, called "Park induction", reads

$$\Phi(I) \sqsubseteq I \qquad \text{implies} \qquad \mathrm{lfp}\,\Phi \sqsubseteq I,$$

i.e., for a candidate upper bound I we check $\Phi(I) \sqsubseteq I$ (for a suitable partial order $\sqsubseteq$) to prove that I is indeed an upper bound on the least fixed point, and hence on the sought-after expected value.

For *lower bounds*, a simple proof principle analogous to Park induction, namely

$$I \sqsubseteq \Phi(I) \qquad \text{implies} \qquad I \sqsubseteq \mathrm{lfp}\,\Phi$$

is *unsound* in general. *Sound* rules, on the other hand, often suffer from the fact that either f needs to be *bounded*, or that one has to find the *limit of some sequence*, as well as the sequence itself, rendering those rules conceptually much more involved than Park induction.

One of our main contributions in this part is to provide relatively *simple* side conditions that can be added to the (unsound) implication above such that the implication becomes true, i.e.,

$$I \sqsubseteq \Phi(I) \wedge \begin{array}{c}\text{some side}\\ \text{conditions}\end{array} \quad \text{implies} \quad I \sqsubseteq \operatorname{lfp} \Phi\,.$$

In particular, our side conditions are simple in the sense that (a variation of) Φ needs to be applied to a candidate I only a *finite* number of times, which is beneficial for potential *automation*.

The need for verifying lower bounds on expected values is quite natural: First of all, they help to assess the quality and tightness of upper bounds. Moreover, giving *total correctness* guarantees for randomized algorithms amounts to lower-bounding the correctness probability, e.g., in order to establish membership in complexity classes like RP and PP.

In addition to expected values of random variables at termination, lower bounds on expected runtimes are also of significant interest: if such bounds depend on secret variables, they may compromise the secret, thus allowing for timing side-channel attacks; "very large" lower bounds could indicate potential denial-of-service attacks. In order to enable practicable reasoning about lower bounds on expected runtimes, we show how our inductive lower bound rule carries over to expected runtimes.

Our new inductive rules are stated in terms of so-called expectation transformers [MM05] and rely on the notions of *uniform integrability*, *martingales*, *conditional difference boundedness*, and the *Optional Stopping Theorem* from the theory of stochastic processes. However, we do not only *make use* of these notions in order to prove soundness of our induction rule, but instead establish tight connections in terms of these notions between expectation transformers and certain canonical stochastic processes. In particular, we build upon this connection to study exactly how inductive proof rules for both upper and lower bounds can be understood in the realm of these stochastic processes and vice versa. We see those connections between the theories of expectation transformers and stochastic processes as a stepping stone for applying further results from stochastic process theory to probabilistic loop analysis and possibly also vice versa.

As another contribution, we revisit one of the few existing rules for lower bounds due to [MM05], which gives *sufficient* criteria for a candidate being a lower bound

on the expected value of a *bounded* function f. We show that their rule is also a consequence of uniform integrability and we are moreover able to generalize their rule to a *necessary* and sufficient criterion.

Verifying the termination behavior of a probabilistic program itself is another crucial ingredient of probabilistic verification. Here, classes of programs where almost-sure termination[1] can be *decided* are of special interest since they allow for sound *and complete* methods to reason about their termination behavior. In Part II, we have already studied classes of non-probabilistic loops where (non-)termination is (semi-)decidable and upper bounds on the number of loop iterations can always be computed. Loops form the bottleneck of termination analysis in probabilistic verification as well. Clearly, in general almost-sure termination is undecidable and closed forms for the looping times of loops, i.e., the number of loop iterations, are not computable. Thus, decidability results can only be obtained for suitably restricted forms of loops.

We present a class of probabilistic loops where almost-sure termination is decidable (so-called *cpa-loops*) which includes many examples that are often regarded in the literature on probabilistic loops. So while other approaches are concerned with *incomplete* techniques to analyze termination and complexity, we investigate classes of probabilistic loops where one can *decide* almost-sure termination, *always* find complexity bounds, and even compute the expected looping time *exactly*. Our decision procedure could be integrated into general tools for termination and complexity analysis of probabilistic loops: As soon as one has to investigate a sub-loop that falls into our class, one can use the decision procedure to compute its exact looping time. Our contributions provide a starting point for such results.

However, to analyze (possibly large) probabilistic programs arising from real-world applications, fully automated approaches for analyzing termination or complexity are needed. There exist several approaches and tools for automatic complexity analysis of non-probabilistic programs, e.g., [Alb+11; Alb+19a; Bro+16; Car+17; Flo16; Fro+20; HAH12; HDW17; HS15; SSH17; SZV17]. While most of them rely on basic techniques like *ranking functions* (see, e.g., [Ali+10; BDG19; BG14; BG17; BMS05a; BMS05b; BMS05d; PR04]), they usually combine these basic techniques in sophisticated ways. For example, in [Bro+16] the authors developed a modular approach for automated complexity analysis of so-called non-probabilistic transition systems, based on an alternation between finding symbolic runtime bounds for program parts and using them to infer bounds on the sizes of variables in such parts. So each analysis step is restricted to a small part of the program. The implementation of this approach in KoAT [Bro+16] is one of the leading tools for complexity analysis [Gie+19b].

[1] This means termination with probability one.

While there exist several adaptions of basic techniques like ranking functions to *probabilistic programs* (see e.g., [ACN18; ALY20; BG05; BG06; CFN20; Cha+18; CNZ17; CS13; FC19; FH15; HFC18; Hua+19; Moo+21; Wan+19]), most of the sophisticated frameworks for complexity analysis have not been adapted to probabilistic programs yet, and there are only few powerful tools available which analyze the expected runtimes of probabilistic programs automatically [AMS20; NCH18; Wan+19; WKH20].

In this part, we adapt the modular approach for runtime analysis for non-probabilistic transition systems [Bro+16] to probabilistic transition systems. Moreover, our experimental evaluation shows that the implementation of this approach is especially well-suited for analyzing (possibly large) probabilistic transition systems which only contain few probabilistic parts. These systems might correspond to the typical application of randomization in real-world programs.

13.2 Structure of This Part

In Ch. 14, we formally introduce the syntax and semantics of so-called pGCL-programs. Here, we do not only recapitulate well-known semantics but also present our new results on connections between some of those semantics. These insights are crucial for our results in Ch. 15 and 16.

In Ch. 15, we present inductive proof rules for lower bounds on expected outcomes of probabilistic loops. To this end, we connect our results from Ch. 14 to certain concepts from probability theory. Additionally, we use our rule for lower bounds on expected outcomes to develop the first inductive proof rule for verifying lower bounds on the expected runtime[2] of probabilistic loops.

Ch. 16 is devoted to developing a class of probabilistic loops where almost-sure termination is decidable, and both upper *and* lower bounds on the expected looping time[3] can always be inferred. Based on these bounds we present the first algorithm to compute closed forms for the *exact* expected looping time of these loops, i.e., we show that the exact expected looping time is always computable.

In Ch. 17, we study so-called probabilistic transition systems (PITSs). We do not only introduce their syntax and semantics but also compare them to pGCL-programs as considered in Ch. 14 to 16. We discuss why for every pGCL-program (under some side conditions) their exists an equivalent PITS. Still, not every PITS is equivalent to some pGCL-program since PITSs allow for infinite non-deterministic sampling, whereas pGCL-programs do not. The main contribution of Ch. 17 then is

[2] This differs from the looping time since the runtime of the body of the loop is taken into account.
[3] The runtime of a loop from this class can be computed easily from the looping time.

an (incomplete) modular approach for inferring upper bounds on expected runtimes of PITSs.

We discuss work related to our results in Ch. 15 to 17 in Ch. 18. Finally, we conclude this part and also consider possible directions for future work in Ch. 19.

App. A contains a brief introduction into probability theory and all missing proofs can be found in App. B.

Chapter 14

Syntax and Semantics of Probabilistic Programs

In this chapter, we introduce probabilistic programs as used in this thesis. First of all, we use a probabilistic guarded command language à la Dijkstra's guarded commands (see [Dij75]) to represent probabilistic programs (Sect. 14.1). We then present operational semantics for probabilistic programs via Markov decision processes (Sect. 14.2) and denotational semantics via expectation transformers (Sect. 14.3). Finally, we focus on a special semantics for probabilistic *loops* which builds upon so-called stochastic processes (Sect. 14.4). However, we do not only introduce this semantics but also present our new results on its connection to expectation transformers (Thm. 14.4.25 and Cor. 14.4.26 to 14.4.28). These insights are so far only partially published.

14.1 Syntax of Probabilistic Programs via pGCL

In this section, we introduce a syntax for so-called *probabilistic* programs. A probabilistic program can be seen as a classical program in which program variables can also be assigned a value drawn from discrete probability distributions or a (possibly biased) coin can be flipped to determine the continuation of a program. We denote such probabilistic programs by using guarded commands. As usual, a *probability distribution* on $\mathbb{Z}$ is a mapping $pr : \mathbb{Z} \to \mathbb{R}$ with $pr(v) \in [0,1]$ for all $v \in \mathbb{Z}$ and $\sum_{v \in \mathbb{Z}} pr(v) = 1$.

Definition 14.1.1 (pGCL-Programs, [GKM14; Kam19; MM05]). Let $\mathcal{V}$ be a finite

P	$\longrightarrow$	**skip**	Effectless Program
	|	$x \leftarrow e$	Deterministic Assignment
	|	$x \leftsquigarrow D$	Probabilistic Assignment
	|	$\{\,\mathbf{P}\,\}\ [p]\ \{\,\mathbf{P}\,\}$	Probabilistic Choice
	|	$\{\,\mathbf{P}\,\} \,\square\, \{\,\mathbf{P}\,\}$	Non-Deterministic Choice
	|	**if** $(\,\varphi\,)$ **then** $\{\,\mathbf{P}\,\}$ **else** $\{\,\mathbf{P}\,\}$	Conditional Statement
	|	$\mathbf{P} \fatsemi \mathbf{P}$	Sequential Composition
	|	**while** $(\,\varphi\,)$ **do** $\{\,\mathbf{P}\,\}$	Guarded While Loop

Table 14.1: Grammar for pGCL

set of variables and let $\mathrm{Expr}(\mathcal{V})$ be the set of all arithmetic expressions over the variables $\mathcal{V}$. The set of *states* is the set $\Sigma = \{s \mid s : \mathcal{V} \to \mathbb{Z}\}$.

The set of pGCL-*programs* over $\mathcal{V}$ is the set of programs generated by the *probabilistic guarded command language* (pGCL) described as a grammar in Table 14.1. Here, $x \in \mathcal{V}$, $e \in \mathrm{Expr}(\mathcal{V})$, $p \in \mathrm{Expr}(\mathcal{V})$ describes a probability (i.e., $s(p) \in [0,1]$ for all states s where we take into account that s might be restricted via conditions), D is a *distribution function*, i.e., $D(s)$ is a probability distribution over $\mathbb{Z}$ for all $s \in \Sigma$, and φ is a Boolean formula over $\mathrm{Expr}(\mathcal{V})$. If $x \in \mathcal{V}$ is assigned $e \in \mathrm{Expr}(\mathcal{V})$ in a deterministic assignment, we require $s(e) \in \mathbb{Z}$ for all $s \in \Sigma$.

A program without non-deterministic choices is *purely probabilistic*.

Purely probabilistic programs are also referred to as "tame" (see, e.g., [Kam19]). Clearly, probabilistic choice is just syntactic sugar and can be modeled by using a probabilistic assignment and a conditional statement. Nevertheless, it is usually shorter to use this construct directly instead of the mentioned workaround.

Note that the syntax for polynomial loops from Def. 4.1.1 covered loops over some ring $\mathcal{S} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}_{\mathbb{A}}\}$ whose condition is a Boolean combination of polynomial inequations and whose body consists of a single update of the vector of variables by a vector of polynomials. In contrast, Def. 14.1.1 covers arbitrary pGCL-programs featuring nested or consequential loops, non-determinism, as well as discrete probability distributions where the variables range over the integers. Still, every polynomial loop over the integers can of course be modeled using pGCL.

while $(h \leq t)$ **do {**
$\quad \{\, h \leftsquigarrow h + \mathrm{Unif}[0 \ldots 10] \,\} \; \left[\tfrac{1}{2}\right] \; \{\, \mathbf{skip} \,\} \,;$
$\quad t \leftarrow t + 1$
}

Figure 14.1: Race Between Tortoise and Hare

$$\{\, x \leftarrow 1 \,\} \,\square\, \Big\{\, \mathbf{while}\,(\,\mathrm{true}\,)\,\mathbf{do}\,\{\,\mathbf{skip}\,\}\,\Big\}$$

Figure 14.2: Program With Bounded Non-Determinism

Example 14.1.2. Consider the well-known program which models the race between a tortoise and a hare (see, e.g., [CS13; Kam+18; NCH18]) which is depicted in pGCL-syntax in Fig. 14.1. As long as the hare (variable h) is not in front of the tortoise (variable t) the tortoise does one step in each iteration. With probability $\frac{1}{2}$, the hare stays at its position and with probability $\frac{1}{2}$ it does a random number of steps uniformly chosen between 0 and 10, i.e., $h + \mathrm{Unif}[0 \ldots 10]$ denotes the distribution function where when evaluated in state s the values $s(h) + 0, \ldots, s(h) + 10$ all have probability $\frac{1}{11}$. The race ends when the hare is in front of the tortoise. This program is purely probabilistic.

In a probabilistic choice, the first argument is executed with probability p, while the second argument is executed with probability $1 - p$. In contrast, in a non-deterministic choice we only know that *one* of both statements is executed. This notion of non-determinism reflects so-called *bounded* or *finite* non-determinism.

Example 14.1.3. The program depicted in Fig. 14.2 contains bounded non-determinism. Either the variable x is set to 1 or an infinite loop is executed. We have no information which of both statements is executed, while in a probabilistic choice we have the information with which probability the statements are executed. Thus, in this part "non-determinism" refers to "presence of non-deterministic choices".

Intuitively, given some initial state s a purely probabilistic pGCL-program $\mathbf{P}$ computes a (sub)*distribution* over output states. So while the computation of a de-

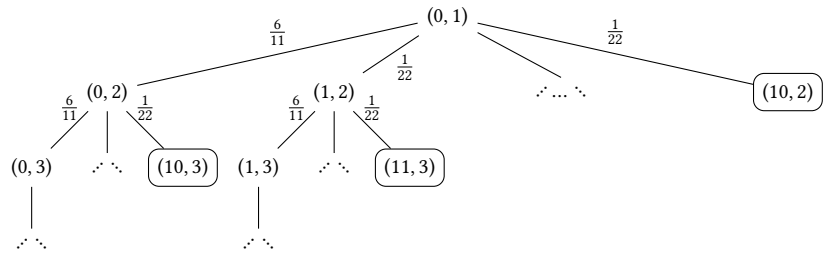


Figure 14.3: Excerpt of Computation Tree of Fig. 14.1 on Input $(0, 1)$

terministic program (i.e., without probabilistic or non-deterministic behavior) can be graphically captured by a (possibly infinite) path, the computation of a purely probabilistic pGCL-program has to be captured by a tree of possibly infinite depth.

Example 14.1.4. Consider again the race between tortoise and hare from Ex. 14.1.2. Here, a program state can be depicted by a tuple $(h, t) \in \mathbb{Z}^2$ where the first entry is the value assigned to the variable h and the second entry is the value assigned to the variable t. Fig. 14.3 depicts an intuitive visualization of the program in Fig. 14.2 started in the initial state $(0, 1)$. Here, every level of the tree reflects the state after one execution of the body of the loop. The edges in the tree are labeled with the probabilities to get from one state to the other. So for example, the probability to end in state $(0, 2)$ after the first loop iteration is $\frac{6}{11}$: **skip** is executed with probability $\frac{1}{2}$, with probability $\frac{1}{2} \cdot \frac{1}{11}$ the value 0 is drawn from the uniform distribution and $\frac{1}{2} + \frac{1}{2} \cdot \frac{1}{11} = \frac{6}{11}$. Moreover, the probability of a path is the product of the probabilities of its edges, e.g., the probability to end in state $(1, 3)$ after two executions of the loop body is $\frac{1}{22} \cdot \frac{6}{11} = \frac{3}{121}$.

The boxed states mark *final* states, i.e., they violate the condition of the loop and thus, the program terminates when reaching such a state.

Of course, the tree in Fig. 14.3 explained in Ex. 14.1.4 is only an *intuitive* visualization which does not define a proper operational semantics of a pGCL-program. For example, this sketch cannot be generalized to sequential loops. Nevertheless, there is indeed an operational semantics via computation-trees for (arbitrary) pGCL-programs (see, e.g., [Kam19]). However, we will restrict ourselves to probabilistic loops later on so that the idea of "i-th level corresponds to i-th iteration" is sufficient for our needs.

Still, we will introduce a formal operational semantics via so-called Markov decision processes for pGCL-programs to define the different notions of termination in the next section. As said in [Kam19], the operational semantics via Markov-decision processes is "potentially more compact" than the computation tree semantics, e.g., there are cases where the MDP semantics yields a finite operational structure, whereas the computation tree semantics yields an infinite one (see, e.g., [Kam19]).

14.2 An Operational Semantics for pGCL-Programs

In this section, we introduce an operational semantics for pGCL-programs based on so-called *Markov decision processes*, MDPs for short. MDPs are widely used in probabilistic verification. We introduce them based on [BK08; Put94]. If not explicitly stated otherwise, all concepts related to MDPs are taken from these sources.

Before we start with the main part, we introduce regular expressions of sets since they ease readability when dealing with (in)finite sequences of states.

Definition 14.2.1 (Regular Expressions of Sets)**.** Let A and B be some sets and $n \in \mathbb{N}$. We then define the following notation for *regular expressions of sets*

- $AB = \{ab \mid a \in A, b \in B\}$
- $A^0 = \{\varepsilon\}$, $A^{n+1} = AA^n$
- $A^* = \bigcup_{n \in \mathbb{N}} A^n$
- $A^+ = \bigcup_{n \in \mathbb{N}_{\geq 1}} A^n$
- $A^\omega = \{f \mid f : \mathbb{N} \to A\}$.

Here, ε is the empty sequence with $\varepsilon\vartheta = \vartheta$ for all sequences ϑ with arbitrary entries, and $\vartheta\varepsilon = \vartheta$ for all finite sequences with arbitrary entries.

If A is a singleton, we omit the braces, e.g., we write a^ω instead of $\{a\}^\omega$.

Example 14.2.2. Consider the set $A = \{(1,2),(1,0)\} \subseteq \mathbb{Z}^2$. For example, $\varepsilon \in A^0$, $(1,2)(1,0)(1,0)(1,2) \in A^4$, and $(1,0)(1,0)(1,0)(1,0)\cdots \in A^\omega$.

The semantics of pGCL-programs we present here is taken from [GKM14; Kam+18; Kam19] and adapted to the syntax which we defined in the last section.

Definition 14.2.3 (Markov Decision Processes)**.** A *(discrete time) Markov decision process (MDP)* $\mathcal{M} = (\mathcal{S}, \mathfrak{s}_0, \mathcal{A}, \mathcal{P})$ consists of

- a countable set of states $\mathcal{S}$,
- an initial state $\mathfrak{s}_0 \in \mathcal{S}$,
- a countable set of actions $\mathcal{A}$,
- a transition probability function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ such that for all states $\mathfrak{s} \in \mathcal{S}$ and all actions $\mathfrak{a} \in \mathcal{A}$ we have $\sum_{\mathfrak{s}' \in \mathcal{S}} \mathcal{P}(\mathfrak{s}, \mathfrak{a}, \mathfrak{s}') = 1$.

A (discrete time) *Markov chain (DTMC)* is an MDP where $\mathcal{A}$ is a singleton. In a Markov chain, we omit the actions, i.e., we write $\mathcal{M} = (\mathcal{S}, \mathfrak{s}_0, \mathcal{P})$ and $\mathcal{P}(\mathfrak{s}, \mathfrak{s}')$ instead of $\mathcal{P}(\mathfrak{s}, \mathfrak{a}, \mathfrak{s}')$ where $\mathfrak{a}$ is the unique action.

We now define the operational semantics of a pGCL-program $\mathbf{P}$ via MDPs. To avoid confusion, we denote states of programs, i.e., variable assignments, by a lowercase s in italics, i.e., s, and states of MDPs by a calligraphic lowercase s, i.e., $\mathfrak{s}$.

Definition 14.2.4 (MDP Semantics for pGCL-Programs)**.** Let $\mathbf{P}$ be a pGCL-program and $s_0 \in \Sigma$. The *operational MDP* of $\mathbf{P}$ started in s_0 denoted by $\mathcal{M}^{\mathbf{P}}_{s_0} = (\mathcal{S}, \mathfrak{s}_0, \mathcal{A}, \mathcal{P})$ is given by

- set of states $\mathcal{S} = \left\{ (\mathbf{P}', s) \;\middle|\; \mathbf{P}' \text{ a pGCL-program or } \mathbf{P}' = \bot, s \in \Sigma \right\}$
- initial state $\mathfrak{s}_0 = (\mathbf{P}, s_0)$
- set of actions $\mathcal{A} = \{L, N, R\}$
- transition probability function $\mathcal{P}$ defined by the inference rules in Table 14.2. Here, the notation $s[x \mapsto z]$ refers to the state s' such that $s'(x) = z$ and $s'(y) = s(y)$ for all $x \neq y \in \mathcal{V}$. Moreover, we indicate that the probabilities in a probabilistic choice or the distributions in a probabilistic assignment might depend on the values of variables in the program.

A state of the form $(\bot, s) \in \mathcal{S}$ is called *final*.

So the states in this MDP correspond to "commands still to be executed" and the current state of the program, i.e., the assignment of the program variables to integers. Moreover, loops are unfolded until the values the program variables take violate the

α	$\mathfrak{z} \xrightarrow{\mathcal{P}(\mathfrak{z},\alpha,\mathfrak{z}')} \mathfrak{z}'$
N	$(\bot, s) \xrightarrow{1} (\bot, s)$
N	$(\mathbf{skip}, s) \xrightarrow{1} (\bot, s)$
N	$(x \leftarrow e, s) \xrightarrow{1} (\bot, s[x \mapsto s(e)])$
N	$\dfrac{(D(s))(z) = p}{(x \leftsquigarrow D, s) \xrightarrow{p} (\bot, s[x \mapsto z])}$
N	$(\{\, \mathbf{P}_1 \,\}\ [p]\ \{\, \mathbf{P}_2 \,\}, s) \xrightarrow{s(p)} (\mathbf{P}_1, s)$
N	$(\{\, \mathbf{P}_1 \,\}\ [p]\ \{\, \mathbf{P}_2 \,\}, s) \xrightarrow{1-s(p)} (\mathbf{P}_2, s)$
L	$(\{\, \mathbf{P}_1 \,\} \,\square\, \{\, \mathbf{P}_2 \,\}, s) \xrightarrow{1} (\mathbf{P}_1, s)$
R	$(\{\, \mathbf{P}_1 \,\} \,\square\, \{\, \mathbf{P}_2 \,\}, s) \xrightarrow{1} (\mathbf{P}_2, s)$
N	$\dfrac{s \vDash \varphi}{(\mathbf{if}\ (\varphi)\ \mathbf{then}\ \{\, \mathbf{P}_1 \,\}\ \mathbf{else}\ \{\, \mathbf{P}_2 \,\}, s) \xrightarrow{1} (\mathbf{P}_1, s)}$
N	$\dfrac{s \nvDash \varphi}{(\mathbf{if}\ (\varphi)\ \mathbf{then}\ \{\, \mathbf{P}_1 \,\}\ \mathbf{else}\ \{\, \mathbf{P}_2 \,\}, s) \xrightarrow{1} (\mathbf{P}_2, s)}$
N	$(\bot \fatsemi \mathbf{P}, s) \xrightarrow{1} (\mathbf{P}, s)$
N	$\dfrac{(\mathbf{P}_1, s_1) \xrightarrow{p} (\mathbf{P}_2, s_2)}{(\mathbf{P}_1 \fatsemi \mathbf{P}_3, s) \xrightarrow{p} (\mathbf{P}_2 \fatsemi \mathbf{P}_3, s_2)}$
N	$\dfrac{s \vDash \varphi}{(\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\, \mathbf{P} \,\}, s) \xrightarrow{1} (\mathbf{P} \fatsemi \mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\, \mathbf{P} \,\}, s)}$
N	$\dfrac{s \nvDash \varphi}{(\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\, \mathbf{P} \,\}, s) \xrightarrow{1} (\bot, s)}$

Table 14.2: Inference Rules for the Transition Probability Function $\mathcal{P}$ of $\mathcal{M}^{\mathbf{P}}_{s_0}$

condition. Thus, the number of states reachable (with a positive probability) from the initial state easily becomes infinite when unbounded loops are present. The actions are used to model non-determinism: N is the default action and L and R indicate which statement in a non-deterministic choice is to be executed. The final states correspond to termination since no statements are left to be executed.

Example 14.2.5. Consider again the pGCL-program $\mathbf{P}$ of the race of tortoise and hare from Fig. 14.1 and Ex. 14.1.2. An excerpt of its MDP with initial program state $s_0 = (0, 1)$ is given in Fig. 14.4. A final state is indicated by a box with rounded corners and grey background. Here, we abbreviate the program $\mathbf{P}$ to **while** $(h \leq t)$ **do** $\{ \dots \}$ in all states besides the initial state to increase readability. Moreover, we left out states of the form $(\bot \fatsemi \mathbf{P}, s)$ because from such a state we continue to the state $(\mathbf{P}, s)$ with probability 1, i.e., instead of $(\bot \fatsemi \mathbf{P}, s)$ we directly use $(\mathbf{P}, s)$ to ease readability. Since $\mathbf{P}$ does not contain non-deterministic choices, this MDP has only the single action N, i.e., it is a DTMC. The initial state of this DTMC consists of the program in Fig. 14.1 and the initial program state s_0. Every state of this DTMC contains as its first entry the commands still to be executed to reach termination. For example, the state $(\bot, s = (10, 2))$ is a final state. Comparing this with the computation tree in Fig. 14.3 we get the following: The probability to reach the program state $s = (0, 2)$ after a single loop iteration, i.e., to reach the state in the bottom left, is $1 \cdot \frac{1}{2} \cdot 1 \cdot 1 + 1 \cdot \frac{1}{2} \cdot \frac{1}{11} \cdot 1 = \frac{6}{11}$ (see, Ex. 14.1.4). Here, we multiplied the probabilities along the two paths leading from the initial state to this state. These are the only paths to this state.

Moreover, the probability to terminate with program state $s = (10, 2)$ is at least $1 \cdot \frac{1}{2} \cdot \frac{1}{11} \cdot 1 \cdot 1 = \frac{1}{22}$ as there might be paths from the initial state to the corresponding final state using some of the infinitely many non-illustrated DTMC states.

However, in the presence of non-deterministic choices the operational MDP has indeed multiple actions.

Example 14.2.6. As another example, consider the program $\mathbf{P}_1$ in Fig. 14.2. Since this program has only the program variable x, we encode states by the integer which is assigned to x. Its operational MDP with initial program state $s = x_0 \in \mathbb{Z}$ is given in Fig. 14.5. As in Ex. 14.2.5, we leave out states of the form $(\bot \fatsemi \mathbf{P}, s)$. The unique final state is indicated by a box with rounded corners and grey background. This MDP is *not* a DTMC as its set of actions $\{L, N, R\}$ is not a singleton.

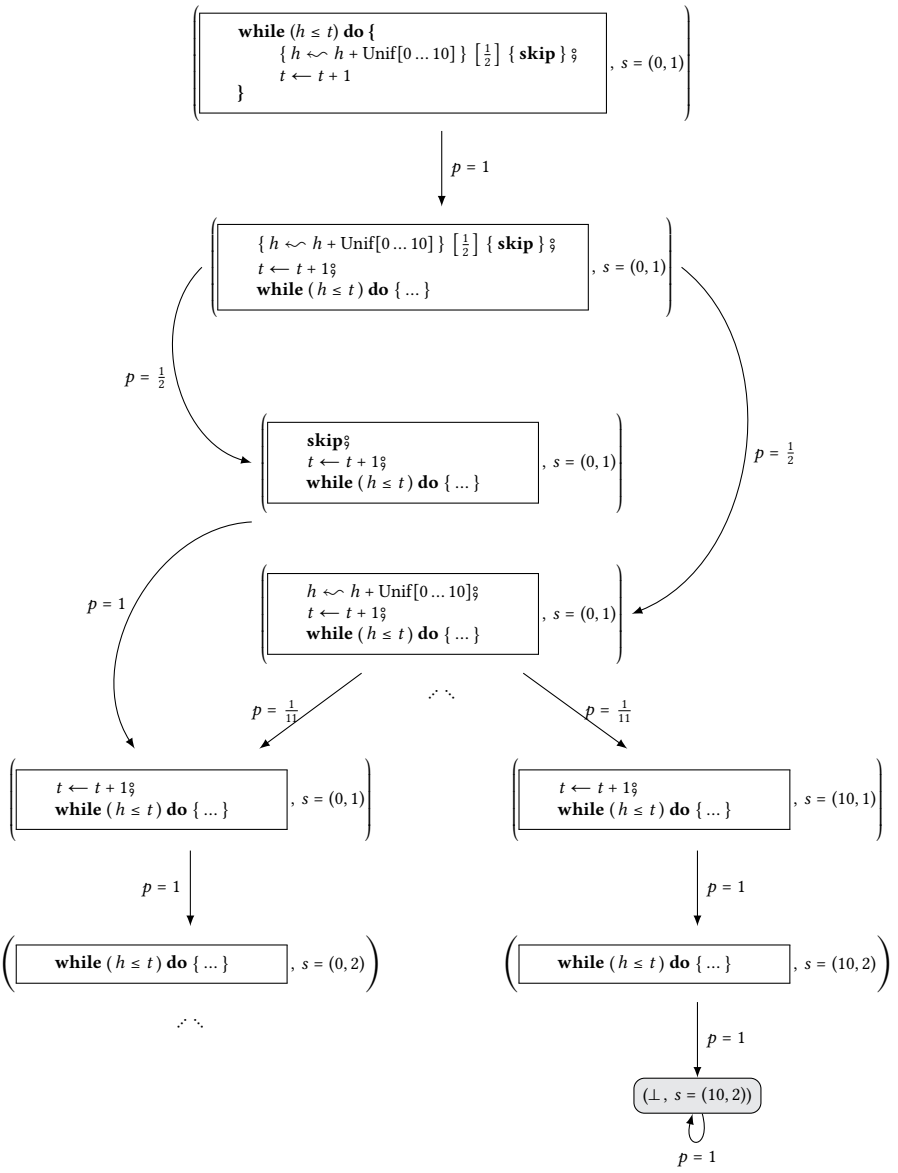


Figure 14.4: Excerpt of MDP for Fig. 14.1

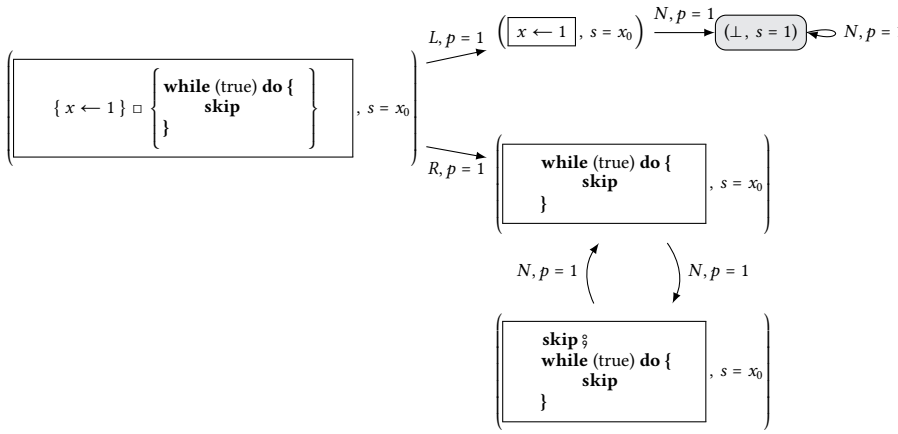


Figure 14.5: MDP for Fig. 14.2

To define the different kinds of termination properly, we first need to explain how to resolve the non-determinism in an MDP. This is done by so-called *schedulers*.

Definition 14.2.7 (Scheduler)**.** Let $\mathcal{M} = (S, \mathfrak{d}_0, \mathcal{A}, \mathcal{P})$ be an MDP. A (deterministic positional) *scheduler* for $\mathcal{M}$ is a mapping $\mathfrak{s} : S \longrightarrow \mathcal{A}$. We denote the set of all schedulers for $\mathcal{M}$ by $\text{Scheds}_{\mathcal{M}}$, where we omit the subscript whenever it is clear from the context.

The Markov chain of $\mathcal{M}$ induced by $\mathfrak{s}$ is $\mathcal{M}_{\mathfrak{s}} = (S, \mathfrak{d}_0, \mathcal{P}_{\mathfrak{s}})$ where for every two states $s, s' \in S$ we have $\mathcal{P}_{\mathfrak{s}}(s, s') = \mathcal{P}(s, \mathfrak{s}(s), s')$.

If $\mathbf{P}$ is a pGCL-program, a scheduler $\mathfrak{s}$ for $\mathbf{P}$ is a scheduler of $\mathcal{M}^{\mathbf{P}}_{s_0}$ where $s_0 \in \Sigma$. The set of schedulers for $\mathbf{P}$ is denoted by $\text{Scheds}_{\mathbf{P}}$. Again, we omit the subscript whenever it is clear from the context.

Thus, by using schedulers we can reduce an MDP to a DTMC. Moreover, note that the initial state of an MDP does not affect the set of schedulers for this MDP since only the set of states is taken into account. This justifies why a scheduler $\mathfrak{s}$ for a pGCL-program $\mathbf{P}$ is independent of the chosen initial program state s_0 since the set of states of $\mathcal{M}^{\mathbf{P}}_{s_0}$ is always the same.

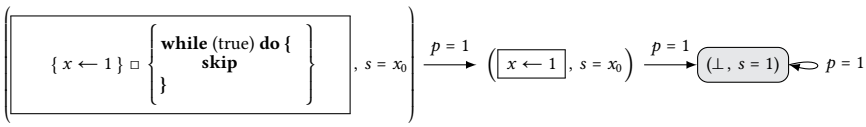


Figure 14.6: DTMC Resulting from Fig. 14.5 and the Scheduler in Ex. 14.2.8

Example 14.2.8. Consider again the operational MDP $\mathcal{M}$ in Fig. 14.5 resulting from the program $\mathbf{P}_1$ in Fig. 14.2. For example, a scheduler of $\mathcal{M}$ (and thus the program) is given by $\mathfrak{s} : S \to \mathcal{A}$ where $\mathfrak{s}(\mathfrak{z}) = L$, if $\mathfrak{z}$ is the initial state of $\mathcal{M}$, and $\mathfrak{s}(\mathfrak{z}) = N$ otherwise, i.e., this scheduler always chooses the left statement in the program in Fig. 14.2. The DTMC $\mathcal{M}_{\mathfrak{s}}$ induced by the scheduler $\mathfrak{s}$ is shown in Fig. 14.6.

There are many different concepts of schedulers which even allow to resolve the non-determinism via a probabilistic choice or to take histories of states into account, i.e., in a state $\mathfrak{z}$ the chosen action may depend on which states have been visited before $\mathfrak{z}$ (see, e.g.,[BK08; Put94]). In the context of pGCL-programs, we use schedulers to define *expected total rewards* of non-negative functions for MDPs with countable state space and countably many actions. In this case, it suffices to only consider deterministic positional schedulers (see [Put94, Prop. 6.2.1, Thm. 7.1.3]) as defined in Def. 14.2.7. So here, a scheduler only takes the current state $\mathfrak{z}$ of the MDP into account.

Before we continue, we introduce the so-called *Iverson bracket*.

Definition 14.2.9 (Iverson Bracket, [Knu92])**.** Let φ be a Boolean formula. The *Iverson bracket* $[\varphi] : \Sigma \to \mathbb{R}_{\geq 0}$ is a mapping such that $[\varphi](s) = 1$ if $s \vDash \varphi$ and $[\varphi](s) = 0$ otherwise. For some set $M \subseteq U$ the Iverson bracket $[M] : U \to \mathbb{R}_{\geq 0}$ is the map defined by $[M](u) = 1$ if $u \in M$ and $[M](u) = 0$ if $M \notin U$ for all $u \in U$.

Using Def. 14.2.7, we can work with DTMCs from now on and lift the definitions to MDPs using schedulers. Since an DTMC does not contain any non-determinism we can define so-called *expected total rewards* for non-negative reward functions.

Definition 14.2.10 (Expected Total Reward)**.** A *reward function* for an MDP with state space S is a function $rew : S \to \overline{\mathbb{R}}_{\geq 0}$ where $\overline{\mathbb{R}}_{\geq 0} = \{r \mid r \in [0, \infty) \vee r = \infty\}$.

» Let $\mathcal{M} = (S, \mathfrak{d}_0, \mathcal{P})$ be a Markov chain with reward function rew and let $T \subseteq S$. The *cumulative reward* of a finite sequence of states $\mathfrak{d}_1 \cdots \mathfrak{d}_n = \pi \in S^*$ is $rew(\pi) = \sum_{k=1}^{n} rew(\mathfrak{d}_k)$. Moreover, we extend $\mathcal{P}$ to paths via $\mathcal{P}(\pi) = \prod_{k=1}^{n-1} \mathcal{P}(\mathfrak{d}_k, \mathfrak{d}_{k+1})$. The *expected total reward of eventually reaching* T denoted by $\mathrm{ExpRew}_{\mathcal{M}}(\diamond T)$ is defined as

$$\mathrm{ExpRew}_{\mathcal{M}}(\diamond T) = \sum_{\pi \in \mathfrak{d}_0 (S \setminus T)^* T} P(\pi) \cdot rew(\pi).$$

If $rew = [T]$, we write $\mathrm{ExpRew}_{\mathcal{M}}(\diamond T) = \Pr_{\mathcal{M}}(\diamond T)$ and call this the *probability of eventually reaching* T.

» Let $\mathcal{M} = (S, \mathfrak{d}_0, \mathcal{A}, \mathcal{P})$ be an MDP with reward function rew. The *maximal expected total reward of eventually reaching* T is defined as

$$\mathrm{ExpRew}^{\max}_{\mathcal{M}}(\diamond T) = \sup_{\mathfrak{s} \in \mathsf{Scheds}} \mathrm{ExpRew}_{\mathcal{M}_{\mathfrak{s}}}(\diamond T).$$

The *minimal expected total reward of eventually reaching* T is defined as

$$\mathrm{ExpRew}^{\min}_{\mathcal{M}}(\diamond T) = \inf_{\mathfrak{s} \in \mathsf{Scheds}} \mathrm{ExpRew}_{\mathcal{M}_{\mathfrak{s}}}(\diamond T).$$

Example 14.2.11. Reconsider the MDP $\mathcal{M}$ in Fig. 14.5 and its DTMC $\mathcal{M}_{\mathfrak{s}}$ in Fig. 14.6 induced by the scheduler $\mathfrak{s}$ from Ex. 14.2.8. Let T be the singleton set containing the initial state and let $rew(\mathfrak{d}) = \max\{s(x), 0\}$ for all states $\mathfrak{d} = (\mathbf{P}, s)$.

Then the expected total reward is $\mathrm{ExpRew}_{\mathcal{M}_{\mathfrak{s}}}(\diamond T) = 2 \cdot \max\{x_0, 0\} + 1$: there is a unique path to T which consists of all states. This path has reward $\max\{x_0, 0\} + \max\{x_0, 0\} + 1$. The self loop has no effect since only finite paths which end in T but never visit T before contribute to $\mathrm{ExpRew}_{\mathcal{M}_{\mathfrak{s}}}(\diamond T)$.

Thus, $\mathrm{ExpRew}^{\max}_{\mathcal{M}}(\diamond T) \geq 2 \cdot \max\{x_0, 0\} + 1$. Indeed, $\mathrm{ExpRew}^{\max}_{\mathcal{M}}(\diamond T) = 2 \cdot \max\{x_0, 0\} + 1$ since if a scheduler $\mathfrak{s}'$ chooses the action R in the initial state of the MDP, there is *no* path with a positive probability which eventually reaches T, i.e., $\mathrm{ExpRew}_{\mathcal{M}_{\mathfrak{s}'}}(\diamond T) = 0$. However, this then yields $\mathrm{ExpRew}^{\min}_{\mathcal{M}}(\diamond T) = 0$.

$$\{\, x \leftarrow 1 \,\} \left[\tfrac{1}{2}\right] \left\{\, \textbf{while}\,(\,\text{true}\,)\,\textbf{do}\,\{\,\textbf{skip}\,\}\,\right\}$$

Figure 14.7: $\mathbf{P}_2$ from Ex. 14.2.13

Using Def. 14.2.10 we can now define output distributions, the probability of termination, and expected runtimes for pGCL-programs.

Definition 14.2.12 (Almost-Sure Termination)**.** Let $\mathbf{P}$ be some pGCL-program with initial program state $s_0 \in \Sigma$ and scheduler $\mathfrak{s}$. The (sub)distribution $\mu^{\mathbf{P}}_{s_0,\mathfrak{s}} : \Sigma \longrightarrow [0,1]$ is defined by

$$\mu^{\mathbf{P}}_{s_0,\mathfrak{s}}(s) = \Pr_{\mathcal{M}^{\mathbf{P}}_{s_0,\mathfrak{s}}}(\diamond\{(\bot, s)\})\,.$$

We call $\left|\mu^{\mathbf{P}}_{s_0,\mathfrak{s}}\right| = \sum_{s\in\Sigma} \mu^{\mathbf{P}}_{s_0,\mathfrak{s}}(s) = \int_\Sigma 1 d\left(\mu^{\mathbf{P}}_{s_0,\mathfrak{s}}\right)$ the *probability of termination* of $\mathbf{P}$ on s_0 and $\mathfrak{s}$. The remaining probability, i.e., $1 - \left|\mu^{\mathbf{P}}_{s_0,\mathfrak{s}}\right|$, is the *probability of non-termination* of $\mathbf{P}$ on s_0 and $\mathfrak{s}$. If $\mathbf{P}$ is purely probabilistic, we omit $\mathfrak{s}$.

If for every $s_0 \in \Sigma$ and $\mathfrak{s} \in \text{Scheds}$ we have $\left|\mu^{\mathbf{P}}_{s_0,\mathfrak{s}}\right| = 1$, $\mathbf{P}$ is called *almost-surely terminating* (AST).

Note that $\mu^{\mathbf{P}}_{\bullet,\bullet} : \Sigma \times \text{Scheds} \longrightarrow \Sigma \longrightarrow [0,1]$. Here, $\mu^{\mathbf{P}}_{s_0,\mathfrak{s}}$ is only a subdistribution since $\left|\mu^{\mathbf{P}}_{s_0,\mathfrak{s}}\right| \leq 1$ as the program $\mathbf{P}$ might not be AST. Of course, $\mu^{\mathbf{P}}_{s_0,\mathfrak{s}}$ is a *distribution* for every $s_0 \in \Sigma$ and $\mathfrak{s} \in \text{Scheds}$ iff $\mathbf{P}$ is AST. Additionally, the term "almost-surely" will become clear in Sect. 15.2 (see Def. 15.2.2).

Example 14.2.13. Consider the programs $\mathbf{P}_1$ in Fig. 14.2 and $\mathbf{P}_2$ in Fig. 14.7 with the single program variable x.

Let $\mathfrak{s}$ be the scheduler for $\mathbf{P}_1$ which always chooses the left statement (see Ex. 14.2.8) and let $\mathfrak{s}'$ be the scheduler which always chooses the right statement in the non-deterministic choice. We have for any initial state s_0

$$\mu^{\mathbf{P}_1}_{s_0,\mathfrak{s}}(s) = \begin{cases} 1, & s(x) = 1 \\ 0, & \text{otherwise,} \end{cases}$$

as in the DTMC induced by $\mathfrak{s}$ (see Fig. 14.6) the only final state is $(\bot, s = 1)$ which is reached with probability 1. Since the right statement in the non-deterministic choice in $\mathbf{P}_1$ never reaches a final state we have

$$\mu^{\mathbf{P}_1}_{s_0,\mathfrak{s}'}(s) = 0,$$

i.e., $\mathbf{P}_1$ is not AST.

In contrast, for $\mathbf{P}_2$ we have for any initial state s_0

$$\mu^{\mathbf{P}_2}_{s_0}(s) = \begin{cases} \frac{1}{2}, & s(x) = 1 \\ 0, & \text{otherwise} \end{cases}$$

since the only final state in the underlying DTMC which is reachable from the initial state is $(\bot, s)$, where $s(x) = 1$, and it is reached with probability $\frac{1}{2}$. Note that $\mathbf{P}_2$ is purely probabilistic and thus, its operational MDP is a DTMC. Moreover, this DTMC is similar to the MDP in Fig. 14.5 where the transitions of the actions L and R are replaced by transitions of the action N which both have probability $\frac{1}{2}$.

Hence, $\mathbf{P}_2$ is also not AST since $\left|\mu^{\mathbf{P}_1}_{s_0}\right| = \frac{1}{2} < 1$ for any $s_0 \in \Sigma$.

However, a program can be AST but this does not give an idea on how long it takes to reach termination. Therefore, we introduce the notion of *expected runtime*.

We define the expected runtime of a pGCL-program via the notion of expected total reward, too. Here, we make use of a *post-runtime* $t : \Sigma \to \overline{\mathbb{R}}_{\geq 0}$, i.e., we define the expected runtime of a pGCL-program $\mathbf{P}$ when assuming that after termination of $\mathbf{P}$ there remains time t. This will later enable compositionality of the so-called expected runtime transformer ert (see Def. 14.3.8).

Definition 14.2.14 (Expected Runtime of a pGCL-Program)**.** Let $\mathbf{P}$ be a pGCL-program with initial state s_0, let $\mathfrak{s}$ be a scheduler for $\mathbf{P}$, and let $t : \Sigma \to \overline{\mathbb{R}}_{\geq 0}$ be a post-runtime. Let rew_t be the reward function for $\mathcal{M}^{\mathbf{P}}_{s_0}$ defined in Table 14.3.

» The *expected runtime of* $\mathbf{P}$ *in* s_0 *for* $\mathfrak{s}$ *followed by time* t is

$$\operatorname{ert}\,[\![\mathbf{P}]\!]\,(t)\,(s_0)^{\mathfrak{s}} = \operatorname{ExpRew}_{\mathcal{M}^{\mathbf{P}}_{s_0,\mathfrak{s}}}(\diamond\{(\bot, s) \mid s \in \Sigma\}).$$

$\mathfrak{d}$	$rew_t(\mathfrak{d})$
$(\bot, s)$	$t(s)$
$(\textbf{skip}, s), (x \leftarrow e, s), (x \leftsquigarrow D, s), (\textbf{if } (\varphi) \textbf{ then } \{ \mathbf{P}_1 \} \textbf{ else } \{ \mathbf{P}_2 \}, s),$ $(\textbf{while } (\varphi) \textbf{ do } \{ \mathbf{P} \}, s), (\{ \mathbf{P}_1 \} [p] \{ \mathbf{P}_2 \}, s)$	1
$(\bot \fatsemi \mathbf{P}, s), (\{ \mathbf{P}_1 \} \square \{ \mathbf{P}_2 \}, s)$	0
$(\mathbf{P}_1 \fatsemi \mathbf{P}_2, s)$	$rew((\mathbf{P}_1, s))$

Table 14.3: Reward Function for Time t

» The (worst-case) *expected runtime of* $\mathbf{P}$ *in* s_0 *followed by time* t is

$$\operatorname{ert}\ [\![\mathbf{P}]\!]\ (t)\,(s_0) = \operatorname{ExpRew}^{\max}_{\mathcal{M}^{\mathbf{P}}_{s_0}}\ (\diamond \{(\bot, s) \mid s \in \Sigma\})\,.$$

» We define the (worst-case) *expected runtime of* $\mathbf{P}$ *in* s_0 by

$$\operatorname{exprt}\ [\![\mathbf{P}]\!]\,(s_0) = \operatorname{ert}\ [\![\mathbf{P}]\!]\ (0)\,(s_0),$$

where 0 is the function mapping every state to 0.

If $\operatorname{ert}\ [\![\mathbf{P}]\!]\ (0)\,(s_0)^{\mathfrak{s}} < \infty$ for every $s_0 \in \Sigma$ and every $\mathfrak{s} \in \text{Scheds}$, then $\mathbf{P}$ is *positive almost-surely terminating* (PAST).

Here, the expected runtime resolves any non-deterministic choice in a worst-case manner, i.e., by taking the supremum over all schedulers. Furthermore, note that we count the runtime of a final state only *once* (i.e., the self loops in the final states $\mathcal{M}^{\mathbf{P}}_{s_0}$ do not have any effect) since we only consider finite paths which end in a final state but do not contain any other final state in Def. 14.2.10.

Here, we chose the reward function in Table 14.3 according to [Kam+18] where basically every statement is counted to consume one time-step besides a non-deterministic choice which is just a virtual construct: in any program run it is resolved by a scheduler. Nonetheless, a probabilistic choice is counted since it contains a "coin-flip". There are of course multiple ways to encode a pGCL-program. For example, probabilistic choices can be modeled also via probabilistic assignments. This then leads to different values for the expected runtime but does not change the asymptotic behavior of the expected runtime. For example, in Ch. 16 we show how to

$y \leftarrow 0 \,⨟$
$x \leftarrow 0 \,⨟$
$a \leftarrow 1 \,⨟$
while $(a = 1)$ **do {**
$\quad \big\{ \{ x \leftarrow x + 1 \} \, [\tfrac{1}{2}] \, \{ a \leftarrow 0 \} \big\} \square \{ y \leftarrow 2^x \cdot x \,⨟ a \leftarrow 0 \}$
$\} \,⨟$
while $(y > 0)$ **do {**
$\quad y \leftarrow y - 1$
}

Figure 14.8: Program $\mathbf{P}_{pns}$

compute the exact expected runtime of so-called constant probability addition loops. However, "runtime" in that case refers to the number of loop iterations which would result in a slightly different version of Table 14.3 where only the states of the form $(\textbf{while}\ (\varphi)\ \textbf{do}\ \{\,\mathbf{P}\,\}, s)$ are counted. We will discuss this "issue" in Sect. 14.4.

Remark 14.2.15. The definition of PAST was coined by [BG05]. However, this notion is somewhat ambiguous when pGCL-programs contain non-determinism: [Kam+18; Kam19] define PAST via exprt $[\![\mathbf{P}]\!](s_0) < \infty$ for all $s_0 \in \Sigma$. The latter notion is called *strong positive almost-sure termination* (SAST) in [ALY20]. For purely probabilistic pGCL-programs PAST and SAST clearly coincide. In case of unbounded non-determinism, PAST and SAST are not equivalent: consider the program which chooses a natural number non-deterministically and then decreases it until zero.

However, even in case of bounded non-determinism PAST and SAST do not coincide: In [ALY20, Ex. 7], it is shown that the program $\mathbf{P}_{pns}$ in Fig. 14.8 is PAST but not SAST. Intuitively, for every scheduler which at some point chooses to execute the right statement in the non-deterministic choice (let us say after at most m iterations) y has at most the value $2^m \cdot m$ after the first loop and thus, the expected runtime of $\mathbf{P}_{pns}$ is at most $C \cdot (2^m \cdot m + m + 1) < \infty$ for some constant $C > 0$ which takes into account how many statements have been executed in the first and the second loop. If the scheduler never executes the right statement, the first loop is expected to be executed twice, while the second loop is not executed at all (as y is still 0). Thus, this program is PAST.

However, it can be shown that for every scheduler which chooses the right statement in the non-deterministic choice after m iterations the expected runtime is at least m and thus, the program is not SAST.

Finally, it is worth noting that one could also choose a different model of schedulers which also allows for resolving non-determinism via distributions (so-called randomized schedulers, see, e.g., [Put94]). Then, the program which chooses a natural number non-deterministically and decreases it until zero is not PAST anymore since a scheduler could resolve the non-determinism with a distribution on the natural numbers whose expected value is infinite. Nevertheless, our technique in Ch. 17 (where also unbounded non-determinism is considered) proves PAST via SAST (see Sect. 17.2.2) and the notions of SAST coincide for the different models of schedulers (see [Put94, Prop. 6.2.1, Thm. 7.1.3]). Moreover, most approaches for proving PAST in related work actually prove SAST (see., e.g., [ACN18; BG05; FC19]). The class of programs for which we present a decision procedure for PAST in Ch. 16 only contains purely probabilistic programs (and thus, SAST and PAST coincide).

Example 14.2.16. Consider again the programs $\mathbf{P}_1$ and $\mathbf{P}_2$ from Ex. 14.2.13. Independent of the initial state we have

$$\operatorname{exprt}\,[\![\mathbf{P}_1]\!]\,(s_0) = \infty = \operatorname{exprt}\,[\![\mathbf{P}_2]\!]\,(s_0)\,.$$

If we consider again the schedulers $\mathfrak{s}$ and $\mathfrak{s}'$ for $\mathbf{P}_1$, then

$$\operatorname{ert}\,[\![\mathbf{P}_1]\!]\,(0)\,(s_0)^{\mathfrak{s}} = 1 \text{ and } \operatorname{ert}\,[\![\mathbf{P}_1]\!]\,(0)\,(s_0)^{\mathfrak{s}'} = \infty.$$

Thus neither $\mathbf{P}_1$ nor $\mathbf{P}_2$ is PAST.

We have already seen that the programs considered in Ex. 14.2.16 are not AST in Ex. 14.2.13. That they are also not PAST is not a coincidence but due to the fact that PAST is stronger than AST.

Lemma 14.2.17 (PAST Implies AST, [BG05])**.** *Let* $\mathbf{P}$ *be a* pGCL*-program which is PAST. Then* $\mathbf{P}$ *is AST.*

However, the converse is not true, i.e., AST is strictly weaker than PAST.

Example 14.2.18. The program $\mathbf{P}$ in Fig. 14.9 captures the classical one-dimensional *random walk*. It is well-known that $\mathbf{P}$ is not PAST (see, e.g., [CNZ17]), i.e., whenever we start in an initial program state s_0 with $s_0(x) > 0$ we cannot expect the program to terminate within finite time. Nonetheless, $\mathbf{P}$ is

while $(x > 0)$ **do** {
    $\{\, x \leftarrow x - 1 \,\} \left[\frac{1}{2}\right] \{\, x \leftarrow x + 1 \,\}$
}

Figure 14.9: One-Dimensional Random Walk

AST (see, e.g., [McI+18]), i.e., the probability of termination is 1 independent of the initial program state.

To sum up, we have seen that the notion of termination from classical programming splits when introducing probabilistic behavior. Even purely probabilistic pGCL-programs can terminate with probability 1 independent of the initial program state but cannot be expected to terminate within finite time. Nonetheless, if a deterministic program (i.e., without probabilistic or non-deterministic behavior) terminates, its runtime can be bounded by a function in the initial program state taking only *finite* values. Whether AST or PAST corresponds to classical termination is a philosophical question and not addressed here. Usually, proving AST is seen as the generalization of proving termination, while proving PAST is seen as computing a bound on the expected runtime complexity of a program.

In Ch. 15, we will also discuss *expected outcomes* of pGCL-programs. The underlying question is as follows: Consider some function $f : \Sigma \to \overline{\mathbb{R}}_{\geq 0}$ and a pGCL-program $\mathbf{P}$. What is the *expected value of* f when evaluated in the final states of $\mathbf{P}$? The function f is referred to as the *post-expectation*, while the function $\mathrm{wp}\,[\![\mathbf{P}]\!]\,(f)$ such that $\mathrm{wp}\,[\![\mathbf{P}]\!]\,(f)\,(s_0)$ is the expected outcome of f when starting the program $\mathbf{P}$ on s_0 is called the *weakest preexpectation* of f.

Definition 14.2.19 (Weakest Preexpectation)**.** Let $\mathbf{P}$ be a pGCL-program with initial state s_0, let $\mathfrak{s}$ be a scheduler for $\mathbf{P}$, and let $f : \Sigma \to \overline{\mathbb{R}}_{\geq 0}$ be a post-expectation. Let rew_f be the reward function for $\mathcal{M}^{\mathbf{P}}_{s_0}$ defined in Table 14.4.

» The *(demonic) weakest preexpectation of* f *w.r.t* $\mathbf{P}$ *in* s_0 is

$$\mathrm{wp}\,[\![\mathbf{P}]\!]\,(f)\,(s_0) = \mathrm{ExpRew}^{\min}_{\mathcal{M}^{\mathbf{P}}_{s_0}}\left(\diamond\,\{(\bot, s) \mid s \in \Sigma\}\right).$$

$\mathfrak{d}$	$rew_f(\mathfrak{d})$
$(\bot, s)$	$f(s)$
$(\mathbf{skip}, s), (x \leftarrow e, s), (x \leftsquigarrow D, s), (\mathbf{if}\ (\varphi)\ \mathbf{then}\ \{\ \mathbf{P}_1\ \}\ \mathbf{else}\ \{\ \mathbf{P}_2\ \}, s),$ $(\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}, s), (\{\ \mathbf{P}_1\ \}\ [p]\ \{\ \mathbf{P}_2\ \}),$ $(\mathbf{P}_1 \fatsemi \mathbf{P}_2, s), (\{\ \mathbf{P}_1\ \} \square \{\ \mathbf{P}_2\ \})$	0

Table 14.4: Reward Function for Post-Expectation f

» The *angelic weakest preexpectation of* f *w.r.t* $\mathbf{P}$ *in* s_0 is

$$\mathrm{awp}\ [\![\mathbf{P}]\!]\ (f)\,(s_0) = \mathrm{ExpRew}^{\max}_{\mathcal{M}^{\mathbf{P}}_{s_0}}\ (\diamond\,\{(\bot, s) \mid s \in \Sigma\})\,.$$

Note that if $\mathbf{P}$ is purely probabilistic and $f = 1$ is the constant function mapping every state to 1, then $\mathrm{wp}\ [\![\mathbf{P}]\!]\ (1)\,(s_0)$ is the probability of termination of $\mathbf{P}$ on input s_0.

Example 14.2.20. Consider again the programs $\mathbf{P}_1$ and $\mathbf{P}_2$ from Ex. 14.2.13 and 14.2.16 and let $f(s) = s(x)$. Independent of the initial state we have

$$\mathrm{wp}\ [\![\mathbf{P}_1]\!]\ (f)\,(s_0) = 0$$

since for the scheduler $\mathfrak{s}'$ no final state is reachable in the underlying MDP (see Fig. 14.5). Nonetheless, we have

$$\mathrm{awp}\ [\![\mathbf{P}_1]\!]\ (f)\,(s_0) = 1$$

since for the scheduler $\mathfrak{s}$ the final state $(\bot, s = 1)$ is reached with probability 1 (see Fig. 14.6). In contrast,

$$\mathrm{awp}\ [\![\mathbf{P}_2]\!]\ (f)\,(s_0) = \mathrm{wp}\ [\![\mathbf{P}_2]\!]\ (f)\,(s_0) = \tfrac{1}{2}$$

since the only final state in the underlying DTMC which is reachable from the initial state is $(\bot, s)$, where $s(x) = 1$, and it is reached with probability $\frac{1}{2}$ (see Ex. 14.2.13).

In contrast to expected runtimes, we resolve the non-determinism demonically (i.e., in the worst-case scenario) via the *minimal* expected reward. This stems from

the just mentioned fact that $\mathsf{wp}\,[\![\mathbf{P}]\!]\,(1)\,(s_0)$ is closely related to the probability of termination: the smaller the probability of termination, the "less useful" is **P**. When interpreting non-determinism as an "adversary" or a "demon" trying to sabotage the program, then it will try to get the probability of termination as small as possible. Nonetheless, the larger the expected runtime of **P**, the "less useful" it is. Thus, in this case an adversary will try to resolve the non-determinism in such a way that the expected runtime becomes as large as possible.

14.3 Denotational Semantics for pGCL-Programs

In this section, we introduce denotational semantics for pGCL-programs via the wp- and the ert-transformer. However, we do not only present these transformers to have a denotational semantics at hand, but we will discuss how to obtain bounds on expected outcomes and expected runtimes of pGCL-programs via these transformers in Ch. 15.

Technically speaking, both transformers are to be seen as quantitative adaptions of so-called *predicate transformers* for probabilistic programs. Predicate transformers have been developed by Dijkstra in his seminal works [Dij75; Dij76] to define a denotational semantics for guarded commands. His idea is usually referred to as inferring *weakest preconditions* for *postconditions*. Here, both pre- and postcondition are predicates. This idea also yields an algorithm to infer so-called Hoare triples from Floyd-Hoare logic [Flo67; Hoa69]. Kozen generalized this idea from predicate transformers to measure transformers [Koz81; Koz85] to enable a quantitative analysis of probabilistic programs which he called *probabilistic Propositional Dynamic Logic* (PPDL). However, he only considered purely probabilistic programs. McIver and Morgan then developed the version of the wp-transformer that we consider in this thesis which in addition to probabilistic choice also considers non-deterministic choice [MM05]. Following this manner, Kaminski et al. developed the so-called ert-transformer to reason about expected runtimes of probabilistic programs in a similar way to the wp-transformer [Kam+18].

While the introduction presented here is enough for our purposes, it is of course far from complete. A thorough introduction including a detailed comparison of the wp-transformer and the ert-transformer with Dijkstra's weakest-precondition calculus as well as Kozen's PPDL can be found in [Kam19].

14.3.1 The Weakest Preexpectation Transformer wp

Dijkstra employs *predicate transformers*, which push a *postcondition* F (a predicate) backward through a non-probabilistic program **P** and yield the *weakest precondition* G

(another predicate) describing the largest set of states such that whenever **P** is started in a state satisfying G, **P** terminates in a state satisfying F. Here, we consider total correctness, i.e., from any state satisfying the weakest precondition G, **P** terminates.

The *weakest preexpectation calculus* on the other hand employs *expectation transformers* which act on real-valued functions called *expectations*, mapping program states to non-negative reals. Kaminski and Katoen also developed a wp-calculus for reasoning about mixed-sign expectations mapping to the *full* extended reals [KK17]. However, we only study the standard case of non-negative expectations.

These transformers push a *postexpectation* f backward through a pGCL-program **P** and yield a *preexpectation* g such that g represents the expected value of f after executing **P**. The term *expectation* was coined by [MM05] and may appear somewhat misleading at first. We *clearly distinguish between expectations and expected values*: An expectation is hence not an expected value, per se. Instead, we can think of an expectation as a *random variable* (see Sect. 14.4 for details).

Definition 14.3.1 (Expectations, [Kam19; MM05])**.** The set of *expectations*, denoted by $\mathbb{F}$, is defined as

$$\mathbb{F} = \left\{ f \mid f : \Sigma \longrightarrow \overline{\mathbb{R}}_{\geq 0} \right\}.$$

We say that $f \in \mathbb{F}$ is *finite* and write $f \ll \infty$ if $f(s) < \infty$ for all $s \in \Sigma$. A partial order $\preceq$ on $\mathbb{F}$ is obtained by pointwise lifting the usual order $\leq$ on $\overline{\mathbb{R}}_{\geq 0}$, i.e.,

$$f_1 \preceq f_2 \quad \text{iff} \quad \forall s \in \Sigma : \quad f_1(s) \leq f_2(s).$$

Suprema and infima are also constructed pointwise. In this way, $(\mathbb{F}, \preceq)$ is a *complete lattice*, i.e., every subset of $\mathbb{F}$ has both a supremum and an infimum w.r.t. $\preceq$ (where we use $\inf \emptyset = \infty$ and $\sup \emptyset = 0$).

We note that this notion of expectations is more general than the one introduced by McIver and Morgan: Their work builds almost exclusively on *bounded* expectations, i.e., non-negative real-valued functions which are bounded from above by some constant, whereas we consider *unbounded* expectations. As a result, we have that $(\mathbb{F}, \preceq)$ forms a complete lattice, whereas McIver and Morgan's space of bounded expectations does not.

Given a pGCL-program **P** and postexpectation $f \in \mathbb{F}$, we are interested in the *expected value* of f evaluated in the final states reached after termination of **P**, i.e., we are interested in the weakest preexpectation $\mathrm{wp}\,[\![\mathbf{P}]\!]\,(f)$ of **P** w.r.t. f as defined in

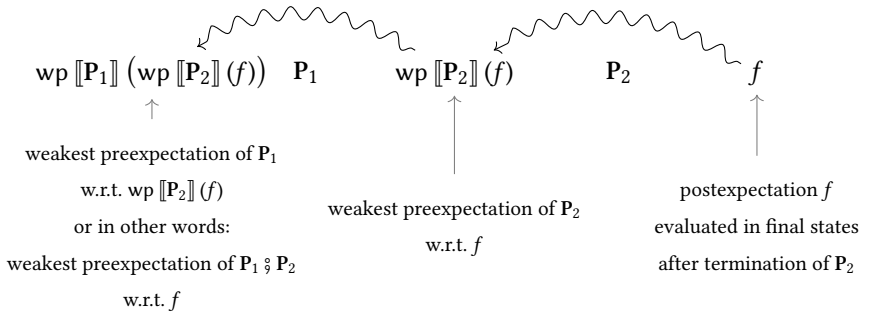


Figure 14.10: Backward-Moving Continuation-Passing Style wp-Transformer

Def. 14.2.19. Put as an equation:

$$\text{wp}\,[\![\mathbf{P}]\!]\,(f)\,(s_0) = \inf_{\mathfrak{s}\in\text{Scheds}} \int_{\Sigma} f \; d\left(\mu^{\mathbf{P}}_{\mathfrak{s},s_0}\right) = \inf_{\mathfrak{s}\in\text{Scheds}} \sum_{s\in\Sigma} f(s)\cdot\mu^{\mathbf{P}}_{\mathfrak{s},s_0}(s)\,.$$

While $\text{wp}\,[\![\mathbf{P}]\!]\,(f)$ in fact represents an expected value, f itself does not. In an analogy to Dijkstra's pre- and postconditions as f is evaluated in the final states after termination of $\mathbf{P}$ it is called the *postexpectation*, and as $\text{wp}\,[\![\mathbf{P}]\!]\,(f)$ is evaluated in the initial states of $\mathbf{P}$ it is called the *preexpectation*.

We now show how to determine weakest preexpectations in a systematic and *compositional* manner by recapitulating the *weakest preexpectation calculus* à la McIver and Morgan. This calculus employs expectation transformers which move backward through the program in a *continuation–passing* style, see Fig. 14.10. If we are interested in the expected value of some postexpectation f after executing the sequential composition $\mathbf{P}_1 \fatsemi \mathbf{P}_2$, we can first determine the weakest preexpectation of $\mathbf{P}_2$ w.r.t. f, i.e., $\text{wp}\,[\![\mathbf{P}_2]\!]\,(f)$. Thereafter, we can use the intermediate result $\text{wp}\,[\![\mathbf{P}_2]\!]\,(f)$ as *postexpectation* to determine the weakest preexpectation of $\mathbf{P}_1$ w.r.t. $\text{wp}\,[\![\mathbf{P}_2]\!]\,(f)$. Overall, this gives the weakest preexpectation of $\mathbf{P}_1 \fatsemi \mathbf{P}_2$ w.r.t. the postexpectation f. Fig. 14.10, which is taken from [Kam19, Fig. 2.3], illustrates the compositional nature of the weakest preexpectation calculus. By induction on the program structure, wp-transformers for all language constructs can be defined.

Definition 14.3.2 (The (a)wp-Transformer [Kam19; MM05])**.** The weakest pre-

P	**wp** $[\![\mathbf{P}]\!]\,(f)$
skip	f
$x \leftarrow e$	$f\,[x/e]$
$x \leftsquigarrow D$	$\lambda s.\ \sum_{z\in\mathbb{Z}}(D(s))(z)\cdot f\,[x/z]\,(s)$
$\{\,\mathbf{P}_1\,\}\ [p]\ \{\,\mathbf{P}_2\,\}$	$p\cdot \mathrm{wp}\,[\![\mathbf{P}_1]\!]\,(f) + (1-p)\cdot \mathrm{wp}\,[\![\mathbf{P}_2]\!]\,(f)$
$\{\,\mathbf{P}_1\,\}\ \square\ \{\,\mathbf{P}_2\,\}$	$\min(\mathrm{wp}\,[\![\mathbf{P}_1]\!]\,(f), \mathrm{wp}\,[\![\mathbf{P}_2]\!]\,(f))$
if (φ) **then** $\{\,\mathbf{P}_1\,\}$ **else** $\{\,\mathbf{P}_2\,\}$	$[\varphi]\cdot \mathrm{wp}\,[\![\mathbf{P}_1]\!]\,(f) + [\neg\varphi]\cdot \mathrm{wp}\,[\![\mathbf{P}_2]\!]\,(f)$
$\mathbf{P}_1 \fatsemi \mathbf{P}_2$	$\mathrm{wp}\,[\![\mathbf{P}_1]\!]\,(\mathrm{wp}\,[\![\mathbf{P}_2]\!]\,(f))$
while (φ) **do** $\{\,\mathbf{P}\,\}$	$\mathrm{lfp}\ {}^{\mathrm{wp}}\Phi_{\langle\varphi,\mathbf{P}\rangle,f}$

$${}^{\mathrm{wp}}\Phi_{\langle\varphi,\mathbf{P}\rangle,f}(X) = [\neg\varphi]\cdot f + [\varphi]\cdot \mathrm{wp}\,[\![\mathbf{P}]\!]\,(X) \qquad \text{characteristic function}$$

Table 14.5: Rules for the wp-Transformer

$$\{\,h \leftarrow h+3\,\}\ \left[\tfrac{1}{2}\right]\ \{\,\mathbf{skip}\,\}$$

Figure 14.11: Simplified Loop Body of Fig. 14.1

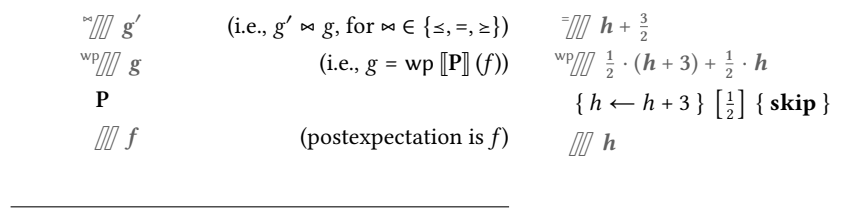
expectation transformer for the pGCL-programs defined via pGCL

$$\mathrm{wp}:\ \mathrm{pGCL} \longrightarrow \mathbb{F} \longrightarrow \mathbb{F}$$

is defined according to the rules in Table 14.5. Here $f\,[x/z]\,(s) = f(s[x \mapsto s(z)])$.

We call the function ${}^{\mathrm{wp}}\Phi_{\langle\varphi,\mathbf{P}\rangle,f}$ the *characteristic function* of **while** (φ) **do** $\{\,\mathbf{P}\,\}$ w.r.t. f. Its least fixed point is understood in terms of $\preceq$. We omit wp, φ, $\mathbf{P}$, or f from Φ whenever they are clear from the context.

Similarly, we define the awp-transformer for the angelic weakest precondition. Its rules are identical to the ones in Table 14.5 where wp is replaced by awp and instead of using min in case of a non-deterministic choice we use max.



(a) Style for wp-Annotations

(b) wp-Annotations for Ex. 14.3.3

Figure 14.12: Annotations for Weakest Preexpectations: It is more intuitive to read these from the bottom to the top.

Example 14.3.3 (Applying the wp-Calculus)**.** Consider the pGCL-program $\mathbf{P}$ given in Fig. 14.11 which is a simplified version of the body of the loop in Fig. 14.1. Suppose we are interested in the expected value of h after execution of $\mathbf{P}$. To this end, we determine $\text{wp}\,[\![\mathbf{P}]\!]\,(h)$. Using the annotation style shown in Fig. 14.12(a), we can annotate the program $\mathbf{P}$ as shown in Fig. 14.12(b) by using the rules from Table 14.5. At the top, we read off the weakest preexpectation of $\mathbf{P}$ w.r.t. h, namely $h + \frac{3}{2}$. This tells us that the expected value of h after termination of $\mathbf{P}$ on initial state s_0 is $s_0(h) + \frac{3}{2}$.

The wp-transformer satisfies what is sometimes called *healthiness conditions* [Hin+16; Kei15; MM05] or *homomorphism properties* [BW98]:

Theorem 14.3.4 (Healthiness Conditions [Kam19; MM05])**.** *Let* $\mathbf{P}$ *be a* pGCL*-program,* $s_0 \preceq s_1 \preceq \cdots$ *a monotonic sequence in* $\mathbb{F}$*,* $f, g \in \mathbb{F}$*, and* $r \in \mathbb{R}_{\geq 0}$*. Then:*

(i) **Continuity:** $\text{wp}\,[\![\mathbf{P}]\!]\,\left(\sup_{n\in\mathbb{N}} s_n\right) = \sup_{n\in\mathbb{N}} \text{wp}\,[\![\mathbf{P}]\!]\,(s_n)$.

(ii) **Strictness:** $\text{wp}\,[\![\mathbf{P}]\!]$ *is strict, i.e.,* $\text{wp}\,[\![\mathbf{P}]\!]\,(0) = 0$.

(iii) **Monotonicity:** $f \preceq g$ *implies* $\text{wp}\,[\![\mathbf{P}]\!]\,(f) \preceq \text{wp}\,[\![\mathbf{P}]\!]\,(g)$.

(iv) **Superlinearity:** $\text{wp}\,[\![\mathbf{P}]\!]\,(r \cdot f + g) \succeq r \cdot \text{wp}\,[\![\mathbf{P}]\!]\,(f) + \text{wp}\,[\![\mathbf{P}]\!]\,(g)$.

Here, Thm. 14.3.4 ensures that the characteristic function ${}^{\text{wp}}\Phi_{\langle\varphi,\mathbf{P}\rangle,f}$ indeed has a least fixed point (see also Thm. 15.1.5), i.e., the wp-transformer is well defined.

Moreover, continuity refers to continuity w.r.t. a complete lattice D: a function $\Phi : D \longrightarrow D$ is *continuous* if for every monotonic increasing sequence $(s_n)_{n\in\mathbb{N}}$ in D we have $\Phi\left(\sup_{n\in\mathbb{N}} s_n\right) = \sup_{n\in\mathbb{N}} \Phi(s_n)$. Note that a similar result also holds for the awp-transformer with the only difference that it is sublinear, i.e., the inequation-sign has to be turned around in Thm. 14.3.4 (iv). Thus, if **P** is purely probabilistic, wp is linear. Indeed, the (a)wp-calculus is *sound*:

Theorem 14.3.5 (Soundness of the (a)wp-Calculus, [GKM14; Kam19])**.** *The* (a)wp*-calculus is sound, i.e., the operational semantics defined in Def. 14.2.19 and the denotational semantics of* (a)wp *via the* (a)wp*-calculus defined in Def. 14.3.2 coincide.*

Example 14.3.6. Let us reconsider Ex. 14.2.20. First of all, the loop

$$\textbf{while}\,(\,\text{true}\,)\,\textbf{do}\,\{\,\textbf{skip}\,\}$$

has the characteristic function ${}^{\text{wp}}\Phi_x = {}^{\text{awp}}\Phi_x = \Phi$ with $\Phi(X) = X$ for every $X \in \mathbb{F}$, i.e., the identity function, and thus, its least fixed point is 0. So

$$\text{wp}\,[\![\textbf{while}\,(\,\text{true}\,)\,\textbf{do}\,\{\,\textbf{skip}\,\}]\!]\,(x) = \text{awp}\,[\![\textbf{while}\,(\,\text{true}\,)\,\textbf{do}\,\{\,\textbf{skip}\,\}]\!]\,(x) = 0.$$

The wp annotations for $\mathbf{P}_1$ and $\mathbf{P}_2$ can be found in Fig. 14.13. Thus, we have inferred the results from Ex. 14.2.20 algorithmically.

14.3.2 The Expected Runtime Transformer ert

Similar to the weakest preexpectation calculus we define the so-called expected runtime calculus via the ert-transformer. Again, we only introduce the basic results needed for the following chapters. For more details, we refer to [Kam+18; Kam19].

First of all, one could think that introducing a new transformer for reasoning about expected runtimes is in fact superfluous: why not introduce a fresh variable and use it as a counter? Its expected value should then be the expected runtime of the underlying program. Unfortunately, this is *unsound* as illustrated in the following example which is taken from [Kam+18].

Counterexample 14.3.7. Consider the program

$$\mathbf{P} = \textbf{while}\,(\,\text{true}\,)\,\textbf{do}\,\{\,\textbf{skip}\,\}.$$

$\overset{=}{/\!/\!/}\ \mathbf{0}$
$\overset{\text{wp}}{/\!/\!/}\ \min(1,0)$
$\{\, x \leftarrow 1 \,\} \square \{\, \textbf{while}\,(\text{true})\,\textbf{do}\,\{\,\textbf{skip}\,\}\,\}$
$/\!/\!/\ x$

(a) wp-Annotations for $\mathbf{P}_1$

$\overset{=}{/\!/\!/}\ \mathbf{1}$
$\overset{\text{awp}}{/\!/\!/}\ \max(1,0)$
$\{\, x \leftarrow 1 \,\} \square \{\, \textbf{while}\,(\text{true})\,\textbf{do}\,\{\,\textbf{skip}\,\}\,\}$
$/\!/\!/\ x$

(b) awp-Annotations for $\mathbf{P}_1$

$\overset{=}{/\!/\!/}\ \frac{1}{2}$
$\overset{\text{wp}}{/\!/\!/}\ \frac{1}{2}\cdot 1 + \frac{1}{2}\cdot 0$
$\{\, x \leftarrow 1 \,\}\ \left[\tfrac{1}{2}\right]\ \{\, \textbf{while}\,(\text{true})\,\textbf{do}\,\{\,\textbf{skip}\,\}\,\}$
$/\!/\!/\ x$

(c) wp-Annotations for $\mathbf{P}_2$

Figure 14.13: (a)wp-Annotations for Ex. 14.3.6

Clearly, exprt $[\![\mathbf{P}]\!](s_0) = \infty$ for all $s_0 \in \Sigma$. Let us now introduce a fresh variable c which we use as a counter for the statements. Here, every iteration of the loop leads to an increase of 2 since we need one unit of time for evaluating the condition of the loop and the **skip** command also consumes one unit of time. This program is stated in Fig. 14.14(a).

The wp annotations are given in Fig. 14.14(b). Here, we have used

$$\text{wp}\,[\![c \leftarrow c + 2]\!]\,(X) = X\,[c/c+2]$$

and thus

$$^{\text{wp}}\Phi_c(X) = [\neg\text{true}]\cdot c + [\text{true}]\cdot X\,[c/c+2] = X\,[c/c+2]$$

which has 0 as a fixed point, i.e., $\text{lfp}\,{}^{\text{wp}}\Phi_c(X) = 0$. Hence, the expected value of the counter is 0 independent of the initial state, i.e., this does *not* reflect the expected

$c \leftarrow 0 \,⨾$
while (true) **do** { $c \leftarrow c + 2$ }

(a) Introducing a Counter c for **P**

$\overset{\text{wp}}{/\!/\!/}\ \mathbf{0}$
$c \leftarrow 0 \,⨾$
$\overset{\text{wp}}{/\!/\!/}\ \mathbf{0}$
while (true) **do** { $c \leftarrow c + 2$ }
$/\!/\!/\ c$

(b) wp-Annotations for Expected Value of c

Figure 14.14: Unsoundness of wp for Expected Runtimes [Kam+18]

runtime: whereas non-terminating computations of the program do not contribute to the expected value of c, they indeed contribute to the expected runtime.

To resolve this issue, we introduce the ert-transformer from [Kam+18], which is similar to the wp-transformer: Given program **P** and post*runtime* $t \in \mathbb{F}$, we are interested in the *expected time* it takes to first execute **P** and then let time t pass (where t is evaluated in the final states reached after termination of **P**). Again, the behavior (and the runtime) of **P** depends on its input, so we are actually interested in a *function* $g \in \mathbb{F}$ mapping *initial* states s_0 to the respective expected time.

Similarly to weakest preexpectations, expected runtimes can be determined in a systematic and *compositional* manner by means of the ert-calculus:

Definition 14.3.8 (The ert-Transformer [Kam+18; Kam19])**.** The expected runtime transformer for pGCL-programs

$$\text{ert} : \text{pGCL} \longrightarrow \mathbb{F} \longrightarrow \mathbb{F}$$

is defined according to the rules given in Table 14.6. We call the function ${}^{\text{ert}}\Phi_{\langle \varphi, \mathbf{P} \rangle, t}$ the ert-*characteristic function* of the loop **while** (φ) **do** { **P** } w.r.t. t. Its least fixed point is understood in terms of the partial order $\preceq$. We again usually omit ert, φ, C, or t from Φ whenever they are clear from the context.

$\mathbf{P}$	$\mathbf{ert}\,[\![\mathbf{P}]\!]\,(t)$
skip	$1 + t$
$x \leftarrow e$	$1 + t\,[x/e]$
$x \leftsquigarrow D$	$\lambda s.\ 1 + \sum_{z\in\mathbb{Z}}(D(s))(z) \cdot f\,[x/z]\,(s)$
$\{\,\mathbf{P}_1\,\}\ [p]\ \{\,\mathbf{P}_2\,\}$	$1 + p \cdot \mathrm{ert}\,[\![\mathbf{P}_1]\!]\,(t) + (1-p) \cdot \mathrm{ert}\,[\![\mathbf{P}_2]\!]\,(t)$
$\{\,\mathbf{P}_1\,\} \square \{\,\mathbf{P}_2\,\}$	$\max(\mathrm{ert}\,[\![\mathbf{P}_1]\!]\,(t), \mathrm{ert}\,[\![\mathbf{P}_2]\!]\,(t))$
if (φ) **then** $\{\,\mathbf{P}_1\,\}$ **else** $\{\,\mathbf{P}_2\,\}$	$1 + [\varphi] \cdot \mathrm{ert}\,[\![\mathbf{P}_1]\!]\,(t) + [\neg\varphi] \cdot \mathrm{ert}\,[\![\mathbf{P}_2]\!]\,(t)$
$\mathbf{P}_1 \fatsemi \mathbf{P}_2$	$\mathrm{ert}\,[\![\mathbf{P}_1]\!]\,(\mathrm{ert}\,[\![\mathbf{P}_2]\!]\,(t))$
while (φ) **do** $\{\,\mathbf{P}\,\}$	$\mathrm{lfp}\ {}^{\mathrm{ert}}\Phi_{\langle\varphi,\mathbf{P}\rangle,\,t}$

$$ {}^{\mathrm{ert}}\Phi_{\langle\varphi,\mathbf{P}\rangle,\,t}(X) = 1 + [\neg\varphi]\cdot t + [\varphi]\cdot \mathrm{ert}\,[\![\mathbf{P}]\!]\,(X) \qquad \text{characteristic function} $$

Table 14.6: Rules for the ert-Transformer

Example 14.3.9 (Applying the ert-Calculus)**.** Consider again the simple pGCL-program $\mathbf{P}$ given in Fig. 14.11. We can infer $\mathrm{ert}\,[\![\mathbf{P}]\!]\,(0)\,(s_0) = \mathrm{exprt}\,[\![\mathbf{P}]\!]\,(s_0) = 2$ for all $s_0 \in \Sigma$ as shown in Fig. 14.15.

Additionally we can also infer that the expected runtime of the loop

$$\textbf{while}\,(\,\mathrm{true}\,)\,\textbf{do}\,\{\,\textbf{skip}\,\}$$

is infinite: the ert-characteristic function is given by

$$ {}^{\mathrm{ert}}\Phi_0(X) = 1 + [\neg\mathrm{true}]\cdot 0 + [\mathrm{true}]\cdot X = 1 + X $$

whose only fixed point is ∞.

Similar to Thm. 14.3.4, the ert-transformer satisfies the following healthiness conditions which again ensure the well-definedness of the ert-transformer for loops, i.e., the existence of the least fixed point.

Theorem 14.3.10 (Healthiness Conditions for ert [Kam+18; Kam19])**.** *Let* $\mathbf{P}$ *be a* pGCL*-program,* $s_0 \preceq s_1 \preceq \cdots$ *a monotonic sequence in* $\mathbb{F}$*,* $f, g \in \mathbb{F}$*, and* $r \in \mathbb{R}_{\geq 0}$*. Then:*

(i) **Continuity:** $\mathrm{ert}\,[\![\mathbf{P}]\!]\,\left(\sup_{n\in\mathbb{N}}\ s_n\right) = \sup_{n\in\mathbb{N}}\ \mathrm{ert}\,[\![\mathbf{P}]\!]\,(s_n)$.

$$
\begin{aligned}
&\overset{=}{/\!/\!/}\ 2\\
&\overset{\text{ert}}{/\!/\!/}\ 1 + \tfrac{1}{2}\cdot(1 + 0\,[h/h+3]) + \tfrac{1}{2}\cdot(1+0)\\
&\quad \{\, h \leftarrow h+3 \,\}\ \left[\tfrac{1}{2}\right]\ \{\, \textbf{skip} \,\}\\
&/\!/\!/\ 0
\end{aligned}
$$

Figure 14.15: ert-Annotations for Ex. 14.3.9

(ii) **Monotonicity:** $f \preceq g$ *implies* $\text{ert}\,[\![\mathbf{P}]\!]\,(f) \preceq \text{ert}\,[\![\mathbf{P}]\!]\,(g)$.

Again, the operational semantics and the denotational semantics for expected runtimes coincide.

Theorem 14.3.11 (Soundness of the ert-Calculus,[Kam+18; Kam19])**.** *The* ert*-calculus is sound, i.e., the operational semantics defined in Def. 14.2.14 and the denotational semantics of* ert *via the* ert*-calculus defined in Def. 14.3.8 coincide.*

Finally, the following theorem describes the connection of the expected runtime transformer and the weakest preexpectation transformer for purely probabilistic pGCL-programs which we will use in Sect. 15.6 to generalize our results from the wp-transformer to the ert-transformer.

Theorem 14.3.12 (Connection between ert and wp, [Kam+18])**.** *Let* $\mathbf{P}$ *be a purely probabilistic* pGCL*-program and* $t \in \mathbb{F}$*. Then*

$$\text{ert}\,[\![\mathbf{P}]\!]\,(t) = \text{ert}\,[\![\mathbf{P}]\!]\,(0) + \text{wp}\,[\![\mathbf{P}]\!]\,(t) = \text{exprt}\,[\![\mathbf{P}]\!] + \text{wp}\,[\![\mathbf{P}]\!]\,(t)\,,$$

i.e., the expected runtime of $\mathbf{P}$ *whenever after termination of* $\mathbf{P}$ *time* t *passes is the expected runtime of* $\mathbf{P}$ *plus the expected value of* t *after termination of* $\mathbf{P}$*.*

As we have seen, both the wp-calculus and the ert-calculus require a computation of least fixed points for the characteristic functions of loops. As already said, computing these fixed points is in general impossible. Still, one can try to approximate these fixed points from below or above. While we will discuss rules for over- and under-approximating such least fixed points in detail in Ch. 15, we already introduce the key-concept of invariants here. The reason is that we need this notion in the next section to explain how to construct a stochastic process from a probabilistic loop. Here, we only define wp-invariants but we use this concept also for ert.

Kind	Relation
Invariant	$\mathrm{wp}\Phi_f(I) = I$
Superinvariant	$\mathrm{wp}\Phi_f(I) \preceq I$
Subinvariant	$\mathrm{wp}\Phi_f(I) \succeq I$

Table 14.7: Different Kinds of Invariants

while (φ) **do** { **P** }

Figure 14.16: Probabilistic Loop

Definition 14.3.13 (wp-Invariants)**.** Let $f, I \in \mathbb{F}$. We call I a wp-*(super/sub)invariant* w.r.t. f of **while** (φ) **do** { **P** } if the corresponding relation depicted in Table 14.7 holds.

14.4 Denotational Semantics for Probabilistic Loops via Stochastic Processes

In the last sections, we have introduced a syntax for probabilistic programs and discussed both operational and denotational semantics for them. In this section, we focus on so-called *probabilistic loops* and discuss denotational semantics via stochastic processes. A probabilistic loop **L** has the form in Fig. 14.16 where φ is an arbitrary condition and **P** is an arbitrary (not necessarily loop-free) pGCL-program.

Definition 14.4.1 (Probabilistic Loop)**.** A *probabilistic loop* is a pGCL-program of the form in Fig. 14.16. Here, we decompose the state space Σ into $\Sigma = \Sigma_\varphi \uplus \Sigma_{\neg\varphi}$ where $\Sigma_\varphi = \{s \in \Sigma \mid s \vDash \varphi\}$ and $\Sigma_{\neg\varphi} = \{s \in \Sigma \mid s \nvDash \varphi\}$.

In this thesis, we require **P** to be AST.

Example 14.4.2. Again, consider the race between tortoise and hare in Fig. 14.1. It is a probabilistic loop whose body is itself loop-free, i.e., its body is AST.

Clearly, probabilistic loops are a sub-class of general pGCL-programs and therefore, the semantics introduced in Sect. 14.2 and 14.3 apply to them. Nevertheless, probabilistic loops form the bottleneck of verification of pGCL-programs: while applying wp and ert to loop-free programs is straightforward, programs including loops require the computation of a least fixed point which is in general not computable. Thus, we introduce an additional denotational semantics via so-called stochastic processes for these loops. Moreover, in Sect. 14.4.2 we also present results on the connection of the semantics via expectation transformers and the semantics via stochastic processes which are unpublished so far. We further deepen this connection in Ch. 15.

Since every pGCL-program is, operationally, a countable MDP (see Sect. 14.2), there is also a denotational semantics via stochastic processes for general pGCL-programs which uses the standard construction for MDPs (see, e.g., [Var85]). However, introducing this semantics only for probabilistic loops is sufficient for our needs.

14.4.1 Canonical Probability Space

We begin with constructing a canonical probability measure and space corresponding to the execution of a probabilistic loop. As already said, the construction presented here is similar to the standard construction for MDPs (see, e.g., [Var85]). For further details on measure and probability theory we refer to App. A and [Bau11a; Bau11b; GS01; Kal02].

Definition 14.4.3 (σ-Field, Measurable Space)**.** Let Ω be some set. A set $\mathcal{F} \subseteq \mathfrak{P}(\Omega)$ of subsets of Ω is a *σ-field* of Ω if $\Omega \in \mathcal{F}$ and $\mathcal{F}$ is closed under complement and countable unions. We call $(\Omega, \mathcal{F})$ a *measurable space*.

In our setting, a probabilistic loop induces a *canonical* measurable space:

Definition 14.4.4 (Loop Space)**.** A probabilistic loop induces a unique measurable space $(\Omega^{\text{loop}}, \mathcal{F}^{\text{loop}})$ with sample space $\Omega^{\text{loop}} = \Sigma^{\omega}$, i.e., it is the set of all *infinite* sequences of program states (so-called *runs*). For $\vartheta \in \Omega^{\text{loop}}$, we denote by $\vartheta[n]$ the n-th state in the sequence ϑ (starting with index 0). The σ-field $\mathcal{F}^{\text{loop}}$ is the smallest σ-field that contains all *cylinder sets* $\text{Cyl}(\pi) = \left\{ \pi\vartheta \;\middle|\; \vartheta \in \Omega^{\text{loop}} \right\}$ for

all finite prefixes $\pi \in \Sigma^+$, denoted by

$$\mathcal{F}^{\text{loop}} = \left\langle \left\{ \text{Cyl}(\pi) \mid \pi \in \Sigma^+ \right\} \right\rangle_\sigma .$$

A detailed introduction of the notation $\langle \cdot \rangle_\sigma$ is given in App. A.

Example 14.4.5. In the setting of Ex. 14.4.2, $\text{Cyl}((1,5)(1,6)(6,7)) \subseteq (\mathbb{Z}^2)^\omega$ consists of all infinite sequences starting with $(1,5)(1,6)(6,7)$.

Intuitively, a run $\vartheta \in \Omega^{\text{loop}}$ is an infinite sequence of states $\vartheta = s_0 s_1 s_2 s_3 \cdots$ where s_0 represents the initial state on which the loop is started and s_i is a state that could be reached after i iterations of the loop. Obviously, some sequences in Ω^{loop} may not actually be *admissible* by the loop.

We next develop a canonical probability measure corresponding to the execution of a probabilistic loop, which assign the measure 0 to *non-admissible runs*. We start with considering a single loop iteration. Since the body $\mathbf{P}$ of the loop is AST, it induces the *family* of distributions (see Def. 14.2.12)

$$\mu^{\mathbf{P}}_{\bullet,\bullet} : \Sigma \times \text{Scheds} \longrightarrow \Sigma \longrightarrow [0,1],$$

where $\mu^{\mathbf{P}}_{s,\mathfrak{s}}(s')$ is the probability that after *one* iteration of $\mathbf{P}$ on s, when resolving the non-determinism with $\mathfrak{s}$, the program is in state s'. A probabilistic loop then induces a *family* of so-called probability measures on $(\Omega^{\text{loop}}, \mathcal{F}^{\text{loop}})$.

Definition 14.4.6 (Probability Measure)**.** Let $(\Omega, \mathcal{F})$ be a measurable space. A *measure* is a mapping $m : \mathcal{F} \longrightarrow \overline{\mathbb{R}}_{\geq 0}$ such that

$$m(\emptyset) = 0 \quad \text{and} \quad m\left(\biguplus_{i \in \mathbb{N}} F_i\right) = \sum_{i \in \mathbb{N}} m(F_i) .$$

If $m(\Omega) = 1$, then m is a *probability measure*.

The family of probability measures induced by a probabilistic loop is parameterized by the initial state of the loop and the chosen scheduler.

Using the distributions $\mu^{\mathbf{P}}_{\bullet,\bullet}$ above, we first define the probability of a finite non-empty prefix of a run, i.e., for $\pi \in \Sigma^+$.

Definition 14.4.7 (Loop Measure for Finite Prefixes)**.** The family of distributions $p^{\text{loop}}_{\cdot,\cdot} : \Sigma \times \text{Scheds} \to \Sigma^+ \to [0,1]$ on Σ^+ is defined by

(i) $p^{\text{loop}}_{s,\mathfrak{s}}(s') = [s = s']$

(ii) $p^{\text{loop}}_{s,\mathfrak{s}}(\pi s' s'') = \begin{cases} p^{\text{loop}}_{s,\mathfrak{s}}(\pi s') \cdot [s'' = s'], & \text{if } s' \in \Sigma_{\neg\varphi} \\ p^{\text{loop}}_{s,\mathfrak{s}}(\pi s') \cdot \mu^{\text{P}}_{s',\mathfrak{s}}(s''), & \text{if } s' \in \Sigma_{\varphi}. \end{cases}$

A prefix $\pi \in \Sigma^+$ is *admissible* for $s \in \Sigma$ and $\mathfrak{s} \in \text{Scheds}$ if $p^{\text{loop}}_{s,\mathfrak{s}}(\pi) > 0$. A run is admissible for $s \in \Sigma$ and $\mathfrak{s} \in \text{Scheds}$ if *all* its finite prefixes are admissible.

Here, $p^{\text{loop}}_{s,\mathfrak{s}}(\pi)$ is the probability that π is the sequence of states reached after the first loop iterations, when starting the loop in state s and resolving the non-determinism with scheduler $\mathfrak{s}$.

Example 14.4.8. In the context of Ex. 14.4.5, we have $p^{\text{loop}}_{(1,5)}((1,5)(1,6)(6,7)) = 1 \cdot \frac{6}{11} \cdot \frac{1}{22} = \frac{3}{121}$, i.e., this prefix is admissible for $(1,5)$. For example, the prefix $(0,5)(3,9)$ is non-admissible for $(1,5)$.

Using the family $p^{\text{loop}}_{\cdot,\cdot}$, we now obtain a canonical probability measure on the loop space. This is an application of "Kolmogorov's Extension Theorem" [Fel71].

Lemma 14.4.9 (Loop Measure, [Fel71])**.** *There exists a unique family of probability measures* $\mathbb{P}^{loop}_{\cdot,\cdot} : \Sigma \times \text{Scheds} \to \mathcal{F}^{loop} \to [0,1]$ *such that for each* $\pi \in \Sigma^+$ *we have*

$$\mathbb{P}^{loop}_{s,\mathfrak{s}}(\text{Cyl}(\pi)) = p^{loop}_{s,\mathfrak{s}}(\pi).$$

Example 14.4.10. We have $\mathbb{P}^{\text{loop}}_{(1,5)}(\text{Cyl}((1,5)(1,6)(6,7))) = \frac{3}{121}$ for the loop in Fig. 14.1 by Ex. 14.4.8.

We now turn to random variables and their expected values. Here, we use integrals for arbitrary measures. An introduction into this topic is given in App. A and [Bau11a].

Definition 14.4.11 (Random Variable)**.** A mapping $X : \Omega \to \overline{\mathbb{R}}_{\geq 0}$ on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is called $(\mathcal{F}\text{-})$*measurable* or *random variable* if for any open set $U \subseteq \overline{\mathbb{R}}_{\geq 0}$ its preimage lies in $\mathcal{F}$, i.e., $X^{-1}(U) \in \mathcal{F}$. A mapping $X : \Omega \to$

$\overline{\mathbb{N}} = \mathbb{N} \cup \{\omega\}$ is a random variable if $X^{-1}(\{n\}) \in \mathcal{F}$ for any $n \in \mathbb{N}$. The *expected value* $\mathbb{E}(X)$ of a random variable X is defined as $\mathbb{E}(X) = \int_\Omega X d\mathbb{P}$. If X takes only countably many values we have

$$\mathbb{E}(X) = \int_\Omega X d\mathbb{P} = \sum_{r \in X(\Omega)} \mathbb{P}\left(X^{-1}(\{r\})\right) \cdot r.$$

For the computation of expected values we identify ω with ∞.

We saw that a probabilistic loop gives rise to a unique canonical measurable space $(\Omega^{\text{loop}}, \mathcal{F}^{\text{loop}})$ and to a family of probability measures $\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}$ parameterized by the initial state s on which the loop is started and the scheduler $\mathfrak{s}$ which is used for resolving the non-determinism. We now define a corresponding parameterized expected value operator $\mathbb{E}^{\text{loop}}_{\cdot,\cdot}$.

Definition 14.4.12 (Expected Value $\mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}$ for Loops)**.** Let $X : \Omega^{\text{loop}} \to \overline{\mathbb{R}}_{\geq 0}$ (resp. $X : \Omega^{\text{loop}} \to \overline{\mathbb{N}}$) be a random variable. The *expected value of* X w.r.t. the loop measure $\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}$, parameterized by state $s \in \Sigma$ and scheduler $\mathfrak{s} \in \text{Scheds}$, is defined by

$$\mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}(X) = \int_{\Omega^{\text{loop}}} X \, d\left(\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\right).$$

Next, we define a random variable that corresponds to the number of iterations that the loop does until it terminates.

Definition 14.4.13 (Looping Time)**.** The mapping

$$T^{\varphi} : \Omega^{\text{loop}} \to \overline{\mathbb{N}}, \ \vartheta \mapsto \inf\left\{ n \in \mathbb{N} \;\middle|\; \vartheta[n] \in \Sigma_{\neg\varphi} \right\}$$

is a random variable and called the *looping time* of a probabilistic loop of the form in Fig. 14.16. Here, $\inf \emptyset = \omega$.

Note that this definition is the generalization of Def. 8.1.1 to probabilistic loops, where we also consider non-terminating runs.

Example 14.4.14. In the setting of Ex. 14.4.5, we have $T^{h \leq t}((1,5)(1,6)(8,7)(8,7)\cdots) = 2$, while $T^{h \leq t}((1,5)(1,6)(6,7)(6,8)(6,9)\cdots) = \omega$ (i.e., this sequence always sat-

isfies the condition of the loop as the j-th entry is $(6, 5 + j)$ for $j \geq 3$).

The canonical σ-field $\mathcal{F}^{\text{loop}}$ contains infinite runs. But after n iterations of the loop we only know the first $n + 1$ states $s_0 \cdots s_n$ of a run. Gaining knowledge in this successive fashion can be captured by a so-called *filtration* of the σ-field $\mathcal{F}^{\text{loop}}$.

Definition 14.4.15 (Filtration, Stochastic Process)**.** Let $(\Omega, \mathcal{F})$ be a measurable space. A *filtration* of $\mathcal{F}$ is a sequence $(\mathcal{F}_n)_{n\in\mathbb{N}}$ of σ-fields of Ω where $\mathcal{F}_n \subseteq \mathcal{F}_{n+1}$ and $\mathcal{F}_n \subseteq \mathcal{F}$ for each $n \in \mathbb{N}$.

A *stochastic process* $\mathbf{X} = (X_n)_{n\in\mathbb{N}}$, i.e., a sequence of random variables, is *adapted to* $(\mathcal{F}_n)_{n\in\mathbb{N}}$ if X_n is $\mathcal{F}_n$-measurable.

Thus, a filtration approximates $\mathcal{F}$ from below.

Definition 14.4.16 (Loop Filtration)**.** The sequence $(\mathcal{F}_n^{\text{loop}})_{n\in\mathbb{N}}$ is a filtration of $\mathcal{F}^{\text{loop}}$ where

$$\mathcal{F}_n^{\text{loop}} = \left\langle \left\{ \text{Cyl}(\pi) \;\middle|\; \pi \in \Sigma^{n+1} \right\} \right\rangle_\sigma,$$

i.e., $\mathcal{F}_n^{\text{loop}}$ is the smallest σ-field containing $\left\{ \text{Cyl}(\pi) \;\middle|\; \pi \in \Sigma^{n+1} \right\}$.

Note that here $\mathcal{F}^{\text{loop}} = \bigcup_{n\in\mathbb{N}} \mathcal{F}_n^{\text{loop}}$ which is *not* the case for general filtrations.

Next, we recall the notion of *stopping times* from probability theory.

Definition 14.4.17 (Stopping Time)**.** For a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with filtration $(\mathcal{F}_n)_{n\in\mathbb{N}}$, a random variable $T : \Omega \to \overline{\mathbb{N}}$ is called a *stopping time* w.r.t. $(\mathcal{F}_n)_{n\in\mathbb{N}}$ if for every $n \in \mathbb{N}$ we have $T^{-1}(\{n\}) = \{\vartheta \in \Omega \mid T(\vartheta) = n\} \in \mathcal{F}_n$.

Let us reconsider the looping time T^φ and the filtration $(\mathcal{F}_n^{\text{loop}})_{n\in\mathbb{N}}$. In order to decide for a run $\vartheta = s_0 s_1 \cdots \in \Omega^{\text{loop}}$ whether its looping time is n, we only need to consider the states $s_0 \cdots s_n$. Hence, $(T^\varphi)^{-1}(\{n\}) \in \mathcal{F}_n^{\text{loop}}$ for any $n \in \mathbb{N}$ and thus, we have the following corollary.

Corollary 14.4.18. *The random variable T^φ is a* stopping time *w.r.t. $(\mathcal{F}_n^{loop})_{n\in\mathbb{N}}$.*

Note that T^φ does *not* reflect the actual runtime of a probabilistic loop of the form in Fig. 14.16 as it does not take the runtime of the body $\mathbf{P}$ into account. Instead, T^φ only counts the number of *loop iterations* of the "outer loop".

```
while (a ≠ 0) do {
    { a = 0 } [1/2] { skip } ;
    while (x > 0) do {
        { x ← x + 1 } [1/2] { x ← x − 1 } ;
    }
}
```

Figure 14.17: $\mathbf{L}_{olp}$

Example 14.4.19. Consider the race of tortoise and hare in Fig. 14.1. Here, in terms of the ert-semantics, every iteration of the loop consumes four units of time: one for checking whether the condition holds, one for executing the probabilistic choice, one for executing the argument of the probabilistic choice and one for increasing the value of t. Thus, the actual runtime of this loop on some run ϑ is given by $4 \cdot T^{h \leq t}(\vartheta) + 1$. Here, the addend 1 stems from the fact that after the last loop iteration we have to check the condition *once* again to conclude that the loop has terminated. So in this example, the looping time and the actual runtime are only asymptotically equal. Note that this is always the case if the body is itself loop-free.

In Ch. 15, we will present results on verifying lower bounds on expected outcomes and expected runtimes of pGCL-programs. Here, we will sometimes require that the expected value of the looping time is finite. This is a weaker requirement then a finite expected runtime as it only applies to the number of iterations of the outermost loop (since loops contained in **P** are not considered).

Example 14.4.20. Consider the loop $\mathbf{L}_{olp}$ in Fig. 14.17. As the expected runtime of the inner loop is infinite (see Ex. 14.2.18), the overall expected runtime of $\mathbf{L}_{olp}$ is infinite whenever $a \neq 0$ and $x > 0$ initially. Still, the expected looping time, i.e., the expected number of iterations of the outer loop, is indeed finite (see Ex. 15.3.13).

14.4.2 Canonical Stochastic Process

In Ch. 15, we will discuss the connection between weakest preexpectations and stochastic processes in detail. Here, we already present a part of this methodology since it is also used in Ch. 16. As mentioned before, restricted versions of Thm. 14.4.25 and Cor. 14.4.26 to 14.4.28 have been published in [Har+20]. However, while we only considered purely probabilistic pGCL-programs in [Har+20], here we present our unpublished results Thm. 14.4.25 and Cor. 14.4.26 to 14.4.28 to enable a proper discussion on how to transfer our results in [Har+20] to pGCL-programs featuring also non-deterministic choices (see Sect. 15.7).

Let $f, I \in \mathbb{F}$. We now present a canonical stochastic process that captures approximating wp $[\![\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}]\!]\,(f)$ using the expectation I. Later on, we will choose I to be an invariant.

Definition 14.4.21 (Induced Stochastic Process)**.** The *stochastic process induced by* I, denoted $\mathbf{X}^{f,I} = (X_n^{f,I})_{n\in\mathbb{N}}$, is given by

$$X_n^{f,I} : \Omega^{\text{loop}} \to \overline{\mathbb{R}}_{\geq 0},\ \vartheta \mapsto \begin{cases} f\left(\vartheta\left[T^{\varphi}(\vartheta)\right]\right), & \text{if } T^{\varphi}(\vartheta) \leq n \\ I(\vartheta[n+1]), & \text{otherwise.} \end{cases}$$

Now, in what sense does the stochastic process $\mathbf{X}^{f,I}$ capture approximating the weakest preexpectation of our loop w.r.t. f by expectation I? $X_n^{f,I}$ takes as argument a run ϑ of the loop and assigns to ϑ a value as follows: If the loop has reached a final state within n iterations, i.e., a state from $\Sigma_{\neg\varphi}$, it returns the value of the postexpectation f evaluated in that final state. If no such final state is reached within n steps, it simply *approximates the remainder of the run*, i.e.,

$$\vartheta[0] \cdots \vartheta[n]\ \underbrace{\vartheta[n{+}1]\,\vartheta[n{+}2]\,\vartheta[n{+}3]\,\cdots}_{},$$

by returning the value of the expectation I evaluated in $\vartheta[n{+}1]$. We observe that $X_n^{f,I}$ needs at most the first $n+2$ states of a run to determine its value. Thus, $X_n^{f,I}$ is not $\mathfrak{F}_n^{\text{loop}}$-measurable but $\mathfrak{F}_{n+1}^{\text{loop}}$-measurable, as there exist runs that agree on the first $n+1$ states but yield different images under $X_n^{f,I}$. Hence, we shift $(\mathfrak{F}_n^{\text{loop}})_{n\in\mathbb{N}}$ by one.

Definition 14.4.22 (Shifted Loop Filtration)**.** The filtration $(\mathcal{G}_n^{\text{loop}})_{n\in\mathbb{N}}$ of $\mathfrak{F}^{\text{loop}}$ is

defined by

$$\mathcal{G}_n^{\text{loop}} = \mathfrak{F}_{n+1}^{\text{loop}} = \left\langle \left\{ \text{Cyl}(\pi) \;\middle|\; \pi \in \Sigma^{n+2} \right\} \right\rangle_\sigma .$$

T^φ is also a stopping time w.r.t. $(\mathcal{G}_n^{\text{loop}})_{n\in\mathbb{N}}$ as $(T^\varphi)^{-1}(\{n\}) \in \mathfrak{F}_n^{\text{loop}} \subseteq \mathfrak{F}_{n+1}^{\text{loop}} = \mathcal{G}_n^{\text{loop}}$.

Since we shifted the filtration by one, we get the following lemma.

Lemma 14.4.23 (Adaptedness of $\mathbf{X}^{f,I}$)**.** *$\mathbf{X}^{f,I}$ is adapted to $(\mathcal{G}_n^{loop})_{n\in\mathbb{N}}$.*

The loop space, the loop measure, and the induced stochastic process $\mathbf{X}^{f,I}$ are not defined by induction on the number of steps performed in the program. The loop space, for instance, contains *all* infinite sequences of states, whether they are admissible by the loop or not. The loop measure filters out the non-admissible runs and gives them probability 0.

Reasoning by invariants and characteristic functions, on the other hand, *is* inductive, as we will discuss in detail in Ch. 15. We thus relate iterating a characteristic function on I to the stochastic process $\mathbf{X}^{f,I}$. For this, let ${}^{\text{wp}}\Phi_f$ again be the wp-characteristic function of a loop of the form in Fig. 14.16 w.r.t. f, i.e.,

$${}^{\text{wp}}\Phi_f(X) = [\neg\varphi]\cdot f + [\varphi]\cdot \text{wp}\,[\![\mathbf{P}]\!]\,(X)\,.$$

We now develop a first connection between the stochastic process $\mathbf{X}^{f,I}$ and ${}^{\text{wp}}\Phi_f$, which involves the notion of conditional expected values w.r.t. a σ-field (see, e.g., [Bau11b; GS01]), for which we provide some preliminaries here.

Definition 14.4.24 (Conditional Expected Value)**.** Let $(\Omega, \mathfrak{F}, \mathbb{P})$ be a probability space and $\mathcal{G} \subseteq \mathfrak{F}$ a σ-field of Ω. The *conditional expected value* $\mathbb{E}(X \mid \mathcal{G}) : \Omega \to \overline{\mathbb{R}}_{\geq 0}$ is a $\mathcal{G}$-measurable *map* such that for every $G \in \mathcal{G}$ we have

$$\mathbb{E}(X\cdot[G]) = \mathbb{E}(\mathbb{E}(X \mid \mathcal{G})\cdot[G])\,.$$

Therefore, restricted to the set G the conditional expected value $\mathbb{E}(X \mid \mathcal{G})$ and X have the same expected value. Hence, $\mathbb{E}(X \mid \mathcal{G})$ is a random variable that is like X, but for elements that are indistinguishable in the subfield $\mathcal{G}$, i.e., they either are both contained or none of them is contained in a $\mathcal{G}$-measurable set, $\mathbb{E}(X \mid \mathcal{G})$ "distributes the value of X equally" (see also App. A.6 for more details).

Using this concept, we can now relate ${}^{\text{wp}}\Phi_f$ to an expected value. However, in the case the loop contains non-deterministic choices, this relation needs to take into account the scheduler for resolving the non-determinism. In this case, we observe that

wp resolves the non-determinism "ad-hoc" by taking a minimum over the different possibilities (i.e., it extracts the "worst-case"), whereas in a probability space we have fixed a scheduler. Thus, in this case, we have to take the infimum over all schedulers.

Theorem 14.4.25 (Relating $\mathbf{X}^{f,I}$ and ${}^{\mathrm{wp}}\Phi_f$)**.** *For any $n \in \mathbb{N}$ and any $s \in \Sigma$ we have*

$$\inf_{\mathfrak{s} \in \mathrm{Scheds}} \mathbb{E}^{loop}_{s,\mathfrak{s}}\left(X^{f,I}_{n+1} \,\middle|\, \mathcal{G}^{loop}_n\right) = X^{f,{}^{\mathrm{wp}}\Phi_f(I)}_n .$$

Note that both sides in Thm. 14.4.25 are mappings of type $\Omega^{\mathrm{loop}} \longrightarrow \overline{\mathbb{R}}_{\geq 0}$, i.e., the infimum is to be understood as taken pointwise. Moreover, in the proof of Thm. 14.4.25 we have to choose the value of the conditional expectation on null sets appropriately, see App. B.21.

A similar result holds for awp where we take the *supremum* over all schedulers since in the awp-calculus non-determinism is resolved via taking the maximum.

Corollary 14.4.26. *For any $n \in \mathbb{N}$ and any $s \in \Sigma$*

$$\sup_{\mathfrak{s} \in \mathrm{Scheds}} \mathbb{E}^{loop}_{s,\mathfrak{s}}\left(X^{f,I}_{n+1} \,\middle|\, \mathcal{G}^{loop}_n\right) = X^{f,{}^{\mathrm{awp}}\Phi_f(I)}_n .$$

Intuitively, Thm. 14.4.25 and Cor. 14.4.26 express the following: Consider some cylinder $\mathrm{Cyl}(\pi) \in \mathcal{G}^{\mathrm{loop}}_n$, i.e., $\pi = s_0 \cdots s_{n+1} \in \Sigma^{n+2}$ is a sequence of states of length $n+2$. Then, the expected value of $X^{f,I}_{n+1}$ under $\mathbb{P}^{\mathrm{loop}}_{s,\mathfrak{s}}$ is at least the expected value of $X^{f,{}^{\mathrm{wp}}\Phi_f(I)}_n$ and at most the expected value of $X^{f,{}^{\mathrm{awp}}\Phi_f(I)}_n$ under this measure on $\mathrm{Cyl}(\pi)$ independent of the initial state s and the scheduler $\mathfrak{s}$ of the loop. Depending on whether we take the infimum or the supremum over all schedulers, we achieve (in some sense) equality of the expected values. Of course, if $\mathbf{P}$ is purely probabilistic, equality holds in Thm. 14.4.25 as wp = awp.

Thm. 14.4.25 enables us to explain how iterating ${}^{\mathrm{wp}}\Phi_f$ on I relates to an expected value, thus revealing the inductive structure inside the induced stochastic process:

Corollary 14.4.27 (Relating Expected Values of $\mathbf{X}^{f,I}$ and Iterations of ${}^{\mathrm{wp}}\Phi_f$)**.** *For any $n \in \mathbb{N}$, any $\mathfrak{s} \in$ Scheds, and any $s \in \Sigma$*

$$\mathbb{E}^{loop}_{s,\mathfrak{s}}\left(X^{f,I}_n\right) \geq {}^{\mathrm{wp}}\Phi^{n+1}_f(I)(s).$$

Again, the same result with inverted inequality sign holds for awp.

Corollary 14.4.28 (Relating Expected Values of $\mathbf{X}^{f,I}$ and Iterations of ${}^{\mathrm{awp}}\Phi_f$)**.** *For any $n \in \mathbb{N}$, any $\mathfrak{s} \in$ Scheds, and any $s \in \Sigma$*

$$\mathbb{E}^{loop}_{s,\mathfrak{s}}\left(X^{f,I}_n\right) \leq {}^{\mathrm{awp}}\Phi^{n+1}_f(I)(s).$$

Kind	Relation
Martingale	$\mathbb{E}\left(X_{n+1} \mid \mathfrak{F}_n\right) = X_n$
Supermartingale	$\mathbb{E}\left(X_{n+1} \mid \mathfrak{F}_n\right) \leq X_n$
Submartingale	$\mathbb{E}\left(X_{n+1} \mid \mathfrak{F}_n\right) \geq X_n$

Table 14.8: Different Kinds of Martingales

For purely probabilistic pGCL-programs wp and awp coincide, so *equality* also holds in Cor. 14.4.27 and Thm. 14.4.25 if **P** is purely probabilistic.

Finally, we introduce a methodology which is widely used in verification of probabilistic programs: *martingales* (see, e.g., [Kal02]). A martingale is a process which induces a monotonic sequence of its expected values.

Definition 14.4.29 (Martingale)**.** Let $\mathbf{X} = (X_n)_{n\in\mathbb{N}}$ be a stochastic process on a probability space $(\Omega, \mathfrak{F}, \mathbb{P})$ adapted to a filtration $(\mathfrak{F}_n)_{n\in\mathbb{N}}$ of $\mathfrak{F}$. Moreover, let $\mathbb{E}\left(X_n\right) < \infty$ for all $n \in \mathbb{N}$. Then $\mathbf{X}$ is called a *(sub/ super)martingale* w.r.t. $(\mathfrak{F}_n)_{n\in\mathbb{N}}$ if the corresponding relation in Table 14.8 holds.

We will use martingales to prove correctness of rules for inferring bounds on expected values and expected runtimes. Moreover, we will discuss the relation between invariants (visualized in Table 14.7) and martingales (visualized in Table 14.8) in Ch. 15. Additionally, we will also rely on martingale theory to prove bounds on the expected looping time of certain probabilistic loops in Ch. 16.

Chapter 15

Lower Bounds in Probabilistic Program Verification

In this chapter, we present rules for inferring lower bounds on expected outcomes and expected runtimes of probabilistic programs. More precisely, for a probabilistic loop $\mathbf{L}$ we present how to under-approximate $\mathsf{wp}\,[\![\mathbf{L}]\!]\,(f)$ and $\mathsf{ert}\,[\![\mathbf{L}]\!]\,(f)$ where $f \in \mathbb{F}$. Since both the wp- and the ert-calculus are compositional, this yields a way to obtain lower bounds for arbitrary probabilistic programs.

Here, we first discuss existing rules for bounding expected outcomes and expected runtimes of probabilistic loops (Sect. 15.1). We then further study the connection of (purely) probabilistic loops and stochastic processes in Sect. 15.2 using our results from Sect. 14.4. In Sect. 15.3, we present our Optional Stopping Theorem (Thm. 15.3.11) for purely probabilistic loops. This enables us to generalize existing rules for lower bounds on *bounded* expected outcomes (Sect. 15.4) and to discuss why upper bounds are conceptually simpler to verify than lower bounds (Sect. 15.5). Additionally, we can use our Optional Stopping Theorem to develop the first inductive proof rule for verifying lower bounds on the expected runtime of purely probabilistic loops (Thm. 15.6.1 in Sect. 15.6). We close this chapter with a discussion on how to generalize our inductive proof rules for lower bounds to probabilistic loops also featuring non-deterministic choices (Sect. 15.7).

15.1 Bounds on Weakest Preexpectations

For *loop-free* programs $\mathbf{P}$, it is generally straightforward to determine weakest pre-expectations, simply by applying the rules in Table 14.5, which guide us along the

syntax of **P**, see Ex. 14.3.3. Weakest preexpectations of loops, on the other hand, are generally non-computable least fixed points and we often have to content ourselves with some *approximation* of those fixed points.

For us, a sound approximation is either a *lower* or an *upper* bound on the least fixed point. There are in principle two challenges: (1) finding a candidate bound and (2) verifying that the candidate is indeed an upper or lower bound. Problem (1) is often tackled by setting up a template, e.g., a (linear) polynomial with yet-to-determine coefficients. Using solutions to problem (2), one then sets up a Boolean formula such that when instantiating the coefficients in the template with a satisfying assignment of the formula one obtains a correct bound (see, e.g., [ALY20; Cha+18; Fro+20; NCH18; PR04; Wan+19]). Moreover, in Ch. 17, we present a framework for computing upper bounds on the expected runtime and the expected size of so-called probabilistic transition systems where we not only discuss problem (2) but also problem (1). However, in this chapter we mainly focus on problem (2).

15.1.1 Upper Bounds

The *Park induction* principle provides us with a very convenient proof rule for verifying upper bounds. In general, this principle reads as follows:

Theorem 15.1.1 (Park Induction, [Par69]). *Let $(D, \sqsubseteq)$ be a complete lattice and let $\Phi : D \longrightarrow D$ be continuous. Then Φ has a least fixed point in D and for any $I \in D$,*

$$\Phi(I) \sqsubseteq I \qquad \textit{implies} \qquad \operatorname{lfp} \Phi \sqsubseteq I.$$

For Park induction, it would even suffice for Φ to be monotonic, but we consider continuous functions throughout this chapter.

In the realm of weakest preconditions, Park induction gives rise to the following induction principle:

Corollary 15.1.2 (Park Induction for wp, [Kam19; Koz85]). *Let ${}^{\mathrm{wp}}\Phi_f$ be the characteristic function of a probabilistic loop of the form in Fig. 14.16 w.r.t. postexpectation f. Then for any superinvariant $I \in \mathbb{F}$ we have $\operatorname{lfp} {}^{\mathrm{wp}}\Phi_f \preceq I$, i.e.,*

$${}^{\mathrm{wp}}\Phi_f(I) \preceq I \quad \textit{implies} \quad \mathrm{wp}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](f) \preceq I.$$

The striking power of Park induction is its simplicity: Once an appropriate candidate I is found (even though this is usually not an easy task), all we have to do is push it through the characteristic function ${}^{\mathrm{wp}}\Phi_f$ *once* and check whether we went down in our underlying partial order. If this is the case, we have verified that I is indeed an upper bound on the least fixed point and thus on the sought-after weakest preexpectation. Note that this principle works similarly for awp and ert.

while $(a \neq 0)$ **do** {
  $\{\, a \leftarrow 0 \,\}\ \left[\frac{1}{2}\right]\ \{\, b \leftarrow b+1 \,\}$
}

Figure 15.1: $\mathbf{L}_{geo}$

$\overset{\bowtie}{/\!/\!/}\ I$ (see Fig. 14.12(a))
$\overset{\Phi}{/\!/\!/}\ g$ (i.e., $g = [\neg\varphi]\cdot f + [\varphi]\cdot I''$)
while (φ) **do** {
  $\overset{\bowtie}{/\!/\!/}\ I''$ (see Fig. 14.12(a))
  $\overset{\text{wp}}{/\!/\!/}\ I'$ (see Fig. 14.12(a))
  P
  $[\![\ I$ (i.e., we employ I)
}
$/\!/\!/\ f$ (postexpectation is f)

(a) Annotation Style for Loops Using an Invariant I

$\overset{=}{/\!/\!/}\ b + [a \neq 0]$
$\overset{\Phi}{/\!/\!/}\ [a=0]\cdot b + [a\neq 0]\cdot\left(b + \frac{1}{2}\cdot(1+[a\neq 0])\right)$
while $(a \neq 0)$ **do** {
  $\overset{=}{/\!/\!/}\ b + \frac{1}{2}\cdot(1+[a\neq 0])$
  $\overset{\text{wp}}{/\!/\!/}\ \frac{1}{2}\cdot(b+[0\neq 0]+b+1+[a\neq 0])$
  $\{\, a \leftarrow 0 \,\}\ \left[\frac{1}{2}\right]\ \{\, b \leftarrow b+1 \,\}$
  $[\![\ b + [a \neq 0]$
}
$/\!/\!/\ b$

(b) wp-Annotations for Ex. 15.1.3

Figure 15.2: Annotation Style for Loops Using Invariants and Annotations for Ex. 15.1.3: Inside the loop, we push an invariant I (provided externally, denoted by $[\![\ I$) through the loop body and thus obtain I'' which is (possibly an over- or under-approximation of) $\text{wp}\,[\![\mathbf{P}]\!]\,(I)$. Here, "invariant" also refers to sub- and super-invariants. Above the loop head, we then annotate $g = [\neg\varphi]\cdot f + [\varphi]\cdot I''$. In the first line, we establish $g \bowtie I$ for $\bowtie \in \{\preceq, =, \succeq\}$. Note that if $\bowtie \in \{\preceq, =\}$, we have established the precondition of Cor. 15.1.2 since we have then overall established

$$^{\text{wp}}\Phi_f(I) = [\neg\varphi]\cdot f + [\varphi]\cdot \text{wp}\,[\![\mathbf{P}]\!]\,(I) \preceq [\neg\varphi]\cdot f + [\varphi]\cdot I'' = g \preceq I.$$

For reasoning about lower bounds, we will later employ $\bowtie = \succeq$.

Example 15.1.3. Consider the loop $\mathbf{L}_{geo}$, given in Fig. 15.1 where we assume $b \in \mathbb{N}$. Suppose we aim at an upper bound on the expected value of b after executing $\mathbf{L}_{geo}$. Using the annotation style of Fig. 15.2(a), we can annotate the loop $\mathbf{L}_{geo}$ as shown in Fig. 15.2(b), using invariant $I = b + [a \neq 0]$, establishing $\Phi_b(I) = I$ so in particular $\Phi_b(I) \preceq I$ and by Cor. 15.1.2 establishing $\mathrm{wp}\,[\![\mathbf{L}_{geo}]\!]\,(b) \preceq b + [a \neq 0]$. So the expected value of b after termination of $\mathbf{L}_{geo}$ on s is at most $s(b) + [a \neq 0]\,(s)$.

Let us explain why Park induction is sound using the so-called *Tarski-Knaster Fixed Point Theorem*:

Theorem 15.1.4 (Tarski-Knaster Fixed Point Theorem, [Kna28; Tar55])**.** *Let $(D, \sqsubseteq)$ be a complete lattice and $\Phi : D \longrightarrow D$ be continuous. Then the set of fixed points of Φ is a complete lattice. Here, $\inf\{U \in D \mid \Phi(U) \sqsubseteq U\}$ is the least fixed point of Φ and $\sup\{I \in D \mid I \sqsubseteq \Phi(I)\}$ is the greatest fixed point of Φ.*

Again, for this theorem it would even suffice for Φ to be only monotonic.

In our setting, the characterization of the least fixed point in Thm. 15.1.4 proves soundness of Park induction (recall that Φ is continuous, see Thm. 14.3.4). For making a comparison to the lower bound case which we consider later, let us introduce the so-called *Tarski-Kantorovich Principle*:

Theorem 15.1.5 (Tarski-Kantorovich Principle, [JGP00])**.** *Let $(D, \sqsubseteq)$ be a complete lattice, $\Phi : D \longrightarrow D$ be continuous, and let $I \in D$ such that $I \sqsubseteq \Phi(I)$. Then the sequence $I \sqsubseteq \Phi(I) \sqsubseteq \Phi^2(I) \sqsubseteq \Phi^3(I) \sqsubseteq \cdots$ is an* ascending chain *that* converges *to an element*

$$\Phi^{\omega}(I) = \lim_{n \longrightarrow \omega} \Phi^n(I) = \sup\{\Phi^n(I) \mid n \in \mathbb{N}\} \in D,$$

which is a fixed point of Φ. In particular, $\Phi^{\omega}(I)$ is the least fixed point of Φ that is $\sqsupseteq I$.

The well-known Kleene Fixed Point Theorem (see, e.g., [LNS82]), which states that $\mathrm{lfp}\,\Phi = \Phi^{\omega}(\bot)$ where $\bot$ is the least element of D, is a special case of the Tarski-Kantorovich Principle.

15.1.2 Lower Bounds

For verifying lower bounds, we do not have a rule as simple as Park induction available. In particular, for a given complete lattice $(D, \sqsubseteq)$ and continuous function $\Phi : D \longrightarrow D$, the rule for simple lower induction

$$I \sqsubseteq \Phi(I) \qquad \text{implies} \qquad I \sqsubseteq \mathrm{lfp}\,\Phi$$

while $(a \neq 0)$ **do** {
  $\{\, a \leftarrow 0 \,\} \left[\frac{1}{2}\right] \{\, b \leftarrow b+1 \,\}$;
  $k \leftarrow k+1$
}

Figure 15.3: $\mathbf{L}_{cex}$

is *unsound* in general, i.e., a subinvariant does *not* necessarily yield a lower bound. I being a subinvariant or superinvariant thus expresses mainly its *inductive* nature, namely that I is comparable with $\Phi(I)$ w.r.t. the partial order $\sqsubseteq$.

An explanation why simple lower induction is unsound is as follows: By Thm. 15.1.5, we know from $I \sqsubseteq \Phi(I)$ that $\Phi^{\omega}(I)$ is the *least fixed point of* Φ *that is greater than or equal to* I. Since $\Phi^{\omega}(I)$ is a fixed point, $\Phi^{\omega}(I) \sqsubseteq \operatorname{gfp} \Phi$ holds, where $\operatorname{gfp} \Phi$ denotes the greatest fixed point of Φ, but we do not know how I compares to $\operatorname{lfp} \Phi$. We only know that if indeed $I \sqsubseteq \operatorname{lfp} \Phi$ and $I \sqsubseteq \Phi(I)$, iterating Φ on I also converges to $\operatorname{lfp} \Phi$, i.e.,

$$I \sqsubseteq \operatorname{lfp} \Phi \quad \text{and} \quad I \sqsubseteq \Phi(I) \qquad \text{implies} \qquad \Phi^{\omega}(I) = \operatorname{lfp} \Phi.$$

If, however, $I \sqsubseteq \Phi(I)$ and I is *strictly greater* than $\operatorname{lfp} \Phi$, the fixed point $\Phi^{\omega}(I)$ is also strictly greater than $\operatorname{lfp} \Phi$, contradicting soundness of simple lower induction.

While we just illustrated by means of the Tarski-Kantorovich principle why the simple lower induction rule is not sound in general, we should note that the rule is not per se absurd: So-called *metering functions* [Fro+20] basically employ simple lower induction to verify *lower bounds* on runtimes of non-probabilistic programs, see e.g., [Kam19, Thm. 7.18]. For weakest preexpectations, however, simple lower induction is unsound, as illustrated in the following example.

Counterexample 15.1.6. Consider the loop $\mathbf{L}_{cex}$ in Fig. 15.3 with $b, k \in \mathbb{N}$. The characteristic function for postexpectation b is

$$^{\mathrm{wp}}\Phi_b(X) = [a = 0] \cdot b + [a \neq 0] \cdot \tfrac{1}{2} \cdot \big(X\,[a/0] + X\,[b/b+1]\big)\,[k/k+1]\,.$$

As in Ex. 15.1.3, $\mathrm{wp}\,[\![\mathbf{L}_{cex}]\!]\,(b) \;\preceq\; b + [a \neq 0]$. In particular, this upper bound is independent of k.

Let us consider $I' = b + [a \neq 0] \cdot (1 + 2^k)$, which *does* depend on k. Indeed, one can check that $I' = \Phi_b(I')$, i.e., I' is an invariant, so in particular a subinvariant,

$$
\begin{array}{l}
\overset{=}{/\!/\!/}\ b + [a \neq 0] \cdot (1 + 2^k) \\
\overset{\Phi}{/\!/\!/}\ [a = 0] \cdot b + [a \neq 0] \cdot \left(b + \tfrac{1}{2} \cdot \left(1 + [a \neq 0] \cdot \left(1 + 2^{k+1}\right)\right)\right) \\
\textbf{while}\,(\,a \neq 0\,)\ \textbf{do}\ \{ \\
\quad \overset{=}{/\!/\!/}\ b + \tfrac{1}{2} \cdot \left(1 + [a \neq 0] \cdot \left(1 + 2^{k+1}\right)\right) \\
\quad \overset{\text{wp}}{/\!/\!/}\ \tfrac{1}{2} \cdot \left(b + [0 \neq 0] \cdot \left(1 + 2^{k+1}\right) + b + 1 + [a \neq 0] \cdot \left(1 + 2^{k+1}\right)\right) \\
\quad \{\, a \leftarrow 0 \,\}\ \left[\tfrac{1}{2}\right]\ \{\, b \leftarrow b + 1 \,\}\ ; \\
\quad \overset{\text{wp}}{/\!/\!/}\ b + [a \neq 0] \cdot \left(1 + 2^{k+1}\right) \\
\quad k \leftarrow k + 1 \\
\quad [\!]\!]\ b + [a \neq 0] \cdot \left(1 + 2^k\right) \\
\} \\
/\!/\!/\ b
\end{array}
$$

Figure 15.4: wp-Annotations for I' in Counterex. 15.1.6

see Fig. 15.4. If the simple lower induction rule were sound, we would immediately conclude that I' is a *lower bound* on $\text{wp}\,[\![\mathbf{L}_{cex}]\!]\,(b)$, which is clearly false as otherwise

$$I' \preceq \text{wp}\,[\![\mathbf{L}_{cex}]\!]\,(b) \preceq b + [a \neq 0] \prec I'. \quad \lightning$$

The only sound existing rules for verifying lower bounds on possibly unbounded postexpectations f require reasoning on limits of sequences:

Theorem 15.1.7 (ω-Invariants [AP09; Jon90; Kam+18; Kam19])**.** *Let $(I_n)_{n\in\mathbb{N}}$ be a sequence of expectations such that $I_0 = 0$, and for all $n \in \mathbb{N}$ we have $I_n \preceq I_{n+1}$ and $I_{n+1} \preceq {}^{\text{wp}}\Phi_f(I_n)$. Then we call the sequence an* (wp-)ω-subinvariant *and we have*

$$\sup_{n\in\mathbb{N}} I_n \preceq \text{wp}\,[\![\textbf{while}\,(\,\varphi\,)\ \textbf{do}\ \{\,\mathbf{P}\,\}]\!]\,(f)\,.$$

Of course, the same results also hold for lower bounds on awp and ert. The main limitations of Thm. 15.1.7 are to establish $I_{n+1} \preceq {}^{\text{wp}}\Phi_f(I_n)$ for all $n \in \mathbb{N}$, i.e., we have

to apply ${}^{\mathrm{wp}}\Phi_f$ in total ω times, and to determine $\sup_{n\in\mathbb{N}} I_n$. Moreover, in [Kam19] the author states: "*in my personal experience, we have never encountered a case where we found a* wp-ω-*subinvariant* I_n *that truly under-approximated* ${}^{\mathrm{wp}}\Phi_n^f(0)$. *Instead, we were always able to prove* $I_n = {}^{\mathrm{wp}}\Phi_n^f(0)$."

15.1.3 Problem Statement

The purpose of this chapter is to present a *sound lower induction rule* of the following form: Let ${}^{\mathrm{wp}}\Phi_f$ be the characteristic function of a probabilistic loop of the form in Fig. 14.16 w.r.t. f and let $I \in \mathbb{F}$. Then

$$I \preceq {}^{\mathrm{wp}}\Phi_f(I) \wedge \begin{array}{c}\text{some side}\\ \text{conditions}\end{array} \qquad \text{implies} \qquad I \preceq \mathrm{lfp}\, {}^{\mathrm{wp}}\Phi_f\,.$$

We still want our lower induction rule to be simple in the sense that checking the side conditions should be conceptually as simple as checking $I \preceq {}^{\mathrm{wp}}\Phi_f(I)$. Intuitively, we want to apply the semantics of the loop body only *finitely often*, not ω times, to avoid reasoning about limits of sequences or anything alike as in Thm. 15.1.7. We provide such side conditions in the main result of this chapter, Thm. 15.3.11, which transfers the Optional Stopping Theorem from probability theory (see Thm. 15.3.5) to weakest preexpectation reasoning.

15.1.4 Uniform Integrability

We now present a sufficient and necessary criterion to under-approximate the least fixed points that we seek for. Let again ${}^{\mathrm{wp}}\Phi_f$ be the characteristic function of a probabilistic loop of the form in Fig. 14.16 w.r.t. f. Thm. 14.3.4 implies that ${}^{\mathrm{wp}}\Phi_f$ is continuous and monotonic.

Let us consider a *subinvariant* I, i.e., $I \preceq {}^{\mathrm{wp}}\Phi_f(I)$. If we iterate ${}^{\mathrm{wp}}\Phi_f$ on I ad infinitum, the Tarski-Kantorovich principle (Thm. 15.1.5) guarantees that we converge to some fixed point ${}^{\mathrm{wp}}\Phi_f^\omega(I)$ that is $\succeq I$. From continuity of ${}^{\mathrm{wp}}\Phi_f$ and Thm. 15.1.5, one can easily show that ${}^{\mathrm{wp}}\Phi_f^\omega(I)$ coincides with $\mathrm{lfp}\, {}^{\mathrm{wp}}\Phi_f$ *if and only if* I itself was already $\preceq \mathrm{lfp}\, {}^{\mathrm{wp}}\Phi_f$, i.e.:

Theorem 15.1.8 (Subinvariance and Lower Bounds)**.** *For any* subinvariant I*, we have*

$$ {}^{\mathrm{wp}}\Phi_f^\omega(I) = \mathrm{lfp}\, {}^{\mathrm{wp}}\Phi_f \qquad \textit{iff} \qquad I \preceq \mathrm{lfp}\, {}^{\mathrm{wp}}\Phi_f.$$

More generally, for *any* expectation X (not necessarily a sub- or superinvariant), if iterating ${}^{\mathrm{wp}}\Phi_f$ on X converges to the least fixed point of ${}^{\mathrm{wp}}\Phi_f$, we call X *uniformly integrable for* f:

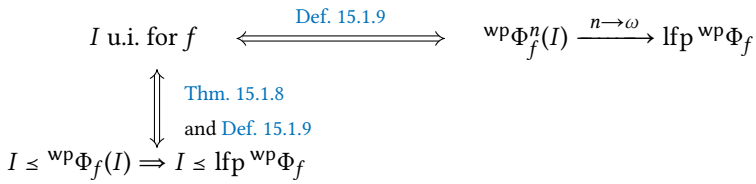


Figure 15.5: Uniform Integrability of Expectations and Lower Bounds

Definition 15.1.9 (Uniform Integrability of Expectations)**.** Given a probabilistic loop, an expectation $X \in \mathbb{F}$ is called *uniformly integrable (u.i.) for* $f \in \mathbb{F}$ if $\lim_{n\to\omega} \mathrm{wp}\Phi_f^n(X)$ exists and

$$\lim_{n\to\omega} \mathrm{wp}\Phi_f^n(X) = \mathrm{lfp}\,\mathrm{wp}\Phi_f.$$

Example 15.1.10. Let $f \in \mathbb{F}$ and $\mathbf{L}$ some probabilistic loop with body $\mathbf{P}$. Then the expectation 0 is uniformly integrable for f: by Thm. 15.1.5 we have

$$\lim_{n\to\omega} \mathrm{wp}\Phi_f^n(0) = \mathrm{lfp}\,\mathrm{wp}\Phi_f.$$

So far, we have thus established the diagram in Fig. 15.5 which we will gradually extend over the next two sections.

Uniform integrability [GS01] – a notion originally from probability theory – will be essential for the Optional Stopping Theorem in Sect. 15.3.2. While we have studied the function $\mathrm{wp}\Phi_f$ solely from an expectation transformer point of view and defined a purely expectation-theoretical notion of uniform integrability without any reference to probability theory in this section, we continue our study of the function $\mathrm{wp}\Phi_f$ from a stochastic process point of view in Sect. 15.2 which we have started in Sect. 14.4. As already seen, stochastic processes are not inductive per se, whereas expectation transformers make heavy use of induction. Still, we have already rediscovered the inductiveness also in the realm of stochastic processes via conditional expectations (see Thm. 14.4.25 and Cor. 14.4.26). We will now also see how our notion of uniform integrability corresponds to uniform integrability in its original sense.

15.2 Expectations and the Stopped Canonical Process

In this section, we continue our study of the connection of the denotational semantics for probabilistic loops of the form Fig. 14.16 via the wp-transformer and stochastic processes which we have already started in Sect. 14.4. Thus, we again fix two expectations $f, I \in \mathbb{F}$ and consider the stochastic process $\mathbf{X}^{f,I}$ (see Def. 14.4.21) for a fixed probabilistic loop $\mathbf{L}$ as in Fig. 14.16 (recall that we assume that the body is AST). Here, $X_n^{f,I}$ captures the following behavior: If a run ϑ terminates within the first n loop iterations, $X_n^{f,I}(\vartheta) = f(\vartheta[T^\varphi(\vartheta)])$ and otherwise $X_n^{f,I}(\vartheta) = I(\vartheta[n+1])$.

In Cor. 14.4.27 and 14.4.28 we have compared the expected value of the random variable $X_n^{f,I}$ under the probability measure $\mathbb{P}_{s,\mathbf{S}}^{\text{loop}}$ with iterating ${}^{\text{wp}}\Phi_f$ a total of $n+1$ times on I. Intuitively, ${}^{\text{wp}}\Phi_f^{n+1}$ represents allowing for at most $n+1$ evaluations of the condition of the loop. For any state $s \in \Sigma$, the number ${}^{\text{wp}}\Phi_f^{n+1}(I)(s)$ is composed of

(i) f's (worst-case) average value on the final states of those runs ϑ starting in s such that one of the states $\vartheta[0]$, ..., $\vartheta[n+1]$ violates the condition, and

(ii) I's (worst-case) average value on the $(n+2)$-nd states of those runs ϑ starting in s such that *all* of the states $\vartheta[0]$, ..., $\vartheta[n+1]$ *satisfy* the condition.

We now want to take n to the limit by considering all possible numbers of iterations of the body of the loop. We show that this corresponds to evaluating the stochastic process $\mathbf{X}^{f,I}$ at the time when our loop terminates, i.e., the looping time T^φ. Therefore, we define this concept of stopping the process at a "random time" formally. Here, we choose the value of the stopped process to be zero on non-terminating runs in accordance to the wp-calculus since only terminating runs contribute to wp $[\![\mathbf{L}]\!](f)$.

Definition 15.2.1 (Stopped Canonical Process)**.** The mapping

$$X_{T^\varphi}^{f,I} : \Omega^{\text{loop}} \longrightarrow \overline{\mathbb{R}}_{\geq 0}, \vartheta \mapsto \begin{cases} X_{T^\varphi(\vartheta)}^{f,I}(\vartheta) = f(\vartheta[T^\varphi(\vartheta)]), & \text{if } T^\varphi(\vartheta) \in \mathbb{N} \\ 0, & \text{otherwise} \end{cases}$$

is the *stopped process*, corresponding to $\mathbf{X}^{f,I}$ stopped at stopping time T^φ. As this mapping is independent of I, we write $X_{T^\varphi}^{f}$ instead of $X_{T^\varphi}^{f,I}$.

The stopped process now corresponds exactly to the quantity we want to reason about. Given some run ϑ let us assume that $T^\varphi(\vartheta) \in \mathbb{N}$, i.e., ϑ terminates:

$$\vartheta = \vartheta[0] \cdots \vartheta[T^\varphi(\vartheta)]\, \vartheta[T^\varphi(\vartheta)+1] \cdots$$

In this case, the stopped process $X^{f,I}_{T^\varphi}$ applied to ϑ evaluates the postexpectation f in the final state after termination, i.e.,

$$X^{f,I}_{T^\varphi}(\vartheta) = f(\vartheta[T^\varphi(\vartheta)]).$$

However, for a non-terminating run ϑ, i.e., $T^\varphi(\vartheta) = \omega$, we have $X^{f,I}_{T^\varphi}(\vartheta) = 0$ as there exists no final state and thus no final state in which to evaluate f. As said before, this is in line with the (a)wp-calculus: here, only terminating runs contribute to the (angelic) weakest preexpectation.

We now show that the limit of the induced stochastic process $\mathbf{X}^{f,I}$ corresponds to the stopped process $X^f_{T^\varphi}$. For the next lemma, we first recall another definition from probability theory (see, e.g., [Bau11b; GS01; Kal02]).

Definition 15.2.2 (Almost-Sure Property)**.** A statement α over runs holds *almost-surely* in the probability space $(\Omega^{\text{loop}}, \mathfrak{F}^{\text{loop}}, \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}})$, if the set of all elements of the sample space satisfying α has probability 1, i.e.,

$$\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\left(\left\{\, \vartheta \in \Omega^{\text{loop}} \;\middle|\; \vartheta \text{ satisfies } \alpha \,\right\}\right) = 1.$$

Lemma 15.2.3 (Convergence of $\mathbf{X}^{f,I}$ to $X^f_{T^\varphi}$)**.** *The stochastic process* $\mathbf{X}^{f,I} \cdot \left[(T^\varphi)^{-1}(\mathbb{N})\right]$ *converges pointwise to* $X^f_{T^\varphi}$*, i.e., for all* $\vartheta \in \Omega^{loop}$ *we have*

$$\lim_{n\to\omega} X^{f,I}_n(\vartheta) \cdot \left[(T^\varphi)^{-1}(\mathbb{N})\right](\vartheta) \;=\; X^f_{T^\varphi}(\vartheta).$$

So if the underlying probabilistic loop is AST, $\mathbf{X}^{f,I}$ *converges to* $X^f_{T^\varphi}$ *almost-surely w.r.t. the probability measure* $\mathbb{P}^{loop}_{s,\mathfrak{s}}$ *for any* $s \in \Sigma$ *and* $\mathfrak{s} \in \mathsf{Scheds}$.

Intuitively, the factor $\left[(T^\varphi)^{-1}(\mathbb{N})\right](\vartheta)$ selects those runs ϑ where the looping time T^φ is finite. If the loop is AST, this factor can be neglected because then $\left[(T^\varphi)^{-1}(\mathbb{N})\right]$ *is* the constant function 1 for the probability measures $\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}$. In any case, (i.e., whether the looping time is almost-surely finite or not) the expected value of the *stopped process* is strongly related to the (angelic) weakest preexpectation of our loop with respect to the postexpectation f: only terminating runs are taken into account by both $X^f_{T^\varphi}$ and $\mathrm{lfp}^{\,\mathsf{wp}}\Phi_f$ (resp. $\mathrm{lfp}^{\,\mathsf{awp}}\Phi_f$) when computing the expected value of f after termination of the loop. From Cor. 14.4.27 and Lemma 15.2.3 we get the first main result of this chapter:

Theorem 15.2.4 (Weakest Preexpectation and Expected Value of Stopped Process)**.** *Let* $\mathfrak{s} \in \mathsf{Scheds}$ *and* $\mathbf{L}$ *be a probabilistic loop of the form in Fig. 14.16 whose body is AST.*

$$\mathsf{wp}\,[\![\mathbf{L}]\!]\,(f) = \mathrm{lfp}^{\,\mathsf{wp}}\Phi_f \preceq \lambda s.\ \mathbb{E}^{loop}_{s,\mathfrak{s}}\left(X^f_{T^\varphi}\right) \preceq \mathrm{lfp}^{\,\mathsf{awp}}\Phi_f = \mathsf{awp}\,[\![\mathbf{L}]\!]\,(f)\,.$$

Thm. 15.2.4 relates our sought-after least fixed point with the expected value of the stopped canonical process. As explained before, given any terminating run ϑ applying $X^f_{T^\varphi}$ to this run yields the value of f at termination of ϑ. Thus, when taking the expected value of $X^f_{T^\varphi}$ w.r.t. $\mathbb{P}^{\mathrm{loop}}_{s,\mathfrak{s}}$ we get the expected value of f after termination of the loop when the loop is started in s and non-deterministic choices are resolved by $\mathfrak{s}$. However, in the realm of weakest preexpectation transformers we do not have an equivalent to schedulers. We either resolve non-deterministic choices by taking min (in case of wp), i.e., we take the infimum over all schedulers, or we resolve them by taking max (in case of awp), i.e., we take the supremum over all schedulers. Therefore, $\mathsf{wp}\,[\![\mathbf{L}]\!]\,(f)\,(s)$ is at most $\mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^f_{T^\varphi}\right)$ and $\mathsf{awp}\,[\![\mathbf{L}]\!]\,(f)\,(s)$ is at least $\mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^f_{T^\varphi}\right)$ for any $s \in \Sigma$ and $\mathfrak{s} \in \mathsf{Scheds}$.

Note that in case of purely probabilistic loops awp and wp coincide, i.e., in this case the least fixed point *is* the expected value of the stopped canonical process. This is what will allow us to later apply the Optional Stopping Theorem. Moreover, it is crucial for deriving our generalization of an existing rule for lower bounds (see Sect. 15.4) and the connection of upper bounds to the Lemma of Fatou (see Sect. 15.5) for purely probabilistic loops.

15.3 Inductive Proof Rules for Lower Bounds on Weakest Preexpectations

In this section, we develop inductive proof rules for lower bounds on weakest preexpectations by using the results of Sect. 14.4 and 15.2 and the Optional Stopping Theorem (Thm. 15.3.5). Recall that we have fixed a loop of the form in Fig. 14.16, a finite postexpectation $f \in \mathbb{F}$, the corresponding characteristic function $^{\mathsf{wp}}\Phi_f$, and another finite expectation I which will play the role of a subinvariant later.

Here, we restrict ourselves to *purely probabilistic* loops, i.e., the body $\mathbf{P}$ does not contain non-deterministic choices so wp and awp coincide. We will discuss the reason for this restriction in Sect. 15.7. To give a simplified justification, for transferring the Optional Stopping Theorem to probabilistic programs we need *equality* in Thm. 15.2.4.

15.3.1 Uniform Integrability

As we will see in Sect. 15.3.2, *uniform integrability* of a certain stochastic process is the central aspect of the Optional Stopping Theorem (Thm. 15.3.5). In probability theory, uniform integrability means that taking the expected value and taking the limit of a converging stochastic process commute (see, e.g., [GS01]).

Definition 15.3.1 (Uniform Integrability of Stochastic Processes)**.** Let $\mathbf{X} = (X_n)_{n\in\mathbb{N}}$ be a stochastic process on a probability space $(\Omega, \mathfrak{F}, \mathbb{P})$ such that $\lim_{n\to\omega} X_n$ exists almost-surely. The process $\mathbf{X}$ is *uniformly integrable (u.i.)* if

$$\mathbb{E}\left(\lim_{n\to\omega} X_n\right) = \lim_{n\to\omega} \mathbb{E}\left(X_n\right).$$

The following counterexample from [GS01] illustrates that uniform integrability of a stochastic process cannot be taken for granted.

Counterexample 15.3.2. Consider a stochastic process $\mathbf{Y} = (Y_n)_{n\in\mathbb{N}}$ on a probability space $(\Omega, \mathfrak{F}, \mathbb{P})$ with $\mathbb{P}\left(Y_n = n\right) = \frac{1}{n} = 1 - \mathbb{P}\left(Y_n = 0\right)$. Then $\mathbb{E}\left(Y_n\right) = 1$. Moreover, $\mathbf{Y}$ converges almost-surely to the constant function which maps everything to 0. Therefore, $\mathbf{Y}$ is not u.i. since

$$\lim_{n\to\omega} \mathbb{E}\left(Y_n\right) = \lim_{n\to\omega} 1 = 1 \neq 0 = \mathbb{E}\left(\lim_{n\to\omega} Y_n\right).$$

Note that our notion of uniform integrability of expectations from Def. 15.1.9 coincides with uniform integrability of the corresponding induced stochastic process. In the following result, it suffices that $\mathbb{P}_s^{\text{loop}}\left((T^{\varphi})^{-1}\left(\mathbb{N}\right)\right) = 1$ for any $s \in \Sigma$. But this is equivalent to AST as we required the body of the loop to be AST.

Corollary 15.3.3 (Uniform Integrability of Expectations and Stochastic Processes)**.** *Let the underlying probabilistic loop be AST. Then I is uniformly integrable for f (in the sense of Def. 15.1.9) iff the induced stochastic process $\mathbf{X}^{f,I}$ is uniformly integrable w.r.t. $\mathbb{P}_s^{loop}$ (in the sense of Def. 15.3.1) for every $s \in \Sigma$, i.e.,*

$$\lim_{n\to\omega} {}^{\text{wp}}\Phi_f^n(I) = \text{lfp}\,{}^{\text{wp}}\Phi_f \quad \textit{iff} \quad \mathbb{E}_s^{loop}\left(\lim_{n\to\omega} X_n^{f,I}\right) = \lim_{n\to\omega} \mathbb{E}_s^{loop}\left(X_n^{f,I}\right) \textit{ for all } s \in \Sigma.$$

Cor. 15.3.3 justifies the naming in Def. 15.1.9: an expectation I is uniformly integrable for f iff its induced process $\mathbf{X}^{f,I}$ is uniformly integrable. So, we can now

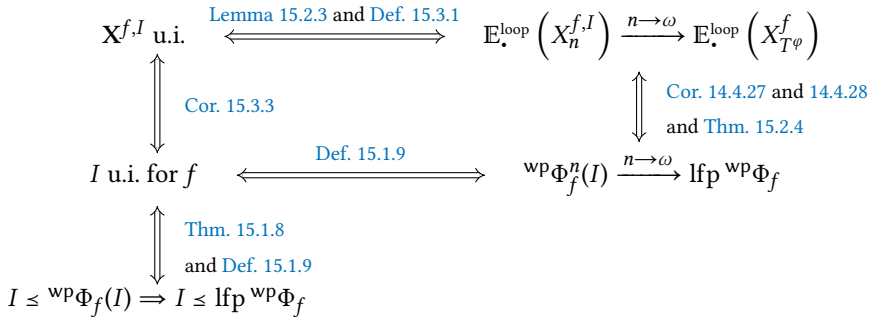


Figure 15.6: Correspondence Between wp-Transformer and Stochastic Processes

extend the diagram in Fig. 15.5 for a purely probabilistic and almost-surely terminating loop as shown in Fig. 15.6.

Uniform integrability is very hard to verify in general, both in the realm of stochastic processes as well as in the realm of expectation transformers. Thus, one usually tries to find *sufficient* criteria for uniform integrability that are easier to verify. The very idea of the Optional Stopping Theorem is to provide such sufficient criteria for uniform integrability which then allow deriving a lower bound, as we will discuss in the next subsection.

15.3.2 The Optional Stopping Theorem of Weakest Preexpectations

We first introduce the Optional Stopping Theorem from probability theory. It builds upon the concept of a submartingale (see Def. 14.4.29), which is a stochastic process that induces a monotonically increasing sequence of its expected values. As already mentioned, submartingales are closely related to subinvariants. In fact, I being a *subinvariant* (plus some side conditions) gives us that the stochastic process induced by I is a *submartingale*.

Lemma 15.3.4 (Subinvariant Induces Submartingale)**.** *Let I be a* subinvariant*, i.e., $I \preceq \text{wp}\Phi_f(I)$, such that $\text{wp}\Phi^n_f(I) \ll \infty$ for every $n \in \mathbb{N}$, that is, $\text{wp}\Phi^n_f(I)$ only takes finite values. Then the induced stochastic process $\mathbf{X}^{f,I}$ is a submartingale w.r.t. $(\mathcal{G}^{loop}_n)_{n\in\mathbb{N}}$.*

Given a submartingale $\mathbf{X} = (X_n)_{n\in\mathbb{N}}$ and a stopping time T, the goal of the Optional Stopping Theorem is to prove a lower bound for the expected value of X_n at the stopping time T. To this end, we define a stochastic process $(X_{n\wedge T})_{n\in\mathbb{N}}$

where for any $\vartheta \in \Omega$, $X_{n\wedge T}(\vartheta) = X_n(\vartheta)$ if n is at most the stopping time $T(\vartheta)$ and otherwise, $X_{n\wedge T}(\vartheta) = X_{T(\vartheta)}(\vartheta)$. Hence, $\mathbb{E}\left(\lim_{n\to\omega} X_{n\wedge T}\right)$ is the expected value of $\mathbf{X}$ at the stopping time T. The Optional Stopping Theorem asserts that the first component X_0 of the stochastic process $\mathbf{X}$ is a lower bound for $\mathbb{E}\left(\lim_{n\to\omega} X_{n\wedge T}\right)$ provided that $(X_{n\wedge T})_{n\in\mathbb{N}}$ is uniformly integrable. Moreover, the Optional Stopping Theorem provides a collection of criteria that are sufficient for uniform integrability of $(X_{n\wedge T})_{n\in\mathbb{N}}$. The version of the Optional Stopping Theorem presented here is a combination of [GS01, Thm. 12.3.(1)], [GS01, Thm. 12.4.(11)], [GS01, Thm. 12.5.(1)], [GS01, Thm. 12.5.(2)], and [GS01, Thm. 12.5.(9)].

Theorem 15.3.5 (Optional Stopping Theorem, [GS01])**.** *Let $(X_n)_{n\in\mathbb{N}}$ be a submartingale and T be a stopping time on a probability space $(\Omega, \mathfrak{F}, \mathbb{P})$ w.r.t. a filtration $(\mathfrak{F}_n)_{n\in\mathbb{N}}$. Then $\mathbf{X}_{\wedge T} = (X_{n\wedge T})_{n\in\mathbb{N}}$ defined by*

$$X_{n\wedge T} : \Omega \longrightarrow \overline{\mathbb{R}}_{\geq 0}, \vartheta \mapsto X_{\min(n,T(\vartheta))}(\vartheta),$$

is also a submartingale w.r.t. $(\mathfrak{F}_n)_{n\in\mathbb{N}}$. If $\mathbf{X}_{\wedge T}$ converges almost-surely and is u.i.,

$$\mathbb{E}(X_0) = \mathbb{E}(X_{0\wedge T}) \leq \lim_{n\to\omega} \mathbb{E}(X_{n\wedge T}) = \mathbb{E}\left(\lim_{n\to\omega} X_{n\wedge T}\right).$$

If one *of the following conditions holds, $\mathbf{X}_{\wedge T}$ converges almost-surely and is u.i.:*

(i) T is almost-surely bounded, i.e., there is a constant $N \in \mathbb{N}$ such that $\mathbb{P}(T \leq N) = 1$.

(ii) $\mathbb{E}(T) < \infty$ and there is a constant $c \in \mathbb{R}_{\geq 0}$ such that for all $n \in \mathbb{N}$

$$\mathbb{E}\left(|X_{n+1} - X_n| \;\middle|\; \mathfrak{F}_n\right) \leq c \qquad \text{holds almost-surely}.$$

(iii) There is a constant $c \in \mathbb{R}_{\geq 0}$ such that $X_{n\wedge T} \leq c$ holds almost-surely for all $n \in \mathbb{N}$.

Our goal is to transfer the Optional Stopping Theorem from probability theory to the realm of weakest preexpectations in order to obtain inductive proof rules for lower bounds on weakest preexpectations. So far, we have introduced the looping time T^{φ} (Def. 14.4.13) which is a stopping time w.r.t. $(\mathcal{G}_n^{\text{loop}})_{n\in\mathbb{N}}$, presented the connection of subinvariants and submartingales (Lemma 15.3.4), and defined the concept of uniform integrability also for expectations (Def. 15.1.9). Hence, the only missing ingredient is a proper connection of expectations to the condition "$\mathbb{E}\left(|X_{n+1} - X_n| \;\middle|\; \mathfrak{F}_n\right) \leq c$" in Thm. 15.3.5 (ii). To translate this concept to expectations, we require that the expectation I has a certain shape depending on the postexpectation f.

Definition 15.3.6 (Harmonization)**.** An expectation I *harmonizes with* $f \in \mathbb{F}$ if there is an expectation $I' \in \mathbb{F}$ such that

$$I = [\neg\varphi] \cdot f + [\varphi] \cdot I'.$$

Def. 15.3.6 reflects that in final states $s \in \Sigma_{\neg\varphi}$ of the loop the invariant I evaluates to $f(s)$. For an invariant I to harmonize with postexpectation f is a minor restriction on the shape of I. It is usually easy to choose an I that takes the value of f for states in which the loop is not executed at all. Furthermore, our goal is to bound $\mathrm{lfp}\,{}^{\mathrm{wp}}\Phi_f$ from below and clearly, $\mathrm{lfp}\,{}^{\mathrm{wp}}\Phi_f(s) = f(s)$ whenever $s \in \Sigma_{\neg\varphi}$. Hence, bounding the least fixed point with an expectation not harmonizing with f is "off-topic". Moreover, performing one iteration of ${}^{\mathrm{wp}}\Phi_f$ obviously brings *any* expectation "into shape":

Corollary 15.3.7. *For any $J \in \mathbb{F}$, ${}^{\mathrm{wp}}\Phi_f(J)$ harmonizes with f.*

The actual criterion that connects "$\mathbb{E}\left(|X_{n+1} - X_n| \;\middle|\; \mathfrak{F}_n\right) \leq c$" with the invariant I is called *conditional difference boundedness* (see also [FH15]):

Definition 15.3.8 (Conditional Difference Boundedness)**.** Let $I \in \mathbb{F}$ with $I \ll \infty$. We define the function $H : \mathbb{F} \longrightarrow \mathbb{F}$ and the expectation $\Delta I \in \mathbb{F}$ as

$$H(X) = [\varphi] \cdot \mathrm{wp}\,[\![\mathbf{P}]\!]\,(X) \qquad \text{and} \qquad \Delta I = \lambda s.\ \big(H\big(|I - I(s)|\big)(s)\big)\,.$$

The expectation I is called *conditionally difference bounded (c.d.b.)* if for some constant $c \in \mathbb{R}_{\geq 0}$

$$\Delta I(s) \leq c \text{ holds for all } s \in \Sigma.$$

Here, $I \ll \infty$ is needed to ensure well-definedness of ΔI. The expectation ΔI expresses the *expected change of* I within one loop iteration. So, if I is c.d.b. it is expected to change at most by a constant in one loop iteration.

Example 15.3.9. Reconsider the loop $\mathbf{L}_{cex}$ shown in Fig. 15.3 from Counterex. 15.1.6 and expectation $I = b + [a \neq 0]$. We check *conditional difference boundedness* of I, using the function H given by

$$H(X) = [a \neq 0] \cdot \tfrac{1}{2} \cdot \big(X\,[a/0] + X\,[b/b+1]\big)\,[k/k+1]\,.$$

We then check the following for any $s \in \Sigma$:

$$\begin{aligned}
&\Delta I(s)\\
&= \left([a \neq 0] \cdot \tfrac{1}{2} \cdot \left(|I - I(s)|\,[a/0] + |I - I(s)|\,[b/b+1]\right)[k/k+1]\right)(s)\\
&= \left([a \neq 0] \cdot \tfrac{1}{2} \cdot \left(|b + [0 \neq 0] - s(b) - [a \neq 0](s)|\right.\right.\\
&\quad \left.\left. + |b + 1 + [a \neq 0] - s(b) - [a \neq 0](s)|\right)\right)(s)\\
&= [a \neq 0](s) \cdot \tfrac{1}{2} \cdot \left(|s(b) - s(b) - [a \neq 0](s)|\right.\\
&\quad \left. + |s(b) + 1 + [a \neq 0](s) - s(b) - [a \neq 0](s)|\right)\\
&= [a \neq 0](s) \cdot \tfrac{1}{2} \cdot \left(|-1| + |1|\right) \;\leq\; 1\,.
\end{aligned}$$

Thus, I is c.d.b. by the constant 1. In contrast, the invariant $I' = b + [a \neq 0] \cdot (1 + 2^k)$ from Counterex. 15.1.6 is not c.d.b. Indeed, we get for any $s \in \Sigma$:

$$\begin{aligned}
&\Delta I'(s)\\
&= \left([a \neq 0] \cdot \tfrac{1}{2} \cdot \left(\left|I' - I'(s)\right|[a/0] + \left|I' - I'(s)\right|[b/b+1]\right)[k/k+1]\right)(s)\\
&= [a \neq 0](s) \cdot \tfrac{1}{2} \cdot \left(\left|s(b) + [0 \neq 0](s) \cdot (2^{s(k)+1} + 1) - (s(b) + [a \neq 0](s) \cdot (2^{s(k)} + 1))\right|\right.\\
&\quad \left. + \left|s(b) + 1 + [a \neq 0](s) \cdot (2^{s(k)+1} + 1) - (s(b) + [a \neq 0](s) \cdot (2^{s(k)} + 1)(s))\right|\right)\\
&= [a \neq 0](s) \cdot \tfrac{1}{2} \cdot \left(2^{s(k)} + 1 + 1 + 2^{s(k)}\right)\\
&= [a \neq 0](s) \cdot \left(2^{s(k)} + 1\right)
\end{aligned}$$

So we have $\Delta I' = \lambda s.\,[a \neq 0](s) \cdot (2^{s(k)} + 1)$ which is unbounded.

Finally, we can connect the expected change of I to a property of the stochastic process $\mathbf{X}^{f,I}$. This is the second major result of this chapter.

Theorem 15.3.10 (Expected Change of I). *Let $I \ll \infty$ harmonize with $f \ll \infty$. For any $s \in \Sigma$*

$$\mathbb{E}_s^{loop}\left(\left|X_{n+1}^{f,I} - X_n^{f,I}\right| \,\middle|\, \mathcal{G}_n^{loop}\right) = X_n^{0,\Delta I}\,.$$

The stochastic process $\mathbf{X}^{f,I}$ induced by I exhibits an interesting correspondence: If ΔI is bounded by a constant c (i.e., if I is c.d.b.), so is $X_n^{0,\Delta I}$ and thus, Thm. 15.3.10 ensures that precondition (ii) of the Optional Stopping Theorem (Thm. 15.3.5) is

fulfilled. Note that Thm. 15.3.10 depends crucially on the fact that $I \ll \infty$ as otherwise the well-definedness of the expectation ΔI cannot be ensured.

Now Lemma 15.3.4 allows us to use the Optional Stopping Theorem from probability theory (Thm. 15.3.5) to prove an *Optional Stopping Theorem for weakest preexpectations*, which collects sufficient conditions for uniform integrability. In particular, due to Thm. 15.3.10, *our* Optional Stopping Theorem shows that our notion of conditional difference boundedness is an (easy-to-check) sufficient criterion for uniform integrability and hence, for ensuring that a subinvariant is indeed a lower bound for the weakest preexpectation under consideration. After stating the theorem, we discuss the intuition of its parts in more detail.

Theorem 15.3.11 (Optional Stopping Theorem for Weakest Preexpectation Reasoning)**.** *Consider a purely probabilistic loop of the form in Fig. 14.16 where* $\mathbf{P}$ *is AST. Let* $I \ll \infty$ *be a subinvariant w.r.t. the postexpectation* $f \ll \infty$ *(i.e.,* $I \preceq {}^{\mathrm{wp}}\Phi_f(I)$*). I is u.i. for f iff I is a* lower bound*, i.e.,*

$$I \text{ is u.i. for } f \quad \text{iff} \quad I \preceq \operatorname{lfp} {}^{\mathrm{wp}}\Phi_f = \mathrm{wp}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](f)\,.$$

I is u.i. *for f if one of the following three conditions holds:*

(i) The looping time T^{φ} is almost-surely bounded, i.e., for every state $s \in \Sigma$ there exists a constant $N(s) \in \mathbb{N}$ with $\mathbb{P}^{loop}_s(T^{\varphi} \preceq N(s)) = 1$, and ${}^{\mathrm{wp}}\Phi^n_f(I) \ll \infty$ for every $n \in \mathbb{N}$.

(ii) The expected looping time $\mathbb{E}^{loop}_s(T^{\varphi})$ is finite for every initial state $s \in \Sigma$, I harmonizes with f, ${}^{\mathrm{wp}}\Phi_f(I) \ll \infty$, and I is conditionally difference bounded.

(iii) Both f and I are bounded and the loop is AST.

We can now extend the diagram in Fig. 15.6 to the one in Fig. 15.7 connecting the realm of stochastic processes (on the right) and the realm of expectation transformers (on the left) for a purely probabilistic and almost-surely terminating loop. The respective Optional Stopping Theorems provide the sufficient criteria for uniform integrability, which is marked by the dashed implications.

Let us elaborate on the different cases of *our* Optional Stopping Theorem (Thm. 15.3.11): Case (i) yields an alternative proof for the technique of so-called metering functions by [Fro+20] for *deterministic terminating* loops. As for the severity of the finiteness condition "${}^{\mathrm{wp}}\Phi^n_f(I) \ll \infty$ for every $n \in \mathbb{N}$", note that if the body $\mathbf{P}$ is loop-free, this condition is vacuously satisfied as I itself is finite and cannot become infinite by finitely many iterations of ${}^{\mathrm{wp}}\Phi_f$. If $\mathbf{P}$ contains loops, we can establish the finiteness condition by finding a finite superinvariant U with

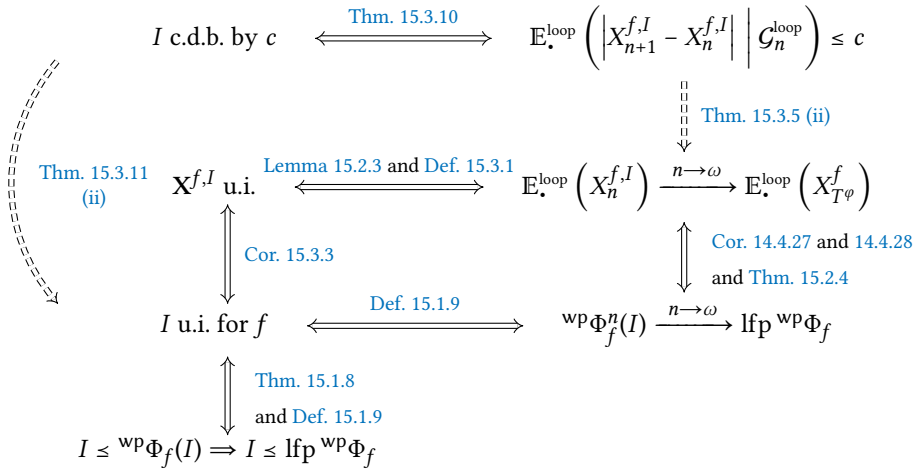


Figure 15.7: Correspondence Between wp-Transformer and Stochastic Processes With Optional Stopping Theorems

$I \preceq U \ll \infty$. In this case, we can also guarantee ${}^{\text{wp}}\Phi^{n}_{f}(I) \ll \infty$: The reason is that we have $U \succeq {}^{\text{wp}}\Phi^{n}_{f}(U)$ for all $n \in \mathbb{N}$ by the monotonicity of ${}^{\text{wp}}\Phi_{f}$ (Thm. 14.3.4). Then, $U \succeq I$ implies ${}^{\text{wp}}\Phi^{n}_{f}(U) \succeq {}^{\text{wp}}\Phi^{n}_{f}(I)$ also by the monotonicity of ${}^{\text{wp}}\Phi_{f}$, which gives us $\infty \gg U \succeq {}^{\text{wp}}\Phi^{n}_{f}(U) \succeq {}^{\text{wp}}\Phi^{n}_{f}(I)$.

Case (ii) applies whenever the outer loop is expected to be executed finitely often. In particular, this holds if the entire loop terminates positively almost-surely (i.e., within finite expected *runtime*). Here, the finiteness condition "${}^{\text{wp}}\Phi_{f}(I) \ll \infty$" is almost negligible: since we have to ensure that I is a subinvariant we have to determine ${}^{\text{wp}}\Phi_{f}(I)$ anyway and thus can simply read off whether it is finite or not.

To the best of our knowledge, Cases (i) and (ii) are the first simple induction rules for lower bounds that do not require restricting to bounded postexpectations f. While the requirements on the termination behavior of the loop gradually weaken along (i) $\longrightarrow$ (ii) $\longrightarrow$ (iii), the requirements on the subinvariant I become stricter.

Finally, Case (iii) yields an alternative proof of the result of [MM05] on inductive lower bounds for bounded expectations in case of AST. We will elaborate on this special case in Sect. 15.4 and present a generalization of the rule from [MM05].

When comparing the cases of Thm. 15.3.5 and Thm. 15.3.11, we notice that

```
while (true) do {
    skip
}
```

Figure 15.8: $\mathbf{L}_{na}$

Thm. 15.3.5 (iii) has no restrictions on the stopping time, whereas Thm. 15.3.11 (iii) requires almost-sure termination. This might spark some hope that AST is not needed in Thm. 15.3.11 (iii), but the following counterexample shows that this is not the case:

Counterexample 15.3.12. Consider the loop $\mathbf{L}_{na}$ in Fig. 15.8 together with the bounded postexpectation $f = 1$, i.e., we are interested in the termination probability which is obviously 0. The corresponding characteristic function is given by

$$\Phi_1(X) = [\neg\text{true}] \cdot 1 + [\text{true}] \cdot \text{wp}\,[\![\mathbf{skip}]\!]\,(X) = X,$$

i.e., Φ_1 is the identity map. Trivially, the bounded expectation $I = 1$ is a fixed point of Φ_1, thus in particular I is a subinvariant. Clearly, I is not a lower bound on the actual termination probability, i.e., on $\text{lfp}\,\Phi_1$. If the condition of almost-sure termination in Thm. 15.3.11 (iii) could be weakened, it has to be ensured that for any loop of the form Fig. 14.16 with almost-surely terminating body $\mathbf{P}$ and postexpectation $f = 1$, $I = 1$ is a lower bound only if the loop terminates almost-surely. Note that for any probabilistic loop with almost-surely terminating body 1 is always a subinvariant for ${}^{\text{wp}}\Phi_1$. Hence, any property which might replace AST in Thm. 15.3.11 (iii) has to be at least as strong as AST.

We reconsider Counterex. 15.1.6 illustrating unsoundness of simple lower induction and do *sound* lower induction instead.

Example 15.3.13. Let us continue Ex. 15.3.9, where we have checked that for the loop $\mathbf{L}_{cex}$ the expectation $I = b + [a \neq 0]$ is c.d.b. by 1. It is easy to check that I is a fixed point of the characteristic function ${}^{\text{wp}}\Phi_b$ w.r.t. the postexpectation b, which by Park induction gives us a finite upper bound on the least fixed point of ${}^{\text{wp}}\Phi_b$ (see Ex. 15.1.3). But up to now, we could not prove that I is indeed equal to the least fixed point. Using Thm. 15.3.11, we can now do this.

First of all, we already have ${}^{\mathrm{wp}}\Phi_b(I) = I \ll \infty$ and since I is a fixed point it is also a subinvariant. Secondly, exprt $[\![\mathbf{L}_{cex}]\!] \preceq 1 + 6 \cdot [a \neq 0] \ll \infty$ and thus, the expected looping time is also finite, which can be verified by Park induction for the ert-calculus. Here, we have

$${}^{\mathrm{ert}}\Phi_0(X) = 1 + [a \neq 0] \cdot \left(1 + \tfrac{1}{2} \cdot (1 + X\,[a, k/0, k+1] + 1 + X\,[b, k/b+1, k+1])\right)$$

Hence, we have

$$\begin{aligned}
&{}^{\mathrm{ert}}\Phi_0(1 + 6 \cdot [a \neq 0]) \\
&= 1 + [a \neq 0] \cdot \Big(1 + \tfrac{1}{2} \cdot \big(1 + (1 + 6 \cdot [a \neq 0])\,[a, k/0, k+1] \\
&\quad + 1 + (1 + 6 \cdot [a \neq 0])\big)\,[b, k/b+1, k+1]\Big) \\
&= 1 + [a \neq 0] \cdot \left(1 + \tfrac{1}{2} \cdot (1 + (1 + 6 \cdot [0 \neq 0]) + 1 + (1 + 6 \cdot [a \neq 0]))\right) \\
&= 1 + [a \neq 0] \cdot \left(1 + \tfrac{1}{2} \cdot (1 + 1 + 1 + 1 + 6)\right) \\
&= 1 + [a \neq 0] \cdot (1 + 5) \\
&= 1 + [a \neq 0] \cdot 6.
\end{aligned}$$

Finally, $I = b + [a \neq 0] = [\neg(a \neq 0)] \cdot b + [a \neq 0] \cdot (b+1)$ harmonizes with b. Hence, the preconditions of Thm. 15.3.11 (ii) are satisfied and I is indeed a lower bound on $\mathrm{lfp}\,{}^{\mathrm{wp}}\Phi_b$. Since I is a fixed point, *it is the least fixed point*, i.e., we have even proved

$$\mathrm{wp}\,[\![\mathbf{L}_{cex}]\!]\,(b) = I.$$

In Counterex. 15.1.6, we have seen that $I' = b + [a \neq 0] \cdot \left(1 + 2^k\right)$ is also a subinvariant but not a lower bound on $\mathrm{lfp}\,{}^{\mathrm{wp}}\Phi_b = \mathrm{wp}\,[\![\mathbf{L}_{cex}]\!]\,(b)$. Note that I' harmonizes with b as well. Since we have seen that I' is not c.d.b. in Ex. 15.3.9, I' does not satisfy the preconditions of Thm. 15.3.11 (ii). Hence, our proof rule sorts out this subinvariant. Note that neither (i) (as the looping time is unbounded) nor (iii) (as neither b nor I' are bounded) are applicable.

There are some other variants of the Optional Stopping Theorem which include further sufficient criteria for uniform integrability (see., e.g., [FC19; Wan+19; WHR21]). We have not included these criteria here since they do not have an expectation-transformer based counterpart. Nevertheless, in the following example we show that these criteria are orthogonal to the ones we have used here.

while $(a \neq 0)$ **do** {
  $\{\, a \leftarrow 0 \,\} \left[\frac{2^b-1}{2^b}\right] \{\, b \leftarrow 2 \cdot b \,\}$
}

Figure 15.9: $\mathbf{L}_{upd}$

Example 15.3.14. Let us consider the loop $\mathbf{L}_{upd}$ with $b \in \mathbb{N}_{\geq 1}$. The characteristic function for $f = b$ of $\mathbf{L}_{upd}$ is given by

$$^{\mathrm{wp}}\Phi_b(X) = [a = 0] \cdot b + [a \neq 0] \cdot \left(\tfrac{2^b-1}{2^b} \cdot X\,[a/0] + \tfrac{1}{2^b} \cdot X\,[b/2 \cdot b] \right).$$

Similarly to Ex. 15.3.13, we have $\mathrm{exprt}\,[\![\mathbf{L}_{upd}]\!] \preceq 1 + 6 \cdot [a \neq 0]$. In particular, $\mathbf{L}_{upd}$ has a finite expected looping time.

Consider $I = f = b = [a = 0] \cdot b + [a \neq 0] \cdot b$, which clearly harmonizes with f. We have

$$\begin{aligned}
&{}^{\mathrm{wp}}\Phi_b(I) \\
&= [a = 0] \cdot b + [a \neq 0] \cdot \left(\tfrac{2^b-1}{2^b} \cdot b + \tfrac{1}{2^b} \cdot 2 \cdot b \right) \\
&= [a = 0] \cdot b + [a \neq 0] \cdot \left(\tfrac{2^b-1+2}{2^b} \cdot b \right) \\
&= [a = 0] \cdot b + [a \neq 0] \cdot \left(\left(1 + \tfrac{1}{2^b} \right) \cdot b \right) \\
&= I + [a \neq 0] \cdot \left(\tfrac{1}{2^b} \cdot b \right) \\
&\succeq I.
\end{aligned}$$

So I is a subinvariant of ${}^{\mathrm{wp}}\Phi_b$ and ${}^{\mathrm{wp}}\Phi_b(I) \ll \infty$. Moreover, we have

$$\begin{aligned}
&\Delta I(s) \\
&= \left([a \neq 0] \cdot \left(\tfrac{2^{b-1}}{2^b} \cdot |I - I(s)|\,[a/0] + \tfrac{1}{2^b} \cdot |I - I(s)|\,[b/2 \cdot b] \right) \right)(s) \\
&= [a \neq 0]\,(s) \cdot \left(\tfrac{2^{s(b)}-1}{2^{s(b)}} \cdot \left| s(b) - s(b) \right| + \tfrac{1}{2^{s(b)}} \cdot \left| 2 \cdot s(b) - s(b) \right| \right)
\end{aligned}$$

$$= [a \neq 0](s) \cdot \tfrac{1}{2^{s(b)}} \cdot |s(b)|$$
$$= [a \neq 0](s) \cdot \tfrac{s(b)}{2^{s(b)}} \cdot \leq 1$$

Consequently, I is c.d.b. by 1. Hence, we can apply Thm. 15.3.11 (ii) which proves that I is uniformly integrable for f, i.e., it is a lower bound on $\operatorname{lfp}^{\mathrm{wp}}\Phi_f$.

In [WHR21, Thm. 4.4] the authors state an extension of the Optional Stopping Theorem Thm. 15.3.5 which among some other conditions requires that for each $n \in \mathbb{N}$ the random variable $X_n^{f,I}$ can be bounded by a polynomial in n. In personal correspondence, they conjectured that this extension generalizes Thm. 15.3.11 (ii). We now show that [WHR21, Thm. 4.4] cannot be used to prove uniform integrability of the induced submartingale $\mathbf{X}^{f,I}$ thus rejecting this conjecture.

Let $n \in \mathbb{N}_{\geq 1}$ and let ϑ be an admissible run of the loop starting in some initial state s_0 satisfying $s_0(a) = 1$ and $s_0(b) = b_0 > 0$ such that $T^{(a\neq 0)}(\vartheta) = n$. More precisely, we have $\vartheta[m](a) = \begin{cases} 1, & \text{if } m < n \\ 0, & \text{if } m = n \end{cases}$ and $\vartheta[m](b) = \begin{cases} 2^m \cdot b_0, & \text{if } m < n \\ 2^{n-1} \cdot b_0, & \text{if } m = n \end{cases}$. So, we have $X_n^{f,I}(\vartheta) = f(\vartheta[n]) = 2^{n-1} \cdot b_0$, which cannot be bounded by a polynomial in n. Moreover, $\mathbb{P}_{s_0}^{\text{loop}}\left(T^{(a\neq 0)} = n\right) > 0$ by construction of the loop. So, $X_n^{f,I}$ cannot be bounded almost-surely by a polynomial in n. Consequently, [WHR21, Thm. 4.4] is not applicable and hence not a generalization of Thm. 15.3.11 (ii).

Moreover, this example also shows that conditional difference boundedness is a weaker requirement than difference boundedness (see., e.g., [CNZ17; FC19; Wan+19]), which would translate to bounding $\left|X_{n+1}^{f,I} - X_n^{f,I}\right|$ by a constant or a polynomial in n. Again, consider a run ϑ of the form described above but this time with $T^{(a\neq 0)}(\vartheta) = n + 3$. Then we have

$$\begin{aligned} &\left|X_{n+1}^{f,I}(\vartheta) - X_n^{f,I}(\vartheta)\right| \\ &= |I(\vartheta[n+2]) - I(\vartheta[n+1])| \\ &= \left|2^{n+2} \cdot b_0 - 2^{n+1} \cdot b_0\right| \\ &= \left|2^{n+1} \cdot b_0\right| \\ &= 2^{n+1} \cdot b_0, \end{aligned}$$

which can neither be bounded by a constant nor by a polynomial in n, i.e., the process $\mathbf{X}^{f,I}$ is not difference bounded.

We have stated that Thm. 15.3.11 only provides *sufficient* criteria for uniform

while $(a \neq 0)$ **do** {
  $\{\, a \leftarrow 0 \,\} \left[\frac{1}{2}\right] \{\, b \leftarrow 2 \cdot b \,\}$
}

Figure 15.10: $\mathbf{L}_{doub}$

integrability. Thus, we conclude this section with an example showing the limitations of our rule. Further case studies demonstrating the effectiveness of our proof rule are provided in [Har+19, App. A].

Example 15.3.15. Let us consider the loop $\mathbf{L}_{doub}$ in Fig. 15.10 with $b \in \mathbb{N}_{\geq 1}$. The characteristic function for $f = b$ of $\mathbf{L}_{doub}$ is given by

$$ {}^{\mathrm{wp}}\Phi_b(X) = [a = 0] \cdot b + [a \neq 0] \cdot \tfrac{1}{2} \cdot \Big(X\,[a/0] + X\,[b/2 \cdot b] \Big). $$

Now consider the sequence of expectations $I_n = [a = 0] \cdot b + [a \neq 0] \cdot \frac{n \cdot b}{2}$. Then $I_n = {}^{\mathrm{wp}}\Phi_f^{n+1}(0)$: $I_0 = [a = 0] \cdot b = {}^{\mathrm{wp}}\Phi_f(0)$. Furthermore,

$$ \begin{aligned} & {}^{\mathrm{wp}}\Phi_f(I_n) \\ = \; & [a = 0] \cdot b + [a \neq 0] \cdot \tfrac{1}{2} \cdot \Big(I_n\,[a/0] + I_n\,[b/2 \cdot b] \Big) \\ = \; & [a = 0] \cdot b + [a \neq 0] \cdot \tfrac{1}{2} \cdot \Big(b + n \cdot b \Big) = I_{n+1} \succeq I_n. \end{aligned} $$

Hence, $I_0 \preceq I_1 \preceq \cdots$ and $[a = 0] \cdot b + [a \neq 0] \cdot \infty = \lim_{n \to \omega} I_n = \mathrm{lfp}\, {}^{\mathrm{wp}}\Phi_f$ by the Tarski-Kantorovich Principle (Thm. 15.1.5). Moreover, each of the subinvariants I_n is a lower bound on the least fixed point, i.e., $I_n \preceq \mathrm{lfp}\, {}^{\mathrm{wp}}\Phi_f = \mathrm{wp}\, [\![\mathbf{L}_{doub}]\!]\,(f)$ (note that the I_n induce an ω-subinvariant, see Thm. 15.1.7).

Similarly to Ex. 15.3.13 and 15.3.14 we have $\mathrm{exprt}\, [\![\mathbf{L}_{doub}]\!] \preceq 1 + 6 \cdot [a \neq 0]$. Clearly, Thm. 15.3.11(i) and (iii) are not applicable since neither the looping time nor the postexpectation is bounded. Let $n \geq 2$. Then Thm. 15.3.11 (ii) is not applicable either because ΔI_n is unbounded, i.e., I_n is not conditionally difference

bounded: let $s \in \Sigma$.

$$
\begin{aligned}
&\Delta I_n(s) \\
&= \left([a \neq 0] \cdot \tfrac{1}{2} \cdot \left(|I_n - I_n(s)|\,[a/0] + |I_n - I_n(s)|\,[b/2 \cdot b] \right) \right)(s) \\
&= [a \neq 0](s) \cdot \tfrac{1}{2} \cdot \Bigg(\Bigg| [0 = 0](s) \cdot s(b) + [0 \neq 0](s) \cdot \frac{n \cdot s(b)}{2} \\
&\quad - \left([a = 0](s) \cdot s(b) + [a \neq 0](s) \cdot \frac{n \cdot s(b)}{2} \right) \Bigg| \\
&\quad + \Bigg| [a = 0](s) \cdot 2 \cdot s(b) + [a \neq 0](s) \cdot \frac{n \cdot 2 \cdot s(b)}{2} \\
&\quad - \left([a = 0](s) \cdot s(b) + [a \neq 0](s) \cdot \frac{n \cdot s(b)}{2} \right) \Bigg| \Bigg) \\
&= [a \neq 0](s) \cdot \tfrac{1}{2} \cdot \left(\left| s(b) - \left([a \neq 0](s) \cdot \frac{n \cdot s(b)}{2} \right) \right| \right. \\
&\quad \left. + \left| [a \neq 0](s) \cdot \frac{n \cdot 2 \cdot s(b)}{2} - \left([a \neq 0](s) \cdot \frac{n \cdot s(b)}{2} \right) \right| \right) \\
&= [a \neq 0](s) \cdot \tfrac{1}{2} \cdot \left(\left| s(b) - \frac{n \cdot s(b)}{2} \right| + \left| n \cdot s(b) - \frac{n \cdot s(b)}{2} \right| \right) \\
&= [a \neq 0](s) \cdot \tfrac{1}{2} \cdot \left(\left| \frac{(2 - n) \cdot s(b)}{2} \right| + \left| \frac{n \cdot s(b)}{2} \right| \right) \\
&= [a \neq 0](s) \cdot \tfrac{1}{2} \cdot \frac{(2 \cdot n - 2) \cdot s(b)}{2} \\
&= [a \neq 0](s) \cdot \frac{(n - 1) \cdot s(b)}{2}
\end{aligned}
$$

which is unbounded as $s(b) \in \mathbb{N}_{\geq 1}$. Hence, Thm. 15.3.11 (ii) *cannot* be applied and thus *cannot* prove that I_n is a lower bound on $\mathrm{wp}\,[\![\mathbf{L}_{doub}]\!]\,(f)$.

15.4 Lower Bound Rules by McIver and Morgan

In Sect. 15.3.2, we briefly mentioned the rules for lower bounds for *bounded* expectations by [MM05] which are restated in Thm. 15.4.1 below. To the best of our knowledge, before our Thm. 15.3.11 these were the only existing *inductive* proof rules for lower

bounds on weakest preexpectations. Again, in this section we restrict ourselves to purely probabilistic loops since we need this for Thm. 15.3.11 to hold. Nevertheless, Thm. 15.4.1 also holds for loops containing non-deterministic choices. The version of Thm. 15.4.1 presented here is a combination of [MM05, Lemma 2.4.1],[MM05, Eq. (2.12)], [MM05, Lemma 7.7.6], [MM05, Thm. 7.3.3], and [MM05, Thm. B.2.2].

Theorem 15.4.1 (Lower Bounds for Bounded Preexpectations, [MM05]). *Let $f \in \mathbb{F}$ be a **bounded** postexpectation. Furthermore, let $I' \in \mathbb{F}$ be a **bounded** expectation such that the harmonizing expectation $I \in \mathbb{F}$ given by $I = [\neg\varphi] \cdot f + [\varphi] \cdot I'$ is a subinvariant of a loop of the form in Fig. 14.16 w.r.t. f. Finally, let $T = \text{wp}\,[\![\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}]\!]\,(1)$ be the termination probability of the loop.*

(i) If $I = [G]$ for some predicate G, then $T \cdot I \preceq \text{wp}\,[\![\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}]\!]\,(f)$.

(ii) If $[G] \preceq T$ for some predicate G, then $[G] \cdot I \preceq \text{wp}\,[\![\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}]\!]\,(f)$.

(iii) If $\varepsilon \cdot I \preceq T$ for some $\varepsilon > 0$, then $I \preceq \text{wp}\,[\![\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}]\!]\,(f)$.

Thm. 15.4.1 does not make any assumptions on the termination behavior of the loop, so it is in principle also possible to analyze loops with termination probability < 1. Still, we should note that in the cases Thm. 15.4.1 (ii) and (iii) the proof in [MM05] of the statement for the non-trivial case, i.e., the case in which the lower bound on the left-hand side does not evaluate to 0 in a given state s, concludes that for this state s the loop indeed terminates with probability 1 by the given preconditions.

It turns out that Thm. 15.4.1 (i) and (ii) can be proved easily from our results from Sect. 15.2 and 15.3 in the case where $\mathbf{P}$ is AST where we do *not* need the restriction that I harmonizes with f. We can also prove Thm. 15.4.1 (iii) but in this case we do *not* need the fact that T is the termination probability but only that $T = \text{wp}\,[\![\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}]\!]\,(f)$ for some *bounded* postexpectation f. Furthermore, if I is a lower bound, by definition $I \preceq \text{wp}\,[\![\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}]\!]\,(f)$. Hence, we have generalized Thm. 15.4.1 (iii) in case of a purely probabilistic loop with almost-surely terminating body to a complete characterization of lower bounds. So, we obtain the following generalized version of Thm. 15.4.1 (iii) in the case where $\mathbf{P}$ is AST and purely probabilistic which is substantially more powerful: it states not only a sufficient condition for a subinvariant to be a lower bound but also a *necessary* condition.

Theorem 15.4.2 (Generalization of Thm. 15.4.1 (iii)). *Let $f \in \mathbb{F}$ be a **bounded** postexpectation. Furthermore, let $I \in \mathbb{F}$ be a **bounded** expectation such that I is a subinvariant of a loop of the form in Fig. 14.16 w.r.t. f where $\mathbf{P}$ is AST and purely probabilistic. There exist $\varepsilon > 0$ and $g \in \mathbb{F}$ **bounded** s.t.*

$$\varepsilon \cdot I \preceq \text{wp}\,[\![\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}]\!]\,(g) \;\; \textit{iff} \;\; I \preceq \text{wp}\,[\![\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}]\!]\,(f)\;.$$

while $(x > 0)$ **do** {
  $\{ x \leftarrow x - 1 \} \left[\frac{1}{3}\right] \{ x \leftarrow x + 1 \}$;
  $y \leftarrow \max(y - 1, 0)$
}

Figure 15.11: $\mathbf{L}_{mm}$

It is worth noting that we have proved this result without making use of Thm. 15.3.11 (iii) but only by using the correspondence between the realm of stochastic processes and expectation transformers as depicted in Fig. 15.6 . In this way, we avoid the restriction to loops which are almost-surely terminating. However, our reasoning in Counterex. 15.3.12 still holds, i.e., Thm. 15.4.2 is, from a theoretical point of view, in general *not* substantially simpler than determining the probability of termination. Nevertheless, in examples Thm. 15.4.2 can indeed be easier to apply than determining the probability of termination.

Example 15.4.3. Let us consider the program $\mathbf{L}_{mm}$ containing an asymmetric random walk in Fig. 15.11 with $x, y \in \mathbb{N}$ and $y \leq 100$. This program is *not* AST (see Cor. 16.3.6) but the body of the loop is indeed AST since it is loop-free. Furthermore, the postexpectation y is bounded. If $y \leq x$ initially, y is 0 after termination of the loop. So, $\mathrm{wp}\,[\![\mathbf{L}_{mm}]\!]\,(y) \geq [y > x] \cdot \left(\frac{1}{3}\right)^x \cdot (y - x) = I$.

Now consider $f = [y \text{ even}] \cdot 200 \cdot y^2 + [y \text{ odd}] \cdot (y+5)^4$. We have $I' \preceq {}^{\mathrm{wp}}\Phi_f(I')$, where $I' = 400 \cdot I$:

$$
\begin{aligned}
& {}^{\mathrm{wp}}\Phi_f(I') \\
= \; & [x = 0] \cdot f + [x > 0] \cdot \Big(\tfrac{1}{3} \cdot I'\,[x, y/x - 1, \max(y - 1, 0)] \\
& + \tfrac{2}{3} \cdot I'\,[x, y/x + 1, \max(y - 1, 0)] \Big) \\
= \; & [x = 0] \cdot f + [x > 0] \cdot 400 \\
& \cdot \Big(\tfrac{1}{3} \cdot [\max(y - 1, 0) > x - 1] \cdot \left(\tfrac{1}{3}\right)^{x-1} \cdot (\max(y - 1, 0) - (x - 1)) \\
& + \tfrac{2}{3} \cdot [\max(y - 1, 0) > x + 1] \cdot \left(\tfrac{1}{3}\right)^{x+1} \cdot (\max(y - 1, 0) - (x + 1)) \Big) \\
= \; & [x = 0] \cdot f + [x > 0] \cdot 400 \cdot \Big(\tfrac{1}{3} \cdot [y - 1 > x - 1] \cdot \left(\tfrac{1}{3}\right)^{x-1} \cdot (y - 1 - (x - 1)) \\
& + \tfrac{2}{3} \cdot [y - 1 > x + 1] \cdot \left(\tfrac{1}{3}\right)^{x+1} \cdot (y - 1 - (x + 1)) \Big)
\end{aligned}
$$

$$
\begin{aligned}
&= [x = 0] \cdot f + [x > 0] \cdot 400 \cdot \Big([y > x] \cdot \left(\tfrac{1}{3}\right)^x \cdot (y - x) \\
&\quad + \tfrac{2}{3} \cdot [y > x + 2] \cdot \left(\tfrac{1}{3}\right)^{x+1} \cdot (y - x - 2) \Big) \\
&= [x = 0] \cdot f + [x > 0] \cdot \Big(I' + \underbrace{400 \cdot \tfrac{2}{3} \cdot [y > x + 2] \cdot \left(\tfrac{1}{3}\right)^{x+1} \cdot (y - x - 2)}_{\geq 0} \Big)
\end{aligned}
$$

If $s(x) > 0$, obviously $I'(s) \leq {}^{\mathrm{wp}}\Phi_f(I')(s)$ by the calculation above. If $s(x) = 0$, we have

$$
\begin{aligned}
&{}^{\mathrm{wp}}\Phi_f(I')(s) \\
&= [x = 0](s) \cdot f(s) \\
&= [y \text{ is even}](s) \cdot 200 \cdot s(y)^2 + [y \text{ is odd}](s) \cdot 200 \cdot (s(y) + 5)^4 \\
&\geq 400 \cdot s(y) \\
&= 400 \cdot [y > x](s) \cdot \left(\tfrac{1}{3}\right)^0 \cdot (s(y) - 0) \\
&= I'(s),
\end{aligned}
$$

as for every even y we have $200 \cdot y^2 \geq 400 \cdot y$ and for every odd y we have $200 \cdot (y + 5)^4 \geq 400 \cdot y$.

As we have $\frac{1}{400} \cdot I' \leq \mathrm{wp}\,[\![\mathbf{L}_{mm}]\!]\,(y)$, we can conclude from Thm. 15.4.2 that $I' \leq \mathrm{wp}\,[\![\mathbf{L}_{mm}]\!]\,(f)$. Note that this is easier than relating I' and the probability of termination as required by Thm. 15.4.1 (iii) since the probability of termination of the loop is independent of y.

Of course, Ex. 15.4.3 is an artificial example. Nevertheless, it shows a strength of our generalization: it makes it easier to reason about bounded expectations which are independent of the probability of termination. However, a drawback of Thm. 15.4.1 remains: one already needs a lower bound, i.e., one has to be able to read off a lower bound directly from the loop.

15.5 Upper Bounds and the Lemma of Fatou

In Sect. 15.1, we saw that Park induction for proving upper bounds does not require additional conditions such as conditional difference boundedness or even boundedness of f or I, respectively. The question arises whether this fact is also explainable using our canonical stochastic process. Indeed, the well-known *Lemma of Fatou* (see, e.g., [Bau11b, Lemma 15.2]) provides such an explanation for *purely probabilistic* loops

(recall that we require the body of the loop to be AST). We present a specialized variant of it which is sufficient for our purpose.

Lemma 15.5.1 (Lemma of Fatou). *Let $(X_n)_{n\in\mathbb{N}}$ be a stochastic process on a probability space $(\Omega, \mathfrak{F}, \mathbb{P})$. Then*

$$\mathbb{E}\left(\lim_{n\to\omega} X_n\right) \leq \lim_{n\to\omega} \mathbb{E}\left(X_n\right).$$

We can now reprove Park induction for wp for a purely probabilistic loop of the form in Fig. 14.16, i.e., the body $\mathbf{P}$ does not contain any non-deterministic choice, using the Lemma of Fatou: Let I be a superinvariant, i.e., ${}^{\mathrm{wp}}\Phi_f(I) \preceq I$. Since for purely probabilistic loops there is no need for considering schedulers, we omit them. By Thm. 14.4.25, the canonical stochastic process $\mathbf{X}^{f,I}$ satisfies

$$\mathbb{E}_s^{\mathrm{loop}}\left(X_{n+1}^{f,I} \,\middle|\, \mathcal{G}_n^{\mathrm{loop}}\right) = X_n^{f,{}^{\mathrm{wp}}\Phi_f(I)} \leq X_n^{f,I}.$$

By applying $\mathbb{E}_s^{\mathrm{loop}}$ on both sides (see Lemma A.6.4), we obtain $\mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I}\right) \geq \mathbb{E}_s^{\mathrm{loop}}\left(X_1^{f,I}\right) \geq \ldots$. This implies

$$\mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I}\right) \geq \mathbb{E}_s^{\mathrm{loop}}\left(X_n^{f,I}\right) \geq \mathbb{E}_s^{\mathrm{loop}}\left(X_n^{f,I} \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right]\right), \tag{15.5.1}$$

as $X_n^{f,I} \geq X_n^{f,I} \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right]$. We conclude

$$\begin{aligned}
&\left(\mathrm{lfp}\,{}^{\mathrm{wp}}\Phi_f\right)(s) \\
&= \mathbb{E}_s^{\mathrm{loop}}\left(X_{T^{\varphi}}^{f}\right) && \text{(by Thm. 15.2.4)} \\
&= \mathbb{E}_s^{\mathrm{loop}}\left(\lim_{n\to\omega} X_n^{f,I} \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right]\right) && \text{(by Lemma 15.2.3)} \\
&\leq \lim_{n\to\omega} \mathbb{E}_s^{\mathrm{loop}}\left(X_n^{f,I} \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right]\right) && \text{(by Lemma 15.5.1)} \\
&\leq \lim_{n\to\omega} \mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I}\right) && \text{(by (15.5.1))} \\
&= \mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I}\right) \\
&= {}^{\mathrm{wp}}\Phi_f(I)(s) && \text{(by Cor. 14.4.27 and 14.4.28)} \\
&\leq I(s), && \text{(since } {}^{\mathrm{wp}}\Phi_f(I) \preceq I\text{)}
\end{aligned}$$

so I is indeed an upper bound on the least fixed point.

Note that here we handle purely probabilistic loops, which are *not* necessarily AST. While I being a superinvariant (plus some side conditions) still implies that $\mathbf{X}^{f,I}$ is a supermartingale, the second part of Lemma 15.2.3 is not applicable, i.e., in general we have $X^f_{T^\varphi} \neq \lim_{n\to\omega} X^{f,I}_n$ if the loop is not AST. So in this case we cannot use classic results from martingale theory. Nevertheless, the Lemma of Fatou combined with Thm. 15.2.4 and the first part of Lemma 15.2.3 provide a connection of Park induction for upper bounds to stochastic processes. Moreover, our reasoning does not translate to loops involving non-determinism since we needed that equality holds in Thm. 14.4.25.

15.6 Lower Bounds on the Expected Runtime

So far, we have developed techniques for verifying lower bounds on weakest preexpectations, i.e., expected values of random variables upon program termination. In this section, we transfer those techniques to verify lower bounds on *expected runtimes* of purely probabilistic pGCL-programs. For this, we employ the ert-transformer as defined in Def. 14.3.8. Again, the behavior (and the runtime) of a pGCL-program depends on its input, so we are actually interested in a *function* $g \in \mathbb{F}$ mapping *initial* states s_0 to the respective expected runtime. In addition, reasoning about upper bounds by Park induction works exactly the same way as for the wp-transformer. For reasoning about lower bounds using subinvariants, we use Thm. 14.3.12 which connects wp and ert. Here, $\mathsf{ert}\,[\![\mathbf{P}]\!]\,(0) = \mathsf{exprt}\,[\![\mathbf{P}]\!]$ is *independent of* t. So, we can combine our rule Thm. 15.3.11 (ii) for lower bounds on wp with Thm. 14.3.12 to establish the first inductive rule for verifying lower bounds on expected runtimes of *purely probabilistic* loops:

Theorem 15.6.1 (Inductive Lower Bounds on Expected Runtimes)**.** *Let $t, I \in \mathbb{F}$ with $t, I \ll \infty$ and let I harmonize with t. Furthermore, let ${}^{\mathsf{ert}}\Phi_t$ be the* ert*-characteristic function of a purely probabilistic loop of the form in Fig. 14.16 with almost-surely terminating body $\mathbf{P}$ w.r.t. t. If I is an* ert*-subinvariant w.r.t. t, ${}^{\mathsf{wp}}\Phi_t(I) \ll \infty$, and I is conditionally difference bounded, we have*

$$I \preceq \mathsf{ert}\,[\![\mathbf{while}\,(\varphi)\,\mathbf{do}\,\{\,\mathbf{P}\,\}]\!]\,(t)\,.$$

We now illustrate the applicability of Thm. 15.6.1 in two case studies: The first deals with the so-called coupon collector's problem which already dates back to the 1930's. The second one illustrates how we can use our rule to *refute* PAST.

$x \leftarrow N$;
while $(x > 0)$ **do** {
  $i \leftarrow N + 1$;
  while $(0 < x < i)$ **do** {
    $i \leftsquigarrow \mathrm{Unif}[1 \dots N]$
  } ;
  $x \leftarrow x - 1$
}

Figure 15.12: $\mathbf{L}_{cc}$

15.6.1 Case Study: Coupon Collector's Problem

Consider the well-known coupon collector's problem [Pól30]: There are N different types of coupons. A collector wants to collect at least one of each type. Each time she buys a new coupon, its type is drawn uniformly at random. How many coupons does she (expectedly) need to buy to have collected at least one coupon of each type?

We can model this problem by the program $\mathbf{L}_{cc}$ given in Fig. 15.12 for some non-zero natural number N, i.e., $N \in \mathbb{N}_{\geq 1}$. Variable x represents the number of uncollected coupon types. The inner loop models the buying of new coupons until an uncollected type is drawn. Here, $\mathrm{Unif}[1 \dots N]$ is the uniform distribution on the natural numbers $1, \dots, N$, i.e., each of these numbers is drawn with probability $\frac{1}{N}$.

The expected runtime of $\mathbf{L}_{cc}$ is proportional to the expected number of coupons the collector needs to buy. We want to prove that $N \cdot \mathcal{H}_N$ is a lower bound on that expected runtime, where $\mathcal{H}_m$ is the m-th *harmonic number*, i.e., $\mathcal{H}_0 = 0$ and $\mathcal{H}_m = \sum_{k=1}^{m} \frac{1}{k}$.

For this purpose, we make use of the expectation

$$I = [0 < x \leq N] \cdot N \cdot \mathcal{H}_x + [N < x] \cdot (N \cdot \mathcal{H}_N + x - N)$$

and show that it is a c.d.b. subinvariant of the outer loop in $\mathbf{L}_{cc}$ w.r.t. the postruntime $t = 0$ and thus a lower bound on its expected runtime. Before we begin proving that I is indeed a lower bound on the expected runtime of the outer loop, let us take a closer look at the meaning of I: If I is a lower bound, the outer loop takes at least expected runtime $N \cdot \mathcal{H}_x$ if x is between 1 and N, and expected runtime $N \cdot \mathcal{H}_N + x - N$ if x is larger than N. In the second case, the too-large x value suggests that we have to collect more coupons than there are different coupons. So we first collect $x - N$ arbitrary "excess coupons" before we enter the "normal coupon collector mode" and

collect the remaining N coupons in expected time $N \cdot \mathcal{H}_N$. Indeed, $N \cdot \mathcal{H}_x$ (without case analysis) is *not* a lower bound on the expected runtime and we would in fact fail to prove its subinvariance.

We make the following annotations, reusing the annotation style of Fig. 15.2(a).

Example 15.6.2.

$$\overset{=}{/\!/\!/}\ 1 + N \cdot \mathcal{H}_N$$
$$\overset{\text{ert}}{/\!/\!/}\ 1 + [0 < N \leq N] \cdot N \cdot \mathcal{H}_N + [N < N] \cdot (N \cdot \mathcal{H}_N + N - N)$$
$$x \leftarrow N\ ;$$
$$\overset{\geq}{/\!/\!/}\ [0 < x \leq N] \cdot N \cdot \mathcal{H}_x + [N < x] \cdot (N \cdot \mathcal{H}_N + x - N)$$
$$\overset{=}{/\!/\!/}\ 1 + [0 < x \leq N] \cdot \left(N \cdot \mathcal{H}_x + 3 + \tfrac{N}{x}\right) + [N < x] \cdot (2 + N \cdot \mathcal{H}_N + x - N)$$
$$\overset{=}{/\!/\!/}\ 1 + [x = 1] \cdot (3 + 2 \cdot N) + [1 < x \leq N] \cdot \left(3 + N \cdot \left(\mathcal{H}_x + \tfrac{1}{x}\right)\right)$$
$$+ [x = N + 1] \cdot (3 + N \cdot \mathcal{H}_N) + [N + 1 < x] \cdot (2 + N \cdot \mathcal{H}_N + x - N)$$
$$\overset{=}{/\!/\!/}\ 1 + [x = 1] \cdot (3 + 2 \cdot N) + [1 < x \leq N] \cdot \left(3 + N \cdot \left(\mathcal{H}_x - \tfrac{1}{x}\right) + \tfrac{2 \cdot N}{x}\right)$$
$$+ [x = N + 1] \cdot \left(3 + N \cdot \left(\mathcal{H}_{N+1} - \tfrac{1}{N+1}\right)\right) + [N + 1 < x] \cdot (2 + N \cdot \mathcal{H}_N + x - N)$$
$$\overset{=}{/\!/\!/}\ 1 + [0 < x] \cdot \left(2 + t + [x \leq N] \cdot \tfrac{2 \cdot N}{x}\right)$$
$$\overset{\Phi}{/\!/\!/}\ 1 + [x \leq 0] \cdot 0 + [0 < x] \cdot \left(2 + t + [0 < x \leq N] \cdot \tfrac{2 \cdot N}{x}\right)$$
while $(0 < x)$ **do** $\{$
$$\overset{=}{/\!/\!/}\ 2 + t + [0 < x \leq N] \cdot \tfrac{2 \cdot N}{x}$$
$$\overset{\text{ert}}{/\!/\!/}\ 1 + 1 + t + [0 < x < N + 1] \cdot 2 \cdot \max\left\{\tfrac{N}{x}, 1\right\}$$
$$i \leftarrow N + 1$$
$$\overset{\text{ert}}{/\!/\!/}\ 1 + t + [0 < x < i] \cdot 2 \cdot \max\left\{\tfrac{N}{x}, 1\right\} \qquad \text{(by Lemma 15.6.3 below)}$$
while $(0 < x < i)$ **do** $\{\ i \leftsquigarrow \text{Unif}[1 \ldots N]\ \}$;
$$\overset{=}{/\!/\!/}\ \underbrace{1 + [1 < x \leq N + 1] \cdot N \cdot \left(\mathcal{H}_x - \tfrac{1}{x}\right) + [N + 1 < x] \cdot (N \cdot \mathcal{H}_N + x - 1 - N)}_{=t}$$
$$\overset{\text{ert}}{/\!/\!/}\ 1 + [0 < x - 1 \leq N] \cdot N \cdot \mathcal{H}_{x-1} + [N < x - 1] \cdot (N \cdot \mathcal{H}_N + x - 1 - N)$$
$$x \leftarrow x - 1$$
$$[\!]\ [0 < x \leq N] \cdot N \cdot \mathcal{H}_x + [N < x] \cdot (N \cdot \mathcal{H}_N + x - N)$$

}
/// 0

For the inner loop, we make use of the following lemma, for which we also give a detailed proof in the following:

Lemma 15.6.3. *Let $t \in \mathbb{F}$ be a runtime (i.e. an expectation) such that t does* not *depend on program variable i. Then the following expected runtime annotation is valid:*

$$
\begin{aligned}
&/// \; 1 + t + [0 < x < i] \cdot 2 \cdot \max\left\{\tfrac{N}{x}, 1\right\} \\
&\textbf{while}\ (0 < x < i)\ \textbf{do}\ \{\ i \leftsquigarrow \mathrm{Unif}[1 \dots N]\ \} \\
&/// \; t
\end{aligned}
$$

Proof. We employ [Bat+18, Thm. 4] for so-called t-independent and identically distributed loops (t-i.i.d. loops for short) (see [Bat+18, Def. 5]). In order to verify the t-i.i.d.-ness of the loop **while** $(0 < x < i)$ **do** $\{\ i \leftsquigarrow \mathrm{Unif}[1 \dots N]\ \}$, we have to establish that neither

$$
\mathrm{wp}\,[\![i \leftsquigarrow \mathrm{Unif}[1 \dots N]]\!]\,([0 < x < i]) = \frac{1}{N} \cdot \sum_{m=1}^{N} [0 < x < m] = [0 < x] \cdot \max\left\{1 - \tfrac{x}{N}, 0\right\} \tag{\dagger}
$$

nor

$$
\begin{aligned}
&\mathrm{wp}\,[\![i \leftsquigarrow \mathrm{Unif}[1 \dots N]]\!]\,([\neg(0 < x < i)] \cdot t) \\
&= \mathrm{wp}\,[\![i \leftsquigarrow \mathrm{Unif}[1 \dots N]]\!]\,([\neg(0 < x < i)]) \cdot t && \text{(by [Bat+18, Lemma 1])} \\
&= \mathrm{wp}\,[\![i \leftsquigarrow \mathrm{Unif}[1 \dots N]]\!]\,(1 - [0 < x < i]) \cdot t \\
&= \left(\mathrm{wp}\,[\![i \leftsquigarrow \mathrm{Unif}[1 \dots N]]\!]\,(1) - \mathrm{wp}\,[\![i \leftsquigarrow \mathrm{Unif}[1 \dots N]]\!]\,([0 < x < i])\right) \cdot t \\
&\qquad\qquad \text{(by [Kam19, Cor. 4.22])} \\
&= \left(1 - [0 < x] \cdot \max\left\{1 - \tfrac{x}{N}, 0\right\}\right) \cdot t \\
&= \left([x \leq 0] + [0 < x] \cdot \min\left\{\tfrac{x}{N}, 1\right\}\right) \cdot t && (\ddagger)
\end{aligned}
$$

depend on variable i which is indeed the case by the assumption that t does not depend on i. Additionally to t-i.i.d.-ness, [Bat+18, Thm. 4] requires establishing that

$$
\mathrm{ert}\,[\![i \leftsquigarrow \mathrm{Unif}[1 \dots N]]\!]\,(0) = 1
$$

does not depend on variable i and that the body of the loop terminates almost-surely, i.e.,

$$
\mathrm{wp}\,[\![i \leftsquigarrow \mathrm{Unif}[1 \dots N]]\!]\,(1) = 1\ .
$$

Both conditions are obviously true.

Having established all preconditions of [Bat+18, Thm. 4], we can now make the following ert-annotations:

$$/\!/\!/\ 1 + t + [0 < x < i] \cdot 2 \cdot \max\left\{\tfrac{N}{x}, 1\right\}$$

$$/\!/\!/\ 1 + [0 < x < i] \cdot \left(\frac{2}{\min\left\{\frac{x}{N}, 1\right\}} + t\right) + [\neg(0 < x < i)] \cdot t$$

$$/\!/\!/\ 1 + [0 < x < i] \cdot \frac{1 + 1 + \min\left\{\frac{x}{N}, 1\right\} \cdot t}{\min\left\{\frac{x}{N}, 1\right\}} + [\neg(0 < x < i)] \cdot t \qquad \text{(see (‡))}$$

$$/\!/\!/\ 1 + [0 < x < i] \cdot \frac{1 + \text{ert}\ [\![\cdots]\!]\ (0) + \text{wp}\ [\![\cdots]\!]\ ([\neg(0 < x < i)] \cdot t)}{1 - \max\left\{1 - \frac{x}{N}, 0\right\}} + [\neg(0 < x < i)] \cdot t$$

(by Thm. 14.3.12)

$$/\!/\!/\ 1 + [0 < x < i] \cdot \frac{1 + \text{ert}\ [\![i \leftsquigarrow \text{Unif}[1 \ldots N]]\!]\ ([\neg(0 < x < i)] \cdot t)}{1 - \max\left\{1 - \frac{x}{N}, 0\right\}} + [\neg(0 < x < i)] \cdot t$$

(with $\frac{0}{0} = 0$ by [Bat+18, Thm. 4] and (†))

while $(0 < x < i)$ **do** $\{$

$\quad i \leftsquigarrow \text{Unif}[1 \ldots N]$

$\}$

$/\!/\!/\ t$

Here we left out $i \leftsquigarrow \text{Unif}[1 \ldots N]$ (indicated by $\cdots$) in one step to save space. It is important to note that, again, any loop semantics needed to be applied only a finite number of times. In particular, it was not necessary to find the limit of a sequence or anything alike. □

By our annotations in Ex. 15.6.2, we have shown that I is indeed a runtime subinvariant of the outer loop w.r.t. $t = 0$.

We stress that while in this case we had an *exact* expected runtime for the inner loop available by external techniques, a suitable *under-approximation* of the expected runtime of the inner loop using Thm. 15.6.1 would have worked as well. Hence, our technique *is* generally applicable to nested loops.

At the very top of the annotations in Ex. 15.6.2, we push I over the initial assignment, thus verifying $1 + N \cdot \mathcal{H}_N$ (and hence also $N \cdot \mathcal{H}_N$) as lower bound for the entire expected runtime of $\mathbf{L}_{cc}$.

In order to establish that the subinvariant I is in fact a lower bound, we are still left to prove conditional difference boundedness of I. For this, we first make the

following annotations for wp $[\![\mathit{outer\ loop\ body}]\!]\,(|I - I(s)|)$:

$$
\begin{aligned}
&/\!/\!/\ |I\,[x/x-1] - I(s)| && (I \text{ does not depend on } i)\\
&i \leftarrow N+1\,;\\
&/\!/\!/\ |I\,[x/x-1] - I(s)| && (\text{by AST of inner loop and [Bat+18, Lemma 1]})\\
&\textbf{while}\ (0 < x < i)\ \textbf{do}\ \{\ i \leftsquigarrow \mathrm{Unif}[1 \dots N]\ \}\,;\\
&/\!/\!/\ |I\,[x/x-1] - I(s)|\\
&x \leftarrow x-1\\
&/\!/\!/\ |I - I(s)|
\end{aligned}
$$

We finally bound ΔI:

Example 15.6.4.

$$
\begin{aligned}
&\Delta I(s)\\
=\ &[x>0](s)\cdot|I\,[x/x-1] - I(s)|\,(s) && \text{(see reasoning above)}\\
=\ &[x>0](s)\cdot\big|[0<x-1\le N]\cdot N\cdot\mathcal{H}_{x-1} + [N<x-1]\cdot(N\cdot\mathcal{H}_N + (x-1) - N)\\
&- [0<x\le N](s)\cdot N\cdot\mathcal{H}_{s(x)} - [N<x](s)\cdot(N\cdot\mathcal{H}_N + s(x) - N)\big|(s)\\
=\ &[x>0](s)\cdot\big|[1<x\le N+1](s)\cdot N\cdot\mathcal{H}_{s(x)-1}\\
&+ [N+1<x](s)\cdot(N\cdot\mathcal{H}_N + (s(x)-1) - N) - [0<x\le N](s)\cdot N\cdot\mathcal{H}_{s(x)}\\
&- [N<x](s)\cdot(N\cdot\mathcal{H}_N + s(x) - N)\big|\\
=\ &[x>0](s)\cdot\big|[1<x\le N](s)\cdot N\cdot\mathcal{H}_{s(x)-1} + [x=N+1](s)\cdot N\cdot\mathcal{H}_N\\
&+ [N+1<x](s)\cdot(N\cdot\mathcal{H}_N + (s(x)-1) - N) - [x=1](s)\cdot N\\
&- [1<x\le N](s)\cdot N\cdot\mathcal{H}_{s(x)} - [x=N+1](s)\cdot(N\cdot\mathcal{H}_N + 1)\\
&- [N+1<x](s)\cdot(N\cdot\mathcal{H}_N + s(x) - N)\big|\\
=\ &[x>0](s)\cdot\big|[1<x\le N](s)\cdot N\cdot\left(\mathcal{H}_{s(x)-1} - \mathcal{H}_{s(x)}\right)\\
&+ [x=N+1](s)\cdot(N\cdot\mathcal{H}_N - (N\cdot\mathcal{H}_N + 1))\\
&+ [N+1<x](s)\cdot(N\cdot\mathcal{H}_N + (s(x)-1) - N - (N\cdot\mathcal{H}_N + s(x) - N))\\
&- [x=1](s)\cdot N\big|\\
=\ &[x>0](s)\cdot\left|[1<x\le N](s)\cdot\left(-\tfrac{N}{s(x)}\right) - [N+1\le x](s) - [x=1](s)\cdot N\right|\\
\le\ &[x>0](s)\cdot([1<x\le N](s) + [N+1\le x](s) + [x=1](s))\cdot N
\end{aligned}
$$

while $(x > 0)$ **do** {
$\quad \{\, x \leftarrow x - 1 \,\} \left[\frac{x}{2 \cdot x + 1}\right] \{\, x \leftarrow x + 1 \,\}$
}

Figure 15.13: $\mathbf{L}_{filrw}$

$\leq N$

Hence, ΔI is bounded by a constant as N is constant *within* the program $\mathbf{L}_{cc}$. Finally, we would still have to show ${}^{\mathrm{wp}}\Phi_0(I) \ll \infty$, which can be read off from our proof of the subinvariance of I as ${}^{\mathrm{wp}}\Phi_0(I) \leq {}^{\mathrm{ert}}\Phi_0(I)$ by Thm. 14.3.12. Moreover, I clearly harmonizes with $t = 0$ as $I = [\neg(x > 0)] \cdot 0 + [x > 0] \cdot I$. Note that by Lemma 15.6.3 the inner loop is PAST and thus the body of the outer loop is AST. This concludes our lower bound proof for $\mathbf{L}_{cc}$.

Thus, we have verified that $N \cdot \mathcal{H}_N$ is a lower bound on the expected runtime of the coupon collector's problem. This lower bound enjoys several nice properties: For one, our lower bound is a *tight* asymptotic lower bound. Another fact is that our lower bound is a *strict* lower bound. The actual runtime is a bit higher (see [Kam+16]) as we have omitted some constants. This is, however, a desirable fact as often we are only interested in the asymptotic runtime and do not wish to bother with the constants. Notice further, that we never had to find the limit of any sequence. Loop semantics (be it wp or ert) were all applied only finitely many times in order to verify a tight asymptotic lower bound. All in all, the above example demonstrates the effectiveness of our inductive lower bound rule.

15.6.2 Case Study: Refuting PAST

In this section, we consider the so-called fair-in-the-limit random walk [McI+18] $\mathbf{L}_{filrw}$ which is stated in Fig. 15.13. In [McI+18], it is shown that this program is AST. Whether it is PAST is not discussed. Thus, we use this loop to illustrate that our rule for lower bounds on the expected runtime can also be used to refute PAST.

Let $N \in \mathbb{N}$ be a constant and let us consider the expectation $I_N = [x > 0] \cdot x \cdot N \ll \infty$. We are interested in ert $[\![\mathbf{L}_{filrw}]\!]$ (0). Of course, I harmonizes with $t = 0$.

The ert-characteristic function w.r.t. t is

$$ {}^{\mathrm{ert}}\Phi_0(X) = 1 + [x > 0] \cdot \left(2 + \tfrac{x}{2 \cdot x + 1} \cdot X\,[x/x - 1] + \tfrac{x+1}{2 \cdot x + 1} \cdot X\,[x/x + 1]\right). $$

We then have

$$
\begin{aligned}
&{}^{\mathsf{ert}}\Phi_0\,(I_N)\\
&= 1 + [x > 0] \cdot \left(2 + \tfrac{x}{2\cdot x+1} \cdot I_N\,[x/x-1] + \tfrac{x+1}{2\cdot x+1} \cdot I_N\,[x/x+1]\right)\\
&= 1 + [x > 0] \cdot \left(2 + \tfrac{x}{2\cdot x+1} \cdot [x-1 > 0] \cdot (x-1) \cdot N + \tfrac{x+1}{2\cdot x+1} \cdot [x+1 > 0] \cdot (x+1) \cdot N\right)\\
&= 1 + [x > 0] \cdot \left(2 + \tfrac{x}{2\cdot x+1} \cdot [x > 0] \cdot (x-1) \cdot N + \tfrac{x+1}{2\cdot x+1} \cdot [x > 0] \cdot (x+1) \cdot N\right)\\
&= 1 + [x > 0] \cdot \left(2 + [x > 0] \cdot N \cdot \left(\tfrac{x^2-x}{2\cdot x+1} + \tfrac{x^2+2\cdot x+1}{2\cdot x+1}\right)\right)\\
&= 1 + [x > 0] \cdot \left(2 + N \cdot \left(\tfrac{2\cdot x^2+x+1}{2\cdot x+1}\right)\right)\\
&= 1 + [x > 0] \cdot \left(2 + N \cdot \left(x + \tfrac{1}{2\cdot x+1}\right)\right)\\
&\geq [x > 0] \cdot x \cdot N = I_N
\end{aligned}
$$

Thus, I_N is an ert-subinvariant w.r.t. $t = 0$ Moreover we have

$$
\begin{aligned}
&\Delta I_N(s)\\
&= \left(\tfrac{x}{2\cdot x+1} \cdot [x > 0]\,|[x-1 > 0] \cdot N \cdot (x-1) - [x > 0]\,(s) \cdot N \cdot s(x)|\right)(s)\\
&\quad + \left(\tfrac{x+1}{2\cdot x+1} \cdot [x > 0]\,|[x+1 > 0] \cdot N \cdot (x+1) - [x > 0]\,(s) \cdot N \cdot s(x)|\right)(s)\\
&= \left(\tfrac{s(x)}{2\cdot s(x)+1} \cdot [x > 0]\,(s)\,|[x > 0]\,(s) \cdot N \cdot (s(x)-1) - [x > 0]\,(s) \cdot N \cdot s(x)|\right)\\
&\quad + \left(\tfrac{s(x)+1}{2\cdot s(x)+1} \cdot [x > 0]\,(s)\,|[x > 0]\,(s) \cdot N \cdot (s(x)+1) - [x > 0]\,(s) \cdot N \cdot s(x)|\right)\\
&= \tfrac{s(x)}{2\cdot s(x)+1} \cdot [x > 0]\,(s) \cdot |N \cdot (-1)| + \tfrac{s(x)+1}{2\cdot s(x)+1} \cdot [x > 0]\,(s) \cdot |N| \leq N
\end{aligned}
$$

Therefore, I_N is also c.d.b. because N is a constant. As the body of the loop $\mathbf{L}_{filrw}$ is itself loop-free, we have ${}^{\mathsf{wp}}\Phi_0\,(I_N) \ll \infty$. Thus, we have

$$I_N \leq \mathsf{ert}\ [\![\mathbf{L}_{filrw}]\!]\ (0)\,.$$

Since this proof works independent of $N \in \mathbb{N}$ we conclude

$$\sup_{N\in\mathbb{N}} I_N = [x > 0] \cdot \infty \leq \mathsf{ert}\ [\![\mathbf{L}_{filrw}]\!]\ (0)\,.$$

Thus, $\mathbf{L}_{filrw}$ is *not* PAST.

While the proof of showing that I_N is a lower bound is quite simple and involves only finite applications of the semantics of the body of the loop, the final reasoning of course involves an argument on the "asymptotic behavior" of I_N in N. Still, this example shows that our rule can also be used to refute PAST.

while $(a \neq 0)$ **do** {
  $\{\, a \leftarrow 0 \,\}\ \left[\frac{1}{2}\right]\ \{\, \{\, b \leftarrow b+1 \,\} \,\square\, \{\, b \leftarrow b+2 \,\} \,\}$;
  $k \leftarrow k+1$
}

Figure 15.14: $\mathbf{L}_{cexn}$

It is worth noting that the sequence of invariants $(I_N)_{N\in\mathbb{N}}$ presented here does *not* form an ert-ω-subinvariant (see Thm. 15.1.7): the condition $I_{N+1} \preceq {}^{\text{ert}}\Phi_0(I_N)$ is false in general. For example, consider $N = 1$ and $x = 5$. Then $I_{N+1}(x) = I_2(5) = 10$ but

$$\begin{aligned} {}^{\text{ert}}\Phi_0(I_1)(5) &= 1 + [x > 0] \cdot \left(2 + 1 \cdot \left(x + \tfrac{1}{2\cdot x+1}\right)\right)(5) \\ &= 1 + \left(2 + 1 \cdot \left(5 + \tfrac{1}{2\cdot 5+1}\right)\right) \\ &= 1 + \left(2 + 1 \cdot \left(5 + \tfrac{1}{11}\right)\right) \\ &= 8 + \tfrac{1}{11} < 10. \end{aligned}$$

Thus, this sequence of invariants indeed relies on our rule Thm. 15.6.1 for proving that each of its members is a lower bound on exprt $[\![\mathbf{L}_{filrw}]\!]$.

However, we will see in Ch. 16 that for $\mathbf{L}_{filrw}$ there is an even simpler proof for refuting PAST which fits into a more general concept (see Ex. 16.1.17). Nevertheless, we chose this example for illustrating how to refute PAST since it is on the one hand a concise example but on the other hand involves non-constant probabilities which are usually difficult to handle.

15.7 Discussion of Non-Determinism

In Ch. 14, we have introduced the concept of bounded non-determinism in probabilistic programs. So far, we have presented proof rules for lower bounds on expected outcomes and expected runtimes for purely probabilistic loops only. We now discuss our results in the presence of non-deterministic choices. Again, we fix a loop of the form in Fig. 14.16 where $\mathbf{P}$ is AST, a finite postexpectation $f \in \mathbb{F}$, and a finite expectation $I \in \mathbb{F}$.

Example 15.7.1. Consider the loop $\mathbf{L}_{cexn}$ in Fig. 15.14 which is an adaption of

the loop $\mathbf{L}_{cex}$ in Fig. 15.3 (we have studied $\mathbf{L}_{cex}$ in Counterex. 15.1.6 and Ex. 15.3.9 and 15.3.13).

Similar to Ex. 15.3.9 and 15.3.13, we would like to prove that $I = b + [a \neq 0]$ is a lower bound on $\mathsf{wp}\,[\![\mathbf{L}_{cex}]\!]\,(b)$. The wp-characteristic function of $\mathbf{L}_{cexn}$ is given by

$$\begin{aligned}{}^{\mathsf{wp}}\Phi_b(X) &= [a = 0] \cdot b \\ &\quad + [a \neq 0] \cdot \tfrac{1}{2} \cdot \left(X\,[a/0] + \min\left\{X\,[b/b+1],\, X\,[b/b+2]\right\}\right)[k/k+1]\,.\end{aligned}$$

Indeed ${}^{\mathsf{wp}}\Phi_b(I) = I$ can be shown similarly to Ex. 15.1.3.

First of all, Lemma 15.2.3 is already formulated for arbitrary loops whose body is AST. However, equality in Thm. 15.2.4 does not hold anymore. Intuitively, the reason is that the random variable $X^f_{T^\varphi}$ represents both the intuition behind wp *and* awp: For any terminating run, it evaluates the postexpectation at the moment when termination is reached. The only difference is that when regarding wp, a non-deterministic choice is resolved via taking a minimum, whereas awp resolves non-determinism by taking a maximum. However, equality in Thm. 15.2.4 is crucial for the proof of Cor. 15.3.3 as it connects the notion of uniform integrability in expectation-transformer semantics (Def. 15.1.9) and probability theory (Def. 15.3.1).

As already indicated in Thm. 14.4.25 and Cor. 14.4.26, our correspondence in Fig. 15.6 cannot be shown when probabilistic loops involve non-deterministic choices. The simple reason is that the concept of a "scheduler" has no counterpart in the realm of expectation-transformers since it is predetermined by the transformer how the non-deterministic choice is resolved, i.e., either via a worst-case or via a best-case assumption. Thus, we can only maintain some of these correspondences when taking an infimum or a supremum over all schedulers. We conjecture the following.

Conjecture 15.7.2. *Let* $\mathbf{L}$ *be a probabilistic loop as in Fig. 14.16. We have*

(i)

$$\lambda s.\ \inf_{\mathfrak{s}\in\mathsf{Scheds}} \mathbb{E}^{loop}_{s,\mathfrak{s}}\left(X^{f,I}_n\right) = {}^{\mathsf{wp}}\Phi^{n+1}_f(I).$$

(ii)

$$\mathrm{lfp}\,{}^{\mathsf{wp}}\Phi_f = \lambda s.\ \inf_{\mathfrak{s}\in\mathsf{Scheds}} \mathbb{E}^{loop}_{s,\mathfrak{s}}\left(X^{f}_{T^\varphi}\right) = \mathsf{wp}\,[\![\mathbf{L}]\!]\,(f)$$

(iii)

$$\lambda s.\ \sup_{\mathfrak{s}\in\mathsf{Scheds}} \mathbb{E}^{loop}_{s,\mathfrak{s}}\left(X^{f,I}_n\right) = {}^{\mathsf{awp}}\Phi^{n+1}_f(I).$$

(iv)

$$\operatorname{lfp}{}^{\mathrm{awp}}\Phi_f = \lambda s.\ \sup_{\mathfrak{s}\in\mathsf{Scheds}} \mathbb{E}^{loop}_{s,\mathfrak{s}}\left(X^f_{T^\varphi}\right) = \mathrm{awp}\ [\![\mathbf{L}]\!]\ (f)$$

Let us elaborate on Conj. 15.7.2. As already indicated, it is not possible to just adapt our proofs from the purely probabilistic case to obtain Conj. 15.7.2. Still, we could prove the similar looking statements in Thm. 14.4.25 and Cor. 14.4.26. The reason is that Thm. 14.4.25 and Cor. 14.4.26 are statements regarding *measurable functions* w.r.t. the measurable space $(\Omega^{\mathrm{loop}}, \mathcal{F}^{\mathrm{loop}})$ which is *independent* of the chosen initial state s and especially *independent* of the chosen scheduler $\mathfrak{s}$. In Conj. 15.7.2, we have to prove results regarding operations from probability spaces which arise from fixing an initial state s and some scheduler $\mathfrak{s}$. Moreover, we cannot make use of Thm. 14.4.25 and Cor. 14.4.26 since taking an expected value and taking an infimum (resp. a supremum) do in general *not* commute. Nevertheless, Thm. 14.4.25 and Cor. 14.4.26 make Conj. 15.7.2 seem reasonable.

In the following, we show that if Conj. 15.7.2 (i) and (ii) hold, Thm. 15.3.11 (i) to (iii) are still sound, i.e., we have the following theorem.

Theorem 15.7.3 (Generalized Optional Stopping Theorem for Weakest Preexpectation Reasoning)**.** *Consider a loop of the form in Fig. 14.16 where* $\mathbf{P}$ *is AST and assume that Conj. 15.7.2 (i) and (ii) hold. Let* $I \not\equiv \infty$ *be a* wp*-subinvariant w.r.t. the postexpectation* $f \not\equiv \infty$ *(i.e.,* $I \preceq {}^{\mathrm{wp}}\Phi_f(I)$*). If one of the following statements holds,* I *is a lower bound, i.e.,* $I \preceq \mathrm{wp}\ [\![\mathbf{L}]\!]\ (f)$ *and thus,* I *is u.i. for* f*.*

(i) The looping time T^φ *is almost-surely bounded, i.e., for every state* $s \in \Sigma$ *and scheduler* $\mathfrak{s} \in \mathsf{Scheds}$ *there exists a constant* $N(s,\mathfrak{s}) \in \mathbb{N}$ *with* $\mathbb{P}^{loop}_{s,\mathfrak{s}}\left(T^\varphi \preceq N(s,\mathfrak{s})\right) = 1$ *and* ${}^{\mathrm{awp}}\Phi^n_f(I) \not\equiv \infty$ *for every* $n \in \mathbb{N}$*.*

(ii) The expected looping time $\mathbb{E}^{loop}_{s,\mathfrak{s}}\left(T^\varphi\right)$ *is finite for every initial state* $s \in \Sigma$ *and scheduler* $\mathfrak{s} \in \mathsf{Scheds}$*,* I *harmonizes with* f*,* ${}^{\mathrm{awp}}\Phi_f(I) \not\equiv \infty$*, and* I *is worst-case conditionally difference bounded.*

(iii) Both f *and* I *are bounded and Fig. 14.16 is AST.*

We now show that independent of Conj. 15.7.2 the results on the crucial ingredients of Thm. 15.3.5, i.e., "submartingale" (Lemma 15.3.4) and "conditional difference boundedness" (Thm. 15.3.10) can be generalized to the non-deterministic case.

We begin with a generalized version of Lemma 15.3.4. To ensure that $\mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^{f,I}_n\right)$ is finite for every $n \in \mathbb{N}$ (see Def. 14.4.29), we have to make use of awp, which resolves a non-deterministic choice via taking a maximum. Thus, whenever ${}^{\mathrm{awp}}\Phi^n_f(I) \not\equiv \infty$

we can ensure that independent of the scheduler in the underlying probability space $X_n^{f,I}$ has a finite expected value.

Lemma 15.7.4 (Subinvariant Induces Submartingale)**.** *Let I be a* wp-subinvariant*, i.e., $I \preceq {}^{\mathsf{wp}}\Phi_f(I)$ such that ${}^{\mathsf{awp}}\Phi_f^n(I) \ll \infty$ for every $n \in \mathbb{N}$, that is, ${}^{\mathsf{awp}}\Phi_f^n(I)$ only takes finite values. Then the induced stochastic process $\mathbf{X}^{f,I}$ is a submartingale w.r.t. $(\mathcal{G}_n^{loop})_{n\in\mathbb{N}}$.*

The other crucial concept for our inductive proof rule for lower bounds is conditional difference boundedness as defined in Def. 15.3.8. This concept has to be generalized to work in the presence of non-determinism.

Definition 15.7.5 (Worst-Case Conditional Difference Boundedness)**.** Let $I \in \mathbb{F}$ with $I \ll \infty$. We define the function $H : \mathbb{F} \to \mathbb{F}$ and the expectation $\Delta I \in \mathbb{F}$ as

$$H(X) = [\varphi] \cdot \mathsf{awp}\ [\![\mathbf{P}]\!]\ (X) \qquad \text{and} \qquad \Delta I = \lambda s.\ \big(H\big(|I - I(s)|\big)(s)\big)\,.$$

The expectation I is called *worst-case conditionally difference bounded* (worst-case c.d.b.) if for some constant $c \in \mathbb{R}_{\geq 0}$

$$\Delta I(s) \preceq c \text{ holds for all } s \in \Sigma.$$

In the absence of non-determinism, wp = awp, i.e., Def. 15.7.5 and Def. 15.3.8 coincide.

Example 15.7.6. Consider again the loop $\mathbf{L}_{cexn}$ in Fig. 15.14 and the invariant $I = b + [a \neq 0] \ll \infty$ from Ex. 15.7.1. We have

$$H(X) = [a \neq 0] \cdot \tfrac{1}{2} \cdot \big(X\,[a/0] + \max\big\{X\,[b/b+1],\ X\,[b/b+2]\big\}\big)\,[k/k+1]\,.$$

Similar to Ex. 15.3.9, it can be shown that I is worst-case c.d.b. However, in contrast to Ex. 15.3.9, ΔI is bounded by 2 since when resolving the non-determinism in an angelic way, i.e., by taking the maximum, the value of I increases by at most 2.

Similar to Ex. 15.3.13 one can show that exprt $[\![\mathbf{L}_{cexn}]\!]$ is *finite* and thus, the expected looping time is finite independent of the scheduler and the initial state. Hence, if Conj. 15.7.2 holds, we conclude wp $[\![\mathbf{L}_{cex}]\!]\,(b) = I$ by Thm. 15.7.3 (ii).

The expectation ΔI expresses the *worst-case* expected change of I within one loop iteration. Here, it is indeed the "worst-case" if we resolve non-determinism by taking the maximum, i.e., the larger the expected change, "the less useful" I is for

our purpose. Thus we make use of awp instead of wp. So, if I is worst-case c.d.b. it is expected to change at most by a constant in one loop iteration (no matter which scheduler is used to resolve the non-determinism).

Counterexample 15.7.7. To mention a case where worst-case conditionally difference boundedness is violated, consider again $\mathbf{L}_{cexn}$ in Fig. 15.14 and $I = b{+}[a \neq 0]$ from Ex. 15.7.1 but assume that the non-deterministic choice in $\mathbf{L}_{cexn}$ would be

$$\{\, b \leftarrow b + 1 \,\} \,\square\, \{\, b \leftarrow 2 \cdot b \,\}\,.$$

Moreover, assume $b \in \mathbb{N}_{\geq 1}$. In this case,

$$H(X) = [a \neq 0] \cdot \tfrac{1}{2} \cdot \left(X\,[a/0] + \max\left\{ X\,[b/b+1],\, X\,[b/2 \cdot b] \right\}\right)[k/k+1]\,.$$

Similar to the calculation in Ex. 15.3.15, one can then conclude that I is *not* worst-case c.d.b. since I is expected to change by at least $\frac{b}{2}$ which is unbounded.

Theorem 15.7.8 (Worst-Case Expected Change of I)**.** *Let $I \ll \infty$ harmonize with $f \ll \infty$. Then for any $s \in \Sigma$ and $\mathfrak{s} \in \mathsf{Scheds}$*

$$\sup_{\mathfrak{s}' \in \mathsf{Scheds}} \mathbb{E}^{loop}_{s,\mathfrak{s}'}\left(\left| X^{f,I}_{n+1} - X^{f,I}_{n} \right| \,\middle|\, \mathcal{G}^{loop}_{n} \right) = X^{0,\Delta I}_{n}.$$

As in Thm. 14.4.25, we have to define the conditional expectation on null sets appropriately, see the proof of Thm. 15.7.8 in App. B.33.

Again, while Thm. 15.7.8 looks similar to Conj. 15.7.2, it is only a statement on measurable functions and therefore independent of the probability measure $\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}$ induced by a fixed state $s \in \Sigma$ and scheduler $\mathfrak{s} \in \mathsf{Scheds}$.

These two results can then be used to prove Thm. 15.7.3 under the precondition that Conj. 15.7.2 (i) and (ii) hold.

When assuming Conj. 15.7.2 (i) and (ii), we can also generalize our proof rule for expected runtimes. In this case, the following slight adaption of Thm. 14.3.12 holds as the ert-transformer also resolves non-deterministic choice via taking the maximum.

Corollary 15.7.9 (Connection Between ert and awp, [Kam+18])**.** *Let $\mathbf{P}$ be any probabilistic program and $t \in \mathbb{F}$. Then*

$$\mathsf{ert}\ [\![\mathbf{P}]\!]\ (t) = \mathsf{ert}\ [\![\mathbf{P}]\!]\ (0) + \mathsf{awp}\ [\![\mathbf{P}]\!]\ (t) = \mathsf{exprt}\ [\![\mathbf{P}]\!] + \mathsf{awp}\ [\![\mathbf{P}]\!]\ (t)\,.$$

So in the presence of non-deterministic choices we have

$$\mathsf{ert}\ [\![\mathbf{P}]\!]\ (t) \geq \mathsf{exprt}\ [\![\mathbf{P}]\!] + \mathsf{wp}\ [\![\mathbf{P}]\!]\,(t)\,.$$

In the proof of Thm. 15.6.1, we use the preconditions to show that

$$\lim_{n\to\omega} {}^{\mathsf{wp}}\Phi_t^n(I)(s) = \mathsf{wp}\,[\![\mathbf{P}]\!]\,(I)\,(s)$$

using Thm. 15.3.11 (ii) which we have just generalized in Thm. 15.7.3 (ii) to arbitrary loops. So this step can be transferred to the non-deterministic case. However, it is only of limited use since in the proof of Thm. 15.6.1 we then conclude that $\lim_{n\to\omega} {}^{\mathsf{ert}}\Phi_t^n(I)(s) = \mathsf{ert}\,[\![\mathbf{P}]\!]\,(I)\,(s)$ by Thm. 14.3.12 which then leads to I being a lower bound as the sequence $\left({}^{\mathsf{ert}}\Phi_t^n(I)(s)\right)_{n\in\mathbb{N}}$ is monotonically increasing. Here, we can only conclude $\lim_{n\to\omega} {}^{\mathsf{ert}}\Phi_t^n(I)(s) \geq \mathsf{ert}\,[\![\mathbf{P}]\!]\,(I)\,(s)$ since awp $\geq$ wp. The reason is that we would need to study lower bounds on awp which is not in the scope of this thesis. Still, if we assume that I is a wp-subinvariant and not only an ert-subinvariant, we can state the following rule when assuming Conj. 15.7.2 (i) and (ii).

Theorem 15.7.10 (Inductive Lower Bounds on Expected Runtimes)**.** *Let $t, I \in \mathbb{F}$ with $t, I \ll \infty$ and let I harmonize with t. Assume Conj. 15.7.2 (i) and (ii). If I is a* wp*-subinvariant w.r.t. t, ${}^{\mathsf{awp}}\Phi_t(I) \ll \infty$, and I is worst-case conditionally difference bounded, we have*

$$I \preceq \mathsf{ert}\,\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right]\,(t)\,.$$

Chapter 16

Computing Exact Expected Looping Times of *cpa*-Loops

In Ch. 15, we have studied proof rules for verifying lower bounds for arbitrary probabilistic loops. In this chapter, we focus on a special form of probabilistic loops, so-called (*c*)*pa*-loops. We show that there is a simple decision procedure for *deciding* (P)AST for *cpa*-loops. Moreover, in case of PAST the expected looping time can be computed *exactly*.

First of all, we discuss a special form of probabilistic loops (so-called *pa*-loops) in Sect. 16.1. Then we use our findings to show that for *cpa*-loops, these are *pa*-loops in which all occurring probabilities are independent of the variables in the loop, there is a very simple decision procedure for termination (Sect. 16.2 and 16.3). In this way, we also obtain asymptotically tight bounds on the expected looping time of any *cpa*-loop. Based on these bounds, we present the first algorithm to compute closed forms for the *exact* expected looping time of such loops (Sect. 16.4).

16.1 Probabilistic Addition Loops

In this section, we introduce a special kind of probabilistic loops which we consider in this chapter. These are so-called *probabilistic addition loops* since they only allow for adding constants to the program variables (besides a single exception of so-called "direct termination"). Let us first define these loops formally.

Definition 16.1.1 (Probabilistic Addition Loop)**.** A *d*-dimensional *probabilistic*

addition loop (pa-loop for short) is a loop of the form in Fig. 16.1(a) where $\vec{x} = (x_1, \dots, x_d)$ for some $d \geq 1$ is a vector of pairwise different variables, $\vec{a}, \vec{c}_1, \dots, \vec{c}_n \in \mathbb{Z}^d$ are vectors of integers, the $\vec{c}_j$ are pairwise distinct, $b \in \mathbb{Z}$, $\bullet$ denotes the scalar product, and $\vec{w} \in \mathbb{Z}^d$ with $\vec{a} \bullet \vec{w} - b \leq 0$. We require that $p_{\vec{c}_j}(\vec{x}), p'(\vec{x}) \in \mathbb{R}_{\mathbb{A}}(\vec{x})$ such that for all $\vec{x} \in \mathbb{Z}^d$ with $\vec{a} \bullet \vec{x} - b > 0$ we have $p_{\vec{c}_j}(\vec{x}), p'(\vec{x}) \geq 0$ for $1 \leq j \leq n$ and $\sum_{j=1}^{n} p_{\vec{c}_j}(\vec{x}) + p'(\vec{x}) = 1$. Here, $\mathbb{R}_{\mathbb{A}}(\vec{x}) = \left\{ \frac{p(\vec{x})}{q(\vec{x})} \;\middle|\; p(\vec{x}), q(\vec{x}) \in \mathbb{R}_{\mathbb{A}}[\vec{x}], q(\vec{x}) \neq 0 \right\}$ is the *field of rational functions* in $\vec{x}$ over $\mathbb{R}_{\mathbb{A}}$. Such a loop has *direct termination* if there is an $\vec{x} \in \mathbb{Z}^d$ with $\vec{a} \bullet \vec{x} - b > 0$ and $p'(\vec{x}) > 0$. The loop is *trivial*, if $\vec{a} = \vec{0}$. If all probabilities are constant, i.e., if there are $p_{\vec{c}_1}, \dots, p_{\vec{c}_n}, p' \in (\mathbb{R}_{\mathbb{A}})_{\geq 0}$ with $p_{\vec{c}_j}(\vec{x}) = p_{\vec{c}_j}$ and $p'(\vec{x}) = p'$ for all $1 \leq j \leq n$ and all $\vec{x} \in \mathbb{Z}^d$, it is a *constant probability addition loop (cpa-loop for short)*.

Such a loop means that the integer variables $\vec{x}$ are changed to $\vec{x} + \vec{c}_j$ with probability $p_{\vec{c}_j}(\vec{x})$. For inputs $\vec{x}$ with $\vec{a} \bullet \vec{x} - b \leq 0$ the loop terminates immediately. Here, we use a slightly different syntax than the one defined via pGCL in Sect. 14.1 to increase readability. The reason is that in pGCL we would need "blocks" of d assignments where each program variable x_i is assigned the corresponding entry of $\vec{x} + \vec{c}_j$ (or $\vec{w}$) to model the body of a *pa*-loop. That each of these blocks is executed with the corresponding probability $p_{\vec{c}_j}(\vec{x})$ (or $p'(\vec{x})$) would then be achieved by using a fresh variable y which is probabilistically assigned a value indicating the "block" and conditional statements. We use the notation [...] in analogy to probabilistic choice (see Table 14.1). This is different from the Iverson bracket notation since $p_{\vec{c}_j}(\vec{x}), p'(\vec{x}) \in \mathbb{R}_{\mathbb{A}}(\vec{x})$ are rational functions and not Boolean formulas.

We restrict ourselves to probabilities from $\mathbb{R}_{\mathbb{A}}(\vec{x})$, i.e., they are fractions of polynomials with real algebraic coefficients since it is unclear how to represent transcendental numbers on computers.

Example 16.1.2. The loop $\mathbf{L}_{race}$ in Fig. 16.1(b) formulates the race of the tortoise and the hare from Fig. 14.1 as a *cpa-loop*. In the condition, we use the scalar product $(-1, 1) \bullet (h, t)$ which stands for $(-1) \cdot h + 1 \cdot t = t - h$. Exactly one of the instructions with numbers in brackets [...] is executed in each loop iteration and the number indicates the probability that the corresponding instruction is chosen. So for example, h does not change and t is incremented by 1 with probability $\frac{6}{11}$, both h and t are incremented by 1 with probability $\frac{1}{22}$, etc. Note that $\mathbf{L}_{race}$ has *no* direct termination (i.e., $p'(\vec{x}) = 0$ for all $\vec{x} \in \mathbb{Z}^d$).

while $(\vec{a} \cdot \vec{x} - b > 0)$ **do** {

$\vec{x} \leftarrow \vec{x} + \vec{c}_1 \quad [p_{\vec{c}_1}(\vec{x})]$

$\vdots$

$\vec{x} \leftarrow \vec{x} + \vec{c}_n \quad [p_{\vec{c}_n}(\vec{x})]$

$\vec{x} \leftarrow \vec{w} \quad [p'(\vec{x})]$

}

(a) *pa*-Loop

while $((-1,1) \cdot (h,t) - (-1) > 0)$ **do** {

$(h,t) \leftarrow (h,t) + (0,1) \quad \left[\frac{6}{11}\right]$

$(h,t) \leftarrow (h,t) + (1,1) \quad \left[\frac{1}{22}\right]$

$(h,t) \leftarrow (h,t) + (2,1) \quad \left[\frac{1}{22}\right]$

$\vdots$

$(h,t) \leftarrow (h,t) + (10,1) \quad \left[\frac{1}{22}\right]$

}

(b) Race of Tortoise and Hare as *pa*-Loop

Figure 16.1: Loops From Def. 16.1.1 and Ex. 16.1.2

In the setting of probabilistic addition loops, the set of states Σ is basically $\Sigma = \mathbb{Z}^d$ and thus, we denote states by $\vec{x} \in \mathbb{Z}^d$. As described in Sect. 14.4, when fixing an initial state $\vec{x}_0 \in \mathbb{Z}^d$ we get the probability space $((\mathbb{Z}^d)^\omega, \mathcal{F}^{\text{loop}}, \mathbb{P}^{\text{loop}}_{\vec{x}_0})$. To indicate a specific loop, we write $\mathbb{P}^{\mathbf{L}}_{\vec{x}_0}$. For example, the probability space arising from $\mathbf{L}_{race}$ is discussed in Ex. 14.4.5, 14.4.8, 14.4.10, and 14.4.14.

In this section, we consider the expected looping time of probabilistic addition loops $\mathbf{L}$. To ease readability, instead of $T^{(\vec{a}\cdot\vec{x}-d>0)}$ we write $T^{\mathbf{L}}$ to denote the looping time of $\mathbf{L}$. Then, the actual runtime in terms of the ert-calculus (see Def. 14.3.8) is only asymptotically equal to the looping time since the ert-calculus also takes the runtime of the body of the loop into account. Still, the runtime of a *pa*-loop is infinite iff $T^{\mathbf{L}}$ is infinite as the body is loop-free. Thus, for reasoning about AST and PAST the looping time is sufficient.

Definition 16.1.3 (Expected Looping Time)**.** Let $\mathbf{L}$ be a d-dimensional *pa*-loop and $\vec{x}_0 \in \mathbb{Z}^d$. We define

$$\text{explt}\,[\![\mathbf{L}]\!]\left(\vec{x}_0\right) = \mathbb{E}^{\mathbf{L}}_{\vec{x}_0}\left(T^{\mathbf{L}}\right).$$

Thus, exprt $[\![\mathbf{L}]\!]$ and explt $[\![\mathbf{L}]\!]$ are asymptotically equal.

Example 16.1.4. By the observations in Sect. 16.3.3, we will infer $\frac{2}{3} \cdot (t-h+1) \leq \text{explt}\,[\![\mathbf{L}_{race}]\!]\,(h,t) \leq \frac{2}{3} \cdot (t-h+1) + \frac{16}{3}$ for $t-h+1 > 0$, see Ex. 16.3.11. So the expected number of steps until termination is finite (and linear in the input variables) and thus, $\mathbf{L}_{race}$ is PAST. Our algorithm in Sect. 16.4 will even be able to compute $\text{explt}\,[\![\mathbf{L}_{race}]\!]\,(h,t)$ exactly, see Ex. 16.4.15.

If the initial values $\vec{x}_0$ violate the condition, the looping time is clearly 0.

Corollary 16.1.5 (Expected Looping Time for Violating Initial Values)**.** *For a probabilistic addition loop* $\mathbf{L}$ *and any* $\vec{x}_0 \in \mathbb{Z}^d$ *with* $\vec{a} \bullet \vec{x}_0 - b \leq 0$*, we have* $\text{explt}\,[\![\mathbf{L}]\!]\,(\vec{x}_0) = 0$.

To obtain our results, we use a characterization of the expected looping time similar to the ert-transformer defined in Def. 14.3.8 where we count only one unit of time per loop iteration. This then results in searching for the smallest function $f : \mathbb{Z}^d \to \overline{\mathbb{R}_{\geq 0}}$ that satisfies

$$f(\vec{x}) = \sum_{j=1}^{n} p_{\vec{c}_j}(\vec{x}) \cdot f(\vec{x}+\vec{c}_j) + p'(\vec{x}) \cdot f(\vec{w}) + 1 \quad \text{for all } \vec{x} \text{ with } \vec{a} \bullet \vec{x} - b > 0, \tag{16.1.1}$$

i.e., we search for the smallest fixed point of the characteristic looping time function which transforms the left-hand side of (16.1.1) into its right-hand side.

Definition 16.1.6 (Characteristic Looping Time Function)**.** For a probabilistic addition loop $\mathbf{L}$ we define the *characteristic looping time function* $\Phi_{\langle \mathbf{L} \rangle} : (\mathbb{Z}^d \to \overline{\mathbb{R}_{\geq 0}}) \to (\mathbb{Z}^d \to \overline{\mathbb{R}_{\geq 0}})$ where for any $f : \mathbb{Z}^d \to \overline{\mathbb{R}_{\geq 0}}$:

$$\Phi_{\langle \mathbf{L} \rangle}(f)(\vec{x}) = \begin{cases} 1 + \sum_{j=1}^{n} p_{\vec{c}_j}(\vec{x}) \cdot f(\vec{x}+\vec{c}_j) + p'(\vec{x}) \cdot f(\vec{w}), & \text{if } \vec{a} \bullet \vec{x} - b > 0 \\ f(\vec{x}), & \text{if } \vec{a} \bullet \vec{x} - b \leq 0 \end{cases}$$

Example 16.1.7. For $\mathbf{L}_{race}$ from Ex. 16.1.2, $\Phi_{\langle \mathbf{L}_{race} \rangle}$ maps any function $f : \mathbb{Z}^2 \to \overline{\mathbb{R}_{\geq 0}}$ to $\Phi_{\langle \mathbf{L}_{race} \rangle}(f)$, where

$$\Phi_{\langle \mathbf{L}_{race} \rangle}(f)(h,t) = \begin{cases} 1 + \frac{6}{11} \cdot f(h,t+1) + \frac{1}{22} \cdot \sum_{j=1}^{10} f(h+j,t+1), & \text{if } t-h+1 > 0 \\ f(h,t), & \text{if } t-h+1 \leq 0. \end{cases} \tag{16.1.2}$$

Similar to Thm. 14.3.10 and 14.3.11, we can state the following corollary which can be proved analogously (see, [Kam+18, Thm. 3.4, Thm. 5.5]) by using Thm. 15.1.5. Therefore, Cor. 16.1.8 recapitulates that the least fixed point of $\Phi_{\langle \mathbf{L} \rangle}$ indeed yields an equivalent characterization of the expected looping time.

Corollary 16.1.8 (Least Fixed Point of $\Phi_{\langle \mathbf{L} \rangle}$ is explt $[\![\mathbf{L}]\!]$)**.** *For any* $\mathbf{L}$ *as in Def. 16.1.1, the characteristic looping time function* $\Phi_{\langle \mathbf{L} \rangle}$ *is continuous. Thus, it has a least fixed point* $\operatorname{lfp} \Phi_{\langle \mathbf{L} \rangle} : \mathbb{Z}^d \to \overline{\mathbb{R}_{\geq 0}}$ *with* $\operatorname{lfp} \Phi_{\langle \mathbf{L} \rangle} = \sup \left\{ \Phi^i_{\langle \mathbf{L} \rangle}(0) \;\middle|\; i \in \mathbb{N} \right\}$*. Moreover, the least fixed point of* $\Phi_{\langle \mathbf{L} \rangle}$ *is the expected looping time of* $\mathbf{L}$*, i.e., for any* $\vec{x}_0 \in \mathbb{Z}^d$*, we have* $\operatorname{lfp} \Phi_{\langle \mathbf{L} \rangle}(\vec{x}_0) = \text{explt } [\![\mathbf{L}]\!] \left(\vec{x}_0\right)$.

Example 16.1.9. By Cor. 16.1.8, the expected looping time explt $[\![\mathbf{L}_{race}]\!]\,(h,t)$ can also be characterized as the smallest function $f : \mathbb{Z}^2 \to \overline{\mathbb{R}_{\geq 0}}$ satisfying $f(h,t) = \Phi_{\langle \mathbf{L}_{race} \rangle}(f)\,(h,t)$ (see (16.1.2)), i.e., as the least fixed point of $\Phi_{\langle \mathbf{L}_{race} \rangle}$.

16.1.1 One-Dimensional *pa*-Loops

In this subsection, we discuss the case $d = 1$, which will become important when studying random walk loops in Sect. 16.3.2. A one-dimensional *pa*-loop has the form shown in Fig. 16.2(a). W.l.o.g. we restrict ourselves to *non-trivial pa*-loops, i.e., $a \neq 0$ since otherwise (P)AST can be directly read off the condition of the loop. Moreover, we restrict ourselves to the case where $p_0(x) < 1$ for all $x > 0$: if $p_0(x) = 1$ for some $x > 0$, the loop does not terminate on this x, and thus, the loop is not AST.

It turns out that we can restrict ourselves to loops with $a = 1$ and $b = 0$.

Theorem 16.1.10 (Reduction to Standardized Condition)**.** *Let* $\mathbf{L}$ *be the loop in Fig. 16.2(a) with* $a \neq 0$*,* $\mathbf{L}_t$ *be the loop in Fig. 16.2(b), and* $g : \mathbb{R} \to \mathbb{R}, x \mapsto a \cdot x + b$.

(i) For every finite prefix $\pi \in \mathbb{Z}^+$ *and any* $x_0 \in \mathbb{Z}$ *we have*

$$\mathbb{P}^{\mathbf{L}}_{x_0}(\mathrm{Cyl}(\pi)) = \mathbb{P}^{\mathbf{L}_t}_{g(x_0)}(\mathrm{Cyl}(g(\pi)))\,,$$

where $g(\pi)$ *denotes the application of* g *to each entry in* π.

(ii) For every run $\vartheta \in \mathbb{Z}^\omega$ *we have* $T^{\mathbf{L}}(\vartheta) = T^{\mathbf{L}_t}(g(\vartheta))$

(iii) For every $j \in \mathbb{N}$ *we have* $\mathbb{P}^{\mathbf{L}}_{x_0}\left(T^{\mathbf{L}} = j\right) = \mathbb{P}^{\mathbf{L}_t}_{g(x_0)}\left(T^{\mathbf{L}_t} = j\right)$.

while $(a \cdot x - b > 0)$ **do {**

$x \leftarrow x + c_1 \quad [p_{c_1}(x)]$

$\vdots$

$x \leftarrow x + c_n \quad [p_{c_n}(x)]$

$x \leftarrow w \quad [p'(x)]$

}

(a) One-Dimensional *pa*-Loop

while $(x > 0)$ **do {**

$x \leftarrow x + a \cdot c_1 \quad \left[p_{c_1}\left(\frac{x+b}{a}\right)\right]$

$\vdots$

$x \leftarrow x + a \cdot c_n \quad \left[p_{c_n}\left(\frac{x+b}{a}\right)\right]$

$x \leftarrow a \cdot w - b \quad \left[p'\left(\frac{x+b}{a}\right)\right]$

}

(b) Reduction From Thm. 16.1.10

while $(x > 0)$ **do {**

$x \leftarrow x + m \quad [p_m(x)]$

$\vdots$

$x \leftarrow x + 1 \quad [p_1(x)]$

$x \leftarrow x \quad [p_0(x)]$

$x \leftarrow x - 1 \quad [p_{-1}(x)]$

$\vdots$

$x \leftarrow x - k \quad [p_{-k}(x)]$

$x \leftarrow w \quad \left[p'(x)\right]$

}

(c) Standard Form

Figure 16.2: One-Dimensional *pa*-Loops

Proof. We prove the statements separately.

(i) We prove the result by induction on the length k of π. If $k = 1$, then $\pi = z \in \mathbb{Z}$, i.e., $\mathrm{Cyl}(\pi) = z\mathbb{Z}^\omega$, and we have

$$\mathbb{P}^{\mathbf{L}}_{x_0}(\mathrm{Cyl}(\pi)) = [x_0 = z] = [g(x_0) = g(z)] = \mathbb{P}^{\mathbf{L}_t}_{g(x_0)}(\mathrm{Cyl}(g(\pi)))$$

as g is bijective.

Now assume that we have proved correctness of the result for all prefixes of length k. Moreover, the claim is trivial for prefixes that do not start with x_0 since in this case both sides are 0. Let $\pi' = x_0 \cdots x_k$.

Let q be the probability to enter x_k from x_{k-1} in $\mathbf{L}$. We show that q is also the

probability to enter $g(x_k)$ from $g(x_{k-1})$ in $\mathbf{L}_t$. Then we have

$$\begin{aligned}
&\mathbb{P}^{\mathbf{L}}_{x_0}\left(\mathrm{Cyl}\left(\pi'\right)\right)\\
&= \mathbb{P}^{\mathbf{L}}_{x_0}\left(\mathrm{Cyl}\left(x_0 \cdots x_k\right)\right) \cdot q && \text{(by Lemma 14.4.9)}\\
&= \mathbb{P}^{\mathbf{L}_t}_{g(x_0)}\left(\mathrm{Cyl}\left(g(x_0) \cdots g(x_k)\right)\right) \cdot q && \text{(by I.H.)}\\
&= \mathbb{P}^{\mathbf{L}_t}_{g(x_0)}\left(\mathrm{Cyl}\left(g(\pi')\right)\right). && \text{(by Lemma 14.4.9)}
\end{aligned}$$

Note that g is bijective as a is non-zero and $g^{-1}(x) = \frac{x+b}{a}$.

If $a \cdot x_{k-1} \cdot b \leq 0$, i.e., $g(x_{k-1}) \leq 0$ we have $[x_{k-1} = x_k] = q$ and thus $[g(x_{k-1}) = g(x_k)] = q$ which is the probability to enter $g(x_k)$ from $g(x_{k-1})$ in $\mathbf{L}_t$.

Otherwise we have

$$\begin{aligned}
q = &\sum_{j=1}^{n} \left[x_k = x_{k-1} + c_j\right] \cdot p_j(x_{k-1})\\
&+ [x_k = w] \cdot p'(x_{k-1})\\
= &\sum_{j=1}^{n} \left[g(x_k) = g(x_{k-1}) + a \cdot c_j\right] \cdot p_j(g^{-1}(g(x_{k-1})))\\
&+ [g(x_k) = g(w)] \cdot p'(g^{-1}(g(x_{k-1})))\\
= &\sum_{j=1}^{n} \left[g(x_k) = g(x_{k-1}) + a \cdot c_j\right] \cdot p_j\left(\tfrac{g(x_{k-1})+b}{a}\right)\\
&+ [g(x_k) = g(w)] \cdot p'\left(\tfrac{g(x_{k-1})+b}{a}\right),
\end{aligned}$$

which is indeed the probability to enter $g(x_k)$ from $g(x_{k-1})$ in $\mathbf{L}_t$. Note that in the sums above at most two addends are non-zero as w and $x_{k-1} + c_j$ might coincide. Moreover, since all the c_i are pairwise distinct, the same holds for $a \cdot c_i$ as a is non-zero.

(ii) We have

$$a \cdot x - b > 0 \iff g(x) > 0.$$

Since the condition of $\mathbf{L}$ is $a \cdot x - b > 0$ and the condition of $\mathbf{L}_t$ is $x > 0$, the result follows.

(iii) This follows from the two previous results.

□

Thus, the termination behavior of the loop in Fig. 16.2(a) and the termination behavior of the loop in Fig. 16.2(b) are equivalent. Therefore, we can assume that a one-dimensional *pa*-loop has the form in Fig. 16.2(c): here, we assume that $m > 0$ implies $p_m(x) > 0$ and $k > 0$ implies $p_k(x) > 0$ for at least one $x \in \mathbb{Z}_{>0}$ and that $w \leq 0$.

For such loops, the so-called *drift* plays a crucial role, i.e., the expected value of the change of the program variable in one loop iteration when ignoring the part including direct termination, see [BG05].

Definition 16.1.11 (Drift)**.** Let $\mathbf{L}$ be a one-dimensional *pa*-loop as in Fig. 16.2(c). Its *drift* is the function $\mu_{\mathbf{L}} : \mathbb{Z} \longrightarrow \mathbb{R}$ with

$$\mu_{\mathbf{L}}(x) = [x > 0] \cdot \sum_{j=-k}^{m} j \cdot p_j(x)$$

The definition of the drift is justified by the following result where we use the stochastic process defined in Def. 14.4.21. Since we have only one program variable x and the condition is $x > 0$, its expected outcome is of course of interest. Thus, we would like to use the wp-calculus to reason about this value. However, this is not yet possible as x might take *negative* values and for wp-reasoning we need *non*-negative values. However, in any admissible run of a 1-dimensional *pa*-loop, the value of x is at least $-k+1$. Hence, we shift the value of x by $k-1$, i.e., we consider the expectation f with $f(x) = [x \geq -(k-1)] \cdot (x + k - 1)$ to fix this gap. In the following, we denote by $\mathbf{X}$ the stochastic process $\mathbf{X}^{f,f}$.

Lemma 16.1.12. *Let* $\mathbf{L}$ *be a one-dimensional pa-loop* without *direct termination. Then for every* $n \in \mathbb{N}$

$$X_n^{f,f+\mu_{\mathbf{L}}} = \mathbb{E}_{x_0}^{\mathbf{L}}\left(X_{n+1} \mid \mathcal{G}_n^{loop}\right)$$

Proof. This result follows from Thm. 14.4.25. First of all, *pa*-loops are purely proba-

bilistic, i.e., we have no schedulers in Thm. 14.4.25. Moreover,

$$\begin{aligned}
&{}^{\mathrm{wp}}\Phi_f(f)\\
&= [x \leq 0] \cdot [x \geq -(k-1)] \cdot (x+k-1)\\
&\quad + [x > 0] \cdot \sum_{j=-k}^{m} p_j(x) \cdot [x+j \geq -(k-1)] \cdot (x+j+k-1)\\
&= [0 \geq x \geq -(k-1)] \cdot (x+k-1)\\
&\quad + [x > 0] \cdot \left(\sum_{j=-k}^{m} p_j(x) \cdot (x+k-1) + \sum_{j=-k}^{m} p_j(x) \cdot j \right)\\
&= [0 \geq x \geq -(k-1)] \cdot (x+k-1) + [x > 0] \cdot (x+k-1+\mu_{\mathbf{L}}(x))\\
&= f + \mu_{\mathbf{L}}.
\end{aligned}$$

This proves the result. □

We now present two technical results concerning the process $\mathbf{X}$. These results then imply sufficient criteria for proving/refuting (P)AST for one-dimensional *pa*-loops. Here, the first result is obvious since in each iteration of a one-dimensional *pa*-loop without direct termination as in Fig. 16.2(c), the value changes in each loop iteration by some addend contained in $[-k, m]$, i.e., $\mathbf{X}$ is difference bounded (see Ex. 15.3.14).

Lemma 16.1.13. *Let* $\mathbf{L}$ *be a one-dimensional pa-loop without direct termination and* $n \in \mathbb{N}$. *We have almost-surely*

$$|X_{n+1} - X_n| \leq \max(m, k).$$

The following lemma is based on [Küs19], a Bachelor's thesis which has been supervised by the author of this thesis.

Lemma 16.1.14. *Let* $\mathbf{L}$ *be a one-dimensional non-trivial pa-loop without direct termination. For every* $x_0 \in \mathbb{Z}$ *and* $z \in \mathbb{Z}_{>0}$ *we have*

$$\mathbb{P}^{\mathbf{L}}_{x_0}\left(\mathbb{Z}^* z^\omega\right) = 0.$$

Proof. Fix, $n, i \in \mathbb{N}$ and consider the set $A_{n,z,i} = \{\vartheta \in \mathbb{Z}^\omega \mid \vartheta[j] = z \text{ for all } n \leq j \leq n+i\}$. Then $\mathbb{P}^{\mathbf{L}}_{x_0}\left(A_{n,z,i}\right) \leq p_0(z)^i$ by Lemma 14.4.9. Thus,

$$\sum_{i \in \mathbb{N}} \mathbb{P}^{\mathbf{L}}_{x_0}\left(A_{n,z,i}\right) \leq \sum_{i \in \mathbb{N}} p_0(z)^i = \frac{1}{1-p_0(z)} < \infty$$

by convergence of the geometric series since $p_0(z) < 1$. By the Lemma of Borel-Cantelli (Lemma A.2.1), we have $\mathbb{P}^{\mathbf{L}}_{x_0}\left(\bigcap_{s\in\mathbb{N}}\bigcup_{i\in\mathbb{N}_{\geq s}} A_{n,z,i}\right) = 0$. Hence,

$$\mathbb{P}^{\mathbf{L}}_{x_0}\left(\bigcup_{n\in\mathbb{N}}\left(\bigcap_{s\in\mathbb{N}}\bigcup_{i\in\mathbb{N}_{\geq s}} A_{n,z,i}\right)\right) \leq \sum_{n\in\mathbb{N}} \mathbb{P}^{\mathbf{L}}_{x_0}\left(\bigcap_{s\in\mathbb{N}}\bigcup_{i\in\mathbb{N}_{\geq s}} A_{n,z,i}\right) = 0.$$

This proves the result: We have $\bigcap_{s\in\mathbb{N}}\bigcup_{i\in\mathbb{N}_{\geq s}} A_{n,z,i} = \mathbb{Z}^n z^\omega$, i.e.,

$$\bigcup_{n\in\mathbb{N}}\left(\bigcap_{s\in\mathbb{N}}\bigcup_{i\in\mathbb{N}_{\geq s}} A_{n,z,i}\right) = \mathbb{Z}^* z^\omega.$$

□

Why are these two results interesting for us? Lemma 16.1.13 is a precondition for using existing results to refute (P)AST. Lemma 16.1.14 is used in combination with the Supermartingale Convergence Theorem (Thm. A.7.2): remember that the process $\mathbf{X}$ only takes integer values. Thus, the process $\mathbf{X}$ can only converge almost-surely if for each $\vartheta \in \mathbb{Z}^\omega$ the integer sequence $(X_j(\vartheta))_{j\in\mathbb{N}}$ stabilizes. In Lemma 16.1.14 we proved that $\mathbf{X}$ cannot stabilize on values greater than $k-1$ (remember that we had to add $k-1$ to ensure non-negativity of the expectation f above). Thus, if $\mathbf{X}$ converges almost-surely, it stabilizes on a value $\leq k-1$ and due to the shift of $k-1$, the value of x is then ≤ 0. Thus, the underlying *pa*-loop has to *terminate* almost-surely, i.e., it is AST.

Now the behavior of the drift leads to sufficient criteria for the termination behavior of a one-dimensional non-trivial *pa*-loop.

Theorem 16.1.15 (Sufficient Criteria for Proving and Refuting (P)AST)**.** *Let* $\mathbf{L}$ *be a one-dimensional non-trivial pa-loop without direct termination and* $u > 0$.

(i) If $\mu_{\mathbf{L}}(x) \leq 0$ *for all* $x > 0$, $\mathbf{L}$ *is AST.*

(ii) If $\mu_{\mathbf{L}}(x) \leq -u$ *for all* $x > 0$, $\mathbf{L}$ *is PAST.*

(iii) If $\mu_{\mathbf{L}}(x) \geq 0$ *for all* $x > 0$, $\mathbf{L}$ *is not PAST.*

(iv) If $\mu_{\mathbf{L}}(x) \geq u$ *for all* $x > 0$, $\mathbf{L}$ *is not AST.*

Proof. Fix some initial state $x_0 > 0$ and consider the induced probability measure $\mathbb{P}^{\mathbf{L}}_{x_0}$. Clearly, $\mathbf{X}$ is non-negative and $\mathbb{E}^{\mathbf{L}}_{x_0}(X_n)$ is finite for every n by Cor. 14.4.27 as the body of $\mathbf{L}$ is loop-free and $f \ll \infty$. In the third and fourth case, we use a stochastic process,

which takes also non-positive values but is bounded from above. Although we have not discussed such processes, their analysis is equivalent to non-negative processes and the basic concepts such as martingales (Def. 14.4.29) can be defined analogously.

(i) By Lemma 16.1.12, $\mathbf{X}$ is a supermartingale. By Thm. A.7.2, $\mathbf{X}$ converges with probability 1. Hence, Lemma 16.1.14 implies that $\mathbf{X}$ almost-surely stabilizes on a value $\leq k-1$, i.e., if started in x_0, the loop $\mathbf{L}$ terminates almost-surely.

(ii) In this case, $\frac{\mathbf{X}}{u}$ is a ranking supermartingale for $T^{\mathbf{L}}$ (see Def. B.45.11), i.e., (in some sense) $\mathbb{E}^{\mathbf{L}}_{x_0}\left(\frac{X_{n+1}}{u} \mid \mathcal{G}^{\text{loop}}_n\right) \leq \frac{X_n}{u} - 1$. Thus, $\mathbb{E}^{\mathbf{L}}_{x_0}\left(T^{\mathbf{L}}\right) < \infty$, i.e., the loop is PAST (see Thm. B.45.16).

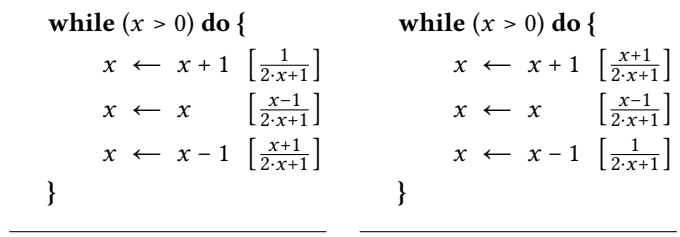
(iii) By Lemmas 16.1.12 and 16.1.13, $-\mathbf{X}+k-1$ is a difference bounded supermartingale. Additionally, whenever $T^{\mathbf{L}}(\vartheta) = j$, then $\vartheta[j] \leq 0$, i.e., $-X_j(\vartheta)+k-1 = -(\vartheta[j]+k-1)+k-1 \geq 0$. Thus, $-\mathbf{X}+k-1$ is a so-called repulsing supermartingale for $T^{\mathbf{L}}$. Moreover, as $\mu_{\mathbf{L}}(x) \geq 0$ for all $x > 0$, the probability that $x > 0$ is increased in an iteration is positive. Thus, $\mathbb{P}^{\mathbf{L}}_{x_0}(-X_0+k-1<0) > 0$. By the relaxed version of [CNZ17, Thm. 7] in [Moo+21, Thm. 5], $\mathbb{E}^{\mathbf{L}}_{x_0}\left(T^{\mathbf{L}}\right) = \infty$, i.e., the loop is not PAST.

(iv) By Lemmas 16.1.12 and 16.1.13, $\frac{-\mathbf{X}+k-1}{u}$ is a difference bounded 1-ranking supermartingale. As in the previous case, it is repulsing and $\mathbb{P}^{\mathbf{L}}_{x_0}\left(\frac{-X_0+k-1}{u} < 0\right) > 0$, i.e., it is a repulsing 1-ranking supermartingale for $T^{\mathbf{L}}$. By the relaxed version of [CNZ17, Thm. 6] in [Moo+21, Thm. 5], $\mathbb{P}^{\mathbf{L}}_{x_0}\left((T^{\mathbf{L}})^{-1}(\mathbb{N})\right) < 1$, i.e., the loop is not AST.

This concludes the proof. □

Example 16.1.16. Consider the loops $\mathbf{L}_{past}$ in Fig. 16.3(a) and $\mathbf{L}_{nast}$ in Fig. 16.3(b). We have $\mu_{\mathbf{L}_{past}}(x) = -\frac{x}{2\cdot x+1} = -\mu_{\mathbf{L}_{nast}}(x)$ for $x > 0$. Moreover, for all $x > 0$ we have $\mu_{\mathbf{L}_{past}}(x) \leq -\frac{1}{3}$ and thus $\mu_{\mathbf{L}_{nast}}(x) \geq \frac{1}{3}$. By Thm. 16.1.15, $\mathbf{L}_{past}$ is PAST and $\mathbf{L}_{nast}$ is not AST. While these programs are of course "toy" examples, they involve non-constant probabilities which are usually difficult to handle.

In Sect. 15.6.2, we have shown how to use our results from Ch. 15 to refute PAST for the fair-in-the-limit random walk. Using Thm. 16.1.15, we can show this directly.



(a) $\mathbf{L}_{past}$ (b) $\mathbf{L}_{nast}$

Figure 16.3: Loops From Ex. 16.1.16

Example 16.1.17. Consider the fair-in-the-limit random walk $\mathbf{L}_{filrw}$ in Fig. 15.13. Its drift is $\mu_{\mathbf{L}_{filrw}}(x) = [x > 0] \cdot \frac{1}{2 \cdot x+1} \geq 0$, i.e., this loop is not PAST by Thm. 16.1.15.

However, $\mathbf{L}_{filrw}$ is AST so a non-negative drift does not suffice to refute AST.

Nevertheless, we should note that the idea illustrated in Sect. 15.6.2 can also be applied to more complex loops, while Thm. 16.1.15 only works for one-dimensional *pa*-loops.

Still, for $x > 0$ the drift of a one-dimensional non-trivial *pa*-loop is a rational function, i.e., the case analysis in Thm. 16.1.15 is far from complete and only covers the most basic behavior. For example, the following result can be found in [Küs19]:

Theorem 16.1.18 (Eventually Non-Positive Drift Implies AST)**.** *Let* $\mathbf{L}$ *be a one-dimensional non-trivial pa-loop without direct termination. If there is some* $c > 0$ *such that* $\mu_{\mathbf{L}}(x) \leq 0$ *for all* $x > c$*, then* $\mathbf{L}$ *is AST.*

16.2 Expected Looping Time of *pa*-Loops with Direct Termination

We start with stating a decidability result for the case where for all $\vec{x}$ with $\vec{a} \cdot \vec{x} - b > 0$ the probability $p'(\vec{x})$ for direct termination is at least p' for some $p' > 0$. Intuitively, these loops have a looping time whose distribution is closely related to the geometric distribution with parameter p' (which has expected value $\frac{1}{p'}$). By using the alternative characterization of $\mathsf{explt}\ [\![\mathbf{L}]\!]$ in Cor. 16.1.8, we obtain that such loops are *always PAST*

while $((-1,1)\cdot(h,t)-(-1)>0)$ **do {**
$(h,t) \leftarrow (h,t)+(0,1)\ \left[\frac{9}{10}\right]$
$(h,t) \leftarrow (8,7)\ \left[\frac{1}{10}\right]$
}

(a) $\mathbf{L}_{direct}$

while $(x>0)$ **do {**
$x \leftarrow x+1\ \left[\frac{x}{x+1}\right]$
$x \leftarrow 0\ \left[\frac{1}{x+1}\right]$
}

(b) $\mathbf{L}_{npast}$

Figure 16.4: Loops From Ex. 16.2.2 and Counterex. 16.2.3

and their expected looping time is indeed *bounded by the constant* $\frac{1}{p'}$. This result will be used in Sect. 16.4 when computing the *exact* expected looping time of such loops. The more involved case where $p'(\vec{x})=0$ is considered in Sect. 16.3.2 and 16.3.3.

Theorem 16.2.1 (Expected Looping Time for Loops With Direct Termination)**.** *Let* $\mathbf{L}$ *be a pa-loop where there is a* $p'>0$ *such that* $p'(\vec{x})\geq p'$ *for all* $\vec{x}\in\mathbb{Z}^d$ *with* $\vec{a}\cdot\vec{x}>b$. *Then* $\mathbf{L}$ *is PAST and its expected looping time is at most* $\frac{1}{p'}$, *i.e.,* $\mathsf{explt}\,[\![\mathbf{L}]\!]\,(\vec{x}_0)\leq\frac{1}{p'}$ *if* $\vec{a}\cdot\vec{x}_0-b>0$, *and* $\mathsf{explt}\,[\![\mathbf{L}]\!]\,(\vec{x}_0)=0$ *if* $\vec{a}\cdot\vec{x}_0-b\leq 0$.

Example 16.2.2. Consider the variant $\mathbf{L}_{direct}$ of $\mathbf{L}_{race}$ (see Fig. 16.1(b)) in Fig. 16.4(a), where in each iteration the hare either does nothing with probability $\frac{9}{10}$ or one directly reaches a configuration where the hare is ahead of the tortoise. By Thm. 16.2.1, this loop is PAST and for all $(h,t)\in\mathbb{Z}^2$ it is $\mathsf{explt}\,[\![\mathbf{L}_{direct}]\!]\,(h,t)\leq\frac{1}{\frac{1}{10}}=10$, i.e., independent of the initial state it takes at most 10 loop iterations on average. In Sect. 16.4, it will turn out that 10 is indeed the exact expected looping time, see Ex. 16.4.13.

It turns out that the precondition of Thm. 16.2.1 cannot be weakened since there are loops with direct termination that are not PAST.

Counterexample 16.2.3. Consider the loop $\mathbf{L}_{npast}$ in Fig. 16.4(b). For every $x\in\mathbb{Z}$ with $x>0$ we have $\frac{1}{x+1}>0$ but there is no $p'>0$ such that $\frac{1}{x+1}>p'$ for all $x\in\mathbb{Z}_{>0}$. Thus, Thm. 16.2.1 cannot be applied. However, the loop $\mathbf{L}_{npast}$ is *not*

PAST: Consider $\Phi_{\langle \mathbf{L}_{npast} \rangle}$. For $x > 0$ we have

$$\Phi_{\langle \mathbf{L}_{npast} \rangle}(f)(x) = 1 + \tfrac{x}{x+1} \cdot f(x+1) + \tfrac{1}{x+1} \cdot f(0).$$

Since the least fixed point of $\Phi_{\langle \mathbf{L}_{npast} \rangle}$ is the expected looping time $\mathrm{explt}\,[\![\mathbf{L}_{npast}]\!]$ and $\mathrm{explt}\,[\![\mathbf{L}_{npast}]\!](0) = 0$ by Cor. 16.1.5, $\mathrm{explt}\,[\![\mathbf{L}_{npast}]\!]$ is a solution of

$$f(x) = 1 + \tfrac{x}{x+1} \cdot f(x+1)$$

for $x > 0$. Now it turns out that the only non-negative function f satisfying this equation for every positive x is given by the infinite series $\sum_{k \in \mathbb{N}} \frac{x}{x+k}$. By using the divergence of the harmonic series, we obtain that $\sum_{k \in \mathbb{N}} \frac{x}{x+k} = \infty$ for every $x > 0$ (see [Küs19] for details). So for each $x > 0$ we have $\mathrm{explt}\,[\![\mathbf{L}_{npast}]\!](x) = \sum_{k \in \mathbb{N}} \frac{x}{x+k} = \infty$ and thus, this loop is not PAST.

16.3 Expected Looping Times of *cpa*-Loops

The results of Sect. 16.2 already cover *cpa*-loops with direct termination. Now we present a very simple decision procedure for termination of *cpa*-loops without direct termination (Sect. 16.3.2) and show how to infer their asymptotic expected looping times (Sect. 16.3.3). This will be needed for the computation of exact expected looping times in Sect. 16.4.

We begin this section with reducing (P)AST and the expected looping time of general *cpa*-loops (with or without direct termination) to a simpler class of loops.

16.3.1 Reduction to Random Walk Loops

In this section, we show that we can restrict ourselves to *random walk loops*, i.e., loops with a single variable x and condition $x > 0$.

Definition 16.3.1 (Random Walk Loop)**.** A one-dimensional *cpa*-loop $\mathbf{L}$ of the form in Fig. 16.2(c) is called a *random walk loop*. Thus, all probabilities are constants, i.e., they do not depend on the value of the variable x.

Def. 16.3.2 shows how to transform any *cpa*-loop as in Def. 16.1.1 into a random walk loop. The idea is to replace the vector of variables $\vec{x}$ by a single variable x that stands for $\vec{a} \bullet \vec{x} - b$. Thus, the condition $\vec{a} \bullet \vec{x} - b > 0$ now becomes $x > 0$. Moreover, a change from $\vec{x}$ to $\vec{x} + \vec{c}_j$ now becomes a change from x to $x + \vec{a} \bullet \vec{c}_j$.

$$
\begin{array}{l}
\textbf{while } (\vec{a} \cdot \vec{x} - b > 0) \textbf{ do } \{ \\
\quad \vec{x} \leftarrow \vec{x} + \vec{c}_1 \quad [p_{\vec{c}_1}] \\
\quad \vdots \\
\quad \vec{x} \leftarrow \vec{x} + \vec{c}_n \quad [p_{\vec{c}_n}] \\
\quad \vec{x} \leftarrow \vec{w} \qquad [p'] \\
\}
\end{array}
$$

(a) d-Dimensional *cpa*-Loop

$$
\begin{array}{l}
\textbf{while } (x > 0) \textbf{ do } \{ \\
\quad x \leftarrow x + m_{\mathbf{L}} \quad \left[p^{rdw}_{m_{\mathbf{L}}}\right] \\
\quad \vdots \\
\quad x \leftarrow x - k_{\mathbf{L}} \quad \left[p^{rdw}_{-k_{\mathbf{L}}}\right] \\
\quad x \leftarrow w_{\mathbf{L}} \qquad [p'] \\
\}
\end{array}
$$

(b) Transformation via $rdw_{\mathbf{L}}$

Figure 16.5: Illustration of $rdw_{\mathbf{L}}$ From Def. 16.3.2

Definition 16.3.2 (Transforming *cpa*-Loops to Random Walk Loops)**.** Let $\mathbf{L}$ be a d-dimensional *cpa*-loop as in Fig. 16.5(a). Let $rdw_{\mathbf{L}}$ be the map $rdw_{\mathbf{L}} : \mathbb{Z}^d \to \mathbb{Z}$ with $rdw_{\mathbf{L}}(\vec{z}) = \vec{a} \cdot \vec{z} - b$. Let $rdw_{\mathbf{L}}(\vec{w}) = w_{\mathbf{L}}$. Let $k_{\mathbf{L}}, m_{\mathbf{L}} \in \mathbb{N}$ be minimal such that $-k_{\mathbf{L}} \leq \vec{a} \cdot \vec{c}_j \leq m_{\mathbf{L}}$ for all $1 \leq j \leq n$ with $p_{c_j} > 0$. For $-k_{\mathbf{L}} \leq u \leq m_{\mathbf{L}}$, we define $p^{rdw}_u = \sum_{1 \leq j \leq n, \vec{a} \cdot \vec{c}_j = u} p_{\vec{c}_j}$. This results in the random walk loop $\mathbf{L}^{rdw}$ in Fig. 16.5(b).

Example 16.3.3. For the loop $\mathbf{L}_{race}$ of Ex. 16.1.2, the mapping $rdw_{\mathbf{L}_{race}} : \mathbb{Z}^2 \to \mathbb{Z}$ is $rdw_{\mathbf{L}_{race}}(h,t) = (-1,1) \cdot (h,t) - (-1) = t - h + 1$. Hence we obtain the random walk loop $\mathbf{L}^{rdw}_{race}$ in Fig. 16.6(a) where $x = rdw_{\mathbf{L}_{race}}(h,t)$ represents the distance between the tortoise and the hare.

Approaches based on supermartingales (e.g., [ACN18; BG05; Cha+18; CNZ17; CS13; FH15]) use mappings similar to $rdw_{\mathbf{L}}$ in order to infer a real-valued term which over-approximates the expected looping time (see also the concept of probabilistic ranking functions in Sect. 17.3.1). However, in the following (non-trivial) theorem we show that our transformation is not only an over- or under-approximation, but the termination behavior and the expected looping time of $\mathbf{L}$ and $\mathbf{L}^{rdw}$ are *identical*. This result is similar to Thm. 16.1.10, where we showed that w.l.o.g. we can restrict ourselves to one-dimensional *pa*-loops with $a = 1$, $b = 0$. In Thm. 16.3.4, we show that we can restrict ourselves to *cpa*-loops with $d = 1$, $a = 1$, $b = 0$. While Thm. 16.1.10

while $(x > 0)$ **do** {
$x \leftarrow x + 1 \quad \left[\frac{6}{11}\right]$
$x \leftarrow x \quad \left[\frac{1}{22}\right]$
$x \leftarrow x - 1 \quad \left[\frac{1}{22}\right]$
$\vdots$
$x \leftarrow x - 9 \quad \left[\frac{1}{22}\right]$
}

(a) $\mathbf{L}^{rdw}_{race}$

while $(x > 0)$ **do** {
$x \leftarrow x \ [1]$
}

(b) Trivial Random Walk Loop

Figure 16.6: Random Walk Loops

relied on the fact that the function $x \mapsto a \cdot x - b$ is bijective in order to transform the non-constant probabilities, we do not need this in case of constant probabilities.

Theorem 16.3.4 (Transformation Preserves Termination Behavior)**.** *Let* $\mathbf{L}$ *be a* d*-dimensional cpa-loop as in Fig. 16.5(a) and let* $\mathbf{L}^{rdw}$ *be the loop in Fig. 16.5(b).*

(i) For every run $\vartheta \in (\mathbb{Z}^d)^\omega$ *we have* $T^{\mathbf{L}}(\vartheta) = T^{\mathbf{L}^{rdw}}(rdw_{\mathbf{L}}(\vartheta))$.

(ii) For every $\vec{x}_0 \in \mathbb{Z}^d$ *and* $j \in \mathbb{N}$ *we have* $\mathbb{P}^{\mathbf{L}}_{\vec{x}_0}\left(T^{\mathbf{L}} = j\right) = \mathbb{P}^{\mathbf{L}^{rdw}}_{rdw_{\mathbf{L}}(\vec{x}_0)}\left(T^{\mathbf{L}^{rdw}} = j\right)$.

So the looping times $T^{\mathbf{L}}$ and $T^{\mathbf{L}^{rdw}}$ are identically distributed w.r.t. $rdw_{\mathbf{L}}$. In particular,

$$\mathbb{P}^{\mathbf{L}}_{\vec{x}_0}\left(\left(T^{\mathbf{L}}\right)^{-1}(\mathbb{N})\right) = \mathbb{P}^{\mathbf{L}^{rdw}}_{rdw_{\mathbf{L}}(\vec{x}_0)}\left(\left(T^{\mathbf{L}^{rdw}}\right)^{-1}(\mathbb{N})\right)$$

and

$$\mathsf{explt}\ [\![\mathbf{L}]\!]\left(\vec{x}_0\right) = \mathbb{E}^{\mathbf{L}}_{\vec{x}_0}\left(T^{\mathbf{L}}\right) = \mathbb{E}^{\mathbf{L}^{rdw}}_{rdw_{\mathbf{L}}(\vec{x}_0)}\left(T^{\mathbf{L}^{rdw}}\right) = \mathsf{explt}\ \left[\!\left[\mathbf{L}^{rdw}\right]\!\right]\left(rdw_{\mathbf{L}}\left(\vec{x}_0\right)\right).$$

We now identify pathological loops that can be disregarded.

Definition 16.3.5 (Trivial *cpa*-Loops)**.** A *cpa*-loop $\mathbf{L}$ is *trivial* if it is trivial as a

pa-loop (see Def. 16.1.1) or if $\mathbf{L}^{rdw}$ is the loop in Fig. 16.6(b).

Note that a random walk loop $\mathbf{L}$ is trivial iff it is the loop in Fig. 16.6(b) since $\mathbf{L} = \mathbf{L}^{rdw}$ holds for random walk loops $\mathbf{L}$. From now on, we exclude trivial *cpa*-loops $\mathbf{L}$ as their termination behavior is obvious: for inputs $\vec{x}_0$ with $\vec{a} \cdot \vec{x}_0 - b > 0$, the loop never terminates (i.e., $\text{explt}\ [\![\mathbf{L}]\!]\ (\vec{x}_0) = \infty$) and for inputs $\vec{x}_0$ with $\vec{a} \cdot \vec{x}_0 - b \leq 0$ we have $\text{explt}\ [\![\mathbf{L}]\!]\ (\vec{x}_0) = 0$. If $\vec{a} = \vec{0}$, the termination behavior just depends on b: if $b < 0$, then $\text{explt}\ [\![\mathbf{L}]\!]\ (\vec{x}_0) = \infty$ for all $\vec{x}_0$ and if $b \geq 0$, then $\text{explt}\ [\![\mathbf{L}]\!]\ (\vec{x}_0) = 0$ for all $\vec{x}_0$.

16.3.2 Deciding Termination

We now present a simple decision procedure for (P)AST of random walk loops $\mathbf{L}$. By the results of Sect. 16.3.1, this also yields a decision procedure for arbitrary *cpa*-loops. If $p' > 0$, Thm. 16.2.1 already shows that $\mathbf{L}$ is PAST and its expected looping time is bounded by the constant $\frac{1}{p'}$. Thus, in the rest of Sect. 16.3.2 we regard *random walk loops without direct termination*, i.e., $p' = 0$.

As the drift $\mu_{\mathbf{L}}$ of a random walk loop $\mathbf{L}$ is a constant for all $x > 0$, Thm. 16.1.15 implies the following decision procedure for (P)AST where we identify $\mu_{\mathbf{L}}$ with the value it takes on positive numbers.

Corollary 16.3.6 (Decision Procedure for (P)AST of Random Walk Loops)**.** *Let* $\mathbf{L}$ *be a non-trivial random walk loop without direct termination.*

- *If* $\mu_{\mathbf{L}} > 0$*, the loop is* not *AST.*
- *If* $\mu_{\mathbf{L}} = 0$*, the loop is AST but* not *PAST.*
- *If* $\mu_{\mathbf{L}} < 0$*, the loop is PAST.*

In fact, the proof of Thm. 16.1.15 implies that the termination behavior of a non-trivial random walk loop is the same for each positive initial value, e.g., if it is not AST, there is a positive probability of non-termination for every positive initial value.

Example 16.3.7. The drift of $\mathbf{L}^{rdw}_{race}$ from Fig. 16.6(a) introduced in Ex. 16.3.3 is $\mu_{\mathbf{L}^{rdw}_{race}} = 1 \cdot \frac{6}{11} + \frac{1}{22} \cdot \sum_{-9 \leq j \leq 0} j = -\frac{3}{2} < 0$. So on average the distance x between the tortoise and the hare decreases by $\frac{3}{2}$ in each loop iteration. Hence by Cor. 16.3.6, $\mathbf{L}^{rdw}_{race}$ is PAST and the following Cor. 16.3.9 implies that $\mathbf{L}_{race}$ as shown in Fig. 14.1 and 16.1(b) is PAST as well.

Let us also come back to the random walk.

Example 16.3.8. Reconsider the loop $\mathbf{L}_{rw}$ in Fig. 14.9. Its drift is $\mu_{\mathbf{L}_{rw}} = 0$ and thus, we obtain the well-known result that $\mathbf{L}_{rw}$ is AST but not PAST by Cor. 16.3.6 (see Ex. 14.2.18).

Corollary 16.3.9 (Decision Procedure for (P)AST of *cpa*-Loops)**.** *For a non-trivial cpa-loop* $\mathbf{L}$*,* $\mathbf{L}$ *is (P)AST iff* $\mathbf{L}^{rdw}$ *is (P)AST. Hence, Thm. 16.2.1 and 16.3.4 and Cor. 16.3.6 yield a decision procedure for AST and PAST of cpa-loops.*

While the proof of Thm. 16.1.15 is based on known results, the formulation of Cor. 16.3.6 shows that there is an extremely *simple* decision procedure for (P)AST of *cpa*-loops, i.e., checking the sign of the drift is much simpler than applying existing (general) techniques for termination analysis of probabilistic programs.

Moreover, it is worth noting that in this thesis we have taken a different approach to proving this result. While in [GGH19b] we have presented a proof based on a connection of random walk loops to random walks from probability theory (see, e.g., [Fel50; GS01; Spi64]), and then using the equivalent result for random walks and so-called hitting times as stated in [Spi64, Thm. 17.1 and Prop. 18.1], we presented a proof based on existing results for martingales here. This alternative idea for proving these results was only sketched briefly in [GGH19b].

16.3.3 Computing Asymptotic Expected Looping Times

It turns out that for random walk loops (and thus by Thm. 16.3.4, also for *cpa*-loops), one can not only decide termination, but one can also infer tight bounds on the expected looping time. Again, the computation of the bounds is very *simple*.

Theorem 16.3.10 (Bounds on the Expected Looping Time of *cpa*-Loops)**.** *Let* $\mathbf{L}$ *be a non-trivial cpa-loop without direct termination which is PAST (i.e.,* $\mu_{\mathbf{L}^{rdw}} < 0$*). Moreover, let* $k_{\mathbf{L}}$ *be obtained according to the transformation in Def. 16.3.2. If* $rdw_{\mathbf{L}}(\vec{x}_0) \leq 0$*,* $\mathsf{explt}\ [\![\mathbf{L}]\!]\ (\vec{x}_0) = 0$*. If* $rdw_{\mathbf{L}}(\vec{x}_0) > 0$*,* $\mathsf{explt}\ [\![\mathbf{L}]\!]\ (\vec{x}_0)$ *is asymptotically linear and it is*

$$-\frac{1}{\mu_{\mathbf{L}^{rdw}}} \cdot rdw_{\mathbf{L}}\ (\vec{x}_0) \leq \mathsf{explt}\ [\![\mathbf{L}]\!]\ (\vec{x}_0) \leq -\frac{1}{\mu_{\mathbf{L}^{rdw}}} \cdot rdw_{\mathbf{L}}\ (\vec{x}_0) + \frac{1-k_{\mathbf{L}}}{\mu_{\mathbf{L}^{rdw}}}.$$

Example 16.3.11. In Ex. 16.3.7, we saw that the program $\mathbf{L}^{rdw}_{race}$ in Fig. 16.6(a) is PAST as it has the drift $\mu_{\mathbf{L}^{rdw}_{race}} = -\frac{3}{2} < 0$. Note that here $k = 9$. Hence by Thm. 16.3.10, we get that whenever $rdw_{\mathbf{L}_{race}}(h,t) = t - h + 1 > 0$, the expected looping time $\mathsf{explt}\ [\![\mathbf{L}_{race}]\!]\ (h,t)$ is between $-\frac{1}{\mu_{\mathbf{L}^{rdw}_{race}}} \cdot rdw_{\mathbf{L}_{race}}(h,t) = \frac{2}{3} \cdot (t-h+1)$

and $-\frac{1}{\mu_{\mathbf{L}^{rdw}_{race}}} \cdot rdw_{\mathbf{L}_{race}}(h,t) + \frac{1-k}{\mu_{\mathbf{L}^{rdw}_{race}}} = \frac{2}{3} \cdot (t-h+1) + \frac{16}{3}$. The same upper bound $\frac{2}{3} \cdot (t-h+1) + \frac{16}{3}$ was inferred in [NCH18] by an incomplete technique based on several inference rules and linear programming solvers. In contrast, Thm. 16.3.10 allows us to read off such bounds directly from the loop. We will come back to this upper bound in Ex. 17.2.5.

16.4 Computing Exact Expected Looping Times

While Thm. 16.2.1 and 16.3.10 state how to deduce the *asymptotic* expected looping time, we now show that based on these results one can compute the looping time of *cpa*-loops *exactly*. In general, whenever it is possible, then inferring the exact looping times of loops is preferable to asymptotic looping times which ignore the "coefficients" of the looping time.

Again, we first consider *random walk* loops and generalize our technique to *cpa*-loops using Thm. 16.3.4 afterwards. Throughout this section, for any random walk loop $\mathbf{L}$, we require that $\mathbf{L}$ is PAST, i.e., that $p' > 0$ (see Thm. 16.2.1) or that the drift $\mu_{\mathbf{L}}$ is negative if $p' = 0$ (see Cor. 16.3.6). Note that whenever $k = 0$ and $\mathbf{L}$ is PAST, then $p' > 0$: if $p' = 0$ and $k = 0$ we have $\mu_{\mathbf{L}} \geq 0$.

To compute $\mathbf{L}$'s expected looping time exactly, we use its characterization as the least fixed point of the characteristic function $\Phi_{\langle \mathbf{L} \rangle}$ (see Cor. 16.1.8), i.e., $\mathsf{eplt}\,[\![\mathbf{L}]\!]$ is the smallest function $f : \mathbb{Z} \to \overline{\mathbb{R}_{\geq 0}}$ satisfying the constraint

$$f(x) = \sum_{j=-k}^{m} p_j \cdot f(x+j) + p' \cdot f(w) + 1 \quad \text{for all } x > 0, \tag{16.4.1}$$

see (16.1.1). Since $\mathbf{L}$ is PAST, f never returns ∞, i.e., $f : \mathbb{Z} \to \mathbb{R}_{\geq 0}$. Note that the smallest function $f : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ that satisfies (16.4.1) also satisfies

$$f(x) = 0 \quad \text{for all } x \leq 0. \tag{16.4.2}$$

Therefore, the constraint (16.4.1) can be simplified to

$$f(x) = \sum_{j=-k}^{m} p_j \cdot f(x+j) + 1 \quad \text{for all } x > 0. \tag{16.4.3}$$

In Sect. 16.4.1, we recapitulate how to compute all solutions of such inhomogeneous recurrence equations (see e.g., [Ela05, Ch. 2]). However, the challenge is to find the *smallest* solution $f : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ of the recurrence equation (16.4.3) to

while $(x > 0)$ **do** {

$x \leftarrow x + 1 \quad \left[\frac{6}{11}\right]$

$x \leftarrow x \quad \left[\frac{1}{11}\right]$

$x \leftarrow x - 1 \quad \left[\frac{1}{22}\right]$

$x \leftarrow x - 2 \quad \left[\frac{7}{22}\right]$

}

Figure 16.7: $\mathbf{L}^{mod}_{race}$

compute explt $[\![\mathbf{L}]\!]$. Therefore, in Sect. 16.4.2 we will exploit the knowledge gained in Thm. 16.2.1 and 16.3.10 to show that there is only a *single* function f that satisfies both (16.4.2) and (16.4.3) *and* is bounded by a constant (if $p' > 0$, see Thm. 16.2.1) resp. by a linear function (if $p' = 0$, see Thm. 16.3.10). This observation then allows us to compute explt $[\![\mathbf{L}]\!]$ exactly. So the crucial prerequisites for this result are Cor. 16.1.8 (which characterizes the expected looping time as the smallest solution of the recurrence equation (16.4.3)), Cor. 16.3.6 (which allows the restriction to negative drift if $p' = 0$), and in particular Thm. 16.2.1 and 16.3.10 (since Sect. 16.4.2 will show that the results of Thm. 16.2.1 and 16.3.10 on the asymptotic looping time can be translated into suitable conditions on the solutions of (16.4.3)).

Example 16.4.1. To illustrate our approach, we use the modified version of $\mathbf{L}^{rdw}_{race}$ (see Fig. 16.6(a)) stated in Fig. 16.7 to ease readability. In Ex. 16.4.15, we will consider the original loop $\mathbf{L}^{rdw}_{race}$ resp. $\mathbf{L}_{race}$ from Ex. 16.3.3 resp. Ex. 16.1.2 again and show its exact expected looping time inferred by the implementation of our approach. In the modified loop $\mathbf{L}^{mod}_{race}$, the distance between the tortoise and the hare still increases with probability $\frac{6}{11}$, but the probability of decreasing by more than two is distributed to the cases where it stays the same and where it decreases by two. We have $p' = 0$ and the drift is $\mu_{\mathbf{L}^{mod}_{race}} = 1 \cdot \frac{6}{11} + 0 \cdot \frac{1}{11} - 1 \cdot \frac{1}{22} - 2 \cdot \frac{7}{22} = -\frac{3}{22} < 0$. So by Cor. 16.3.6, $\mathbf{L}^{mod}_{race}$ is PAST. By Cor. 16.1.8, explt $[\![\mathbf{L}^{mod}_{race}]\!]$ is the smallest function $f : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ satisfying

$$f(x) = \tfrac{6}{11} \cdot f(x+1) + \tfrac{1}{11} \cdot f(x) + \tfrac{1}{22} \cdot f(x-1) + \tfrac{7}{22} \cdot f(x-2) + 1 \quad \text{for all } x > 0. \quad (16.4.4)$$

16.4.1 Finding All Solutions of the Recurrence Equation

Instead of searching for the *smallest* $f : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ satisfying (16.4.3), we first calculate the set of *all* functions $f : \mathbb{Z} \to \mathbb{C}$ that satisfy (16.4.3), i.e., we now consider functions returning arbitrary complex numbers, e.g., they can also return negative or imaginary numbers. Clearly, (16.4.3) is equivalent to

$$0 = \left(\sum_{j=-k}^{-1} p_j \cdot f(x+j)\right) + (p_0 - 1) \cdot f(x) + \left(\sum_{j=1}^{m} p_j \cdot f(x+j)\right) + 1 \quad \text{for all } x > 0. \tag{16.4.5}$$

The set of solutions on $\mathbb{Z} \to \mathbb{C}$ of this linear inhomogeneous recurrence equation is an affine space which can be written as an arbitrary particular solution of the inhomogeneous equation plus any linear combination of $k + m$ linearly independent solutions of the corresponding homogeneous recurrence equation.

We start with computing a solution to the inhomogeneous equation (16.4.5). To this end, we use the bounds for $\mathrm{explt}\,[\![\mathbf{L}]\!](x)$ from Thm. 16.2.1 and 16.3.10 (where we take the upper bound $\frac{1}{p'}$ if $p' > 0$ and the lower bound $-\frac{1}{\mu_{\mathbf{L}}} \cdot x$ if $p' = 0$).

Definition 16.4.2 (Distinguished Solutions)**.** Let $\mathbf{L}$ be a random walk loop which is PAST. We define the distinguished solution $C_{\mathbf{L}} : \mathbb{Z} \to \mathbb{C}$ of (16.4.5) by

$$C_{\mathbf{L}}(x) = \tfrac{1}{p'} \text{ if } p' > 0 \qquad \text{and} \qquad C_{\mathbf{L}}(x) = \left(-\tfrac{1}{\mu_{\mathbf{L}}}\right) \cdot x \text{ if } p' = 0$$

as well as $C_{const} = \frac{1}{p'}$ if $p' > 0$ and $C_{lin} = -\frac{1}{\mu_{\mathbf{L}}}$ if $p' = 0$.

One easily shows that if $p' > 0$, then $C_{\mathbf{L}}(x) = C_{const}$ is a solution of the inhomogeneous recurrence equation (16.4.5) and if $p' = 0$, then $C_{\mathbf{L}}(x) = C_{lin} \cdot x$ solves (16.4.5).

Example 16.4.3. In the loop $\mathbf{L}^{mod}_{race}$ from Fig. 16.7, we have $p' = 0$ and $\mu_{\mathbf{L}^{mod}_{race}} = -\frac{3}{22}$. Hence $C_{\mathbf{L}^{mod}_{race}}(x) = C_{lin} \cdot x = \frac{22}{3} \cdot x$ is a solution of (16.4.4).

After having determined one particular solution of the inhomogeneous recurrence equation (16.4.5), now we compute the solutions of the *homogeneous* recurrence equation which results from (16.4.5) by replacing the add-on "+ 1" with 0. To this end, we consider the corresponding *characteristic polynomial* $\chi_{\mathbf{L}}$.

Definition 16.4.4 (Characteristic Polynomial, [Ela05])**.** Let $\mathbf{L}$ be a non-trivial random walk loop. We define its *characteristic polynomial*

$$\chi_{\mathbf{L}}(\lambda) = \left(\sum_{j=-k}^{-1} p_j \cdot \lambda^{k+j}\right) + (p_0 - 1) \cdot \lambda^k + \left(\sum_{j=1}^{m} p_j \cdot \lambda^{k+j}\right).$$

Note that we restricted ourselves to loops which are PAST, i.e., $p_0 < 1$. Moreover, we required that $m > 0$ (resp. $k > 0$) implies $p_m > 0$ (resp. $p_{-k} > 0$), see Sect. 16.1.1. Hence, the degree of $\chi_{\mathbf{L}}$ is exactly $m + k$ even if $m = 0$ or $k = 0$.

Let $\lambda_1, \ldots, \lambda_c \in \mathbb{C}$ denote the pairwise different (possibly non-real) roots of the characteristic polynomial $\chi_{\mathbf{L}}$. For all $1 \leq j \leq c$, let $\nu_j \in \mathbb{N}_{\geq 1}$ be the multiplicity of the root λ_j. Thus, we have $\nu_1 + \ldots + \nu_c = k + m$.

Then we obtain the following $k + m$ linearly independent solutions of the homogeneous recurrence equation resulting from (16.4.5):

$$x \mapsto \lambda_j^x \cdot x^u \quad \text{for all } 1 \leq j \leq c \text{ and all } 0 \leq u \leq \nu_j - 1$$

So we have the following result.

Theorem 16.4.5 (Solutions of Inhomogeneous Recurrence Equation, [Ela05])**.** *A function $f : \mathbb{Z} \longrightarrow \mathbb{C}$ is a solution of (16.4.3) (resp. (16.4.5)) iff there exist coefficients $a_{j,u} \in \mathbb{C}$ with*

$$f(x) = C_{\mathbf{L}}(x) + \sum_{j=1}^{c} \sum_{u=0}^{\nu_j - 1} a_{j,u} \cdot \lambda_j^x \cdot x^u \quad \text{for all } x > -k, \tag{16.4.6}$$

where $C_{\mathbf{L}}(x)$ is the solution defined in Def. 16.4.2.

The reason for requiring (16.4.6) for all $x > -k$ is that $-k + 1$ is the smallest argument where f's value is taken into account in (16.4.3).

Example 16.4.6. We continue Ex. 16.4.3. The characteristic polynomial for the loop $\mathbf{L}_{race}^{mod}$ from Fig. 16.7 has degree $k + m = 2 + 1 = 3$ and is given by

$$\chi_{\mathbf{L}_{race}^{mod}}(\lambda) = \tfrac{6}{11} \cdot \lambda^3 + \left(\tfrac{1}{11} - 1\right) \cdot \lambda^2 + \tfrac{1}{22} \cdot \lambda + \tfrac{7}{22} = \tfrac{6}{11} \cdot \lambda^3 - \tfrac{10}{11} \cdot \lambda^2 + \tfrac{1}{22} \cdot \lambda + \tfrac{7}{22}.$$

Its roots are $\lambda_1 = 1$, $\lambda_2 = -\frac{1}{2}$, and $\lambda_3 = \frac{7}{6}$. So here, all roots are real numbers and they all have multiplicity 1. Hence, three linearly independent solutions of the homogeneous part of (16.4.4) are the functions $x \mapsto 1^x = 1$, $x \mapsto \left(-\frac{1}{2}\right)^x$,

and $x \mapsto \left(\frac{7}{6}\right)^x$. As seen in Thm. 16.4.5, a function $f : \mathbb{Z} \longrightarrow \mathbb{C}$ satisfies (16.4.4) iff there are $a_1, a_2, a_3 \in \mathbb{C}$ such that

$$\begin{aligned} f(x) &= C_{\mathbf{L}^{mod}_{race}}(x) + a_1 \cdot 1^x + a_2 \cdot \left(-\tfrac{1}{2}\right)^x + a_3 \cdot \left(\tfrac{7}{6}\right)^x \\ &= \tfrac{22}{3} \cdot x + a_1 + a_2 \cdot \left(-\tfrac{1}{2}\right)^x + a_3 \cdot \left(\tfrac{7}{6}\right)^x \text{ for } x > -2. \end{aligned} \tag{16.4.7}$$

16.4.2 Finding the Smallest Solution of the Recurrence Equation

In Sect. 16.4.1, we recapitulated the standard approach for solving inhomogeneous recurrence equations which shows that any function $f : \mathbb{Z} \longrightarrow \mathbb{C}$ that satisfies the constraint (16.4.3) is of the form (16.4.6). Now we present a technique to compute explt $[\![\mathbf{L}]\!]$, i.e., the *smallest non-negative* solution $f : \mathbb{Z} \longrightarrow \mathbb{R}_{\geq 0}$ of (16.4.3). By Thm. 16.2.1 and 16.3.10, this function f is bounded by a constant (if $p' > 0$) resp. linear (if $p' = 0$). So, when representing f in the form (16.4.6), we must have $a_{j,u} = 0$ whenever $|\lambda_j| > 1$. The following lemma shows how many roots with absolute value less or equal to 1 there are (i.e., these are the only roots that we have to consider). It is proved using Rouché's Theorem [Rou62] which allows us to infer the number of roots whose absolute value is below a certain bound. Note that 1 is a root of the characteristic polynomial iff $p' = 0$ since $\sum_{j=-k}^{m} p_j = 1 - p'$.

Lemma 16.4.7 (Number of Roots With Absolute Value ≤ 1)**.** *Let* $\mathbf{L}$ *be a random walk loop that is PAST. Then its characteristic polynomial* $\chi_{\mathbf{L}}$ *has* k *roots* $\lambda \in \mathbb{C}$ *(counted with multiplicity) with* $|\lambda| \leq 1$.

Example 16.4.8. In $\mathbf{L}^{mod}_{race}$ from Fig. 16.7 we have $k = 2$. So by Lemma 16.4.7, $\chi_{\mathbf{L}}$ has exactly two roots with absolute value ≤ 1. Indeed, the roots of $\chi_{\mathbf{L}}$ are $\lambda_1 = 1$, $\lambda_2 = -\frac{1}{2}$, and $\lambda_3 = \frac{7}{6}$, see Ex. 16.4.6. So $|\lambda_3| > 1$, but $|\lambda_1| \leq 1$ and $|\lambda_2| \leq 1$.

Based on Lemma 16.4.7, the following lemma shows that when imposing the restriction that $a_{j,u} = 0$ whenever $|\lambda_j| > 1$, then there is only a *single* function of the form (16.4.6) that also satisfies the constraint (16.4.2).

Lemma 16.4.9 (Unique Solution of (16.4.2) and (16.4.3) with $a_{j,u} = 0$ whenever $|\lambda_j| > 1$)**.** *Let* $\mathbf{L}$ *be a random walk loop that is PAST. Then there is exactly one function* $f : \mathbb{Z} \longrightarrow \mathbb{C}$ *which satisfies both* (16.4.2) *and* (16.4.3) *and has the form* (16.4.6) *with* $a_{j,u} = 0$ *if* $|\lambda_j| > 1$.

The main theorem of Sect. 16.4 now shows how to compute the expected looping time exactly. By Thm. 16.2.1 and 16.3.10 on the bounds for the expected looping time and by Lemma 16.4.9, we no longer have to search for the *smallest* function that satisfies (16.4.2) and (16.4.3), but we just search for *any* solution of (16.4.2) and (16.4.3) which has $a_{j,u} = 0$ whenever $|\lambda_j| > 1$ (because there is just a single such solution and thus, this solution is the expected looping time). So one only has to determine the values of the remaining k coefficients $a_{j,u}$ for $|\lambda_j| \leq 1$, which can be done by exploiting that $f(x)$ has to satisfy both (16.4.2) for all $x \leq 0$ and it has to be of the form (16.4.6) for all $x > -k$. In other words, the coefficients in (16.4.6) must be $a_{j,u} = 0$ whenever $|\lambda_j| > 1$ and must ensure that the function is 0 for $-k+1 \leq x \leq 0$.

Theorem 16.4.10 (Exact Expected Looping Time for Random Walk Loops)**.** *Let* $\mathbf{L}$ *be a random walk loop that is PAST and let* $\lambda_1, \ldots, \lambda_c$ *be the roots of its characteristic polynomial with multiplicities* $\nu_1, \ldots, \nu_c$*. Moreover, let* $C_{\mathbf{L}}(x)$ *be the distinguished solution defined in Def. 16.4.2. Then we have* $\mathsf{explt}\,[\![\mathbf{L}]\!](x) = 0$ *for* $x \leq 0$ *and*

$$\mathsf{explt}\,[\![\mathbf{L}]\!](x) = C_{\mathbf{L}}(x) + \sum_{j=1,\ |\lambda_j|\leq 1}^{c} \sum_{u=0}^{\nu_j-1} a_{j,u} \cdot \lambda_j^x \cdot x^u \quad \text{for } x > 0,$$

where the coefficients $a_{j,u}$ *are the unique solution of the* k *linear equations*

$$0 = C_{\mathbf{L}}(x) + \sum_{j=1,\ |\lambda_j|\leq 1}^{c} \sum_{u=0}^{\nu_j-1} a_{j,u} \cdot \lambda_j^x \cdot x^u \quad \text{for } -k+1 \leq x \leq 0. \tag{16.4.8}$$

So in the special case where $k = 0$*, we have* $\mathsf{explt}\,[\![\mathbf{L}]\!](x) = C_{\mathbf{L}}(x) = C_{const} = \frac{1}{p'}$ *for* $x > 0$.

Thus, the expected looping time $\mathsf{explt}\,[\![\mathbf{L}]\!](x)$ can be computed by summing up the bound $C_{\mathbf{L}}(x)$ and an add-on $\sum_{j=1,|\lambda_j|\leq 1}^{c} \sum_{u=0}^{\nu_j-1} \ldots$ for all $x > 0$. As $C_{\mathbf{L}}(x)$ is an upper bound for $\mathsf{explt}\,[\![\mathbf{L}]\!](x)$ if $p' > 0$ and a lower bound for $\mathsf{explt}\,[\![\mathbf{L}]\!](x)$ if $p' = 0$ by Thm. 16.2.1 and 16.3.10, this add-on is non-positive if $p' > 0$ and non-negative if $p' = 0$.

Example 16.4.11. By Thm. 16.4.10, the expected looping time of the loop $\mathbf{L}_{race}^{mod}$ from Fig. 16.7 is $\mathsf{explt}\,[\![\mathbf{L}_{race}^{mod}]\!](x) = 0$ for $x \leq 0$ and

$$\mathsf{explt}\left[\!\left[\mathbf{L}_{race}^{mod}\right]\!\right](x) = \tfrac{22}{3} \cdot x + a_1 + a_2 \cdot \left(-\tfrac{1}{2}\right)^x \quad \text{for } x > 0, \text{ see (16.4.7)}.$$

The coefficients a_1 and a_2 are the unique solution of the $k = 2$ linear equations

$$0 = \tfrac{22}{3} \cdot 0 + a_1 + a_2 \cdot \left(-\tfrac{1}{2}\right)^0 = a_1 + a_2$$
$$0 = \tfrac{22}{3} \cdot (-1) + a_1 + a_2 \cdot \left(-\tfrac{1}{2}\right)^{-1} = -\tfrac{22}{3} + a_1 - 2 \cdot a_2$$

So $a_1 = \frac{22}{9} = -a_2$, and hence $\mathsf{explt}\,[\![\mathbf{L}^{mod}_{race}]\!](x) = \frac{22}{3} \cdot x + \frac{22}{9} - \frac{22}{9} \cdot \left(-\frac{1}{2}\right)^x$ for $x > 0$.

By Thm. 16.3.4, we can directly lift Thm. 16.4.10 to arbitrary *cpa*-loops $\mathbf{L}$.

Corollary 16.4.12 (Exact Expected Looping Time for *cpa*-loops)**.** *For any cpa-loop its expected looping time can be computed exactly.*

Note that irrespective of the degree of the characteristic polynomial, its roots can always be approximated numerically with any chosen precision. Thus, "exact computation" of the expected looping time in the corollary above means that a closed form for $\mathsf{explt}\,[\![\mathbf{L}]\!]\left(\vec{x}\right)$ can also be computed with any desired precision.

Example 16.4.13. Reconsider Ex. 16.2.2 where we discussed the loop $\mathbf{L}_{direct}$ (see Fig. 16.4(a)) with the probability $p' = \frac{1}{10}$ for direct termination. $\mathbf{L}_{direct}$ is PAST and its expected looping time is *at most* $\frac{1}{p'} = 10$, see Ex. 16.2.2. The random walk loop $\mathbf{L}^{rdw}_{direct}$ in Fig. 16.8(a) is obtained by the transformation of Def. 16.3.2. As $k = 0$, by Thm. 16.4.10 we obtain $\mathsf{explt}\,[\![\mathbf{L}^{rdw}_{direct}]\!](x) = \frac{1}{p'} = 10$ for $x > 0$. By Thm. 16.3.4, this implies that for (h, t) with $rdw_{\mathbf{L}_{direct}}(h, t) = t - h + 1 > 0$ it is

$$\mathsf{explt}\,[\![\mathbf{L}_{direct}]\!](h, t) = \mathsf{explt}\,\left[\!\!\left[\mathbf{L}^{rdw}_{direct}\right]\!\!\right]\left(rdw_{\mathbf{L}_{direct}}(h, t)\right) = 10,$$

i.e., 10 is indeed the *exact* expected looping time of $\mathbf{L}_{direct}$.

Note that Thm. 16.4.10 and Cor. 16.4.12 imply that for any $\vec{x}_0 \in \mathbb{Z}^d$, the expected looping time $\mathsf{explt}\,[\![\mathbf{L}]\!]\left(\vec{x}_0\right)$ of a *cpa*-loop $\mathbf{L}$ that is PAST is always an *algebraic* number since we restricted ourselves to algebraic probabilities $p_{\vec{c}_1}, \ldots, p_{\vec{c}_n}, p' \in \mathbb{R}_{\mathrm{A}}$. Thus, one could also compute a closed form for the exact expected looping time $\mathsf{explt}\,[\![\mathbf{L}]\!]$ using a representation with algebraic numbers instead of numerical approximations. The following example shows that this is not true for general *pa*-loops.

Counterexample 16.4.14. Consider the loop $\mathbf{L}_{nalg}$ in Fig. 16.8(b). In [Küs19], it

```
while (x > 0) do {
    x ← x + 1  [9/10]
    x ← 0      [1/10]
}
```

(a) $\mathbf{L}^{rdw}_{direct}$

```
while (x > 0) do {
    x ← x + 1  [1/(x+1)]
    x ← 0      [x/(x+1)]
}
```

(b) $\mathbf{L}_{nalg}$

Figure 16.8: Loops from Ex. 16.4.13 and Counterex. 16.4.14

is shown that the expected looping time of $\mathbf{L}_{nalg}$ for $x > 0$ is given by

$$\text{explt } [\![\mathbf{L}_{nalg}]\!](x) = \sum_{s \in \mathbb{N}} \frac{x!}{(x+s)!}$$

where $x!$ denotes the factorial $x! = \prod_{j=1}^{x} j$. By standard results from mathematics (see, e.g., [Abr74]), one can conclude that $\text{explt } [\![\mathbf{L}_{nalg}]\!](1) = e - 1 \notin \mathbb{R}_{\mathbb{A}}$.

While the exact computation of the expected looping time of a random walk loop $\mathbf{L}$ according to Thm. 16.4.10 may yield a representation of $\text{explt } [\![\mathbf{L}]\!]$ with possibly non-real numbers, one can easily obtain a more intuitive representation of $\text{explt } [\![\mathbf{L}]\!]$ that uses real numbers only. As the characteristic polynomial $\chi_{\mathbf{L}}$ only has real coefficients, whenever $\chi_{\mathbf{L}}$ has a complex root λ of multiplicity ν, its conjugate $\overline{\lambda}$ is also a root of $\chi_{\mathbf{L}}$ with the same multiplicity ν. So the pairwise different roots $\lambda_1, \ldots, \lambda_c$ can be distinguished into pairwise different real roots $\lambda_1, \ldots, \lambda_s$, and into pairwise different non-real complex roots $\lambda_{s+1}, \overline{\lambda_{s+1}}, \ldots, \lambda_{s+t}, \overline{\lambda_{s+t}}$, where $c = s + 2 \cdot t$.

For any coefficients $a_{j,u}, a'_{j,u} \in \mathbb{C}$ with $j \in \{s+1, \ldots, s+t\}$ and $u \in \{0, \ldots, \nu_j - 1\}$ there exist coefficients $b_{j,u}, b'_{j,u} \in \mathbb{C}$ such that

$$a_{j,u} \cdot \lambda_j^x + a'_{j,u} \cdot \overline{\lambda_j}^x = b_{j,u} \cdot \text{Re}(\lambda_j^x) + b'_{j,u} \cdot \text{Im}(\lambda_j^x)$$

holds for all $x \in \mathbb{Z}$. More precisely, $b_{j,u} = a_{j,u} + a'_{j,u}$ and $b'_{j,u} = (a_{j,u} - a'_{j,u}) \cdot i$. So any linear combination of the functions $\lambda_j^x \cdot x^u$ and $\overline{\lambda_j}^x \cdot x^u$ can be replaced by a linear combination of the functions $\text{Re}(\lambda_j^x) \cdot x^u$ and $\text{Im}(\lambda_j^x) \cdot x^u$. In this way, one obtains $k + m$ linearly independent *real* solutions of the corresponding homogeneous recurrence equation. Hence, we now get the following representation of the expected looping

time by Thm. 16.4.10:

$$\text{explt}\,[\![\mathbf{L}]\!](x) = \begin{cases} C_{\mathbf{L}}(x) + \sum_{j=1,\,|\lambda_j|\leq 1}^{s} \sum_{u=0}^{\nu_j-1} a_{j,u}\cdot\lambda_j^x\cdot x^u \\ \qquad + \sum_{j=s+1,\,|\lambda_j|\leq 1}^{s+t} \sum_{u=0}^{\nu_j-1} \left(b_{j,u}\cdot \mathrm{Re}(\lambda_j^x) + b'_{j,u}\cdot \mathrm{Im}(\lambda_j^x)\right)\cdot x^u, & \text{for } x > 0 \\ 0, & \text{for } x \leq 0 \end{cases} \tag{16.4.9}$$

As $\text{explt}\,[\![\mathbf{L}]\!]$ is real-valued, $\lambda_j^x \in \mathbb{R}$ for $j \in \{1,\dots,s\}$, and $\mathrm{Re}(\lambda_j^x), \mathrm{Im}(\lambda_j^x) \in \mathbb{R}$ for $j \in \{s+1,\dots,s+t\}$, all $a_{j,u}$ for $j \in \{1,\dots,s\}$ and all $b_{j,u}, b'_{j,u}$ for $j \in \{s+1,\dots,s+t\}$ are real numbers. So when considering the coefficients $a_{j,u}, a'_{j,u}$ of λ_j^x and $\overline{\lambda}_j^x$ computed by Thm. 16.4.10, we have $b_{j,u} = a_{j,u} + a'_{j,u}$ and $b'_{j,u} = (a_{j,u} - a'_{j,u})\cdot i$ for $j \in \{s+1,\dots,s+t\}$. This means that $a'_{j,u}$ is the conjugate of $a_{j,u}$, i.e., $a'_{j,u} = \overline{a_{j,u}}$ and thus $b_{j,u} = 2\cdot\mathrm{Re}(a_{j,u})$ and $b'_{j,u} = -2\cdot\mathrm{Im}(a_{j,u})$.

To compute $\mathrm{Re}(\lambda_j^x)$ and $\mathrm{Im}(\lambda_j^x)$, we consider the polar representation of the non-real roots λ_j, i.e., for $j \in \{s+1,\dots,s+t\}$ let $\lambda_j = r_j\cdot e^{\theta_j\cdot i}$ with $r_j \in \mathbb{R}_{>0}$ and $\theta_j \in (0, 2\pi)$. Then $\lambda_j^x = r_j^x\cdot e^{\theta_j\cdot i\cdot x}$, $\mathrm{Re}(\lambda_j^x) = r_j^x\cdot\cos(\theta_j\cdot x)$, and $\mathrm{Im}(\lambda_j^x) = r_j^x\cdot\sin(\theta_j\cdot x)$.

Note that in Algorithm 16.4.1 one could also already use the representation in (16.4.9) with $\mathrm{Re}(\lambda_j^x) = r_j^x\cdot\cos(\theta_j\cdot x)$ and $\mathrm{Im}(\lambda_j^x) = r_j^x\cdot\sin(\theta_j\cdot x)$. Then one would only have to solve a system of linear equations over the reals and can compute $b_{j,u}$ and $b'_{j,u}$ directly. Nevertheless, we have to take into account that the system in (16.4.8) is a system of linear equations over the real algebraic numbers and could therefore be solved *exactly*, leading to a representation of the exact expected looping time involving possibly non-real numbers. Still, computing the polar representation then involves non-algebraic numbers and therefore, the polar representation itself cannot be computed exactly. Nonetheless, when directly computing $b_{j,u}$ and $b'_{j,u}$ we have to solve a system over possibly non-algebraic numbers whose solution cannot be computed exactly. Still, in both cases we can compute a representation of the exact expected looping time involving only real numbers with arbitrary precision. To sum up, we obtain Algorithm 16.4.1 to compute the exact expected looping time of a non-trivial *cpa*-loop which is PAST (recall that PAST can be decided by Cor. 16.3.9).

At the end of this section, let us briefly compare the results in this chapter with our findings from Part II. In this chapter, we have shown that for *cpa*-loops there is a simple decision procedure for (P)AST, both upper and lower bounds on the expected looping time can always be inferred, and we can even compute the *exact* expected looping time. In Part II, we have studied *twn*-loops over some ring $\mathcal{S} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}_{\mathrm{A}}\}$ and shown that for these loops both (non-)termination on a given input as well as on

Algorithm 16.4.1: Computing Exact Expected Looping Times

Input: non-trivial *cpa*-loop $\mathbf{L}$ which is PAST
Result: exact expected looping time of $\mathbf{L}$

Transform $\mathbf{L}$ into $\mathbf{L}^{rdw}$ by the transformation of Def. 16.3.2
Compute the solution $C_{\mathbf{L}^{rdw}}(x)$ defined in Def. 16.4.2
Compute the k roots λ of $\chi_{\mathbf{L}^{rdw}}$ (see Def. 16.4.4) with $|\lambda| \leq 1$
Determine the coefficients $a_{j,u}$ by solving the k linear equations in (16.4.8)
return the solution in (16.4.9) **where**
    $b_{j,u} \longleftarrow 2 \cdot \mathrm{Re}(a_{j,u})$ and $b'_{j,u} \longleftarrow -2 \cdot \mathrm{Im}(a_{j,u})$ for $j \in \{s+1, \dots, s+t\}$
    $\mathrm{Re}(\lambda_j^x) \longleftarrow r_j^x \cdot \cos(\theta_j \cdot x)$ and
    $\mathrm{Im}(\lambda_j^x) \longleftarrow r_j^x \cdot \sin(\theta_j \cdot x)$ with $\lambda_j = r_j \cdot e^{\theta_j \cdot i}$ for $j \in \{s+1, \dots, s+t\}$
    $x \longleftarrow rdw_{\mathbf{L}}(\vec{x})$

all inputs are (semi-)decidable and upper bounds on the looping time can always be computed.

Thus, we have shown that both *twn*-loops and *cpa*-loops form (restricted) classes of loops where *complete* methods for analyzing termination exist. Still, *cpa*-loops are *not* a generalization of *twn*-loops. First of all, the variables in *cpa*-loops range over the integers (see Sect. 14.1). Moreover, *cpa*-loops allow for direct termination (which has no counterpart in *twn*-loops) and they only allow for updating the variables by *adding* constants, while in *twn*-loops we allow for a mildly restricted form of *polynomial* updates. Nevertheless, we should note that there is a probabilistic equivalent of *twn*-loops, so-called Prob-solvable loops (see Sect. 18.2). However, to the best of our knowledge there are no other results on classes of probabilistic loops where (P)AST is decidable than the results presented in this chapter.

As shown in [KKM19], termination analysis of probabilistic programs is in general *harder* than termination analysis of non-probabilistic programs. An intuitive reason for this is as follows. For *twn*-loops, we computed a closed form for the iterated update and combined it with the condition of the loop. We then analyzed this expression to obtain our decidability results. In contrast, for a probabilistic loop a closed form for the iterated update is a *distribution* over expressions. To adapt our approach from *twn*-loops to probabilistic loops, one would have to be able to (1) compute this distribution, (2) combine it with the condition of the loop, and (3) reason about this resulting distribution and its expected value. To the best of our knowledge, there exist no (non-trivial) classes of probabilistic loops where a closed form for the distribution is computable, i.e., a solution of problem (1) exists. Note that our decision procedure

for (P)AST of *cpa*-loops (Cor. 16.3.6) was based on results from martingale theory and did *not* rely on a closed form for the iterated update.

16.4.3 Implementation

We implemented Algorithm 16.4.1 in our open-source tool KoAT [Bro+16; KoAT], which was already one of the leading tools for complexity analysis of (non-probabilistic) integer transition systems. The implementation is written in OCaml and uses the Python libraries MpMath [Joh+13] and SymPy [Meu+17] for solving linear equations and for finding the roots of the characteristic polynomial. In addition to the closed form for the exact expected looping time, our implementation can also compute the concrete number of expected loop iterations if the user specifies the initial values of the variables. The implementation of Algorithm 16.4.1 can also be downloaded as a Docker image and accessed via a web interface [Gie+19a]. For further details and a set of benchmarks, we also refer to [Gie+19a].

Example 16.4.15. For the race of tortoise and hare $\mathbf{L}_{race}$ (see Fig. 16.1(b)), our implementation in KoAT computes the following expected looping time within 0.49 s on an Intel Core i7-6500 with 8 GB memory (when selecting a precision of 2 decimal places):

$$\begin{aligned}\operatorname{explt}\,[\![\mathbf{L}_{race}]\!]\,(h,t) \;=\;& 0.049\cdot 0.65^{(t-h+1)}\cdot\sin(2.8\cdot(t-h+1)) - 0.35\cdot 0.65^{(t-h+1)}\cdot\cos(2.8\cdot(t-h+1))\\ &+0.15\cdot 0.66^{(t-h+1)}\cdot\sin(2.2\cdot(t-h+1)) - 0.35\cdot 0.66^{(t-h+1)}\cdot\cos(2.2\cdot(t-h+1))\\ &+0.3\cdot 0.7^{(t-h+1)}\cdot\sin(1.5\cdot(t-h+1)) - 0.39\cdot 0.7^{(t-h+1)}\cdot\cos(1.5\,(t-h+1))\\ &+0.62\cdot 0.75^{(t-h+1)}\cdot\sin(0.83\cdot(t-h+1)) - 0.49\cdot 0.75^{(t-h+1)}\cdot\cos(0.83\cdot(t-h+1))\\ &+\tfrac{2}{3}\cdot(t-h)\;+\;2.3\end{aligned}$$

So when starting in a state with $h = 0$ and $t = 1000$, according to our implementation the number of expected loop iterations is $\operatorname{explt}\,[\![\mathbf{L}_{race}]\!]\,(0,1000) = 670$.

It is worth noting that we also implemented the approach for upper bounds on the expected runtime and the expected size of so-called probabilistic transition systems (PITSs) in KoAT (see Ch. 17). Thus, KoAT contains both techniques and a user can determine via the input if she would like to compute the exact expected looping time of a *cpa*-loop or if she would like to analyze a PITS. Unfortunately, these techniques cannot be combined. The reason is that in Ch. 17 we require so-called weak monotonicity and concavity and the exact expected looping times computed in this section are neither weakly monotonic nor concave. Additionally, the bounds from Sect. 16.3.3 are also not weakly monotonic. Nevertheless, when representing

a *cpa*-loop in the syntax of PITSs, the implementation of the approach in Ch. 17 computes a "monotonic version" of the upper bound from Sect. 16.3.3 (see Ex. 17.1.15 and 17.3.7).

Chapter 17

Upper Bounds via Probabilistic Transition Systems

In the previous chapters of this part, we have studied expected outcomes and expected runtimes of probabilistic programs. Here, we used the syntax of probabilistic programs defined via pGCL (see Sect. 14.1), and their semantics defined by expectation transformers (see Sect. 14.3) and stochastic processes (see Sect. 14.4). In this chapter, we extend probabilistic programs by infinite non-determinism and use a representation as so-called *probabilistic transition systems* (PITSs). We use this concept to present a modular approach for inferring *upper* bounds on the expected runtime. In contrast to Ch. 15 and 16, we do not proceed via backward-analysis but present a *forward-analysis* for whose proof of soundness we again rely on *martingale theory*.

We define the syntax and semantics of PITSs in Sect. 17.1. There, we also compare PITSs with pGCL-programs which we have considered in Ch. 14 to 16. In Sect. 17.2, we recapitulate the notions of non-probabilistic runtime and size bounds from [Bro+16], define random variables reflecting runtime and size for PITSs, and introduce non-probabilistic and expected runtime and size bounds. Based on these concepts, we present our modular approach to infer expected runtime bounds in Sect. 17.3 and to infer expected size bounds in Sect. 17.4. Finally, we present the implementation of our approach in the tool KoAT and compare it with existing tools in Sect. 17.5.

17.1 Probabilistic Transition Systems

In this section, we introduce the syntax of probabilistic transition systems (PITSs)[1] and define their semantics (Sect. 17.1.2). Finally, we briefly comment on the relation between pGCL-programs and PITSs (Sect. 17.1.3).

17.1.1 Syntax

As introduced in Sect. 14.1, we denote by $\mathcal{V}$ a finite set of variables and Σ is the set of all states $s : \mathcal{V} \to \mathbb{Z}$. In the context of expectation transformers, we allowed arbitrary expectations, i.e., mappings from $\Sigma \to \overline{\mathbb{R}}_{\geq 0}$ (see Def. 14.3.1). However, in the context of PITSs we have to restrict ourselves to a certain subset of expectations induced by so-called *bounds*. This is needed to allow for a modular *forward* analysis, i.e., for our approach to be sound we rely on certain properties of these bounds. To avoid confusion, we do not use the term "expectation" in the rest of this chapter.

First of all, we fix a finite set $\mathcal{PV} \subseteq \mathcal{V}$ of *program variables* . Similar to [Bro+16], our bounds induce *weakly monotonically increasing* functions from $\Sigma \to \overline{\mathbb{R}}_{\geq 0}$ which only depend on the program variables, i.e., given a bound $b \in \mathcal{B}$ and two states $s_1, s_2 : \mathcal{V} \to \mathbb{Z}$ such that $s_1(x) = s_2(x)$ for all $x \in \mathcal{PV}$ we have $s_1(b) = s_2(b)$, where $s(b)$ is obtained by instantiating each variable $y \in \mathcal{V}$ in b by $s(y)$. A bound b then induces the function $\Sigma \to \overline{\mathbb{R}}_{\geq 0}, s \mapsto |s|(b)$, where $|s|$ is the state defined by $|s|(x) = |s(x)|$ for all $x \in \mathcal{V}$. Weak monotonic increase means that given a bound b and two states $s_1, s_2 : \mathcal{V} \to \mathbb{Z}$ such that $|s_1| \leq |s_2|$ we have $|s_1|(b) \leq |s_2|(b)$. Such bounds can easily be "composed", i.e., if b_1 and b_2 are both weakly monotonically increasing bounds, so is "$b_1 \circ b_2$".

Definition 17.1.1 (Bounds)**.** The set of *bounds* $\mathcal{B}$ is the smallest set with

(i) $\mathcal{PV} \cup \overline{\mathbb{R}}_{\geq 0} \subseteq \mathcal{B}$

(ii) $b_1 + b_2, b_1 \cdot b_2 \in \mathcal{B}$ and $v^{b_1} \in \mathcal{B}$ for all $b_1, b_2 \in \mathcal{B}$ and $v \in \mathbb{R}_{\geq 1}$.

A bound is *finite* if it does *not* contain ∞. Here, we allow for the syntactic simplification $0 \cdot \infty = \infty \cdot 0 = 0$ which is standard in measure theory.

By Def. 17.1.1, bounds indeed induce weakly monotonically increasing mappings.

[1] The non-probabilistic counterpart of this concept is often referred to as integer transition systems to highlight the domain of the variables. However, in this part we only consider probabilistic programs whose variables take integer values.

Example 17.1.2. If $x \in \mathcal{PV}$, then $1 \in \mathcal{B}$, $x \in \mathcal{B}$, $3 \cdot x + 2^x \in \mathcal{B}$, and $x + \infty \in \mathcal{B}$. Here, all bounds except $x + \infty$ are finite.

Our notion of probabilistic transition systems combines classical (integer) transition systems (as in, e.g., [Bro+16]) with probabilistic concepts such as probabilistic assignments and probabilistic branching. In such a system, we restrict ourselves to conditions of a certain form, so-called *(polynomial) constraints*.

Definition 17.1.3 (Constraints)**.** The set of constraints $\mathcal{C}$ is the smallest set with

(i) $e_1 \leq e_2$ for all polynomials $e_1, e_2 \in \mathbb{Z}[\mathcal{V}]$

(ii) $c_1 \wedge c_2$ for all $c_1, c_2 \in \mathcal{C}$.

In addition to "$\leq$", in examples we also use relations like "$>$", which can be simulated by constraints (e.g., $e_1 > e_2$ is equivalent to $e_2 + 1 \leq e_1$ when regarding *integers*). We also allow the application of states s to constraints c. Then $s(c) \in \{\text{true}, \text{false}\}$ results from evaluating the constraint when instantiating every variable x by $s(x)$.

Example 17.1.4. If $x \in \mathcal{V}$, then $x \leq 0 \in \mathcal{C}$, $2 \cdot x + 1 \leq 0 \in \mathcal{C}$, and $x \leq 0 \wedge 2 \cdot x + 1 \leq 0 \in \mathcal{C}$. If s is a state with $s(x) = 0$, we have $s(x \leq 0) = \text{true}$ but $s(2 \cdot x + 1 \leq 0) = \text{false}$.

In the transitions of a transition system, a program variable $x \in \mathcal{PV}$ can also be updated by adding a value according to a *bounded distribution function* D. Here, for any state s, $D(s)$ is the probability distribution of the values that are added to x.

Definition 17.1.5 (Bounded Distribution Function)**.** Let $\text{Dist}(\mathbb{Z})$ be the set of distributions pr whose expected value $\mathbb{E}(pr) = \sum_{v \in \mathbb{Z}} v \cdot pr(v)$ is well defined and finite, i.e., $\mathbb{E}_{\text{abs}}(pr) = \sum_{v \in \mathbb{Z}} |v| \cdot pr(v) < \infty$. A distribution function $D : \Sigma \to \text{Dist}(\mathbb{Z})$ is *bounded* if there is a finite bound $\mathfrak{E}(D) \in \mathcal{B}$ with $\mathbb{E}_{\text{abs}}(D(s)) \leq |s|\,(\mathfrak{E}(D))$ for all $s \in \Sigma$. The set of all bounded distribution functions is denoted by $\mathcal{D}$.

The implementation of the approach presented in this chapter supports Bernoulli, uniform, geometric, hypergeometric, and binomial distributions (see [MHG21a]). Note that in Def. 17.1.5 *bounded* refers to our set of bounds $\mathcal{B}$ as defined in Def. 17.1.1.

Definition 17.1.6 (PITS)**.** A tuple $\mathbf{T} = (\mathcal{PV}, \mathcal{L}, \mathcal{GT}, \ell_0)$ is a *probabilistic transition system (PITS)* where

(i) $\mathcal{PV} \subseteq \mathcal{V}$ is a finite set of *program variables*

(ii) $\mathcal{L}$ is a finite non-empty set of *locations*

(iii) $\mathcal{GT}$ is a finite non-empty set of *general transitions*. A general transition g is a finite non-empty set of *transitions* $t = (\ell, p, \varphi, \eta, \ell')$, consisting of

a) the *start* and *target locations* $\ell, \ell' \in \mathcal{L}$ of transition t,

b) the *probability* $p \in \mathbb{R}_{\geq 0}$ that transition t is chosen when g is executed,

c) the *condition* $\varphi \in \mathcal{C}$ of t, and

d) the *update function* $\eta : \mathcal{PV} \to \mathbb{Z}[\mathcal{V}] \cup \mathcal{D}$ of t, mapping every program variable to an update polynomial or a bounded distribution function.

All $t \in g$ must have the same start location ℓ and the same condition φ. Thus, we call them the start location and condition *of* g, and denote them by ℓ_g and φ_g. Moreover, the probabilities p of the transitions in g must add up to 1.

(iv) $\ell_0 \in \mathcal{L}$ is a distinguished *initial location*, where no transition has target location ℓ_0. The set of *initial* general transitions $\mathcal{GT}_0 \subseteq \mathcal{GT}$ consists of all general transitions with start location ℓ_0.

The set of *temporary variables* in $\mathbf{T}$ is $\mathcal{TV} = \mathcal{V} \setminus \mathcal{PV}$.

PITs allow for both probabilistic and non-deterministic branching and sampling. Probabilistic branching is modeled by selecting a transition out of a non-singleton general transition. Note that the probability p of a transition t is a real number, i.e., it must not depend on the values of the (program) variables. Non-deterministic branching is represented by several general transitions with the same start location and non-exclusive conditions. Note that both kinds of branching are *finite* since there are only finitely many general transitions and each general transition contains only finitely many transitions. Probabilistic sampling is realized by update functions that map a program variable to a bounded distribution function. Non-deterministic sampling is modeled by updating a program variable with an expression containing temporary variables $u \in \mathcal{TV}$, whose values are non-deterministic (but can be restricted in the condition). Thus, PITSs allow for *infinite* probabilistic sampling and *infinite*

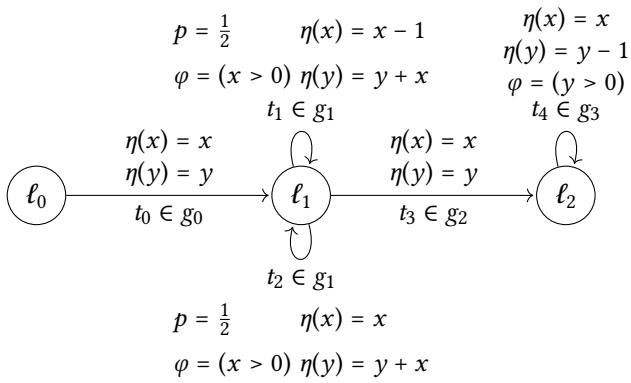


Figure 17.1: PITS $\mathbf{T}_{sq}$ With Non-Deterministic and Probabilistic Branching

non-deterministic sampling. Again, we distinguish between (non-probabilistic) non-determinism and probabilistic constructs as for probabilistic constructs we know with which probability the different possibilities occur, while for non-deterministic constructs we only know that one of the different possibilities occurs (see also Ex. 14.1.3).

Example 17.1.7. Consider the PITS $\mathbf{T}_{sq}$ in Fig. 17.1 with initial location ℓ_0 and the program variables $\mathcal{PV} = \{x, y\}$. Here, let $p = 1$ and $\varphi = \text{true}$ if not stated explicitly. There are four general transitions: $g_0 = \{t_0\}$, $g_1 = \{t_1, t_2\}$, $g_2 = \{t_3\}$, and $g_3 = \{t_4\}$, where g_1 and g_2 represent a non-deterministic branching. When choosing the general transition g_1, the transitions t_1 and t_2 encode a probabilistic branching. If we modified the update η and the condition φ of t_0 to $\eta(x) = u \in \mathcal{TV}$ and $\varphi = (u \geq 0)$, then x would be updated to a non-deterministically chosen non-negative value. In contrast, if $\eta(x) = \text{Geo}(\frac{1}{2})$, then t_0 would update x by adding a value sampled from the geometric distribution with parameter $\frac{1}{2}$, i.e., t_0 would update x to $x + n$ with probability $\left(\frac{1}{2}\right)^n$ where $n \in \mathbb{N}_{\geq 1}$.

17.1.2 Semantics

In this section, we fix a PITS $\mathbf{T}$ as in Def. 17.1.6. We first define the notion of a *configuration* which captures all ingredients that we need to "execute" $\mathbf{T}$.

Definition 17.1.8 (Configurations and Runs). A *configuration* is a tuple (ℓ, t, s), with the current location $\ell \in \mathcal{L}$, the current state $s \in \Sigma$, and the transition t that was evaluated last and led to the current configuration. Let $\mathcal{T} = \bigcup_{g \in \mathcal{GT}} g$ be the set of all transitions in $\mathbf{T}$. Then $\mathsf{Conf} = (\mathcal{L} \uplus \{\ell_\perp\}) \times (\mathcal{T} \uplus \{t_{\text{in}}, t_\perp\}) \times \Sigma$ is the set of all configurations, with a special location $\ell_\perp$ indicating the termination of a run, and special transitions t_{in} (used in the first configuration of a run) and $t_\perp$ (for the configurations of the run after termination). The (virtual) general transition $g_\perp = \{t_\perp\}$ only contains $t_\perp$.

The set of *runs* is Conf^ω and the set of all *finite paths* is Conf^+.

Thus, in contrast to probabilistic loops, a run is not merely an infinite sequence of states (see Def. 14.4.4) but an infinite sequence of configurations. The reason is that in a PITS we do not only need to know the values of the variables but also the location to know how an (admissible) execution of the PITS continues. Moreover, we also need to know which transition led to the current location and the current state. Furthermore, we add a "sink" location $\ell_\perp$, and the virtual transitions t_{in} (to "enter" the initial location ℓ_0) and $t_\perp$ (to "enter" the sink location).

Example 17.1.9. Consider again the PITS $\mathbf{T}_{sq}$ in Fig. 17.1. Let $s_0(x) = 1$ and $s_0(y) = 0$. Then $(\ell_0, t_{\text{in}}, s_0) \in \mathsf{Conf}$. A finite path is given by $(\ell_0, t_{\text{in}}, s_0)(\ell_1, t_0, s_0)$. For $s_1(x) = 0, s_1(y) = 1$ and $s_2(x) = 0, s_2(y) = 0$,

$$(\ell_0, t_{\text{in}}, s_0)(\ell_1, t_0, s_0)(\ell_1, t_1, s_1)(\ell_2, t_2, s_1)(\ell_2, t_3, s_2)(\ell_2, t_\perp, s_2)(\ell_2, t_\perp, s_2)\cdots$$

is a run of $\mathbf{T}_{sq}$ as well as

$$(\ell_0, t_{\text{in}}, s_0)(\ell_0, t_{\text{in}}, s_0)(\ell_0, t_{\text{in}}, s_0)\cdots.$$

Of course, the latter run is only a virtual run and does not correspond to a proper execution of $\mathbf{T}_{sq}$. Thus, the probability measure which we define later on will sort out such runs (i.e., such runs will be non-admissible).

To resolve non-determinism, we also make use of schedulers (see Def. 14.2.7). As for pGCL-programs, deterministic positional schedulers suffice for PITSs.

Definition 17.1.10 (Schedulers for PITSs). A function $\mathfrak{s} : \mathsf{Conf} \longrightarrow (\mathcal{GT} \uplus \{g_\perp\}) \times \Sigma$ is a *scheduler* if for every configuration $c = (\ell, t, s) \in \mathsf{Conf}$, $\mathfrak{s}(c) = (g, s')$ implies:

(i) $s(x) = s'(x)$ for all $x \in \mathcal{PV}$.

(ii) ℓ is the start location ℓ_g of the general transition g.

(iii) $s'(\varphi_g) = \text{true}$ for the condition φ_g of the general transition g.

(iv) $g = g_\perp$ and $s' = s$, if $\ell = \ell_\perp$, $t = t_\perp$, or no $g' \in \mathcal{GT}$, $s' \in \Sigma$ satisfy (i) to (iii).

The set of all schedulers is denoted by Scheds.

So to continue an evaluation in state s, (i) the scheduler chooses a state s' that agrees with s on all program variables, but where the values for temporary variables are chosen non-deterministically. After instantiating the temporary variables, (ii) the scheduler selects a general transition that starts in the current location ℓ and (iii) whose condition is satisfied, if such a general transition exists. Otherwise, (iv) $\mathfrak{s}$ chooses $g_\perp = \{t_\perp\}$ and leaves the state s unchanged.

Similar to the construction for loops in Sect. 14.4.1, we assign a probability space to a PITS via the cylinder set construction based on [Var85].

Definition 17.1.11 (Measurable Space for PITS)**.** The canonical measurable space is given by $(\Omega^{\text{pt}}, \mathcal{F}^{\text{pt}})$ where $\Omega^{\text{pt}} = \mathsf{Conf}^\omega$ and

$$\mathcal{F}^{\text{pt}} = \left\langle \left\{ \pi \in \mathsf{Conf}^+ \;\middle|\; \mathrm{Cyl}(\pi) \right\} \right\rangle_\sigma .$$

To every scheduler $\mathfrak{s}$ and initial state s_0, one can assign a unique probability measure, which captures the probabilistic evaluation behavior of PITSs. To this end, we first define a family of one-step distributions

$$\mu^{\text{pt}}_{\bullet,\bullet} : \mathsf{Conf} \longrightarrow \mathsf{Scheds} \longrightarrow \mathsf{Conf} \longrightarrow [0,1]$$

on configurations. Fix a scheduler $\mathfrak{s}$. Let $(\ell', t', s') \in \mathsf{Conf}$ and $\mathfrak{s}(\ell', t', s') = (g, \tilde{s})$.

In the following, we always have to consider the state $\tilde{s}$ instead of the state s' to determine the probability of the next configurations since the temporary variables are updated by the scheduler. If $g \in \mathcal{GT}$, i.e., $\tilde{s}(\varphi_g) = \text{true}$, and ℓ' is the start location of g, for every $t = (\ell', p, \varphi_g, \eta, \ell) \in g$ we define:

$$\mu^{\text{pt}}_{(\ell',t',s'),\mathfrak{s}}(\ell, t, s) = p \cdot \prod_{x \in \mathcal{PV}} \mathrm{q}_{\tilde{s},t,s(x)}(x) \cdot \prod_{u \in \mathcal{TV}} [\tilde{s}(u) = s(u)] ,$$

where for any transition t, $\tilde{s} \in \Sigma$, $x \in \mathcal{PV}$, and $v \in \mathbb{Z}$, $\mathrm{q}_{\tilde{s},t,v}(x)$ expresses the probability that during the evaluation of t in state $\tilde{s}$ the variable x gets updated to the value v:

$$\mathrm{q}_{\tilde{s},(_,_,_,\eta,_),v}(x) = \begin{cases} 1, & \text{if } \eta(x) \in \mathbb{Z}[\mathcal{V}] \text{ and } \tilde{s}(\eta(x)) = v \\ p', & \text{if } \eta(x) \in \mathcal{D} \text{ and } \eta(x)(\tilde{s})(\tilde{s}(x) - v) = p' \\ 0, & \text{otherwise} \end{cases}$$

Note that if $\eta(x)$ is a distribution function D, the value sampled according to $D(\tilde{s})$ must be *added* to the current value $\tilde{s}(x)$ of x. Moreover, the updates in a transition $t = (_, _, _, \eta, _)$ take place *in parallel*, e.g., if $x, y \in \mathcal{PV}$, then $\eta(x)$ does not affect $\eta(y)$ as both only access the values $\tilde{s}$ of the variables *before* the execution of t. Additionally, we require that the values of the temporary variables which the scheduler chooses in $\tilde{s}$ are "saved" by s. For all other configurations c, we define $\mu^{\mathrm{pt}}_{(\ell',t',s'),\mathfrak{s}}(c) = 0$. If the system has already *terminated* (i.e., $g = g_\perp$ and thus $\tilde{s} = s'$), we define:

$$\mu^{\mathrm{pt}}_{(\ell',t',s'),\mathfrak{s}}(c) = \begin{cases} 1, & \text{if } c = (\ell_\perp, t_\perp, s') \\ 0, & \text{otherwise} \end{cases}$$

Example 17.1.12. Consider the PITS $\mathbf{T}_{\mathrm{sq}}$ from Fig. 17.1, but where the update $\eta(x) = x$ of t_0 is changed to $\eta(x) = u \in \mathcal{TV}$. Let $\mathfrak{s}$ be the scheduler that selects $u = 5$ and that always chooses g_1 over g_2 (when possible), and let $s_1 \in \Sigma$ with $s_1(x) = 5$, $s_1(u) = 5$, and $s_1(y) = s_0(y)$. Then we have:

$$\mu^{\mathrm{pt}}_{(\ell_0,t_{\mathrm{in}},s_0),\mathfrak{s}}(\ell,t,s) = \begin{cases} 1, & \text{if } \ell = \ell_1, t = t_0, s = s_1 \\ 0, & \text{otherwise} \end{cases}$$

$$\mu^{\mathrm{pt}}_{(\ell_1,t_0,s_1),\mathfrak{s}}(\ell,t,s) = \begin{cases} \frac{1}{2}, & \text{if } \ell = \ell_1, t = t_1, s(x) = 4, s(y) = s_1(y) + 5 \\ \frac{1}{2}, & \text{if } \ell = \ell_1, t = t_2, s(x) = 5, s(y) = s_1(y) + 5 \\ 0, & \text{otherwise} \end{cases}$$

If c' has the form (ℓ_2, t, s) with $s(y) \leq 0$ or $(\ell_\perp, _, s)$ for some state s, the scheduler cannot choose a suitable outgoing transition. Then $\mu^{\mathrm{pt}}_{c',\mathfrak{s}}(c) = 1$ if $c = (\ell_\perp, t_\perp, s)$ and $\mu^{\mathrm{pt}}_{c',\mathfrak{s}}(c) = 0$, otherwise. So the virtual transition $t_\perp$ is used for the infinitely many configurations of a run after termination.

We then define the family of distributions of Conf^+

$$\mathrm{p}^{\mathrm{pt}}_{\bullet,\bullet} : \Sigma \longrightarrow \mathsf{Scheds} \longrightarrow \mathsf{Conf}^+ \longrightarrow [0,1]$$

such that for each path $\pi = c_0 \cdots c_n \in \mathsf{Conf}^+$

$$p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\pi) = [c_0 = (\ell_0, t_{\mathrm{in}}, s_0)] \cdot \prod_{i=1}^{n} \mu^{\mathrm{pt}}_{c_{i-1},\mathfrak{s}}(c_i).$$

Here, we write $p^{\mathrm{pt}}_{s_0,\mathfrak{s}}$ since the initial state s_0 uniquely determines $(\ell_0, t_{\mathrm{in}}, s_0)$. A finite path π is *admissible* for s_0 and $\mathfrak{s}$ if $p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\pi) > 0$ and a run is admissible if all its finite prefixes are admissible. A configuration $c \in \mathsf{Conf}$ is *admissible* for s_0 and $\mathfrak{s}$ if there is an admissible path ending in c.

Similar to Lemma 14.4.9, we obtain a probability measure on $\mathcal{F}^{\mathrm{pt}}$ via "Kolmogorov's Extension Theorem".

Lemma 17.1.13 (PITS Measure, [Fel71]). *There exists a unique family of probability measures* $\mathbb{P}^{pt}_{\bullet,\bullet} : \Sigma \times \mathsf{Scheds} \to \mathcal{F}^{pt} \to [0,1]$ *such that for every* $\pi \in \mathsf{Conf}^+$

$$\mathbb{P}^{pt}_{s_0,\mathfrak{s}}(\mathrm{Cyl}(\pi)) = p^{pt}_{s_0,\mathfrak{s}}(\pi).$$

We denote the associated expected value operator by $\mathbb{E}^{pt}_{s_0,\mathfrak{s}}$. *So for any random variable* $X : \mathsf{Conf}^\omega \to \overline{\mathbb{N}}$, *we have* $\mathbb{E}^{pt}_{s_0,\mathfrak{s}}(X) = \sum_{n\in\mathbb{N}} n \cdot \mathbb{P}^{pt}_{s_0,\mathfrak{s}}(X = n) + \mathbb{P}^{pt}_{s_0,\mathfrak{s}}(X \notin \mathbb{N}) \cdot \infty$.

17.1.3 Comparison of PITSs and Probabilistic Programs

In this section, we compare PITSs and pGCL-programs. First of all, PITSs only support polynomial arithmetic expressions and *bounded* distribution functions, i.e., the expected value has to be bounded by an element from $\mathcal{B}$. These restrictions are crucial for the soundness of the approach we present in this chapter (see Sect. 17.3 and 17.4) and its automation. Thus, whenever a pGCL-program contains expressions using non-polynomial arithmetic or probabilistic assignments with an unbounded distribution function it cannot be encoded as a PITS (it might be possible to use workarounds, e.g., $x \leftarrow 2^x$ can be modeled by assigning a variable y the value 1, then doubling it x-times and finally using $x \leftarrow y$). However, this only concerns probabilistic assignments. A probabilistic choice (even with a *non*-constant probability q) can always be modeled by introducing a variable mimicking a Boolean flag which is then updated w.r.t. the distribution which is 1 with probability q and 0 with probability $1 - q$. This distribution function is trivially bounded since its expected value is at most 1. So for simplicity, we restrict ourselves to pGCL-programs involving only polynomial arithmetic and bounded distribution functions. It is intuitively clear that every such pGCL-program $\mathbf{P}$ can be modeled by a probabilistic transition system $\mathbf{T}_{\mathbf{P}}$, which "computes" the same value:

Remark 17.1.14. Let $\mathbf{P}$ be a pGCL-program over the variables $\mathcal{PV} \subseteq \mathcal{V}$ using only polynomial arithmetic and bounded distribution functions. Then there exists a probabilistic transition system $\mathbf{T_P}$ with program variables $\mathcal{PV}$ such that for every scheduler $\mathfrak{s} \in \mathsf{Scheds}_{\mathbf{P}}$ there is a scheduler $\mathfrak{s}' \in \mathsf{Scheds}_{\mathbf{T_P}}$ such that for every state $s_0 \in \Sigma$ and all $s \in \Sigma$ we have

$$\mu^{\mathbf{P}}_{\mathfrak{s},s_0}(s) = \mathbb{P}^{\mathbf{T_P}}_{s_0}(\mathfrak{s}')\left(\mathsf{Conf}^+(\ell_\perp, t_\perp, s)\mathsf{Conf}^\omega\right),$$

i.e., the probability that $\mathbf{P}$ started in s_0 terminates in s when resolving non-determinism with $\mathfrak{s}$ is identical to the probability that $\mathbf{T_P}$ terminates in a configuration with state s when resolving the non-determinism with $\mathfrak{s}'$. Intuitively, such a PITS can be constructed as an "abstracted" version of the operational MDP of $\mathbf{P}$ (see Sect. 14.2).

We explain this construction with an example.

Example 17.1.15. Reconsider the pGCL-program $\mathbf{P}_{race}$ in Fig. 14.1 which models the race between a tortoise and a hare. This program can be modeled (in the sense of Rem. 17.1.14) by the PITS in Fig. 17.2. To avoid confusion of transitions and the variable "t" occurring in $\mathbf{P}_{race}$ we use the program variables x_h (instead of "h") and x_t (instead of "t").

Location ℓ_0 is a virtual location which does not have a counterpart in $\mathbf{P}_{race}$ and the same holds for general transition g_0. These are only needed since the initial location is not allowed to have incoming transitions. Location ℓ_1 models the beginning of $\mathbf{P}_{race}$. General transition g_1 then checks the condition of the loop since it is only executed whenever $x_h \leq x_t$. Location ℓ_2 corresponds to the beginning of the body of the loop. General transition g_2 mimics the probabilistic choice: with probability $\frac{1}{2}$ either transition t_2 is executed which does not change any of the variables, i.e., this corresponds to **skip**, or transition t_3 is executed which updates x_h by adding a value drawn from the uniform distribution on $\{0, ..., 10\}$. Location ℓ_3 corresponds to the position directly after the probabilistic choice. Finally, general transition g_3 increments the value of x_t by 1 which completes $\mathbf{P}_{race}$.

Of course, there are multiple ways to encode $\mathbf{P}_{race}$ as a PITS. For example, we could have mimicked $\mathbf{P}_{race}$ by the PITS which results from Fig. 17.2 when combining t_1, t_2, and t_4, and t_1, t_3, and t_4 into two self loops at location ℓ_1 and deleting locations ℓ_2 and ℓ_3. However, the PITS in Fig. 17.2 gives an intuition on how to translate *any* pGCL-program into a PITS satisfying the conditions of Rem. 17.1.14. Moreover, the PITS in Fig. 17.2 is indeed an abstract version of the operational MDP of $\mathbf{P}_{race}$ as illustrated in Fig. 14.4 (i.e., where states have been

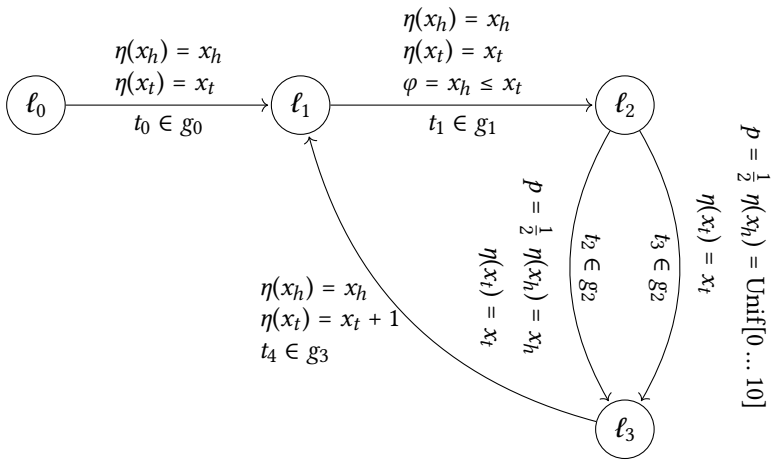


Figure 17.2: PITS Equivalent to Fig. 14.1

"abstracted away").

We cannot model infinite[2] non-deterministic sampling in pGCL-programs. Plotkin proved in [Plo76] that one cannot model infinite non-determinism in a terminating non-probabilistic pGCL-program. Thus, in terms of non-deterministic sampling, PITSs are more expressive than pGCL-programs. However, in Rem. 14.2.15 we have discussed the pGCL-program in Fig. 14.8 which involves non-determinism and is AST where the variable x can take an arbitrary non-negative integer. Nevertheless, in Fig. 14.8 the scheduler *cannot* determine the value that x takes a priori since the value of x at termination also depends on a probabilistic choice. However, in a PITS we can just use the update $\eta(x) = u$ and the condition $\varphi = u \geq 0$. In this way, x is assigned an arbitrary non-negative integer whose value only depends on the scheduler.

17.2 Complexity Bounds

In Sect. 17.2.1, we first recapitulate the concepts of (non-probabilistic) runtime and size bounds for PITSs from [Bro+16]. Then we introduce *expected* runtime and size

[2]This is sometimes also referred to as "unbounded" non-determinism since a set of integers is unbounded (by means of absolute value) iff it is infinite.

bounds in Sect. 17.2.2 and connect them to their non-probabilistic counterparts.

17.2.1 Runtime and Size Bounds

Again, let $\mathbf{T}$ denote the PITS which we want to analyze. Def. 17.2.1 recapitulates the notions of runtime and size bounds from [Bro+16] in our setting. Recall that bounds from $\mathcal{B}$ do not contain temporary variables, i.e., we always try to infer bounds in terms of the initial values of the *program variables*. As usual, $\sup \emptyset = 0$ as all occurring sets only contain non-negative elements. Moreover, as we use the same set of bounds for non-probabilistic bounds and expected bounds we identify $\omega \in \overline{\mathbb{N}}$ and $\infty \in \overline{\mathbb{R}}_{\geq 0}$ as we have done for expected values.

Definition 17.2.1 (Runtime and Size Bounds [Bro+16])**.** A pair $(\mathcal{RB}, \mathcal{SB})$ of mappings $\mathcal{RB} : \mathcal{T} \to \mathcal{B}$ and $\mathcal{SB} : \mathcal{T} \times \mathcal{V} \to \mathcal{B}$ is a (non-probabilistic) *bound pair* if

(i) $\mathcal{RB}$ is a *runtime bound*, i.e., for all transitions $t \in \mathcal{T}$, all states $s_0 \in \Sigma$, and all schedulers $\mathfrak{s} \in \mathrm{Scheds}$, we have

$$|s_0|(\mathcal{RB}(t)) \geq \sup\left\{ |\{i \mid t_i = t\}| \;\middle|\; f = (_, t_0, _) \cdots (_, t_n, _) \wedge p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) > 0 \right\}.$$

(ii) $\mathcal{SB} : \mathcal{T} \times \mathcal{V} \to \mathcal{B}$ is a *size bound*, i.e., for all transitions $t \in \mathcal{T}$, all variables $x \in \mathcal{V}$, all states $s_0 \in \Sigma$, and all schedulers $\mathfrak{s} \in \mathrm{Scheds}$, we have

$$|s_0|(\mathcal{SB}(t,x)) \geq \sup\left\{ |s(x)| \;\middle|\; f = \cdots (_, t, s) \wedge p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) > 0 \right\}.$$

So $\mathcal{RB}(t)$ is a bound on the number of executions of t and $\mathcal{SB}(t,x)$ over-approximates the greatest absolute value that $x \in \mathcal{V}$ takes after the application of the transition t in any admissible finite path. Note that Def. 17.2.1 does *not* apply to t_{in} and $t_\perp$ since they are not contained in $\mathcal{T}$.

We will use such non-probabilistic bound pairs for an initialization of expected bounds (Thm. 17.2.10) and to compute improved expected runtime and size bounds in Sect. 17.3 and 17.4.

Example 17.2.2. The technique of [Bro+16] computes the bound pair $(\mathcal{RB}, \mathcal{SB})$ in Fig. 17.3 for the PITS $\mathbf{T}_{sq}$ in Fig. 17.1 (by ignoring the probabilities of the transitions). Clearly, t_0 and t_3 can only be evaluated once. Since t_1 decrements x and no transition increments it, t_1's runtime is bounded by $|s_0|(x)$. However,

$$\mathcal{RB}(t_i) = \begin{cases} 1, & \text{if } i = 0 \text{ or } i = 3 \\ x, & \text{if } i = 1 \\ \infty, & \text{if } i = 2 \text{ or } i = 4 \end{cases}$$

(a) Runtime Bounds

$$\mathcal{SB}(t_i, x) = \begin{cases} x, & \text{if } i \in \{0,1,2\} \\ 3 \cdot x, & \text{if } i \in \{3,4\} \end{cases}$$

$$\mathcal{SB}(t_i, y) = \begin{cases} y, & \text{if } i = 0 \\ \infty, & \text{if } i \in \{1,2,3,4\} \end{cases}$$

(b) Size Bounds

Figure 17.3: Non-Probabilistic Bound Pair for the PITS in Fig. 17.1

t_2 can be executed arbitrarily often if $s_0(x) > 0$, i.e., the value of y can become arbitrarily large. Thus, the runtimes of t_2 and t_4 are unbounded (i.e., $\mathbf{T}_{sq}$ is not terminating when regarding it as a non-probabilistic system). $\mathcal{SB}(t, x)$ is finite for all transitions t since x is never increased. In contrast, the value of y can be arbitrarily large after all transitions but t_0.

17.2.2 Expected Runtime and Size Bounds

We now define the *expected* runtime for transitions and the expected size of so-called general result variables of a PITS **T**.

Definition 17.2.3 (Expected Runtime, PAST [BG05]). For $g \in \mathcal{GT}$, its *runtime* is the random variable $\mathcal{R}(g)$ where $\mathcal{R} : \mathcal{GT} \to \mathsf{Conf}^\omega \to \overline{\mathbb{N}}$ with

$$\mathcal{R}(g)((_, t_0, _)(_, t_1, _)\cdots) = |\{i \mid t_i \in g\}|.$$

For a state $s_0 \in \Sigma$ and a scheduler $\mathfrak{s} \in \mathrm{Scheds}$, the *expected runtime* of $g \in \mathcal{GT}$ is $\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{R}(g))$ and the *expected runtime* of **T** is $\sum_{g \in \mathcal{GT}} \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{R}(g))$.

If **T**'s expected runtime is finite for every scheduler $\mathfrak{s}$ and every initial state s_0, then **T** is called *positive almost-surely terminating (PAST)*.

So $\mathcal{R}(g)(\vartheta)$ is the number of executions of a transition from g in the run ϑ.

After defining the notions of PAST and expected runtime for a PITS the question arises how these notions relate to the notions for probabilistic programs in the sense of Rem. 17.1.14.

Remark 17.2.4. If $\mathbf{P}$ is a probabilistic program, the PITS $\mathbf{T}_\mathbf{P} = (\mathcal{PV}, \mathcal{L}, \mathcal{GT}, \ell_0)$ arising from Rem. 17.1.14 which is equivalent to $\mathbf{P}$ (its construction is illustrated in Ex. 17.1.15) has *asymptotically* the same expected runtime, i.e., exprt $[\![\mathbf{P}]\!]^{\mathfrak{s}}$ and $\lambda s_0.\ \sum_{g \in \mathcal{GT}} \mathbb{E}^{\mathrm{pt}}_{s_0, \mathfrak{s}'}(\mathcal{R}(g))$ asymptotically coincide where $\mathfrak{s}'$ is the scheduler that mimics $\mathfrak{s}$ in $\mathbf{T}_\mathbf{P}$. (Remember that $\mathbf{T}_\mathbf{P}$ can be assumed to be a "finite abstraction" of the operational MDP of $\mathbf{P}$.) Thus, $\mathbf{P}$ is PAST iff $\mathbf{T}_\mathbf{P}$ is PAST.

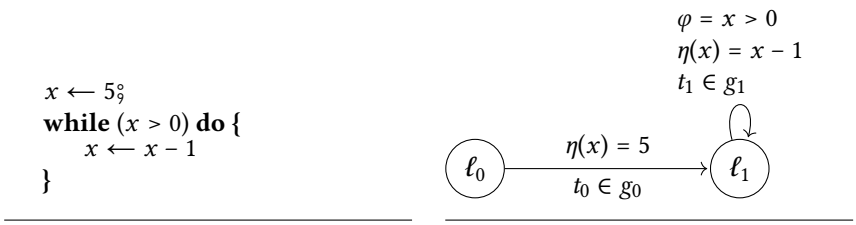
Example 17.2.5. Reconsider Ex. 17.1.15. In Ex. 16.3.11 we have seen that the expected looping time of $\mathbf{P}_{race}$ is bounded from above by $\frac{2}{3} \cdot (t - h + 1) + \frac{16}{3}$ whenever $t > h$ initially. Thus, in this case the expected runtime of $\mathbf{P}_{race}$ is bounded from above by $3 \cdot \left(\frac{2}{3} \cdot (t - h + 1) + \frac{16}{3}\right) + 1 = 2 \cdot t - 2 \cdot h + 19$ (see the discussion before Def. 16.1.3). For the PITS in Fig. 17.2, the approach we present in Sect. 17.3 and 17.4 infers that its expected runtime is bounded from above by

$$3 \cdot \left(\tfrac{2}{3} \cdot (x_t + x_h + 1) + \tfrac{16}{3}\right) + 1 = 2 \cdot x_t + 2 \cdot x_h + 19,$$

whenever x_t and x_h are instantiated with the absolute value they have in the initial state. Here, we had to over-approximate the polynomial $x_t - x_h + 1 \in \mathbb{Z}[\mathcal{V}]$ to $x_t + x_h + 1 \in \mathcal{B}$ to ensure weak monotonic increase.

Moreover, we have briefly discussed the notion of *strong positive almost-sure termination* or *SAST* in Rem. 14.2.15 which refers to $\sup_{\mathfrak{s} \in \mathsf{Scheds}} \sum_{g \in \mathcal{GT}} \mathbb{E}^{\mathrm{pt}}_{s_0, \mathfrak{s}}(\mathcal{R}(g)) < \infty$ for all $s_0 \in \Sigma$ (see [ALY20]). As for probabilistic programs, SAST and PAST clearly differ for PITSs, i.e., SAST is strictly stronger than PAST (see Rem. 14.2.15). However, our notion of expected runtime bounds we present in Def. 17.2.8 yields an upper bound on $\sum_{g \in \mathcal{GT}} \mathbb{E}^{\mathrm{pt}}_{s_0, \mathfrak{s}}(\mathcal{R}(g))$ independent of the scheduler $\mathfrak{s} \in \mathsf{Scheds}$, i.e., it yields an upper bound on $\sup_{\mathfrak{s} \in \mathsf{Scheds}} \sum_{g \in \mathcal{GT}} \mathbb{E}^{\mathrm{pt}}_{s_0, \mathfrak{s}}(\mathcal{R}(g))$. Thus, whenever our approach yields a finite bound on the expected runtime of a PITS, it proves PAST by proving SAST. Nevertheless, as we are interested in a quantitative result, i.e., a bound, we will neither discuss PAST nor SAST in the rest of this chapter.

While non-probabilistic size bounds refer to pairs (t, x) of transitions $t \in \mathcal{T}$ and variables $x \in \mathcal{V}$ (so-called *result variables* in [Bro+16]), we now introduce expected size bounds for *general result variables* (g, ℓ, x), which consist of a general transition g, one of its target locations ℓ, and a program variable $x \in \mathcal{PV}$. So x must not be a temporary variable (which represents non-deterministic sampling and is independent of probabilistic branching and probabilistic sampling) since general result variables are used for *expected* size bounds.



(a) In a Probabilistic Program (b) In a PITS

Figure 17.4: Notions of "Size"

Definition 17.2.6 (Expected Size)**.** The set of *general result variables* is $\mathcal{GRV} = \{(g, \ell, x) \mid g \in \mathcal{GT}, x \in \mathcal{PV}, (_, _, _, _, \ell) \in g\}$. The *size* of $\alpha = (g, \ell, x) \in \mathcal{GRV}$ is the random variable $\mathcal{S}(\alpha)$ where $\mathcal{S} : \mathcal{GRV} \to \mathsf{Conf}^\omega \to \overline{\mathbb{N}}$ with

$$\mathcal{S}(g, \ell, x)\left((\ell_0, t_0, s_0)(\ell_1, t_1, s_1)\cdots\right) = \sup\{|s_i(x)| \mid \ell_i = \ell \wedge t_i \in g\}\,.$$

For a state $s_0 \in \Sigma$ and a scheduler $\mathfrak{s} \in \mathsf{Scheds}$, the *expected size* of α is $\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{S}(\alpha))$.

So for any run ϑ, $\mathcal{S}(g, \ell, x)(\vartheta)$ is the greatest absolute value of x in location ℓ, whenever ℓ was entered with a transition from g.

Again, let us briefly discuss how this notion relates to probabilistic programs and expected outcomes in terms of their expectation transformer semantics as introduced in Sect. 14.3. The answer is simple: this notion does *not* have a counterpart. Not only do result variables have no analogue (a probabilistic program has neither general transitions nor locations) but the random variable $\mathcal{S}(\alpha)$ takes into account *all* values of the variables that occur in an execution of the PITS, whereas for expected outcomes in probabilistic programs we only take the final states into account, i.e., the values of the variables after termination. So, this is not even related to probabilistic behavior but the notions of "size" are unrelated even when no probabilistic behavior occurs. This is illustrated in the next example.

Example 17.2.7. Consider the program **P** in Fig. 17.4(a). We have $\mathrm{wp}\,[\![\mathbf{P}]\!]\,(x) = 0$ since the program terminates whenever $x = 0$. Nonetheless, in the PITS in Fig. 17.4(b), which is equivalent to **P**, $\mathcal{S}(g_1, \ell_1, x)(\vartheta) = 5$ for every run $\vartheta \in \mathsf{Conf}^\omega$,

$$\mathcal{RB}_{\mathbb{E}}(g_i) = \begin{cases} 1, & \text{if } i \in \{0,2\} \\ 2 \cdot x, & \text{if } i = 1 \\ 6 \cdot x^2 + 2 \cdot y, & \text{if } i = 3 \end{cases}$$

(a) Expected Runtime Bounds

$$\mathcal{SB}_{\mathbb{E}}(g_i, _, x) = \begin{cases} x, & \text{if } i = 0 \\ 2 \cdot x, & \text{if } i = 1 \\ 3 \cdot x, & \text{if } i \in \{2,3\} \end{cases}$$

$$\mathcal{SB}_{\mathbb{E}}(g_0, \ell_1, y) = y$$

$$\mathcal{SB}_{\mathbb{E}}(g_2, \ell_2, y) = 6 \cdot x^2 + 2 \cdot y$$

$$\mathcal{SB}_{\mathbb{E}}(g_1, \ell_1, y) = 6 \cdot x^2 + y$$

$$\mathcal{SB}_{\mathbb{E}}(g_3, \ell_2, y) = 12 \cdot x^2 + 4 \cdot y$$

(b) Expected Size Bounds

Figure 17.5: Expected Bound Pair for the PITS in Fig. 17.1

i.e., the maximal "expected" size of x is 5.

We now define bounds on the expected runtime and the expected size which hold *independent* of the scheduler.

Definition 17.2.8 (Expected Runtime and Size Bounds)**.** A pair $(\mathcal{RB}_{\mathbb{E}}, \mathcal{SB}_{\mathbb{E}})$ of mappings $\mathcal{RB}_{\mathbb{E}} : \mathcal{GT} \to \mathcal{B}$ and $\mathcal{SB}_{\mathbb{E}} : \mathcal{GRV} \to \mathcal{B}$ is an *expected bound pair* if

(i) $\mathcal{RB}_{\mathbb{E}}$ is an *expected runtime bound*, i.e., for all $g \in \mathcal{GT}$, all $s_0 \in \Sigma$, and all schedulers $\mathfrak{s}$, we have $|s_0|(\mathcal{RB}_{\mathbb{E}}(g)) \geq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{R}(g))$.

(ii) $\mathcal{SB}_{\mathbb{E}}$ is an *expected size bound*, i.e., for all $\alpha \in \mathcal{GRV}$, all $s_0 \in \Sigma$, and all schedulers $\mathfrak{s}$, we have $|s_0|(\mathcal{SB}_{\mathbb{E}}(\alpha)) \geq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{S}(\alpha))$.

Example 17.2.9. Our techniques from Sect. 17.3 and 17.4 will derive the expected bound pair in Fig. 17.5 for the PITS $\mathbf{T}_{sq}$ (see Fig. 17.1).

While the runtimes of t_2 and t_4 were unbounded in the non-probabilistic case (see Fig. 17.3), we obtain finite bounds on the expected runtimes of $g_1 = \{t_1, t_2\}$ and $g_3 = \{t_4\}$. For example, we can expect x to be non-positive after at most

$|s_0|\,(2 \cdot x)$ iterations of g_1. Based on the above expected runtime bounds, the expected runtime of $\mathbf{T}_{sq}$ is at most $|s_0|\,(\mathcal{RB}_{\mathbb{E}}(g_0) + \ldots + \mathcal{RB}_{\mathbb{E}}(g_3)) = |s_0|\,(2 + 2 \cdot x + 2 \cdot y + 6 \cdot x^2)$, i.e., it is in $\mathcal{O}(n^2)$ where n is the maximal absolute value of the program variables at the start of the program.

The following theorem shows that non-probabilistic bounds can be lifted to expected bounds since they do not only bound the expected value of $\mathcal{R}(g)$ resp. $\mathcal{S}(\alpha)$, but the whole distribution.

Theorem 17.2.10 (Lifting of Bounds)**.** *For a bound pair $(\mathcal{RB}, \mathcal{SB})$, the pair $(\mathcal{RB}_{\mathbb{E}}, \mathcal{SB}_{\mathbb{E}})$ with $\mathcal{RB}_{\mathbb{E}}(g) = \sum_{t \in g} \mathcal{RB}(t)$ and $\mathcal{SB}_{\mathbb{E}}(g, \ell, x) = \sum_{t=(_,_,_,_,\ell) \in g} \mathcal{SB}(t, x)$ is an expected bound pair.*

Here, we over-approximate the maximum of $\mathcal{SB}(t, x)$ for $t = (_, _, _, _, \ell) \in g$ by their sum. For asymptotic bounds, this does not affect precision, since $\max(f, g)$ and $f + g$ have the same asymptotic growth for any non-negative functions f, g.

Example 17.2.11. When lifting the bound pair in Fig. 17.3 to expected bounds according to Thm. 17.2.10, one would obtain $\mathcal{RB}_{\mathbb{E}}(g_0) = \mathcal{RB}_{\mathbb{E}}(g_2) = 1$ and $\mathcal{RB}_{\mathbb{E}}(g_1) = \mathcal{RB}_{\mathbb{E}}(g_3) = \infty$. Moreover, $\mathcal{SB}_{\mathbb{E}}(g_0, \ell_1, x) = x$, $\mathcal{SB}_{\mathbb{E}}(g_1, \ell_1, x) = 2 \cdot x$, $\mathcal{SB}_{\mathbb{E}}(g_2, \ell_2, x) = \mathcal{SB}_{\mathbb{E}}(g_3, \ell_2, x) = 3 \cdot x$, $\mathcal{SB}_{\mathbb{E}}(g_0, \ell_1, y) = y$, and $\mathcal{SB}_{\mathbb{E}}(g_i, _, y) = \infty$ whenever $i \neq 0$. Thus, with these lifted bounds one cannot show that the expected runtime of $\mathbf{T}_{sq}$ is finite, i.e., they are substantially less precise than the finite expected bounds from Fig. 17.5. Our approach will compute such finite expected bounds by repeatedly improving the lifted bounds of Thm. 17.2.10.

17.3 Computing Expected Runtime Bounds

We first present a new variant of probabilistic linear ranking functions in Sect. 17.3.1. Based on this, in Sect. 17.3.2 we introduce our modular technique to infer expected runtime bounds by using expected size bounds.

17.3.1 Probabilistic Linear Ranking Functions

For probabilistic programs, several techniques based on ranking supermartingales have been developed. In this section, we define a class of probabilistic ranking functions that will be suitable for our modular analysis.

We restrict ourselves to ranking functions $\mathfrak{r} : \mathcal{L} \to \mathbb{R}[\mathcal{PV}]_{\text{lin}}$ that map every location to a *linear polynomial* (i.e., of at most degree 1) without temporary variables.

The linearity restriction is common to ease the automated inference of ranking functions. Moreover, this restriction will be needed for the soundness of our technique. Nevertheless, our approach of course also infers non-linear expected runtimes (by combining the linear bounds obtained for different parts of the PITS).

Let $\exp_{\mathfrak{r},g,s}$ denote the expected value of $\mathfrak{r}$ after an execution of $g \in \mathcal{GT}$ in state $s \in \Sigma$. Here, $s_\eta(x)$ is the expected value of $x \in \mathcal{PV}$ after performing the update η in state s. So if $\eta(x) \in \mathcal{D}$, then x's expected value after the update results from adding the expected value of the probability distribution $\eta(x)(s)$:

$$\exp_{\mathfrak{r},g,s} = \sum_{(\ell,p,\varphi,\eta,\ell') \in g} p \cdot s_\eta(\mathfrak{r}(\ell')) \text{ with } s_\eta(x) = \begin{cases} s(\eta(x)), & \text{if } \eta(x) \in \mathbb{Z}[\mathcal{V}] \\ s(x) + \mathbb{E}(\eta(x)(s)), & \text{if } \eta(x) \in \mathcal{D} \end{cases}$$

Definition 17.3.1 (PLRF). Let $\mathcal{GT}_> \subseteq \mathcal{GT}_{\text{ni}} \subseteq \mathcal{GT}$. Then $\mathfrak{r} : \mathcal{L} \to \mathbb{R}[\mathcal{PV}]_{\text{lin}}$ is a *probabilistic linear ranking function (PLRF)* for $\mathcal{GT}_>$ and $\mathcal{GT}_{\text{ni}}$ if for all $g \in \mathcal{GT}_{\text{ni}} \setminus \mathcal{GT}_>$ and $c' \in \mathsf{Conf}$ there is a $\bowtie_{g,c'} \in \{<, \geq\}$ such that for all finite paths $\cdots\ c'\ c$ that are admissible for some $s_0 \in \Sigma$ and $\mathfrak{s} \in \mathsf{Scheds}$, and where $c = (\ell, t, s)$ (i.e., where t is the transition that is used in the step from c' to c), we have:

Boundedness (a): If $t \in g$ for a $g \in \mathcal{GT}_{\text{ni}} \setminus \mathcal{GT}_>$, then $s(\mathfrak{r}(\ell)) \bowtie_{g,c'} 0$.

Boundedness (b): If $t \in g$ for a $g \in \mathcal{GT}_>$, then $s(\mathfrak{r}(\ell)) \geq 0$.

Non-Increase: If $\ell = \ell_g$ for a $g \in \mathcal{GT}_{\text{ni}}$ and $s(\varphi_g) = \text{true}$, then $s(\mathfrak{r}(\ell)) \geq \exp_{\mathfrak{r},g,s}$.

Decrease: If $\ell = \ell_g$ for a $g \in \mathcal{GT}_>$ and $s(\varphi_g) = \text{true}$, then $s(\mathfrak{r}(\ell)) - 1 \geq \exp_{\mathfrak{r},g,s}$.

So if one is restricted to the sub-system with the non-increasing transitions $\mathcal{GT}_{\text{ni}}$, then $\mathfrak{r}(\ell)$ is an upper bound on the expected number of applications of transitions from $\mathcal{GT}_>$ when starting in ℓ. Hence, a PLRF for $\mathcal{GT}_> = \mathcal{GT}_{\text{ni}} = \mathcal{GT}$ would imply that the PITS is PAST (see, e.g., [ACN18; BG06; CFN20; Cha+18]). However, our PLRFs differ from the standard notion of probabilistic ranking functions by considering arbitrary subsets $\mathcal{GT}_{\text{ni}} \subseteq \mathcal{GT}$. This is needed for the modularity of our approach which allows us to analyze parts of the PITS separately (e.g., $\mathcal{GT} \setminus \mathcal{GT}_{\text{ni}}$ is ignored when inferring a PLRF). Thus, our "Boundedness" conditions differ slightly from the corresponding conditions in other definitions. Boundedness (b) requires that $g \in \mathcal{GT}_>$ never leads to a configuration where $\mathfrak{r}$ is negative. Boundedness (a) states that in an admissible path where $g = \{t_1, t_2, \ldots\} \in \mathcal{GT}_{\text{ni}} \setminus \mathcal{GT}_>$ is used for continuing in configuration c', if executing t_1 in c' makes $\mathfrak{r}$ negative, executing t_2 must make $\mathfrak{r}$ negative as well. Thus, such a g can never come before a general transition from $\mathcal{GT}_>$ in an admissible path and hence, g can be ignored when inferring upper bounds on the expected runtimes

for the general transitions in $\mathcal{GT}_>$. This increases the power of our approach and it allows us to consider only *non-negative* random variables in our correctness proofs.

We use SMT solvers to generate PLRFs automatically. Then for "Boundedness", we regard all $s' \in \Sigma$ with $s'(\varphi_g) = \text{true}$ and require "Boundedness" for any state s that is reachable from s'.

Example 17.3.2. Consider again the PITS $\mathbf{T}_{sq}$ in Fig. 17.1 and the sets $\mathcal{GT}_> = \mathcal{GT}_{\text{ni}} = \{g_1\}$ and $\mathcal{GT}'_> = \mathcal{GT}'_{\text{ni}} = \{g_3\}$, which correspond to its two loops.

The function $\mathfrak{r}$ with $\mathfrak{r}(\ell_1) = 2 \cdot x$ and $\mathfrak{r}(\ell_0) = \mathfrak{r}(\ell_2) = 0$ is a PLRF for $\mathcal{GT}_> = \mathcal{GT}_{\text{ni}}$: For every admissible configuration (ℓ, t, s) with $t \in g_1$ we have $\ell = \ell_1$ and $s(\mathfrak{r}(\ell_1)) = 2 \cdot s(x) \geq 0$ since x was positive before (due to g_1's condition) and it was either decreased by 1 or not changed by the update of t_1 resp. t_2. Hence $\mathfrak{r}$ is bounded. Moreover, for $s_1(x) = s(x-1) = s(x) - 1$ and $s_2(x) = s(x)$ we have:

$$\exp_{\mathfrak{r},g,s} = \tfrac{1}{2} \cdot s_1(\mathfrak{r}(\ell_1)) + \tfrac{1}{2} \cdot s_2(\mathfrak{r}(\ell_1)) = 2 \cdot s(x) - 1 = s(\mathfrak{r}(\ell_1)) - 1$$

So $\mathfrak{r}$ is decreasing on g_1 and as $\mathcal{GT}_> = \mathcal{GT}_{\text{ni}}$ also the non-increase property holds.

Similarly, $\mathfrak{r}'$ with $\mathfrak{r}'(\ell_2) = y$ and $\mathfrak{r}'(\ell_0) = \mathfrak{r}'(\ell_1) = 0$ is a PLRF for $\mathcal{GT}'_> = \mathcal{GT}'_{\text{ni}}$.

In our implementation, $\mathcal{GT}_>$ is always a singleton and we let $\mathcal{GT}_{\text{ni}} \subseteq \mathcal{GT}$ be a cycle in the call graph where we find a PLRF for $\mathcal{GT}_> \subseteq \mathcal{GT}_{\text{ni}}$. The next subsection shows how we can then obtain an expected runtime bound for the overall system by searching for suitable ranking functions repeatedly.

Remark 17.3.3. PLRFs are closely related to ert-superinvariants (Def. 14.3.13) for probabilistic loops w.r.t. the postruntime 0. Such a superinvariant $I : \Sigma \to \overline{\mathbb{R}}_{\geq 0}$ takes only non-negative values by definition. Moreover, I has to satisfy $I \geq {}^{\text{ert}}\Phi_0 = 1 + [\varphi] \cdot \text{ert}\,[\![\mathbf{P}]\!]\,(I)$ if φ is the condition and $\mathbf{P}$ the body of the loop. Here, $\text{ert}\,[\![\mathbf{P}]\!]\,(I)$ is the counterpart of $\exp_{\mathfrak{r},g,s}$. Thus, an ert-superinvariant I satisfies conditions similar to Decrease and Boundedness (b). As I takes only non-negative values and considers the complete probabilistic loop (i.e., there is no counterpart of $\mathcal{GT}_{\text{ni}}$ or $\mathcal{GT}_>$) , there are no conditions similar to Boundedness (a) and Non-Increase.

17.3.2 Inferring Expected Runtime Bounds

Our approach to infer expected runtime bounds is based on an underlying (non-probabilistic) bound pair $(\mathcal{RB}, \mathcal{SB})$ which is computed by existing techniques (in our implementation, we use [Bro+16]). To do so, we abstract the PITS to a standard integer transition system by ignoring the probabilities of transitions and replacing

probabilistic with non-deterministic sampling (e.g., the update $\eta(x) = \mathrm{Geo}(\frac{1}{2})$ would be replaced by $\eta(x) = x + u$ with $u \in \mathcal{TV}$, where $u > 0$ is added to the condition). Of course, we usually have $\mathcal{RB}(t) = \infty$ for some transitions t.

We start with the expected bound pair $(\mathcal{RB}_{\mathbb{E}}, \mathcal{SB}_{\mathbb{E}})$ that is obtained by lifting $(\mathcal{RB}, \mathcal{SB})$ as in Thm. 17.2.10. Afterwards, the expected runtime bound $\mathcal{RB}_{\mathbb{E}}$ is improved repeatedly by applying the following Thm. 17.3.8 (and similarly, $\mathcal{SB}_{\mathbb{E}}$ is improved repeatedly by applying Thm. 17.4.7 and 17.4.10 from Sect. 17.4). Our approach alternates the improvement of $\mathcal{RB}_{\mathbb{E}}$ and $\mathcal{SB}_{\mathbb{E}}$, and it uses expected size bounds on "previous" transitions to improve expected runtime bounds, and vice versa.

To improve $\mathcal{RB}_{\mathbb{E}}$, we generate a PLRF $\mathfrak{r}$ for a part of the PITS. To obtain a bound for the *full* PITS from $\mathfrak{r}$, one has to determine which transitions can enter the part of the PITS and from which locations it can be entered.

Definition 17.3.4 (Entry Locations and Transitions)**.** For $\mathcal{GT}_{\mathrm{ni}} \subseteq \mathcal{GT}$ and $\ell \in \mathcal{L}$, the *entry transitions* are $\mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell) = \{g \in \mathcal{GT} \setminus \mathcal{GT}_{\mathrm{ni}} \mid \exists t \in g.\, t = (_, _, _, _, \ell)\}$. Then the *entry locations* are all start locations of $\mathcal{GT}_{\mathrm{ni}}$ whose entry transitions are not empty, i.e., $\mathcal{EL}_{\mathcal{GT}_{\mathrm{ni}}} = \left\{ \ell \;\middle|\; \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell) \neq \emptyset \wedge (\ell, _, _, _, _) \in \bigcup \mathcal{GT}_{\mathrm{ni}} \right\}$. Here, for a set of sets like $\mathcal{GT}_{\mathrm{ni}}$, $\bigcup \mathcal{GT}_{\mathrm{ni}}$ denotes their union, i.e., $\bigcup \mathcal{GT}_{\mathrm{ni}} = \bigcup_{g \in \mathcal{GT}_{\mathrm{ni}}} g$.

Example 17.3.5. For the PITS $\mathbf{T}_{sq}$ in Fig. 17.1 and $\mathcal{GT}_{\mathrm{ni}} = \{g_1\}$, we have $\mathcal{EL}_{\mathcal{GT}_{\mathrm{ni}}} = \{\ell_1\}$ and $\mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell_1) = \{g_0\}$. So the loop formed by g_1 is entered at location ℓ_1 and the general transition g_0 has to be executed before. Similarly, for $\mathcal{GT}'_{\mathrm{ni}} = \{g_3\}$ we have $\mathcal{EL}_{\mathcal{GT}'_{\mathrm{ni}}} = \{\ell_2\}$ and $\mathcal{ET}_{\mathcal{GT}'_{\mathrm{ni}}}(\ell_2) = \{g_2\}$.

Recall that if $\mathfrak{r}$ is a PLRF for $\mathcal{GT}_{>} \subseteq \mathcal{GT}_{\mathrm{ni}}$, in a PITS that is restricted to $\mathcal{GT}_{\mathrm{ni}}$, $\mathfrak{r}(\ell)$ is an upper bound on the expected number of executions of transitions from $\mathcal{GT}_{>}$ when starting in ℓ. Since $\mathfrak{r}(\ell)$ may contain negative coefficients, it is not weakly monotonically increasing in general. Thus, we over-approximate it.

Definition 17.3.6 (Over-Approximating Polynomials by Bounds)**.** Let $e \in \mathbb{R}[\mathcal{PV}]$. Then $\lceil e \rceil \in \mathcal{B}$ denotes the polynomial which arises from e by replacing all coefficients by their absolute value.

Clearly, we have $|s|(\lceil e \rceil) \geq |s|(e)$ for all $s \in \Sigma$.

Example 17.3.7. Reconsider Ex. 17.2.5. To explain why the bound on the expected runtime of Fig. 17.2 computed by the approach presented in this chapter differs slightly from the one obtained in Ex. 16.3.11, we said that we have to ensure weak monotonic increase. Indeed

$$\lceil x_h - x_t + 1 \rceil = \lceil x_h + (-1) \cdot x_t + 1 \rceil = x_h + x_t + 1 \in \mathcal{B}\,.$$

So $\lceil \mathfrak{r}(\ell) \rceil$ is an upper bound on the expected number of executions of transitions from $\mathcal{GT}_{>}$ when starting in ℓ. To turn $\lceil \mathfrak{r}(\ell) \rceil$ into a bound for the full PITS, one has to take into account how often the sub-system with the transitions $\mathcal{GT}_{\text{ni}}$ is reached via an entry transition $h \in \mathcal{ET}_{\mathcal{GT}_{\text{ni}}}(\ell)$ for some $\ell \in \mathcal{EL}_{\mathcal{GT}_{\text{ni}}}$. This can be over-approximated by $\sum_{t=(_,_,_,_,\ell)\in h} \mathcal{RB}(t)$, which is an upper bound on the number of times that transitions in h to the entry location ℓ of $\mathcal{GT}_{\text{ni}}$ are applied in a full run.

The bound $\lceil \mathfrak{r}(\ell) \rceil$ is expressed in terms of the program variables at the entry location ℓ of $\mathcal{GT}_{\text{ni}}$. To obtain a bound in terms of the variables at the *start* of the PITS, one has to take into account which value a program variable x can be expected to have when the sub-system $\mathcal{GT}_{\text{ni}}$ is reached. For every entry transition $h \in \mathcal{ET}_{\mathcal{GT}_{\text{ni}}}(\ell)$, this value can be over-approximated by $\mathcal{SB}_{\mathbb{E}}(h, \ell, x)$. Thus, we have to instantiate each variable x in $\lceil \mathfrak{r}(\ell) \rceil$ by $\mathcal{SB}_{\mathbb{E}}(h, \ell, x)$. Let $\mathcal{SB}_{\mathbb{E}}(h, \ell, \cdot) : \mathcal{PV} \to \mathcal{B}$ be the mapping with $\mathcal{SB}_{\mathbb{E}}(h, \ell, \cdot)(x) = \mathcal{SB}_{\mathbb{E}}(h, \ell, x)$. Hence, $\mathcal{SB}_{\mathbb{E}}(h, \ell, \cdot)(\lceil \mathfrak{r}(\ell) \rceil)$ over-approximates the expected number of applications of $\mathcal{GT}_{>}$ if $\mathcal{GT}_{\text{ni}}$ is entered in location ℓ, where this bound is expressed in terms of the input variables of the system. Here, weak monotonic increase of $\lceil \mathfrak{r}(\ell) \rceil$ ensures that instantiating its variables by an over-approximation of their size yields an over-approximation of the runtime.

Theorem 17.3.8 (Expected Runtime Bounds)**.** *Let $(\mathcal{RB}_{\mathbb{E}}, \mathcal{SB}_{\mathbb{E}})$ be an expected bound pair, $\mathcal{RB}$ a (non-probabilistic) runtime bound, and $\mathfrak{r}$ a PLRF for $\mathcal{GT}_{>} \subseteq \mathcal{GT}_{\text{ni}} \subseteq \mathcal{GT}$. Then $\mathcal{RB}'_{\mathbb{E}} : \mathcal{GT} \to \mathcal{B}$ is an expected runtime bound where*

$$\mathcal{RB}'_{\mathbb{E}}(g) = \begin{cases} \displaystyle\sum_{\substack{\ell \in \mathcal{EL}_{\mathcal{GT}_{\text{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\text{ni}}}(\ell)}} \left(\sum_{t=(_,_,_,_,\ell)\in h} \mathcal{RB}(t) \right) \cdot \left(\mathcal{SB}_{\mathbb{E}}(h, \ell, \cdot)\left(\lceil \mathfrak{r}(\ell) \rceil\right)\right), & \text{if } g \in \mathcal{GT}_{>} \\ \mathcal{RB}_{\mathbb{E}}(g), & \text{if } g \notin \mathcal{GT}_{>} \end{cases}$$

Example 17.3.9. For the PITS T_{sq} in Fig. 17.1, our approach starts with $(\mathcal{RB}_{\mathbb{E}}, \mathcal{SB}_{\mathbb{E}})$ from Ex. 17.2.11 which results from lifting the bound pair in Fig. 17.3. To improve the bound $\mathcal{RB}_{\mathbb{E}}(g_1) = \infty$, we use the PLRF $\mathfrak{r}$ for $\mathcal{GT}_{>} = \mathcal{GT}_{\text{ni}} = \{g_1\}$

from Ex. 17.3.2. By Ex. 17.3.5, we have $\mathcal{EL}_{\mathcal{GT}_{\text{ni}}} = \{\ell_1\}$ and $\mathcal{ET}_{\mathcal{GT}_{\text{ni}}}(\ell_1) = \{g_0\}$ with $g_0 = \{t_0\}$, whose runtime bound is $\mathcal{RB}(t_0) = 1$, see Fig. 17.3. Using the expected size bound $\mathcal{SB}_{\mathbb{E}}(g_0, \ell_1, x) = x$ from Fig. 17.5, Thm. 17.3.8 yields

$$\mathcal{RB}'_{\mathbb{E}}(g_1) = \mathcal{RB}(t_0) \cdot \mathcal{SB}_{\mathbb{E}}(g_0, \ell_1, \cdot)(\lceil \mathfrak{r}(\ell_1) \rceil) = 1 \cdot 2 \cdot x = 2 \cdot x.$$

To improve $\mathcal{RB}_{\mathbb{E}}(g_3)$, we use the PLRF $\mathfrak{r}'$ for $\mathcal{GT}'_{>} = \mathcal{GT}'_{\text{ni}} = \{g_3\}$ from Ex. 17.3.2. As $\mathcal{EL}_{\mathcal{GT}'_{\text{ni}}} = \{\ell_2\}$ and $\mathcal{ET}_{\mathcal{GT}'_{\text{ni}}}(\ell_2) = \{g_2\}$ by Ex. 17.3.5, where $g_2 = \{t_3\}$ and $\mathcal{RB}(t_3) = 1$ (see Fig. 17.3), Thm. 17.3.8 combined with the expected size bound $\mathcal{SB}_{\mathbb{E}}(g_2, \ell_2, y) = 6 \cdot x^2 + 2 \cdot y$ from Fig. 17.5 yields

$$\mathcal{RB}'_{\mathbb{E}}(g_3) = \mathcal{RB}(t_3) \cdot \mathcal{SB}_{\mathbb{E}}(g_2, \ell_2, \cdot)\left(\left\lceil \mathfrak{r}'(\ell_2) \right\rceil\right) = 1 \cdot \mathcal{SB}_{\mathbb{E}}(g_2, \ell_2, y) = 6 \cdot x^2 + 2 \cdot y.$$

So based on the expected size bounds in Fig. 17.5, we have shown how to compute the expected runtime bounds in Fig. 17.5 automatically.

Similar to [Bro+16], our approach relies on combining bounds that one has computed earlier in order to derive new bounds. Here, bounds may be combined *linearly*, bounds may be *multiplied*, and bounds may even be *substituted* into other bounds. But in contrast to [Bro+16], sometimes one may combine *expected* bounds that were computed earlier and sometimes it is only sound to combine *non-probabilistic* bounds: If a new bound is computed by *linear combinations* of earlier bounds, it is sound to use the "expected versions" of these earlier bounds (this is due to the fact that the expected value operator, i.e., taking an integral w.r.t. a certain probability measure, is linear, see Lemma A.3.8 in App. A). However, if two bounds are *multiplied*, it is in general not sound to use their "expected versions". Thus, it would be *unsound* to use the *expected* runtime bounds $\mathcal{RB}_{\mathbb{E}}(h)$ instead of the *non-probabilistic* bounds $\sum_{t=(_,_,_,_,\ell)\in h} \mathcal{RB}(t)$ on the entry transitions in Thm. 17.3.8 as illustrated in Counterex. 17.3.10. An exception is the special case where $\mathfrak{r}(\ell)$ is *constant*. Then, our implementation indeed uses the expected bound $\mathcal{RB}_{\mathbb{E}}(h)$ instead of $\sum_{t=(_,_,_,_,\ell)\in h} \mathcal{RB}(t)$, see the proof of Thm. 17.3.8 in App. B.45 for details and an example (Ex. B.45.27).

Counterexample 17.3.10. Consider the PITS given in Fig. 17.6, where we omitted updates of the form $\eta(x) = x$ to ease readability. We consider $\mathcal{GT}_{\text{ni}} = \mathcal{GT}_{>} = \{g_4\}$ and the PLRF $\mathfrak{r}$ with $\mathfrak{r}(\ell_3) = y$. Clearly, $\mathcal{SB}_{\mathbb{E}}(g_1, \ell_2, x) = \frac{1}{2} \cdot (1 + 3) = 2 = \mathcal{SB}_{\mathbb{E}}(g_2, \ell_3, z) = \mathcal{SB}_{\mathbb{E}}(g_3, \ell_4, y)$ are valid expected size bounds. So for the only entry location ℓ_4 of $\mathcal{GT}_{\text{ni}}$ (i.e., $\mathcal{EL}_{\mathcal{GT}_{\text{ni}}} = \{\ell_4\}$), we have $\mathcal{SB}_{\mathbb{E}}(g_3, \ell_4, y) = 2$. Moreover, $\mathcal{ET}_{\mathcal{GT}_{\text{ni}}}(\ell_4) = \{g_3\}$ and let us assume that we have already inferred $\mathcal{RB}_{\mathbb{E}}(g_3) = 2$ for the only entry

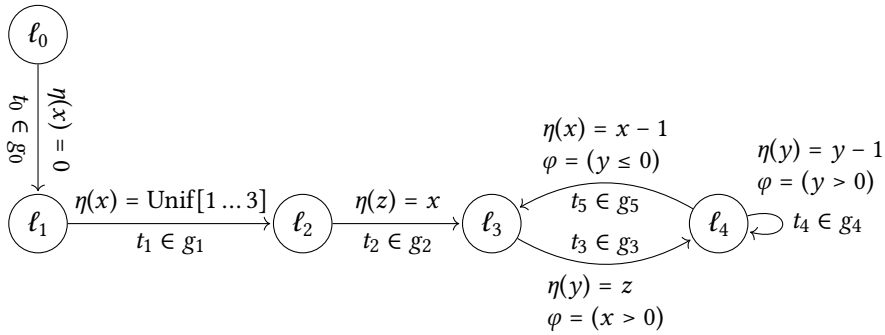


Figure 17.6: PITS Illustrating the Incorrectness of Expected Time Bounds on Entry Transitions

transition g_3. (Note that this "outer loop" decreases x deterministically by 1 and the expected size of x is 2). If Thm. 17.3.8 allowed us to use *expected* runtime bounds for the entry transitions on all entry locations, we would get

$$\begin{aligned}\mathcal{RB}'_{\mathbb{E}}(g_4) &= \sum_{\substack{\ell \in \mathcal{EL}_{\mathcal{GT}_{\mathrm{ni}}}, \\ \tilde{g} \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} \mathcal{RB}_{\mathbb{E}}(\tilde{g}) \cdot (\mathcal{SB}_{\mathbb{E}}(\tilde{g}, \ell, \cdot)([\tau(\ell)])) \\ &= \mathcal{RB}_{\mathbb{E}}(g_3) \cdot \mathcal{SB}_{\mathbb{E}}(g_3, \ell_4, \cdot)([\tau(\ell_4)]) \\ &= 2 \cdot 2 \\ &= 4.\end{aligned}$$

However, if x is incremented by the value $v \in \{1, 2, 3\}$ in g_1 (i.e., it has the value v after the update), g_4 is executed v^2 times. But this means that the expected number of executions of g_4 is

$$\tfrac{1}{3} \cdot \left(1^2 + 2^2 + 3^2\right) = \tfrac{14}{3} = 4 + \tfrac{2}{3} > 4.$$

Thus, using the *expected* runtime bound $\mathcal{RB}_{\mathbb{E}}(g_2) = 2$ does not lead to an upper bound on the expected runtime of g_3, i.e., Thm. 17.3.8 would become unsound. Intuitively, the reason is that the number of executions of $g_3 = \{t_3\}$ in a run and the maximal size of x after executing g_2 are not independent. Nonetheless, when

using the non-probabilistic runtime bound $\mathcal{RB}(t_3) = 3$ instead, which arises from the maximal value that x can take, Thm. 17.3.8 infers the sound expected runtime bound

$$\mathcal{RB}'_{\mathbb{E}}(g_4) = \mathcal{RB}(t_3) \cdot \mathcal{SB}_{\mathbb{E}}(g_3, \ell_4, \cdot)([\mathfrak{r}(\ell_4)]) = 3 \cdot 2 = 6 \;>\; 4 + \tfrac{2}{3}.$$

In general, if bounds $b_1, \dots, b_n$ are *substituted* into another bound b, it is sound to use "expected versions" of the bounds $b_1, \dots, b_n$ if b is *concave*, see, e.g., [ALY20; AMS20; Kal02]. The essential reason is that if a function f is concave, the expected value of the random variable $f(X_1, \dots, X_n)$ is greater or equal than applying f to the expected values of the random variables $X_1, \dots, X_n$, see [Kal02, Lemma 3.5].

A unary function $f : \mathbb{R} \longrightarrow \mathbb{R}$ is *concave* if for any two numbers $v, v' \in \mathbb{R}$ and any r between v and v' (i.e., $r = t \cdot v + (1 - t) \cdot v'$ for $t \in [0, 1]$), the point $(r, f(r))$ is above the line connecting the points $(v, f(v))$ and $(v', f(v'))$. The following definition extends this concept to functions with several arguments.

Definition 17.3.11 (Concavity)**.** A finite bound $b \in \mathcal{B}$ is *concave* if for any $w, w' : \mathcal{PV} \longrightarrow \mathbb{R}_{\geq 0}$ and $t \in [0, 1]$, we have $(t \cdot w + (1 - t) \cdot w')(b) \geq t \cdot w(b) + (1 - t) \cdot w'(b)$.

It turns out that a finite bound is concave iff it is a *linear* polynomial. (The "only if" direction is due to the fact that bounds from $\mathcal{B}$ do not contain negative coefficients, i.e., they do not contain non-linear concave polynomials like $-x^2$.)

Lemma 17.3.12. *A finite bound is concave iff it is a linear polynomial.*

Thus, in Thm. 17.3.8 we may substitute *expected* size bounds $\mathcal{SB}_{\mathbb{E}}(h, \ell, x)$ into $[\mathfrak{r}(\ell)]$ since we restricted ourselves to *linear* ranking functions $\mathfrak{r}$ and hence, $[\mathfrak{r}(\ell)]$ is also linear. Note that in contrast to [ALY20], where a notion of concavity was used to analyze probabilistic term rewriting, a multilinear expression like $x \cdot y$ is not concave when regarding both arguments simultaneously. Hence, it is unsound to use such ranking functions in Thm. 17.3.8. See Counterex. 17.4.9 for a counterexample showing why substituting expected bounds into a non-linear bound is incorrect in general.

As already mentioned at the end of Sect. 16.4, we *cannot* make use of the technique for computing exact expected looping times from Ch. 16 here. The reason is that these expected looping times are *not* concave. Nevertheless, when translating a *cpa*-loop into a PITS as illustrated in Ex. 17.1.15, our approach computes bounds similar to the upper bound from Thm. 16.3.10 (see Ex. 17.2.5).

Finally, it is worth noting that the need for concavity is indeed due to our modular analysis, where parts of the system are analyzed separately, and not due to forward reasoning. For example, when performing a modular analysis of probabilistic programs via expectation transformers (i.e., via backward-reasoning) one also needs concavity to combine the results obtained for different parts of the program (see [AMS20]).

17.4 Computing Expected Size Bounds

In this section, we show how to compute expected size bounds for general result variables. We first compute *local* bounds for one application of a transition (Sect. 17.4.1). To turn them into *global* bounds, we encode the data flow of a PITS in a graph. Sect. 17.4.2 then presents our technique to compute expected size bounds.

17.4.1 Local Change Bounds and the General Result Variable Graph

We first compute a bound on the expected change of a variable during an update. More precisely, for every general result variable (g, ℓ, x) we define a bound $\mathcal{CB}_{\mathbb{E}}(g, \ell, x)$ on the change of the variable x that we can expect in one execution of the general transition g when reaching location ℓ. So we consider all $t = (_, p, _, \eta, \ell) \in g$ and the expected difference between the current value of x and its update $\eta(x)$. However, for $\eta(x) \in \mathbb{Z}[\mathcal{V}]$, $\eta(x) - x$ is not necessarily from $\mathcal{B}$ because it may contain negative coefficients. Thus, we use the over-approximation $\lceil \eta(x) - x \rceil$ (where we always simplify expressions before applying $\lceil \cdot \rceil$, e.g., $\lceil x - x \rceil = \lceil 0 \rceil = 0$). Moreover, $\lceil \eta(x) - x \rceil$ may contain temporary variables. Let $\mathrm{tv}_t : \mathcal{V} \to \mathcal{B}$ instantiate all temporary variables by the largest possible value they can have after evaluating the transition t. Hence, we then use $\mathrm{tv}_t(\lceil \eta(x) - x \rceil)$ instead. For tv_t, we have to use the underlying *non-probabilistic* size bound $\mathcal{SB}$ for the system (as the scheduler determines the value of a temporary variable $y \in \mathcal{TV}$ by non-deterministic sampling w.r.t. the condition of a general transition, i.e., the value of y is independent of probabilistic branching and probabilistic sampling). If x is updated according to a bounded distribution function $D \in \mathcal{D}$, let $\mathfrak{E}(D) \in \mathcal{B}$ denote a finite bound on D, i.e., $\mathbb{E}_{\mathrm{abs}}(D(s)) \leq |s|(\mathfrak{E}(D))$ for all $s \in \Sigma$ (see Def. 17.1.5).

Definition 17.4.1 (Expected Local Change Bound)**.** Let $\mathcal{SB}$ be a size bound. We define the *expected local change bound* $\mathcal{CB}_{\mathbb{E}} : \mathcal{GRV} \to \mathcal{B}$ with $\mathcal{CB}_{\mathbb{E}}(g, \ell, x) =$

$$\sum_{t=(_,p,_,\eta,\ell)\in g} p \cdot \mathrm{ch}_t(\eta(x), x) \text{ where}$$

$$\mathrm{ch}_t(\eta(x), x) = \begin{cases} \mathfrak{E}(D), & \text{if } \eta(x) = D \in \mathcal{D} \\ \mathrm{tv}_t(\lceil \eta(x) - x \rceil), & \text{otherwise} \end{cases} \quad \text{and } \mathrm{tv}_t(y) = \begin{cases} \mathcal{SB}(t, y), & \text{if } y \notin \mathcal{PV} \\ y, & \text{if } y \in \mathcal{PV}. \end{cases}$$

Example 17.4.2. For the PITS $\mathbf{T}_{sq}$ in Fig. 17.1, we have $\mathcal{CB}_{\mathbb{E}}(g_0, _, _) = \mathcal{CB}_{\mathbb{E}}(g_2, _, _) = \mathcal{CB}_{\mathbb{E}}(g_3, \ell_2, x) = 0$ since the respective updates are identities. Moreover,

$$\mathcal{CB}_{\mathbb{E}}(g_1, \ell_1, x) = \tfrac{1}{2} \cdot \lceil (x-1) - x \rceil + \tfrac{1}{2} \cdot \lceil x - x \rceil = \tfrac{1}{2} \cdot 1 + \tfrac{1}{2} \cdot 0 = \tfrac{1}{2}.$$

In a similar way, we obtain $\mathcal{CB}_{\mathbb{E}}(g_1, \ell_1, y) = x$ and $\mathcal{CB}_{\mathbb{E}}(g_3, \ell_2, y) = 1$.

The following theorem shows that for any configuration in a state s' the bound $\mathcal{CB}_{\mathbb{E}}(g, \ell, x)$ is an upper bound on the expected value of $|s(x) - s'(x)|$ if s is the next state obtained when applying g in state s' to reach location ℓ.

Theorem 17.4.3 (Soundness of $\mathcal{CB}_{\mathbb{E}}$)**.** *For any $(g, \ell, x) \in \mathcal{GRV}$, scheduler $\mathfrak{s} \in \mathsf{Scheds}$, and configuration $c' = (_, _, s')$, we have*

$$\left| s' \right| (\mathcal{CB}_{\mathbb{E}}(g, \ell, x)) \geq \sum_{\substack{c=(\ell,t,s)\in \mathsf{Conf}, \\ t\in g}} \mu^{pt}_{c',\mathfrak{s}}(c) \cdot \left| s(x) - s'(x) \right|.$$

By Thm. 17.4.3, $\mathcal{CB}_{\mathbb{E}}(g, \ell, x)$ is indeed an upper bound on the expected change of the value of the variable x after execution of the general transition g with target location ℓ.

Remark 17.4.4. We have seen a similar idea to expected local change bounds in Ch. 15. Here, we discussed the concept of the (worst-case) expected change of an expectation I (see Def. 15.3.8 and 15.7.5, and Thm. 15.3.10 and 15.7.8). By Thm. 17.4.3, expected local change bounds can be seen as a translation of this concept to general result variables of PITSs (see also Lemma B.49.8 in the proof of Thm. 17.4.10).

To obtain *global* bounds from the local bounds $\mathcal{CB}_{\mathbb{E}}(g, \ell, x)$, we construct a *general result variable graph* which encodes the data flow between variables.

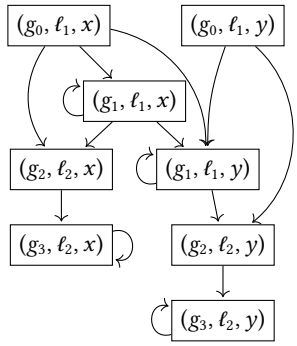


Figure 17.7: General Result Variable Graph for the PITS in Fig. 17.1

Definition 17.4.5 (General Result Variable Graph)**.** Let $\mathrm{pre}(g) = \mathcal{ET}_{\emptyset}(\ell_g)$ be the set of *pre-transitions* of g (see Def. 17.3.4) which lead into g's start location ℓ_g. Moreover, for $\alpha = (g, \ell, x) \in \mathcal{GRV}$ let its *active variables* $\mathrm{actVar}(\alpha)$ consist of all variables occurring in the bound $x + \mathcal{CB}_{\mathbb{E}}(\alpha)$ for α's (local) expected size.

The *general result variable graph* has the set of nodes $\mathcal{GRV}$ and the set of edges $\mathcal{GRVE}$, where

$$\mathcal{GRVE} = \left\{ ((g', \ell', x'), (g, \ell, x)) \;\middle|\; g' \in \mathrm{pre}(g) \wedge \ell' = \ell_g \wedge x' \in \mathrm{actVar}(g, \ell, x) \right\}.$$

Here, we use $\mathrm{actVar}(g, \ell, x) = \{x\}$ if $\mathcal{CB}_{\mathbb{E}}(g, \ell, x) = \infty$.

Example 17.4.6. The general result variable graph for the PITS $\mathbf{T}_{sq}$ in Fig. 17.1 is shown in Fig. 17.7. For $\mathcal{CB}_{\mathbb{E}}$ from Ex. 17.4.2, we have $\mathrm{actVar}(g_1, \ell_1, x) = \{x\}$, as $x + \mathcal{CB}_{\mathbb{E}}(\alpha) = x + \frac{1}{2}$ contains no variable except x. Similarly, $\mathrm{actVar}(g_1, \ell_1, y) = \{x, y\}$, as x and y are contained in $y + \mathcal{CB}_{\mathbb{E}}(g_1, \ell_1, y) = y + x$. For all other $\alpha \in \mathcal{GRV}$, we have $\mathrm{actVar}(_, _, x) = \{x\}$ and $\mathrm{actVar}(_, _, y) = \{y\}$. As $\mathrm{pre}(g_1) = \{g_0, g_1\}$, the graph captures the dependence of (g_1, ℓ_1, x) on (g_0, ℓ_1, x) and (g_1, ℓ_1, x), and of (g_1, ℓ_1, y) on (g_0, ℓ_1, x), (g_0, ℓ_1, y), (g_1, ℓ_1, x), and (g_1, ℓ_1, y). The other edges are obtained in a similar way.

17.4.2 Inferring Expected Size Bounds

We now compute global expected size bounds for the general result variables by considering the SCCs of the general result variable graph separately. As usual, an SCC is a maximal subgraph with a path from each node to every other node. An SCC is *trivial* if it consists of a single node without an edge to itself. We first handle trivial SCCs in Sect. 17.4.2.1 and consider non-trivial SCCs in Sect. 17.4.2.2.

17.4.2.1 Inferring Expected Size Bounds for Trivial SCCs

By Thm. 17.4.3, $x + \mathcal{CB}_{\mathbb{E}}(g, \ell, x)$ is a *local* bound on the expected value of x after applying g once in order to enter ℓ. However, this bound is formulated in terms of the values of the variables immediately before applying g. We now want to compute *global* bounds in terms of the *initial* values of the variables at the start of an execution.

If g is *initial* (i.e., $g \in \mathcal{GT}_0$ since g starts in the initial location ℓ_0), $x + \mathcal{CB}_{\mathbb{E}}(g, \ell, x)$ is already a global bound, as the values of the variables before the application of g are the initial values of the variables at the start of the execution.

Otherwise, the variables y occurring in the local bound $x + \mathcal{CB}_{\mathbb{E}}(g, \ell, x)$ have to be replaced by the values that they can be expected to take in a full run before applying the transition g. Thus, we have to consider all transitions $h \in \mathrm{pre}(g)$ and instantiate every variable y by the maximum of the values that y can be expected to have after applying h. Here, we again over-approximate the maximum by the sum.

If $\mathcal{CB}_{\mathbb{E}}(g, \ell, x)$ is concave (i.e., a linear polynomial, see Lemma 17.3.12), we can instantiate its variables by *expected* size bounds $\mathcal{SB}_{\mathbb{E}}(h, \ell_g, y)$. However, this is unsound if $\mathcal{CB}_{\mathbb{E}}(g, \ell, x)$ is not linear, i.e., not concave (see Counterex. 17.4.9). So in this case, we have to use *non-probabilistic* bounds $\mathcal{SB}(t, y)$ instead.

As in Sect. 17.3, we use an underlying non-probabilistic bound pair $(\mathcal{RB}, \mathcal{SB})$ and start with the expected pair $(\mathcal{RB}_{\mathbb{E}}, \mathcal{SB}_{\mathbb{E}})$ obtained by lifting $(\mathcal{RB}, \mathcal{SB})$ according to Thm. 17.2.10. While Thm. 17.3.8 improves $\mathcal{RB}_{\mathbb{E}}$, we now improve $\mathcal{SB}_{\mathbb{E}}$. Here, the SCCs of the general result variable graph should be treated in topological order. Then one may first improve $\mathcal{SB}_{\mathbb{E}}$ for result variables corresponding to $\mathrm{pre}(g)$, and use that when improving $\mathcal{SB}_{\mathbb{E}}$ for result variables of the form $(g, _, _)$.

Theorem 17.4.7 (Expected Size Bounds for Trivial SCCs)**.** *Let $\mathcal{SB}_{\mathbb{E}}$ be an expected size bound, $\mathcal{SB}$ a (non-probabilistic) size bound, and let $\alpha = (g, \ell, x)$ form a trivial SCC of the general result variable graph. Define the mappings $\mathrm{size}^{\alpha}_{\mathbb{E}}$, $\mathrm{size}^{\alpha} : \mathcal{PV} \to \mathcal{B}$ by*

$$\mathrm{size}^{\alpha}_{\mathbb{E}}(y) = \sum_{h \in \mathrm{pre}(g)} \mathcal{SB}_{\mathbb{E}}(h, \ell_g, y) \quad \text{and} \quad \mathrm{size}^{\alpha}(y) = \sum_{\substack{h \in \mathrm{pre}(g),\\ t=(_,_,_,_,\ell_g) \in h}} \mathcal{SB}(t, y).$$

Then $\mathcal{SB}'_{\mathbb{E}} : \mathcal{GRV} \to \mathcal{B}$ *is an expected size bound, where* $\mathcal{SB}'_{\mathbb{E}}(\beta) = \mathcal{SB}_{\mathbb{E}}(\beta)$ *for* $\beta \neq \alpha$

$$\text{and} \quad \mathcal{SB}'_{\mathbb{E}}(\alpha) = \begin{cases} x + \mathcal{CB}_{\mathbb{E}}(\alpha), & \text{if } g \in \mathcal{GT}_0 \\ \text{size}^{\alpha}_{\mathbb{E}}(x + \mathcal{CB}_{\mathbb{E}}(\alpha)), & \text{if } g \notin \mathcal{GT}_0, \text{ and } \mathcal{CB}_{\mathbb{E}}(\alpha) \text{ is linear} \\ \text{size}^{\alpha}_{\mathbb{E}}(x) + \text{size}^{\alpha}(\mathcal{CB}_{\mathbb{E}}(\alpha)), & \text{if } g \notin \mathcal{GT}_0, \text{ and } \mathcal{CB}_{\mathbb{E}}(\alpha) \text{ is not linear} \end{cases}$$

Example 17.4.8. The general result variable graph in Fig. 17.7 contains 4 trivial SCCs formed by $\alpha_x = (g_0, \ell_1, x)$, $\alpha_y = (g_0, \ell_1, y)$, $\beta_x = (g_2, \ell_2, x)$, and $\beta_y = (g_2, \ell_2, y)$. For all these general result variables, the expected local change bound $\mathcal{CB}_{\mathbb{E}}$ is 0 (see Ex. 17.4.2). Thus, it is linear. Since $g_0 \in \mathcal{GT}_0$, Thm. 17.4.7 yields $\mathcal{SB}'_{\mathbb{E}}(\alpha_x) = x + \mathcal{CB}_{\mathbb{E}}(\alpha_x) = x$ and $\mathcal{SB}'_{\mathbb{E}}(\alpha_y) = y + \mathcal{CB}_{\mathbb{E}}(\alpha_y) = y$.

By treating SCCs in topological order, when handling β_x, β_y, we can assume that we already have $\mathcal{SB}_{\mathbb{E}}(\alpha_x) = x$, $\mathcal{SB}_{\mathbb{E}}(\alpha_y) = y$ and $\mathcal{SB}_{\mathbb{E}}(g_1, \ell_1, x) = 2 \cdot x$, $\mathcal{SB}_{\mathbb{E}}(g_1, \ell_1, y) = 6 \cdot x^2 + y$ (see Fig. 17.5) for the result variables corresponding to $\text{pre}(g_2) = \{g_0, g_1\}$. We will explain in Sect. 17.4.2.2 how to compute such expected size bounds for non-trivial SCCs. Hence, by Thm. 17.4.7 we obtain $\mathcal{SB}'_{\mathbb{E}}(\beta_x) = \text{size}^{\beta_x}_{\mathbb{E}}(x + \mathcal{CB}_{\mathbb{E}}(\beta_x)) = \mathcal{SB}_{\mathbb{E}}(\alpha_x) + \mathcal{SB}_{\mathbb{E}}(g_1, \ell_1, x) = 3 \cdot x$ and $\mathcal{SB}'_{\mathbb{E}}(\beta_y) = \text{size}^{\beta_y}_{\mathbb{E}}(y + \mathcal{CB}_{\mathbb{E}}(\beta_y)) = \mathcal{SB}_{\mathbb{E}}(\alpha_y) + \mathcal{SB}_{\mathbb{E}}(g_1, \ell_1, y) = 6 \cdot x^2 + 2 \cdot y$.

Finally, the following example illustrates the need for linear, i.e., concave, expected local change bounds in Thm. 17.4.7 when substituting expected size bounds.

Counterexample 17.4.9. Consider the PITS $\mathbf{T}_{con}$ in Fig. 17.8. $\mathbf{T}_{con}$ increases x by a random number between 1 and 15 (each with probability $\frac{1}{15}$) before computing its square. So for $\alpha = (g_0, \ell_1, x)$, we have $\mathcal{SB}_{\mathbb{E}}(g_0, \ell_1, x) = x + \mathcal{CB}_{\mathbb{E}}(\alpha) = x + 8$, while $\mathcal{SB}(t_0, x) = x + 15$ is its non-probabilistic counterpart, which is indeed an upper bound on all possible values of x after the update. Now we consider $\beta = (g_1, \ell_2, x)$. Clearly, β forms a trivial SCC in the result variable graph of $\mathbf{T}_{con}$ (in fact, this graph has only trivial SCCs). Depending on whether one uses the expected or the non-probabilistic size bound for g_1's pre-transition g_0, we get $\text{size}^{\beta}_{\mathbb{E}}(x) = \mathcal{SB}_{\mathbb{E}}(g_0, \ell_1, x) = x + 8$ and $\text{size}^{\beta}(x) = \mathcal{SB}(t_0, x) = x + 15$. Note that $\mathcal{CB}_{\mathbb{E}}(\beta) = \lceil x^2 - x \rceil = x^2 + x$, which is clearly not linear and thus not concave. If we were allowed to use the expected size bound for the pre-transition g_0, we would obtain

$$\mathcal{SB}'_{\mathbb{E}}(\beta) = \text{size}^{\beta}_{\mathbb{E}}(x + \mathcal{CB}_{\mathbb{E}}(\beta)) = \text{size}^{\beta}_{\mathbb{E}}(x^2 + 2 \cdot x) = (x+8)^2 + 2 \cdot (x+8) = x^2 + 18 \cdot x + 80.$$

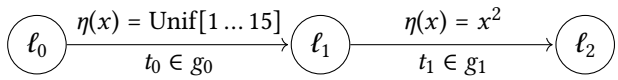


Figure 17.8: PITS $\mathbf{T}_{con}$

However, then $\mathcal{SB}'_{\mathbb{E}}$ would *not* be a correct expected size bound. To see this, consider the initial state s_0 with $s_0(x) = 0$. Here, we have $|s_0|\left(\mathcal{SB}'_{\mathbb{E}}(\beta)\right) = 80$, which is smaller than $\mathbb{E}^{\text{pt}}_{s_0,\mathfrak{s}}(\mathcal{S}(\beta)) = \frac{1}{15} \cdot (1^2 + 2^2 + \dots + 15^2) = 82 + \frac{2}{3}$.

In contrast, applying Thm. 17.4.7 yields

$$\mathcal{SB}'_{\mathbb{E}}(\beta) = \text{size}^{\beta}_{\mathbb{E}}(x) + \text{size}^{\beta}(\mathcal{CB}_{\mathbb{E}}(\beta)) = x + 8 + (x+15)^2 + (x+15) = x^2 + 32 \cdot x + 248.$$

This is a correct expected size bound and indeed, we now have $|s_0|\left(\mathcal{SB}'_{\mathbb{E}}(\beta)\right) = 248 \geq 82 + \frac{2}{3} = \mathbb{E}^{\text{pt}}_{s_0,\mathfrak{s}}(\mathcal{S}(\beta))$.

17.4.2.2 Inferring Expected Size Bounds for Non-Trivial SCCs

Now we handle non-trivial SCCs C of the general result variable graph. An upper bound for the expected size of a variable x when entering C is obtained from $\mathcal{SB}_{\mathbb{E}}(\beta)$ for all general result variables $\beta = (_, _, x)$ which have an edge to C.

To turn $\mathcal{CB}_{\mathbb{E}}(g, \ell, x)$ into a global bound, as in Thm. 17.4.7 its variables y have to be instantiated by the values $\text{size}^{(g,\ell,x)}(y)$ that they can take in a full run before applying a transition from g. Thus, $\text{size}^{(g,\ell,x)}(\mathcal{CB}_{\mathbb{E}}(g, \ell, x))$ is a global bound on the expected change resulting from *one* application of g. To obtain an upper bound for the whole SCC C, we add up these global bounds for all $(g, _, x) \in C$ and take into account how often the general transitions in the SCC are expected to be executed, i.e., we multiply with their expected runtime bound $\mathcal{RB}_{\mathbb{E}}(g)$. So while in Thm. 17.3.8 we improve $\mathcal{RB}_{\mathbb{E}}$ using expected size bounds for previous transitions, we now improve $\mathcal{SB}_{\mathbb{E}}(C)$ using expected runtime bounds for the transitions in C and expected size bounds for previous transitions.

Theorem 17.4.10 (Expected Size Bounds for Non-Trivial SCCs)**.** *Let $(\mathcal{RB}_{\mathbb{E}}, \mathcal{SB}_{\mathbb{E}})$ be an expected bound pair, $(\mathcal{RB}, \mathcal{SB})$ a (non-probabilistic) bound pair, and let $C \subseteq \mathcal{GRV}$ form a non-trivial SCC of the general result variable graph where $\mathcal{GT}_C = \{g \in \mathcal{GT} \mid$*

$(g, _, _) \in C\}$. *Then* $\mathcal{SB}'_{\mathbb{E}} : \mathcal{GRV} \to \mathcal{B}$ *is an expected size bound, where*

$$\mathcal{SB}'_{\mathbb{E}}(\alpha) = \begin{cases} \sum_{(\beta,\alpha)\in\mathcal{GRVE},\beta\notin C,\alpha\in C,\beta=(_,_,x)} \mathcal{SB}_{\mathbb{E}}(\beta) \quad + \\ \sum_{g\in\mathcal{GT}_C} \mathcal{RB}_{\mathbb{E}}(g) \cdot \left(\sum_{\alpha'=(g,_,x)\in C} \mathrm{size}^{\alpha'}\left(\mathcal{CB}_{\mathbb{E}}\left(\alpha'\right)\right) \right), & \text{if } \alpha = (_,_,x) \in C \\ \mathcal{SB}_{\mathbb{E}}(\alpha), & \text{otherwise} \end{cases}$$

Here, we really have to use the *non-probabilistic* size bound $\mathrm{size}^{\alpha'}$ instead of $\mathrm{size}^{\alpha'}_{\mathbb{E}}$ even if $\mathcal{CB}_{\mathbb{E}}(\alpha')$ is linear, i.e., concave. Otherwise, we would multiply the expected values of two random variables which are not independent.

Example 17.4.11. The general result variable graph in Fig. 17.7 contains 4 non-trivial SCCs formed by $\alpha'_x = (g_1, \ell_1, x)$, $\alpha'_y = (g_1, \ell_1, y)$, $\beta'_x = (g_3, \ell_2, x)$, and $\beta'_y = (g_3, \ell_2, y)$. By the results on $\mathcal{RB}_{\mathbb{E}}$, $\mathcal{CB}_{\mathbb{E}}$, $\mathcal{SB}_{\mathbb{E}}$, and $\mathcal{SB}$ from Ex. 17.3.9, 17.4.2, and 17.4.8 and Fig. 17.3, Thm. 17.4.10 yields the expected size bound in Fig. 17.5:

$$\begin{array}{rcll} \mathcal{SB}'_{\mathbb{E}}(\alpha'_x) & = & \mathcal{SB}_{\mathbb{E}}(\alpha_x) + \mathcal{RB}_{\mathbb{E}}(g_1) \cdot \mathrm{size}^{\alpha'_x}(\mathcal{CB}_{\mathbb{E}}(\alpha'_x)) & = & x + 2 \cdot x \cdot \frac{1}{2} = 2 \cdot x \\ \mathcal{SB}'_{\mathbb{E}}(\alpha'_y) & = & \mathcal{SB}_{\mathbb{E}}(\alpha_y) + \mathcal{RB}_{\mathbb{E}}(g_1) \cdot \mathrm{size}^{\alpha'_y}(\mathcal{CB}_{\mathbb{E}}(\alpha'_y)) & = & y + 2 \cdot x \cdot \mathrm{size}^{\alpha'_y}(x) \\ & = & y + 2 \cdot x \cdot \sum_{i\in\{0,1,2\}} \mathcal{SB}(t_i, x) & = & 6 \cdot x^2 + y \\ \mathcal{SB}'_{\mathbb{E}}(\beta'_x) & = & \mathcal{SB}_{\mathbb{E}}(\beta_x) + \mathcal{RB}_{\mathbb{E}}(g_3) \cdot \mathrm{size}^{\beta'_x}(\mathcal{CB}_{\mathbb{E}}(\beta'_x)) & & \\ & = & 3 \cdot x + (6 \cdot x^2 + 2 \cdot y) \cdot 0 & = & 3 \cdot x \\ \mathcal{SB}'_{\mathbb{E}}(\beta'_y) & = & \mathcal{SB}_{\mathbb{E}}(\beta_y) + \mathcal{RB}_{\mathbb{E}}(g_3) \cdot \mathrm{size}^{\beta'_y}(\mathcal{CB}_{\mathbb{E}}(\beta'_y)) & & \\ & = & 6 \cdot x^2 + 2 \cdot y + (6 \cdot x^2 + 2 \cdot y) \cdot 1 & = & 12 \cdot x^2 + 4 \cdot y \end{array}$$

17.5 Evaluation

In this section, we evaluate the approach for inferring upper bounds on the expected runtime of a PITS which we have presented in Sect. 17.3 and 17.4 by comparing with existing tools. First of all, we give details on the implementation of this approach in Sect. 17.5.1. In Sect. 17.5.2, we then present our experimental evaluation.

17.5.1 Implementation

We implemented our analysis in our tool KoAT [Bro+16; KoAT]. As already mentioned in Sect. 16.4.3, KoAT is an open-source tool written in OCaml. The implementation of the approach presented in Sect. 17.3 and 17.4 can also be downloaded as a Docker image and accessed via a web interface [MHG21a]. As explained in Sect. 16.4.3, KoAT also contains the technique for computing exact expected runtimes of *cpa*-loops presented in Ch. 16 and a user can determine via the input if she would like to compute the exact expected runtime of a *cpa*-loop or if she would like to analyze a PITS.

Algorithm 17.5.1: Overall Approach to Infer Bounds on Expected Runtimes

Input: PITS $\mathbf{T} = (\mathcal{PV}, \mathcal{L}, \mathcal{GT}, \ell_0)$
Result: upper bound on the expected runtime of $\mathbf{T}$
1 preprocess the PITS
2 $(\mathcal{RB}, \mathcal{SB}) \leftarrow$ perform non-probabilistic analysis using [Bro+16]
3 $(\mathcal{RB}_{\mathbb{E}}, \mathcal{SB}_{\mathbb{E}}) \leftarrow$ lift $(\mathcal{RB}, \mathcal{SB})$ to an expected bound pair with Thm. 17.2.10
4 sort the SCCs of the general result variable graph in topological order
5 **repeat**
6 **foreach** *SCC C of the general result variable graph* **do**
7 **if** $C = \{\alpha\}$ *is trivial* **then** $\mathcal{SB}'_{\mathbb{E}} \leftarrow$ improve $\mathcal{SB}_{\mathbb{E}}$ for C by Thm. 17.4.7
8 **else** $\mathcal{SB}'_{\mathbb{E}} \leftarrow$ improve $\mathcal{SB}_{\mathbb{E}}$ for C by Thm. 17.4.10
9 **for *all*** $\alpha \in C$ **do** $\mathcal{SB}_{\mathbb{E}}(\alpha) \leftarrow \min\{\mathcal{SB}_{\mathbb{E}}(\alpha), \mathcal{SB}'_{\mathbb{E}}(\alpha)\}$
10 **foreach** *general transition* $g \in \mathcal{GT}$ **do**
11 $\mathcal{RB}'_{\mathbb{E}} \leftarrow$ improve $\mathcal{RB}_{\mathbb{E}}$ for $\mathcal{GT}_{>} = \{g\}$ by Thm. 17.3.8
12 $\mathcal{RB}_{\mathbb{E}}(g) \leftarrow \min\{\mathcal{RB}_{\mathbb{E}}(g), \mathcal{RB}'_{\mathbb{E}}(g)\}$
13 **until** *no bound is improved anymore*
Output: $\sum_{g \in \mathcal{GT}} \mathcal{RB}_{\mathbb{E}}(g)$

Given a PITS, the analysis proceeds as in Algorithm 17.5.1. The preprocessing in Line 1 adds invariants to conditions (using APRON [JM09] to generate (non-probabilistic) invariants), unfolds transitions [BD77], and removes unreachable locations, transitions with probability 0, and transitions with unsatisfiable conditions (using Z3 [MB08]). Here, an invariant does *not* refer to an invariant in the sense of expectation transformers (see Def. 14.3.13) but to a constraint $c \in \mathcal{C}$: we say c is an *invariant* for $g \in \mathcal{GT}$ if in every finite path $f = \cdots(_, t', s')(_, t, _)$ where $t \in g$ that is admissible for some initial state s_0 and scheduler $\mathfrak{s}$ we have $s'(c) = \text{true}$, i.e., c holds whenever g is executed. Thus, w.l.o.g. we can replace the condition φ_g of g by $\varphi_g \wedge c$.

We start by a non-probabilistic analysis and lift the resulting bounds to an initial

expected bound pair (Lines 2 and 3). Afterwards, we first try to improve the expected size bounds using Thm. 17.4.7 and 17.4.10. Here, we process the SCCs of the general result variable graph in topological order, see Line 4. Then, we attempt to improve the expected runtime bounds using Thm. 17.3.8 (if we find a PLRF using Z3). To determine the "minimum" of the previous and the new bound, we use a heuristic which compares polynomial bounds by their degree. While we over-approximated the maximum of expressions by their sum to ease readability in this chapter, KoAT also uses bounds containing "min" and "max" to increase precision.

This alternating modular computation of expected size and runtime bounds is repeated so that one can benefit from improved expected runtime bounds when computing expected size bounds and vice versa. We abort this improvement of expected bounds in Algorithm 17.5.1 if they are all finite (or when reaching a timeout).

17.5.2 Experiments

To assess the power of our approach, we performed an experimental evaluation of our implementation in KoAT. To the best of our knowledge, the only available tools for the inference of upper bounds on the expected runtimes of probabilistic programs are [AMS20; NCH18; Wan+19; WKH20]. The tool of [WKH20] deals with data types and higher-order functions in probabilistic ML programs and does not support programs whose complexity depends on (possibly negative) integers (see [RaML]). Furthermore, the tool of [Moo+21] focuses on proving or refuting (P)AST of probabilistic programs for so-called *Prob-solvable* loops (see Ch. 18), which do not allow for nested or sequential loops or non-determinism. So both [WKH20] and [Moo+21] are orthogonal to our work. We discuss [AMS20; NCH18; Wan+19] below.

We did not compare with the tool of [Wan+19] since [Wan+19] expects the program to be annotated with already computed assertions. But for many of the examples in our experiments, the tool [SSM04] for generating these assertions used by [Wan+19] did not find assertions strong enough to enable a meaningful analysis (and we could not apply APRON [JM09] due to the different semantics of invariants and these assertions).

Instead, we compare KoAT with the tools Absynth [NCH18] and eco-imp [AMS20] which are both based on a *backward-reasoning* approach similar to the ert- and wp-transformer. We ran the tools on all 39 examples from Absynth's evaluation in [NCH18] (except `recursive`, which contains non-tail-recursion and thus cannot be encoded as a PITS), and on the 8 additional examples from the artifact of [NCH18]. Moreover, our collection has 29 additional benchmarks: the 3 examples presented in this chapter, i.e., Fig. 17.1, 17.6, and 17.8 (here, we used a slightly different presentation of the PITS in Fig. 17.6 than in the experiments which does not change the results

$$
\begin{array}{l}
b \leftarrow 0 \,\mathring{;}\\
\textbf{while } (x > 0 \wedge b = 0) \textbf{ do } \{\\
\quad \left\{ \begin{array}{l} y \leftarrow y + x \,\mathring{;}\\ \{\, x \leftarrow x - 1 \,\}\ \left[\tfrac{1}{2}\right]\ \{\, \textbf{skip} \,\} \end{array} \right\} \square\ \{\, b \leftarrow 1 \,\}\\
\}\\
\textbf{while } (y > 0) \textbf{ do } \{\\
\quad y \leftarrow y - 1\\
\}
\end{array}
$$

Figure 17.9: Probabilistic Program Equivalent to Fig. 17.1

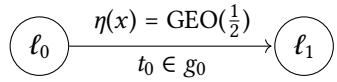
but increases readability), and 11 further examples that illustrate different aspects of PITSs, 5 PITSs based on examples from [NCH18] where we removed assumptions, and 10 PITSs based on benchmarks from the *TPDB* [TPDB] where some transitions were enriched with probabilistic behavior. The *TPDB* is a collection of typical benchmarks used in the annual *Termination and Complexity Competition* [Gie+19b].

Absynth and eco-imp both expect probabilistic programs as inputs. Here, they use a slight generalization of pGCL as a syntax. This includes the possibility to specify a "reward" for a statement (which defaults to zero if not explicitly stated). Thus, both tools do not compute the expected runtime via Table 14.3 but an expected total reward (see Def. 14.2.10) by allowing to change the reward of each statement in Table 14.3. However, they only support finite distributions. We already illustrated how to transform probabilistic programs into PITSs in Rem. 17.1.14. Let us now briefly explain how we transformed PITSs into the probabilistic programs accepted as input by Absynth and eco-imp.

Fig. 17.9 corresponds to the PITS $\mathbf{T}_{sq}$ in Fig. 17.1. When formulating it as a PITS, the general transition g_0 is required since PITSs are not allowed to have a loop containing the initial location ℓ_0. As this restriction has no equivalent in pGCL-programs, the general transition g_0 is missing in Fig. 17.9. General transition g_1 is encoded as a loop. The probabilistic branching between t_1 and t_2 is encoded as a probabilistic choice. Here, we first increase y by the current value of x to access the value of x from the beginning of the iteration (as said before, the updates in a transition are done in parallel, whereas in probabilistic programs we can only model sequential updates). Note that if the condition $x > 0$ of general transition g_1 is satisfied, $\mathbf{T}_{sq}$ chooses between g_1 and g_2 non-deterministically. Thus, we encode this non-deterministic branching via a non-deterministic choice and a fresh variable b (for break), i.e., even

$y \longleftarrow 1$;
while $(y > 0)$ **do {**
  $x \longleftarrow x + y$;
  $\{\, y \longleftarrow 0 \,\} \left[\frac{1}{2}\right] \{\, y \longleftarrow y + 1 \,\}$
}

(a) In the Input of Absynth and eco-imp

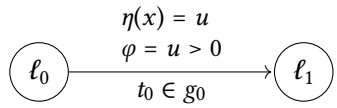


(b) In a PITS

Figure 17.10: Increasing Variable x by a Value Drawn from a Geometric Distribution With Parameter $\frac{1}{2}$

$x \longleftarrow 0$;
$y \longleftarrow 1$;
while $(y > 0)$ **do {**
  $x \longleftarrow x + 1$;
  $\{\, y \longleftarrow 0 \,\} \,\square\, \{$ **skip** $\}$
}

(a) In the Input of Absynth and eco-imp



(b) In a PITS

Figure 17.11: Setting a Variable to an Arbitrary Positive Integer

if $x > 0$, the loop can be left non-deterministically. However, Absynth allows for a statement `break` so that we can avoid the variable b in its input, i.e., instead of $x > 0 \wedge b = 0$ the condition is $x > 0$ and the assignment $b \longleftarrow 1$ is replaced by `break`. Still, eco-imp does not provide such a command and receives Fig. 17.9 as an input.

Both Absynth and eco-imp support the same *finite* distributions as KoAT does but do not support infinite distributions. However, to model the geometric distribution, which is infinite, we proceed as illustrated in Fig. 17.10 and assign the reward 0 to this construct so that it does not contribute to the overall expected reward computed by Absynth and eco-imp.

However, in Sect. 17.1.3 we discussed that it is in general impossible to encode infinite non-deterministic sampling in probabilistic programs. Still, in the input format of Absynth and eco-imp we translate infinite non-deterministic sampling as shown in

Bound	KoAT	Absynth	eco-imp
$\mathcal{O}(1)$	6	6	6
$\mathcal{O}(n)$	32	32	29
$\mathcal{O}(n^2)$	3	8	9
$\mathcal{O}(n^{>2})$	0	0	0
EXP	0	0	0
∞	5	0	2
TO	0	0	0

(a) Results on Benchmarks From [NCH18]

Bound	KoAT	Absynth	eco-imp
$\mathcal{O}(1)$	2	1	2
$\mathcal{O}(n)$	10	3	6
$\mathcal{O}(n^2)$	12	1	6
$\mathcal{O}(n^{>2})$	2	0	0
EXP	1	0	0
∞	2	15	12
TO	0	9	3

(b) Results on our New Benchmarks

Figure 17.12: Experimental Results on Different Benchmarks

Fig. 17.11 and apply reward 0 to this construct (even if the loop does not terminate). In this way, this construct does not contribute to the overall expected reward computed by Absynth and eco-imp and the underlying semantics of both Absynth and eco-imp treat the value of x as "arbitrarily large" which is sound for our examples.

To keep the probabilistic programs in Absynth's input format readable for the larger examples from the TPDB, we made use of tail-recursion. We represent locations as functions and transitions as function calls. In this way, the input of Absynth is as close as possible to the PITS which KoAT receives as input (Absynth allows for function calls in probabilistic programs and thus for recursion). For eco-imp, we used loops instead since eco-imp does not support recursion.

We ran the experiments on an iMac with an Intel i5-2500S CPU and 12 GB of RAM under macOS Sierra for Absynth and NixOS 20.03 for KoAT and eco-imp. A timeout of 5 minutes per example was applied for all tools. The average runtime of successful runs was 4.26 s for KoAT, 3.53 s for Absynth, and just 0.93 s for eco-imp. Fig. 17.12(a) and 17.12(b) show the generated asymptotic bounds, where n is the maximal absolute value of the program variables at the start of an execution. Here, "∞" indicates that no finite time bound could be computed and "TO" means "timeout". The detailed asymptotic results of all tools on all examples can be found at [MHG21a].

Absynth and eco-imp slightly outperform KoAT on the examples from Absynth's collection, while KoAT is considerably stronger than both tools on the additional

benchmarks. In particular, Absynth and eco-imp outperform our approach on examples with nested probabilistic loops. While our modular approach can analyze inner loops separately when searching for probabilistic ranking functions, Thm. 17.3.8 then requires *non-probabilistic* runtime bounds for all transitions entering the inner loop. But these bounds may be infinite if the outer loop has probabilistic behavior itself. Moreover, in contrast to our work and [AMS20], the approach of [NCH18] does not require weakly monotonic bounds.

Nonetheless, KoAT is superior to Absynth and eco-imp on large examples with many loops where only a few transitions have probabilistic behavior (this might correspond to the typical application of randomization in practical programming). Here, we benefit from the modularity of our approach which treats loops independently and combines their bounds afterwards. Absynth and eco-imp also fail for our example in Fig. 17.1 and 17.9, while KoAT infers a quadratic bound. Hence, the tools have particular strengths on orthogonal kinds of examples. Still, our tool KoAT succeeded on 91% of all examples, while Absynth and eco-imp only inferred finite bounds for 68% resp. 77% of the examples.

Chapter 18

Related Work

In this chapter, we discuss the work related to the results of this part. We first discuss work related to Ch. 14 and 15 in Sect. 18.1. In Sect. 18.2, we then present an overview of existing results connected to our contributions in Ch. 16. Finally, we consider the work related to Ch. 17 in Sect. 18.3.

18.1 Work Related to Ch. 14 and 15

The weakest preexpectation calculus goes back to the predicate transformer calculus by [Dij75; Dij76], which provides an important tool for qualitative formal reasoning about non-probabilistic programs. The probabilistic and quantitative analog to predicate transformers for non-probabilistic programs are *expectation transformers* for probabilistic programs. Weakest-preexpectation-style reasoning was first studied in seminal work on probabilistic propositional dynamic logic (PPDL) by [Koz81; Koz85]. Its box- and diamond-modalities provide probabilistic versions of Dijkstra's weakest (liberal) preconditions. Amongst others, [Heh11; Jon90; KK17; MM05; MMS96] have furthered this line of research, e.g., by considering non-determinism, proof rules for bounding preexpectations in the presence of loops, and unbounded mixed-sign expectations. Work towards automation of weakest preexpectation reasoning was carried out, amongst others, by [Che+15; Coc14; Fen+17; Kat+10]. Abstract interpretation of probabilistic programs was studied in this setting by [Mon05]. Additionally, [Bat+21] presents a set of expectations on which wp-reasoning is complete, i.e., if the postexpectation is from this set, the weakest preexpectation is as well.

Rules for bounding weakest preexpectations were considered from very early on. Already [Koz85] provides an induction rule for verifying upper bounds. Pioneering

work on lower bounds by means of limits of sequences was carried out by [Jon90] and later reconsidered by [AP09]. Proof rules that do not make use of limits were studied by [Mor96] and later more extensively in [MM05]. An orthogonal approach to lower bounds by means of bounded model checking was explored by [Jan+16].

Apart from reasoning about expected values of random variables at termination of simple pGCL programs, more advanced expectation-based calculi were invented. For instance, [MM99] uses expectation transformers to reason about temporal logic. More recently, [Olm+18] studies expectation transformers for probabilistic programs with *conditioning*. [Kam+16; Kam+18; KKM20; Olm+16] introduce expectation-based calculi to reason about expected runtimes of probabilistic programs. [Bat+19] presents a quantitative separation logic together with a weakest preexpectation calculus for verifying probabilistic programs with pointer-access to dynamic memory. [Agu+21] introduces a preexpectation calculus for reasoning about probabilistic sensitivity.

In all of the above works, the rules for lower bounds rely on finding limits of sequences as well as the sequences themselves. In particular, the proof of the (exact) expected runtime of the coupon collector's problem by [Kam+16] requires a fairly complicated sequence, whereas our invariant in Sect. 15.6.1 was conceptually simple and thus more informative for a human.

Probabilistic program analysis using martingales was pioneered by [CS13]. Our proof rules in Ch. 15 rely on the notions of *uniform integrability* and *conditional difference boundedness* as well as the *Optional Stopping Theorem*. Previous works have also used these notions. [Bar+16] focuses on synthesizing *exact* martingale expressions. [FH15] develops a type system for uniform integrability in order to prove (P)AST of probabilistic programs and gives upper bounds on the expected runtime. [FC19] gives upper and lower bounds on expected runtimes. [Wan+19] infers upper and lower bounds on the expected cost of probabilistic programs. [CFG16; CNZ17] study almost-sure termination of probabilistic programs. Moreover, [WHR21] presents an extension of the Optional Stopping Theorem which involves reasoning about central moments of the expected runtime to infer tail-bounds.

Besides [Bar+16], the latter works analyze the termination behavior of a probabilistic program, whereas we focus on *general* expected values, e.g., of program variables. Furthermore, we do not only *make use* of uniform integrability and/or conditional difference boundedness of some auxiliary stochastic process in order to prove soundness of our proof rules but establish tight connections between expectation-based reasoning via induction and martingale-based reasoning. Additionally, the extension of the Optional Stopping Theorem presented in [WHR21] is orthogonal to ours as discussed in Ex. 15.3.14.

Work on probabilistic program analysis by specialized kinds of martingales includes [ACN18; Bar+16; BG05; BG06; CFG16; CFN20; Cha+18; CNZ17; CS13; CS14;

FC19; FH15; HFC18; Hua+19; Wan+19]. For instance, regarding expected runtimes of probabilistic (and possibly non-deterministic) programs, [FC19] constructs *difference bounded* (as opposed to *conditionally* difference bounded, which is a strictly weaker requirement, see Ex. 15.3.14) supermartingales which have to correspond to the *exact* asymptotic expected runtimes. In contrast, our rule allows for reasoning about *strict* lower bounds. Moreover, [CNZ17] uses difference bounded repulsing supermartingales to refute (P)AST, which we have tackled exemplary in Sect. 15.6.2. Furthermore, we used [CNZ17] to derive the decision procedure for (P)AST of *cpa*-loops in Ch. 16.

Additionally, [Wan+19] uses an extension of difference boundedness, where the bound does not need to be a constant but can be a polynomial for reasoning about lower and upper bounds on the expected cost of probabilistic programs. This extension is orthogonal to ours as discussed in Ex. 15.3.14.

Finally, [LFR21] discusses upper and lower bounds on probabilities of termination and expected runtimes in the probabilistic λ-calculus via type systems. In particular, this work focuses on the presence of higher-order functions, while we have considered first-order probabilistic programs.

18.2 Work Related to Ch. 16

As already mentioned in Sect. 18.1, many techniques to analyze (P)AST have been developed, e.g., [ACN18; ALY20; AMS20; BG05; BG06; CFG16; CFN20; Cha+18; CNZ17; CS13; CS14; FC19; HFC18; Hua+19; KKM20; McI+18; Moo+21; NCH18; Wan+19], which mostly rely on martingales or invariants for expectation transformers. Indeed, several of these works (see, e.g., [ALY20; BG05; FC19; FH15]) present complete criteria for (P)AST, although (P)AST is undecidable. However, the corresponding automation of these techniques is of course incomplete. In [ACN18; Cha+18] it is shown that for affine probabilistic programs, a super-class of our *cpa*-loops, the existence of a (lexicographic) linear ranking supermartingale is decidable. Still, the existence of a (lexicographic) linear ranking supermartingale is sufficient but not necessary for PAST.

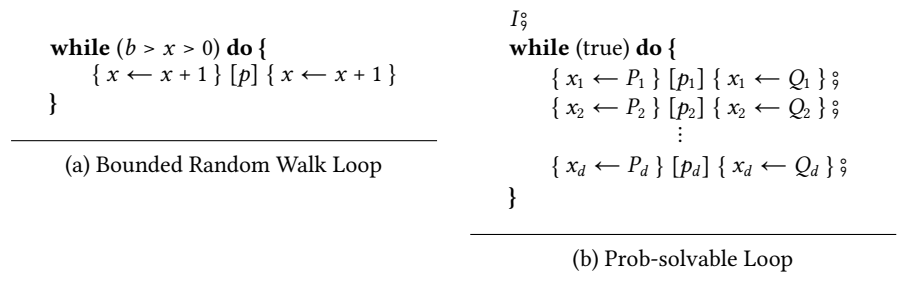
We studied classes of deterministic loops where termination is decidable in Part II and gave a survey on existing decidability results for termination of certain classes of such loops in Ch. 11, e.g., [Bra06; HOW19; Tiw04].

Termination analysis for probabilistic programs is substantially harder than for non-probabilistic ones [KKM19]. Nevertheless, there is some previous work on classes of probabilistic programs where termination is decidable and asymptotic bounds on the expected looping time (or runtime) are computable. For instance, [Brá+13b] considers decidability of (P)AST and bounds on termination probabilities

and expected runtimes for probabilistic pushdown automata, [BBE10; BKK14; Brá+12; Brá+13a; EY09] discuss similar problems for recursive Markov chains, one-counter Markov decision processes, and one-counter stochastic games, and [Brá+19] considers deciding whether the expected runtime is linear for so-called probabilistic vector addition state systems. Additionally, [CFM17] presents an automatic approach for inferring asymptotic upper bounds on the expected looping time by considering uni- and bivariate recurrence equations. While all of these systems are closely related to random walk loops, none of the just mentioned works considers the computation of the *exact* expected looping time (or runtime). Additionally, [KLG20] studies (un)decidability of termination for probabilistic higher-order recursive programs and provides a semi-decision procedure for lower bounding termination probabilities. This is also orthogonal to our work as we study a restricted class of probabilistic loops where termination is indeed decidable.

To the best of our knowledge, Algorithm 16.4.1 presented in Sect. 16.4 is the first which computes a general formula (i.e., a closed form) for the *exact* expected looping time of arbitrary *cpa*-loops. Up to now, such a formula was only known for the very restricted special case of *bounded* simple random walks ([Fel50]), i.e., loops of the form in Fig. 18.1(a) where $1 \geq p \geq 0$ and $b \in \mathbb{Z}$. Note that due to the *two* boundary conditions $x > 0$ and $b > x$, the resulting recurrence equation for the expected looping time of the program only has a *single* solution $f : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ that also satisfies $f(0) = 0$ and $f(b) = 0$. Hence, standard techniques for solving recurrence equations suffice to compute this solution. In contrast, we developed an algorithm to compute the exact expected looping time of *unbounded arbitrary cpa*-loops where the condition of the loop only has *one* boundary condition $x > 0$, i.e., x can grow infinitely large. For that reason, here the challenge is to find an algorithm which computes the *smallest* solution $f : \mathbb{Z} \to \mathbb{R}_{\geq 0}$ of the resulting recurrence equation. We showed that this can be done using the information on the asymptotic bounds of the expected looping time from Sect. 16.2 and 16.3.

Nevertheless, there is also work on inferring exact expected values of expressions in the variables of a probabilistic program, so-called moment-based invariants. These works consider so-called *Prob-solvable* loops and also exploit techniques to compute the solution of a recurrence equation exactly. A Prob-solvable loop is a probabilistic program over the variables $x_1, \dots, x_d$ of the form in Fig. 18.1(b) where I is an initialization phase where every variable is (randomly) assigned a real value, $p_i \in [0, 1]$ are probabilities, and both P_i and Q_i are expressions of the form $a \cdot x_i + g(x_1, \dots, x_{i-1})$ for a real number a and a polynomial $g(x_1, \dots, x_{i-1}) \in \mathbb{R}[x_1, \dots, x_{i-1}]$, i.e., the updates of these loops generalize our *twn*-updates (see Def. 4.1.4). In [BKS19; BKS20b] it is then shown how to compute the expected value of x_i^k (the k-th moment of the variable x_i) after n loop-iterations exactly, where $k, n \in \mathbb{N}$. However, these works



(a) Bounded Random Walk Loop

(b) Prob-solvable Loop

Figure 18.1: Probabilistic Loops Considered in Related Work

are orthogonal to our work in Ch. 16 since they ignore the condition of the loop, i.e., they over-approximate the loop by replacing the condition of the loop by true. Hence, their approach does not consider exact expected looping times. Additionally, these ideas have been applied to analyzing so-called Bayesian networks in [BKS20a]. Again, in Bayesian networks one is interested in expected values of variables, while termination properties are in general not considered.

18.3 Work Related to Ch. 17

As seen in Sect. 18.2, there has been a lot of work on proving PAST and inferring bounds on expected runtimes using supermartingales, e.g., [ACN18; ALY20; BG05; CFG16; CFN20; Cha+18; CNZ17; CS13; FC19; Moo+21; Wan+19]. While these techniques infer one (lexicographic) ranking supermartingale to analyze the complete system, our approach presented in Ch. 17 deals with information flow between different parts of the system and analyzes them separately.

There is also work on modular analysis of AST [ACN18; CFN20; FH15; HFC18; Hua+19]. This differs from our results in Ch. 17 since AST is compositional, in contrast to PAST (see, e.g., [Kam+18; KKM20]).

Ranking supermartingales (i.e., *forward-reasoning*) form a different approach to *backward-reasoning* by expectation transformers (see Sect. 18.1 for work related to expectation transformers). We have studied the connection between expectation transformers for expected outcomes and martingales in Ch. 14 and 15. The connection of expected runtime and size in terms of expectation transformers has been studied in [AMS20; Kam+18; KKM20; Olm+16] (see also Thm. 14.3.12 and Cor. 15.7.9).

To the best of our knowledge, the only available tools for the inference of upper bounds on the expected runtimes of probabilistic programs are [AMS20; NCH18; Wan+19; WKH20]. As already mentioned in Sect. 17.5, [WKH20] deals with data types and higher-order functions in probabilistic ML programs and its tool does not support programs whose complexity depends on (possibly negative) integers. Furthermore, [Moo+21] focuses on proving or refuting (P)AST of *Prob-solvable* loops (they use a slight generalization of the Prob-solvable loops in Fig. 18.1(b) where the condition is a single strict inequation of real polynomials and every variable is updated w.r.t. a distribution over finitely many expressions $a \cdot x_i + g(x_1, \dots, x_{i-1})$). So both [Moo+21; WKH20] are orthogonal to our results in Ch. 17. We have compared our tool KoAT [MHG21a] with the tools of [AMS20; NCH18; Wan+19] in Sect. 17.5.

Chapter 19

Conclusion and Future Work

This part was devoted to the verification of probabilistic programs whose syntax and semantics we have introduced in Ch. 14. As already said in Ch. 13, we interpret probabilistic programs as a model for randomized algorithms. Thus, we focused on verifying properties related to (total) correctness of such algorithms.

We have studied proof rules for lower bounds in probabilistic program verification in Ch. 15. Our rules are *simple* in the sense that the invariants need to be "pushed through the loop semantics" only a *finite* number of times, much like invariants in Floyd-Hoare logic. In contrast, existing rules for lower bounds of unbounded weakest preexpectations required coming up with an infinite *sequence* of invariants, performing induction to prove relative inductiveness of two subsequent invariants, and then — most unpleasantly — finding the limit of this sequence (Thm. 15.1.7). The main results of Ch. 15 are discussed in the following.

We have presented the first *inductive proof rules* (Thm. 15.3.11 (i) and (ii)) for verifying *lower bounds on (possibly unbounded) weakest preexpectations of purely probabilistic loops* using quantitative *invariants*. Our inductive rules are given as an *Optional Stopping Theorem (OST) for weakest preexpectations*. They provide sufficient conditions for the requirement of *uniform integrability* which are much easier to check than uniform integrability in general. Further case studies demonstrating the effectiveness of these rules can be found in [Har+19].

For proving our OST, we resorted to the classical OST from probability theory. However, for most notions that appear in the classical OST, like *uniform integrability* and *conditional difference boundedness*, we were able to find purely expectation-transformer-based counterparts (see Sect. 14.4, 15.2, and 15.3). We thus conjecture that our OST can be proven in purely expectation-theoretic terms, which would most

likely simplify the proof of our OST significantly by avoiding probability theory.

Furthermore, we studied the inductive proof rules for lower bounds on *bounded* weakest preexpectations from [MM05]. Our results gave rise to a generalization of their proof rule to a *sufficient and necessary* criterion for lower bounds (Thm. 15.4.2).

Additionally, we have investigated a measure-theoretical explanation for why verifying upper bounds using domain theoretical Park induction is conceptually simpler (Sect. 15.5). The underlying reason is the well-known *Lemma of Fatou*. This leads us to speculate that Fatou's Lemma could be proved in purely domain theoretical terms, perhaps as an instance of Park induction. A successful attempt at a similar idea is due to [Bar91] who proved that the well-known *Banach Contraction Principle* is a particular instance of the Kleene Fixed Point Theorem.

Moreover, we used the close connection between wp and ert to present the first inductive proof rule for lower bounding expected runtimes (Thm. 15.6.1). As an example to demonstrate the power of this rule, we inferred a non-trivial lower bound on the expected runtime of the famous coupon collector's problem (Sect. 15.6.1). We also showed how to refute PAST with our proof rule in a case-study on the fair-in-the-limit random walk in Sect. 15.6.2.

Finally, we discussed a possible extension of our proof rules for lower bounds to loops involving finite (or bounded) non-determinism in Sect. 15.7. We showed which of our results can be transferred to the case involving non-deterministic choice and presented a proof of soundness of Thm. 15.7.3 based on Conj. 15.7.2. Moreover, we conjecture that a proof for Thm. 15.3.11 in purely expectation-theoretic terms would work for loops involving non-determinism as well and thus would either provide a proof of Conj. 15.7.2 or would make Conj. 15.7.2 obsolete.

Future work of course includes proving Conj. 15.7.2. Moreover, an extension of our proof rules for weakest preexpectation reasoning to recursive programs [Olm+16] and to *mixed-sign* postexpectations is also desirable. For the latter, this will likely yield more appealing proof rules for loops than those provided in [KK17] which currently involve reasoning about sequences. Moreover, we are interested in (partially) automating the synthesis of the quantitative invariants needed in our proof rules.

We discussed termination and complexity for so-called *pa*-loops, a sub-class of probabilistic loops, in Ch. 16. They are based on the connection between the expected looping time of a loop and the smallest solution of its corresponding recurrence equation (Cor. 16.1.8). Here, it is sufficient to analyze the looping time, i.e., the number of iterations of the loop, as the body of the loop is itself loop-free. For our notion of *pa*-loops, if the probability for leaving the loop directly is at least p' for some $p' > 0$, the loop is always PAST and its expected looping time is asymptotically constant (Thm. 16.2.1). Additionally, we have shown that sufficient criteria for proving and refuting (P)AST of one-dimensional *pa*-loops can be obtained by classical results

from martingale theory in Sect. 16.1.1. Based on these results, we have given a very simple decision procedure for AST and PAST of *cpa*-loops (Cor. 16.3.6) and showed that the expected looping time is asymptotically linear if the loop is PAST (Thm. 16.3.10). In Sect. 16.4, we used these insights to present Algorithm 16.4.1 for automatically inferring a closed form for the *exact* expected looping time of *cpa*-loops (i.e., with arbitrarily high precision). We implemented Algorithm 16.4.1 in our tool KoAT [Bro+16; KoAT]. A collection of further examples to demonstrate Algorithm 16.4.1, a web interface, and a Docker image to access our implementation can be found at [Gie+19a].

There are several directions for future work. In Sect. 16.3.1, we reduced *cpa*-loops to random walk loops. In future work, one might consider more advanced reductions in order to extend the class of probabilistic programs where termination and complexity are decidable. Moreover, developing techniques to automatically *over- or under-approximate* the looping time of a loop $\mathbf{L}$ by the looping times of corresponding *cpa*-loops $\mathbf{L}_1$ and $\mathbf{L}_2$ such that $\text{explt } [\![\mathbf{L}_1]\!] (\vec{x}) \preceq \text{explt } [\![\mathbf{L}]\!] (\vec{x}) \preceq \text{explt } [\![\mathbf{L}_2]\!] (\vec{x})$ holds for all $\vec{x} \in \mathbb{Z}^d$ is of high interest. Furthermore, the possibility of a combination of our results from Ch. 16 and Part II should be investigated.

We presented a new modular approach to infer upper bounds on the expected runtimes of probabilistic transition systems in Ch. 17. To this end, we discussed the concepts of non-probabilistic and expected runtime and size (Sect. 17.2) and presented techniques to infer expected runtime and size bounds on parts of the system (Sect. 17.3 and 17.4). These bounds are computed in an alternating fashion and then combined to an overall expected runtime bound. We implemented this approach in our tool KoAT [Bro+16; KoAT] as well. Thus, KoAT does not only contain an implementation of Algorithm 16.4.1 but also an implementation of our modular approach for analyzing probabilistic transition systems. The user can then choose which of both implementations she would like to invoke. Note that Algorithm 16.4.1 is complete, i.e., it always computes the exact expected looping time for *cpa*-loops, whereas the approach presented in Ch. 17 is incomplete. Further details on the implementation of KoAT and its source code can be found at [KoAT].

In our experimental evaluation in Sect. 17.5, the implementation of the approach from Sect. 17.3 and 17.4 in KoAT succeeded on 91% of all examples, while the main other related tools (Absynth and eco-imp) only inferred finite bounds for 68% resp. 77% of the examples. All benchmarks from our experiments as well as a web interface and a Docker image to access our implementation in KoAT can be found at [MHG21a].

In future work, it would be interesting to consider a modular combination of the underlying approaches of KoAT, Absynth, and eco-imp, i.e., a combination of both backward and forward reasoning. In this way, the strengths of all tools could be

combined to an even more powerful fully automatic inference of upper bounds on the expected runtime of probabilistic programs.

Part IV

Postface

Chapter 20

Outlook

At the end of this thesis, we briefly recapitulate our contributions. While we have already discussed possible directions for future work on the particular results of Parts II and III in each part separately (see Ch. 12 and 19), we now give an outlook in a more general manner.

In this thesis, we have studied verification of both non-probabilistic as well as probabilistic programs with a special focus on termination and runtime complexity. Here, we investigated the inference of both upper and lower bounds on worst-case outcomes and runtimes, possible automation, and completeness.

Part II was devoted to so-called *twn*-loops over a ring $\mathcal{S} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}_A, \mathbb{R}\}$, i.e., simple loops whose body consists of a single update of the vector of variables by polynomials (satisfying some side conditions). We showed that for these loops termination is decidable over $\mathcal{S} \in \{\mathbb{R}_A, \mathbb{R}\}$ and non-termination is semi-decidable over $\mathcal{S} \in \{\mathbb{Z}, \mathbb{Q}\}$. Moreover, we proved decidability of the halting problem of *twn*-loops over any ring $\mathcal{S} \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}_A, \mathbb{R}\}$. Additionally, we have proven that upper bounds on the runtime on terminating inputs of such a loop over $\mathcal{S} = \mathbb{Z}$ can *always* be computed.

We have also discussed transformability of arbitrary polynomial loops into *twn*-form. These results were crucial for our analysis of the complexity of deciding termination for certain types of polynomial loops.

In Part III, we have studied probabilistic programs. For probabilistic loops, we have connected their semantics based on expectation transformers to stochastic processes. In this way, we obtained the first inductive proof rules for inferring lower bounds on expected outcomes and the expected runtime of a probabilistic loop. Moreover, we characterized a class of probabilistic loops, so-called *cpa*-loops, where probabilistic termination is decidable, both lower and upper bounds on the expected runtime can

be inferred easily, and the expected runtime can always be computed exactly.

Finally, we have studied so-called probabilistic transition systems. Here, we presented an approach which uses non-probabilistic runtime and size bounds of such a system to compute both expected runtime and expected size bounds in an alternating fashion. In our experimental evaluation, the implementation of this approach outperformed existing tools.

Concerning the main issues for future work, none of the procedures in Part II has been implemented yet. This should be done with a focus on incorporating them into existing automatic termination and runtime complexity analyzers such as AProVE [Gie+17] or KoAT [Bro+16; KoAT]. In this way, our results may contribute to the automatic termination and runtime analysis of larger programs.

While we developed inductive proof rules for lower bounds on expected outcomes and expected runtimes for probabilistic programs in Part III, this is also not yet automated. Therefore, the processors to simplify a deterministic program when computing lower bounds on its runtime from [Fro+20], which are crucial for the performance of the approach in [Fro+20], have to be adapted to probabilistic concepts as well to ensure a good performance of our results on large programs.

As already mentioned, *cpa*-loops are not a generalization of *twn*-loops, e.g., a "non-probabilistic" *cpa*-loop just allows for adding constants to variables. Thus, the possibility of generalizing our results in Part II to classes of probabilistic loops such as so-called Prob-solvable loops (see, e.g., [BKS19; BKS20a; BKS20b; Moo+21]) should be investigated.

Finally, to use our results for analyzing real-world systems, transformations of general programming languages into probabilistic programs or probabilistic transition systems have to be developed, i.e., one should try to generalize the transformations for LLVM [Hen+18] or Java [FG17] into classical (integer) transition systems to also include probabilistic constructs. To the best of our knowledge, so far only results on the complexity analysis of (certain) randomized ML programs [WKH20] exist.

Glossary

AB Concatenations of elements from A and B 149
A^n Sequences of length n with entries from A 149
A^* Finite sequences with entries from A 149
A^+ Finite non-empty sequences with entries from A 149
A^ω Infinite sequences with entries from A 149
$X_n^{f,I}$ Elements of $\mathbf{X}^{f,I}$ 181
$X_{T^\varphi}^f$ Canonical stopped process for $f \in \mathbb{F}$ 193
$X_{\triangleright k}$ Relation in set index 28
$[\cdot/\cdot]$ Substitution 22
$\mathcal{B}$ Set of bounds 258
Ω^{loop} Set of runs for probabilistic loop 176
Σ States $\{s \mid s : \mathcal{V} \to \mathbb{Z}\}$ 146
$\Sigma_{\neg\varphi}$ States for probabilistic loop violating condition φ 174
Σ_φ States for probabilistic loop satisfying condition φ 174
$Tr_\eta(\varphi, \vec{u})$ Transformation of $(\varphi, \vec{u})$ via $\eta \in \mathrm{Aut}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$ 71
$\chi_{\mathbf{L}}$ Characteristic polynomial of random walk loop $\mathbf{L}$ 248
$\mathrm{depdeg}_{\vec{u}}$ Dependency degree w.r.t. $\vec{u}$ 56
$\mathrm{Th}_\exists(\cdot)$ Existential Fragment of First Order Theory 25
ℓ_0 Distinguished initial location of a PITS 260
T^φ Looping time of probabilistic loop with condition φ 178
ΔI Expected change of $I \in \mathbb{F}$ 199
$\mathcal{H}_m$ m-th harmonic number 214
$\mathfrak{s}$ Scheduler for an MDP, a pGCL-program, or a PITS 154
$\mu_{\mathbf{L}}$ Drift of pa-loop $\mathbf{L}$ 234

$\overline{\mathbb{N}}$ Closure of natural numbers $\mathbb{N} \cup \{\omega\}$ 178
$\overline{\mathbb{R}}_{\geq 0}$ Closure of non-negative reals $\{r \mid r \in [0,\infty) \vee r = \infty\}$ 156
$\exists\mathbb{R}$ Decision problems reducible in polynomial time to $\mathrm{Th}_{\exists}(\mathbb{R})$ 88
$\forall\mathbb{R}$ Co-$\exists\mathbb{R}$ 88
$\mathbb{E}(X \mid \mathcal{G})$ Conditional expected value of X w.r.t. $\mathcal{G}$ 182
$\mathbb{E}^{\mathrm{pt}}_{\bullet,\bullet}$ Expected value operator for $\mathbb{P}^{\mathrm{pt}}_{\bullet,\bullet}$ 265
$\mathbb{E}^{\mathrm{loop}}_{\bullet,\bullet}$ Expected value operator for $\mathbb{P}^{\mathrm{loop}}_{\bullet,\bullet}$ 178
$\mathbb{P}^{\mathrm{pt}}_{\bullet,\bullet}$ Probability measures induced by PITS 265
$\mathbb{P}^{\mathrm{loop}}_{\bullet,\bullet}$ Probability measures induced by probabilistic loop 177
$p^{\mathrm{pt}}_{\bullet,\bullet}$ Distributions on Conf^+ induced by PITS 264
$p^{\mathrm{loop}}_{\bullet,\bullet}$ Distributions on Σ^+ induced by probabilistic loop 177
$\mu^{\mathrm{pt}}_{\bullet,\bullet}$ Distributions on Conf induced by PITS 263
$\mu^{\mathbf{P}}_{\bullet,\bullet}$ Distribution induced by $\mathbf{P}$ on Σ 157
explt $[\![\mathbf{L}]\!]$ Expected looping time of *pa*-loop $\mathbf{L}$ 229
exprt $[\![\mathbf{P}]\!]$ Expected runtime of $\mathbf{P}$ 159
$\mathrm{size}^{\alpha}(x)$ Incoming non-probabilistic size bound w.r.t. $\alpha \in \mathcal{GRV}$ for $x \in \mathcal{PV}$ 285
$\mathrm{size}^{\alpha}_{\mathbb{E}}(x)$ Incoming expected size bound w.r.t. $\alpha \in \mathcal{GRV}$ for $x \in \mathcal{PV}$ 285
$\left(\mathcal{F}^{\mathrm{loop}}_n\right)_{n\in\mathbb{N}}$ Filtration induced by probabilistic loop 179
$\left(\mathcal{G}^{\mathrm{loop}}_n\right)_{n\in\mathbb{N}}$ Shifted filtration induced by probabilistic loop 182
$[M]$ Iverson bracket function of set M 155
$[\varphi]$ Iverson bracket function of Boolean formula φ 155
$\mathbb{F}$ Set of expectations $\left\{f \mid f : \Sigma \to \overline{\mathbb{R}}_{\geq 0}\right\}$ 165
$\mathbb{NPE}_{\mathcal{S}}[\vec{x}]$ Normalized poly-exponential expressions over $\mathcal{S}$ in $\vec{x}$ 33
$\mathbb{PE}_{\mathcal{S}}[\vec{x}]$ Poly-exponential expressions over $\mathcal{S}$ in $\vec{x}$ 28
$\mathbb{R}_{\mathbb{A}}$ Real algebraic numbers 21
$\mathbb{R}_{\mathbb{A}}(\vec{x})$ Field of rational functions in $\vec{x}$ with coefficients from $\mathbb{R}_{\mathbb{A}}$ 228
$\mathbf{X}^{f,I}$ Stochastic process for probabilistic loop induced by $f, I \in \mathbb{F}$ 181
$\mathcal{C}$ Set of constraints 259
$\mathcal{D}$ Set of bounded distribution functions 259
$\mathcal{L}$ Set of locations of a PITS 260
$\mathcal{EL}_{\mathcal{GT}_{\mathrm{ni}}}$ Entry locations of $\mathcal{GT}_{\mathrm{ni}}$ 276
$\mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)$ Entry transitions of ℓ w.r.t. $\mathcal{GT}_{\mathrm{ni}}$ 276
$\mathcal{F}^{\mathrm{loop}}$ Cylinder-σ-field for probabilistic loop 176
$\mathcal{GRV}$ Set of general result variables 271
$\mathcal{GT}$ Set of general transitions of a PITS 260
$\mathcal{GT}_{>}$ Decreasing general transitions for PLRF 274
$\mathcal{GT}_{\mathrm{ni}}$ Non-increasing general transitions for PLRF 274
$\mathcal{PV}$ Finite set of program variables 258

$\mathcal{R}$ Runtime of a PITS 269
$\mathcal{S}(\alpha)$ Size of $\alpha \in \mathcal{GRV}$ 271
$\mathcal{S}[\vec{x}]$ Ring of polynomials over $\mathcal{S}$ 21
$\mathcal{TV}$ Temporary variables $\mathcal{V} \setminus \mathcal{PV}$ 260
$\mathcal{T}$ Set of transitions 262
$\mathcal{V}$ Finite set of variables 145
$\mathfrak{P}(\Omega)$ Powerset of Ω 175
$rt_{(\cdot,\cdot)}$ Runtime of a polynomial loop 47
$sth.(\vec{e})$ Stabilization threshold on $\vec{e}$ 40
$\mathrm{Aut}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$ Group of $\mathcal{S}$-automorphisms 70
$\mathrm{End}_{\mathcal{S}}\left(\mathcal{S}[\vec{x}]\right)$ Set of $\mathcal{S}$-endomorphisms 70
Conf Set of configurations of a PITS 262
pre(g) Pre-transitions of g 283
$\preceq$ Point-wise lifting of $\leq$ to $\mathbb{F}$ 165
${}^{\bullet}\Phi_{\langle \varphi, \mathbf{P} \rangle, f}$ $\bullet$-characteristic function of a loop w.r.t. f where $\bullet \in \{\mathsf{wp}, \mathsf{awp}, \mathsf{ert}\}$ 167
pGCL Probabilistic guarded command language 146
Scheds Set of schedulers 154
awp $[\![\mathbf{P}]\!]$ (f) Angelic weakest preexpectation of f w.r.t. $\mathbf{P}$ 163
ert $[\![\mathbf{P}]\!]$ (t) Expected runtime of $\mathbf{P}$ followed by time t 159
ert $[\![\mathbf{P}]\!]$ $(t)^{\mathfrak{s}}$ Expected runtime of $\mathbf{P}$ for $\mathfrak{s}$ followed by time t 158
wp $[\![\mathbf{P}]\!]$ (f) Weakest preexpectation of f w.r.t. $\mathbf{P}$ 162
φ Loop condition 21
ϑ Run of a probabilistic loop or PITS 176
$\vec{q}$ Closed form for $\vec{u}$ 24
$\vec{q}_{norm}$ Normalized closed form of $\vec{u}$ 32
$\vec{u}$ Loop update 21
$\vec{v}_1 \bullet \vec{v}_2$ Scalar product of $\vec{v}_1, \vec{v}_2 \in \mathbb{R}^d$ 228
$f \ll \infty$ Expectation f takes only finite values 165
$o(g)$ Set of functions asymptotically dominated by g 33
$\mathcal{CB}_{\mathbb{E}}(\alpha)$ Expected local change bound of result variable α 282
$\mathcal{RB}$ Runtime bound for PITS 268
$\mathcal{SB}$ Size bound for PITS 268
$\mathcal{RB}_{\mathbb{E}}$ Expected runtime bound for PITS 272
$\mathcal{SB}_{\mathbb{E}}$ Expected size bound for PITS 272
$\Phi_{\langle \mathbf{L} \rangle}$ Characteristic looping time function 230
XP Slicewise polynomial decision problems 92

Index

σ-field, 175
pGCL-program, 146
  almost-sure termination, 157
    positive, 159
    strong positive, 160
  AST, *see* almost-sure termination
  expected runtime, 159
  expected runtime followed by time t, 158, 159
  operational MDP, 150
    final state, 150
  PAST, *see* almost-sure termination, positive
  probability of non-termination, 157
  probability of termination, 157
  purely probabilistic, 146
  state, 146
  weakest preexpectation, 162
    angelic, 163
k-partition, 99
  blocks, 99
  hierarchical, 99

abstract assignment, 120
asymptotic domination, 33

bound, 258
  finite, 258

complete lattice, 165
  continuous function, 169

distribution function, 146
  bounded, 259
DTMC, *see* Markov chain

eigenvalue
  algebraic multiplicity, 95
  geometric multiplicity, 91
  index, 95
expectation, 165
  ω-subinvariant, 190
  c.d.b., *see* conditionally difference bounded
  canonical stochastic process, 181
    stopped process, 193
  conditionally difference bounded, 199, 224
  finite, 165

harmonization, 199
invariant, 174
u.i., *see* uniformly integrable
uniformly integrable, 192

formula
closed, 25
interval condition, 103
linear, 114
quasi sign condition, 103
satisfiable, 25
sign condition, 103
valid, 25

harmonic number, 214
hierarchical expression, 97
base, 97
dominant term, 97
maximal variable, 97
offset, 97
order, 97

Iverson bracket, 28, 155
for a set, 155

Markov chain, 150
cumulative rewards, 156
expected total reward, 156
reachability probability, 156
Markov decision process, 150
maximal expected total reward, 156
minimal expected total reward, 156
reward function, 156
scheduler, 154
MDP, *see* Markov decision process
measurable map, 177
measurable space, 175
filtration, 179
measure, 176
probability measure, 176
almost-sure property, 194
monotonicity threshold, 41

parameterized decision problem, 92
k-termination, 92
slicewise polynomial, 92
PITS, *see* probabilistic integer transition system
poly-exponential expression, 28
normalized, 32
stabilization threshold, 40
polynomial loop, 21
chaining, 27
closed form, 24
linear, 22
uniform, 91
linear-update, 22
real spectrum, 22
linearization via $\vec{w}$, 59
non-negative, 23
runtime, 47
solvable, 24
stabilization, 40
threshold on $\vec{e}$, 40
termination on $\vec{e}$, 23
termination over $\mathcal{S}$, 23
tnn, 23
triangular, 23
twn, 23
dependency degree, 56
twn-transformable, 76
weakly non-linear, 23
probabilistic guarded command language, 146
probabilistic loop, 174
cpa-loop, 228

pa-loop, *see* probabilistic addition loop
cylinder set, 175
looping time, 178
probabilistic addition loop, 227
characteristic looping time function, 230
direct termination, 228
drift, 234
trivial, 228
random walk loop, 240
characteristic polynomial, 248
run, 175
admissible, 176, 177
probabilistic transition system, 260
active variables, 283
almost-sure termination
positive, 269
strong positive, 270
bound pair
expected, 272
non-probabilistic, 268
configuration, 262
admissible, 265
entry locations, 276
entry transitions, 276
expected local change bound, 281
finite path, 262
admissible, 265
general result variable graph, 283
SCC, 284
general transition, 260
condition, 260
invariant, 288
start location, 260
initial general transitions, 260
location, 260
PLRF, *see* probabilistic linear ranking function
pre-transitions, 283
probabilistic linear ranking function, 274
program variables, 260
run, 262
admissible, 265
runtime, 269
expected, 269
runtime bound, 268
expected, 272
scheduler, 262
size, 271
expected, 271
size bound, 268
expected, 272
temporary variable, 260
transition, 260
condition, 260
probability, 260
start location, 260
target location, 260
update function, 260
probability distribution, 145

random variable, 177
conditional expected value, 182
expected value, 178
stopping time, 179
real algebraic number
periodic real, 90
ring
automorphism, 70
degree, 74
linear, 74
endomorphism, 70

set
$\mathrm{Th}_\exists(S, \mathbb{R}_\mathrm{A})$-definable, 73
convex, 125

regular expression, 149
semi-algebraic, 126
update-invariant, 73
stochastic process, 179
adapted to filtration, 179
martingale, 184
submartingale, 184
supermartingale, 184
uniform integrability, 196

transformation
complete, 76, 87

witness
eventual non-termination, 23
non-termination, 23

Bibliography

[AB09] S. Arora and B. Barak. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009.

[Abr74] M. Abramowitz. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables,* Dover Publications, Inc., 1974.

[ABV17] S. Akshay, N. Balaji, and N. Vyas. "Complexity of Restricted Variants of Skolem and Related Problems". In: *Proc. MFCS*. LIPIcs Vol. 83. 2017, 78:1–78:14.

[ACN18] S. Agrawal, K. Chatterjee, and P. Novotný. "Lexicographic Ranking Supermartingales: An Efficient Approach to Termination of Probabilistic Programs". In: *Proc. ACM Program. Lang.* 2.POPL (2018), 34:1–34:32.

[Agu+21] A. Aguirre, G. Barthe, J. Hsu, B. L. Kaminski, J.-P. Katoen, and C. Matheja. "A Pre-Expectation Calculus for Probabilistic Sensitivity". In: *Proc. ACM Program. Lang.* 5.POPL (2021), pp. 1–28.

[Alb+11] E. Albert, P. Arenas, S. Genaim, and G. Puebla. "Closed-Form Upper Bounds in Static Cost Analysis". In: *J. Autom. Reason.* 46.2 (2011), pp. 161–203.

[Alb+19a] E. Albert, M. Bofill, C. Borralleras, E. Martin-Martin, and A. Rubio. "Resource Analysis Driven by (Conditional) Termination Proofs". In: *Theory Pract. Log. Program.* 19.5-6 (2019), pp. 722–739.

[Alb+19b] E. Albert, P. Gordillo, A. Rubio, and I. Sergey. “Running on Fumes - Preventing Out-of-Gas Vulnerabilities in Ethereum Smart Contracts Using Static Resource Analysis”. In: *Proc. VECoS*. Lecture Notes in Computer Science Vol. 11847. 2019, pp. 63–78.

[Alb+20] E. Albert, J. Correas, P. Gordillo, G. Román-Díez, and A. Rubio. “GASOL: Gas Analysis and Optimization for Ethereum Smart Contracts”. In: *Proc. TACAS (2)*. Lecture Notes in Computer Science Vol. 12079. 2020, pp. 118–125.

[Ali+10] C. Alias, A. Darte, P. Feautrier, and L. Gonnord. “Multi-Dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs”. In: *Proc. SAS*. Lecture Notes in Computer Science Vol. 6337. 2010, pp. 117–133.

[ALY20] M. Avanzini, U. D. Lago, and A. Yamada. “On Probabilistic Term Rewriting”. In: *Sci. Comput. Program.* 185 (2020).

[AMS20] M. Avanzini, G. Moser, and M. Schaper. “A Modular Cost Analysis for Probabilistic Programs”. In: *Proc. ACM Program. Lang.* 4.OOPSLA (2020), 172:1–172:30. Implementation available at the web site http://www-sop.inria.fr/members/Martin.Avanzini/software/eco-imp/.

[AP09] P. Audebaud and C. Paulin-Mohring. “Proofs of Randomized Algorithms in Coq”. In: *Sci. Comput. Program.* 74.8 (2009), pp. 568–589.

[Bab+13] D. Babić, B. Cook, A. J. Hu, and Z. Rakamaric. “Proving Termination of Nonlinear Command Sequences”. In: *Formal Aspects Comput.* 25.3 (2013), pp. 389–403.

[Bar+16] G. Barthe, T. Espitau, L. M. F. Fioriti, and J. Hsu. “Synthesizing Probabilistic Invariants via Doob’s Decomposition”. In: *Proc. CAV (1)*. Lecture Notes in Computer Science Vol. 9779. 2016, pp. 43–61.

[Bar91] A. Baranga. “The Contraction Principle as a Particular Case of Kleene’s Fixed Point Theorem”. In: *Discret. Math.* 98.1 (1991), pp. 75–79.

[Bat+18] K. Batz, B. L. Kaminski, J.-P. Katoen, and C. Matheja. “How Long, O Bayesian Network, Will I Sample Thee? - A Program Analysis Perspective on Expected Sampling Times”. In: *Proc. ESOP*. Lecture Notes in Computer Science Vol. 10801. 2018, pp. 186–213.

[Bat+19] K. Batz, B. L. Kaminski, J.-P. Katoen, C. Matheja, and T. Noll. “Quantitative Separation Logic: a Logic for Reasoning about Probabilistic Pointer Programs”. In: *Proc. ACM Program. Lang.* 3.POPL (2019), 34:1–34:29.

[Bat+21] K. Batz, B. L. Kaminski, J.-P. Katoen, and C. Matheja. "Relatively Complete Verification of Probabilistic Programs: An Expressive Language for Expectation-Based Reasoning". In: *Proc. ACM Program. Lang.* 5.POPL (2021), pp. 1–30.

[Bau11a] H. Bauer. *Measure and Integration Theory*. De Gruyter, 2011.

[Bau11b] H. Bauer. *Probability Theory*. De Gruyter, 2011.

[BBE10] T. Brázdil, V. Brozek, and K. Etessami. "One-Counter Stochastic Games". In: *Proc. FSTTCS*. LIPIcs Vol. 8. 2010, pp. 108–119.

[BCF13] M. Brockschmidt, B. Cook, and C. Fuhs. "Better Termination Proving Through Cooperation". In: *Proc. CAV*. Lecture Notes in Computer Science Vol. 8044. 2013, pp. 413–429.

[BD77] R. M. Burstall and J. Darlington. "A Transformation System for Developing Recursive Programs". In: *J. ACM* 24.1 (1977), pp. 44–67.

[BDG19] A. M. Ben-Amram, J. J. Doménech, and S. Genaim. "Multiphase-Linear Ranking Functions and Their Relation to Recurrent Sets". In: *Proc. SAS*. Lecture Notes in Computer Science Vol. 11822. 2019, pp. 459–480.

[BG05] O. Bournez and F. Garnier. "Proving Positive Almost-Sure Termination". In: *Proc. RTA*. Lecture Notes in Computer Science Vol. 3467. 2005, pp. 323–337.

[BG06] O. Bournez and F. Garnier. "Proving Positive Almost Sure Termination Under Strategies". In: *Proc. RTA*. Vol. 4098. Lecture Notes in Computer Science. 2006, pp. 357–371.

[BG14] A. M. Ben-Amram and S. Genaim. "Ranking Functions for Linear-Constraint Loops". In: *J. ACM* 61.4 (2014), 26:1–26:55.

[BG17] A. M. Ben-Amram and S. Genaim. "On Multiphase-Linear Ranking Functions". In: *Proc. CAV*. Lecture Notes in Computer Science Vol. 10427. 2017, pp. 601–620.

[BGM12] A. M. Ben-Amram, S. Genaim, and A. N. Masud. "On the Termination of Integer Loops". In: *ACM Trans. Program. Lang. Syst.* 34.4 (2012), 16:1–16:24.

[BH20] A. M. Ben-Amram and G. W. Hamilton. "Tight Polynomial Worst-Case Bounds for Loop Programs". In: *Log. Methods Comput. Sci.* 16.2 (2020).

[BIK14] M. Bozga, R. Iosif, and F. Konecný. "Deciding Conditional Termination". In: *Log. Methods Comput. Sci.* 10.3 (2014).

[BJK08] A. M. Ben-Amram, N. D. Jones, and L. Kristiansen. "Linear, Polynomial or Exponential? Complexity Inference in Polynomial Time". In: *Proc. CiE*. Lecture Notes in Computer Science Vol. 5028. 2008, pp. 67–76.

[BK08] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[BK12] A. M. Ben-Amram and L. Kristiansen. "On the Edge of Decidability in Complexity Analysis of Loop Programs". In: *Int. J. Found. Comput. Sci.* 23.7 (2012), pp. 1451–1464.

[BKK14] T. Brázdil, S. Kiefer, and A. Kucera. "Efficient Analysis of Probabilistic Programs With an Unbounded Counter". In: *J. ACM* 61.6 (2014), 41:1–41:35.

[BKS19] E. Bartocci, L. Kovács, and M. Stankovic. "Automatic Generation of Moment-Based Invariants for Prob-Solvable Loops". In: *Proc. ATVA*. Lecture Notes in Computer Science Vol. 11781. 2019, pp. 255–276.

[BKS20a] E. Bartocci, L. Kovács, and M. Stankovic. "Analysis of Bayesian Networks via Prob-Solvable Loops". In: *Proc. ICTAC*. Lecture Notes in Computer Science Vol. 12545. 2020, pp. 221–241.

[BKS20b] E. Bartocci, L. Kovács, and M. Stankovic. "Mora - Automatic Generation of Moment-Based Invariants". In: *Proc. TACAS (1)*. Lecture Notes in Computer Science Vol. 12078. 2020, pp. 492–498.

[BM07] A. R. Bradley and Z. Manna. *The Calculus of Computation - Decision Procedures With Applications to Verification*. Springer, 2007.

[BM17] S. Basu and B. Mishra. "Computational and Quantitative Real Algebraic Geometry". In: *Handbook of Discrete and Computational Geometry, Third Edition*. 2017, pp. 969–1002.

[BMS05a] A. R. Bradley, Z. Manna, and H. B. Sipma. "Linear Ranking With Reachability". In: *Proc. CAV*. Lecture Notes in Computer Science Vol. 3576. 2005, pp. 491–504.

[BMS05b] A. R. Bradley, Z. Manna, and H. B. Sipma. "Termination Analysis of Integer Linear Loops". In: *Proc. CONCUR*. Lecture Notes in Computer Science Vol. 3653. 2005, pp. 488–502.

[BMS05c] A. R. Bradley, Z. Manna, and H. B. Sipma. "Termination of Polynomial Programs". In: *Proc. VMCAI*. Lecture Notes in Computer Science Vol. 3385. 2005, pp. 113–129.

[BMS05d] A. R. Bradley, Z. Manna, and H. B. Sipma. "The Polyranking Principle". In: *Proc. ICALP*. Lecture Notes in Computer Science Vol. 3580. 2005, pp. 1349–1361.

[BP16] A. M. Ben-Amram and A. Pineles. "Flowchart Programs, Regular Expressions, and Decidability of Polynomial Growth-Rate". In: *Proc. VPT@ETAPS*. EPTCS Vol. 216. 2016, pp. 24–49.

[BPR06] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics 10. Springer, 2006.

[Brá+12] T. Brázdil, A. Kucera, P. Novotný, and D. Wojtczak. "Minimizing Expected Termination Time in One-Counter Markov Decision Processes". In: *Proc. ICALP (2)*. Lecture Notes in Computer Science Vol. 7392. 2012, pp. 141–152.

[Brá+13a] T. Brázdil, V. Brozek, K. Etessami, and A. Kucera. "Approximating the Termination Value of One-Counter MDPs and Stochastic Games". In: *Inf. Comput.* 222 (2013), pp. 121–138.

[Brá+13b] T. Brázdil, J. Esparza, S. Kiefer, and A. Kucera. "Analyzing Probabilistic Pushdown Automata". In: *Formal Methods Syst. Des.* 43.2 (2013), pp. 124–163.

[Brá+18] T. Brázdil, K. Chatterjee, A. Kucera, P. Novotný, D. Velan, and F. Zuleger. "Efficient Algorithms for Asymptotic Bounds on Termination Time in VASS". In: *Proc. LICS*. 2018, pp. 185–194.

[Brá+19] T. Brázdil, K. Chatterjee, A. Kucera, P. Novotný, and D. Velan. "Deciding Fast Termination for Probabilistic VASS With Nondeterminism". In: *Proc. ATVA*. Lecture Notes in Computer Science Vol. 11781. 2019, pp. 462–478.

[Bra06] M. Braverman. "Termination of Integer Linear Programs". In: *Proc. CAV*. Lecture Notes in Computer Science Vol. 4144. 2006, pp. 372–385.

[Bro+16] M. Brockschmidt, F. Emmes, S. Falke, C. Fuhs, and J. Giesl. "Analyzing Runtime and Size Complexity of Integer Programs". In: *ACM Trans. Program. Lang. Syst.* 38.4 (2016), 13:1–13:50.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[BW98] R.-J. Back and J. von Wright. *Refinement Calculus - A Systematic Introduction*. Graduate Texts in Computer Science. Springer, 1998.

[Cai94] J.-y. Cai. "Computing Jordan Normal Forms Exactly for Commuting Matrices in Polynomial Time". In: *Int. J. Found. Comput. Sci.* 5.3/4 (1994), pp. 293–302.

[Can88] J. F. Canny. "Some Algebraic and Geometric Computations in PSPACE". In: *Proc. STOC*. 1988, pp. 460–467.

[Car+17] Q. Carbonneaux, J. Hoffmann, T. W. Reps, and Z. Shao. "Automated Resource Analysis With Coq Proof Objects". In: *Proc. CAV (2)*. Lecture Notes in Computer Science Vol. 10427. 2017, pp. 64–85.

[CDZ14] T. Colcombet, L. Daviaud, and F. Zuleger. "Size-Change Abstraction and Max-Plus Automata". In: *Proc. MFCS*. Lecture Notes in Computer Science Vol. 8634. 2014, pp. 208–219.

[CFG16] K. Chatterjee, H. Fu, and A. K. Goharshady. "Termination Analysis of Probabilistic Programs Through Positivstellensatz's". In: *Proc. CAV (1)*. Lecture Notes in Computer Science Vol. 9779. 2016, pp. 3–22.

[CFM17] K. Chatterjee, H. Fu, and A. Murhekar. "Automated Recurrence Analysis for Almost-Linear Expected-Runtime Bounds". In: *Proc. CAV (1)*. Lecture Notes in Computer Science Vol. 10426. 2017, pp. 118–139.

[CFN20] K. Chatterjee, H. Fu, and P. Novotný. "Termination Analysis of Probabilistic Programs With Martingales". In: *Foundations of Probabilistic Programming*. 2020, pp. 221–258.

[Cha+18] K. Chatterjee, H. Fu, P. Novotný, and R. Hasheminezhad. "Algorithmic Analysis of Qualitative and Quantitative Termination Problems for Affine Probabilistic Programs". In: *ACM Trans. Program. Lang. Syst.* 40.2 (2018), 7:1–7:45.

[Che+14] H. Y. Chen, B. Cook, C. Fuhs, K. Nimkar, and P. W. O'Hearn. "Proving Nontermination via Safety". In: *Proc. TACAS*. Lecture Notes in Computer Science Vol. 8413. 2014, pp. 156–171.

[Che+15] Y.-F. Chen, C.-D. Hong, B.-Y. Wang, and L. Zhang. "Counterexample-Guided Polynomial Loop Invariant Generation by Lagrange Interpolation". In: *Proc. CAV (1)*. Lecture Notes in Computer Science Vol. 9206. 2015, pp. 658–674.

[Çiç+20] E. Çiçek, M. Bouaziz, S. Cho, and D. Distefano. "Static Resource Analysis at Scale (Extended Abstract)". In: *SAS*. Vol. 12389. Lecture Notes in Computer Science. 2020, pp. 3–6.

[CNZ17] K. Chatterjee, P. Novotný, and D. Zikelic. "Stochastic Invariants for Probabilistic Termination". In: *Proc. POPL*. 2017, pp. 145–160.

[Coc14] D. Cock. "pGCL for Isabelle". In: *Archive of Formal Proofs* (2014).

[Coh69] P. J. Cohen. "Decision Procedures for Real and p-adic Fields". In: *Communications on Pure and Applied Mathematics* 22.2 (1969), pp. 131–151.

[Coh93] H. Cohen. *A Course in Computational Algebraic Number Theory*. Graduate Texts in Mathematics Vol. 138. Springer, 1993.

[Com21] *Companies by Market Capitalization*. 2021. URL: https://companiesmarketcap.com/ (visited on 02/05/2021).

[CS13] A. Chakarov and S. Sankaranarayanan. "Probabilistic Program Analysis With Martingales". In: *Proc. CAV*. Lecture Notes in Computer Science Vol. 8044. 2013, pp. 511–526.

[CS14] A. Chakarov and S. Sankaranarayanan. "Expectation Invariants for Probabilistic Program Loops as Fixed Points". In: *Proc. SAS*. Lecture Notes in Computer Science Vol. 8723. 2014, pp. 85–100.

[Dan47] G. B. Dantzig. *Programming in a Linear Structure*. US Air Force. 1947.

[Das+21] A. Das, S. Balzer, J. Hoffmann, F. Pfenning, and I. Santurkar. "Resource-Aware Session Types for Digital Contracts". In: *Proc. CSF*. 2021, pp. 111–126.

[Dav58] M. D. Davis. *Computability and Unsolvability*. McGraw-Hill Series in Information Processing and Computers. McGraw-Hill, 1958.

[DF99] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.

[Dij75] E. W. Dijkstra. "Guarded Commands, Nondeterminacy and Formal Derivation of Programs". In: *Commun. ACM* 18.8 (1975), pp. 453–457.

[Dij76] E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.

[Dom21] J. J. Doménech. "Termination Analysis of Programs With Complex Control-Flow". PhD Thesis. Complutense University of Madrid, Spain, 2021.

[DR06] W. Dahmen and A. Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. Springer-Lehrbuch. Springer, 2006.

[DX12] L. Dai and B. Xia. "Non-Termination Sets of Simple Linear Loops". In: *Proc. ICTAC*. Lecture Notes in Computer Science Vol. 7521. 2012, pp. 61–73.

[DZ10] M. S. E. Din and L. Zhi. "Computing Rational Points in Convex Semialgebraic Sets and Sum of Squares Decompositions". In: *SIAM J. Optim.* 20.6 (2010), pp. 2876–2889.

[Ede97] A. Edelman. "The Probability that a Random Real Gaussian Matrix has k Real Eigenvalues, Related Distributions, and the Circular Law". In: *J. Multivar. Anal.* 60.2 (1997), pp. 203–232.

[EH96] A. van den Essen and E. Hubbers. “Polynomial Maps With Strongly Nilpotent Jacobian Matrix and the Jacobian Conjecture”. In: *Linear Algebra and its Applications* 247 (1996), pp. 121–132.

[EKS94] A. Edelman, E. Kostlan, and M. Shub. “How Many Eigenvalues of a Random Matrix are Real?” In: *J. Am. Math. Soc.* 7.1 (1994), pp. 247–267.

[Ela05] S. Elaydi. *An Introduction to Difference Equations*. Springer, 2005.

[Ess00] A. van den Essen. *Polynomial Automorphisms and the Jacobian Conjecture*. Springer Basel, 2000.

[Eve+03] G. Everest, A. J. van der Poorten, I. E. Shparlinski, and T. Ward. *Recurrence Sequences*. Vol. 104. Mathematical Surveys and Monographs. American Mathematical Society, 2003.

[EY09] K. Etessami and M. Yannakakis. “Recursive Markov Chains, Stochastic Grammars, and Monotone Systems of Nonlinear Equations”. In: *J. ACM* 56.1 (2009), 1:1–1:66.

[FC19] H. Fu and K. Chatterjee. “Termination of Nondeterministic Probabilistic Programs”. In: *Proc. VMCAI*. Lecture Notes in Computer Science Vol. 11388. 2019, pp. 468–490.

[Fel50] W. Feller. *An Introduction to Probability Theory and its Applications*. Vol. I (First Edition). Probability and Mathematical Statistics. John Wiley & Sons, 1950.

[Fel71] W. Feller. *An Introduction to Probability Theory and its Applications*. Vol. II (Second Edition). Probability and Mathematical Statistics. John Wiley & Sons, 1971.

[Fen+17] Y. Feng, L. Zhang, D. N. Jansen, N. Zhan, and B. Xia. “Finding Polynomial Loop Invariants for Probabilistic Programs”. In: *Proc. ATVA*. Lecture Notes in Computer Science Vol. 10482. 2017, pp. 400–416.

[FG17] F. Frohn and J. Giesl. “Complexity Analysis for Java With AProVE”. In: *Proc. IFM*. Lecture Notes in Computer Science Vol. 10510. 2017, pp. 85–101.

[FG19a] F. Frohn and J. Giesl. “Termination of Triangular Integer Loops is Decidable”. In: *Proc. CAV*. Lecture Notes in Computer Science Vol. 11562. 2019, pp. 269–286.

[FG19b] F. Frohn and J. Giesl. “Proving Non-Termination via Loop Acceleration”. In: *Proc. FMCAD*. 2019, pp. 221–230.

[FH15] L. M. F. Fioriti and H. Hermanns. "Probabilistic Termination: Soundness, Completeness, and Compositionality". In: *Proc. POPL*. 2015, pp. 489–501.

[FHG20] F. Frohn, M. Hark, and J. Giesl. "Termination of Polynomial Loops". In: *Proc. SAS*. Lecture Notes in Computer Science Vol. 12389. 2020, pp. 89–112.

[Flo16] A. Flores-Montoya. "Upper and Lower Amortized Cost Bounds of Programs Expressed as Cost Relations". In: *Proc. FM*. Lecture Notes in Computer Science Vol. 9995. 2016, pp. 254–273.

[Flo67] R. W. Floyd. "Assigning Meanings to Programs". In: Proceedings of Symposia in Applied Mathematics Vol. 19. 1967, pp. 19–32.

[Fro+20] F. Frohn, M. Naaf, M. Brockschmidt, and J. Giesl. "Inferring Lower Runtime Bounds for Integer Programs". In: *ACM Trans. Program. Lang. Syst.* 42.3 (2020), 13:1–13:50.

[Fro20] F. Frohn. "A Calculus for Modular Loop Acceleration". In: *Proc. TACAS*. Lecture Notes in Computer Science Vol. 12078. 2020, pp. 58–76.

[Fru77] M. A. Frumkin. "Polynomial Time Algorithms in the Theory of Linear Diophantine Equations". In: *Proc. FCT*. Lecture Notes in Computer Science Vol. 56. 1977, pp. 386–392.

[GGH19a] J. Giesl, P. Giesl, and M. Hark. "Computing Expected Runtimes for Constant Probability Programs". In: *Proc. CADE*. Lecture Notes in Computer Science Vol. 11716. 2019, pp. 269–286.

[GGH19b] J. Giesl, P. Giesl, and M. Hark. "Computing Expected Runtimes for Constant Probability Programs". In: *CoRR* abs/1905.09544 (2019).

[Gie+17] J. Giesl, C. Aschermann, M. Brockschmidt, F. Emmes, F. Frohn, C. Fuhs, J. Hensel, C. Otto, M. Plücker, P. Schneider-Kamp, T. Ströder, S. Swiderski, and R. Thiemann. "Analyzing Program Termination and Complexity Automatically With AProVE". In: *J. Autom. Reason.* 58.1 (2017), pp. 3–31.

[Gie+19a] J. Giesl, P. Giesl, M. Hark, and T. Küspert. 2019. KoAT for exact expected runtimes: Web interface, binary, Docker image, and examples available at https://aprove-developers.github.io/recurrence/.

[Gie+19b] J. Giesl, A. Rubio, C. Sternagel, J. Waldmann, and A. Yamada. "The Termination and Complexity Competition". In: *Proc. TACAS*. Lecture Notes in Computer Science Vol. 11429. 2019, pp. 156–166.

[Gie95] M. Giesbrecht. "Nearly Optimal Algorithms for Canonical Matrix Forms". In: *SIAM J. Comput.* 24.5 (1995), pp. 948–969.

[GKM14] F. Gretz, J.-P. Katoen, and A. McIver. "Operational Versus Weakest Pre-Expectation Semantics for the Probabilistic Guarded Command Language". In: *Perform. Evaluation* 73 (2014), pp. 110–132.

[GKP94] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science, 2nd Ed.* Addison-Wesley, 1994.

[God05] P. Godefroid. "The Soundness of Bugs is What Matters (Position Statement)". In: *BUGS'2005 (PLDI'2005 Workshop on the Evaluation of Software Defect Detection Tools)*. 2005.

[Gom58] R. Gomory. "Outline of an Algorithm for Integer Solutions to Linear Programs". In: *Bulletin of the American Mathematical Society* 64 (1958), pp. 275–278.

[Gom60] R. Gomory. *An Algorithm for the Mixed Integer Problem*. Tech. rep. The Rand Corporation, 1960.

[Gor+14] A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani. "Probabilistic Programming". In: *Proc. FOSE*. 2014, pp. 167–181.

[GS01] G. Grimmett and D. Stirzaker. *Probability and Random Processes (Third Edition)*. Oxford University Press, 2001.

[Gup+08] A. Gupta, T. A. Henzinger, R. Majumdar, A. Rybalchenko, and R.-G. Xu. "Proving Non-Termination". In: *Proc. POPL*. 2008, pp. 147–158.

[HAH12] J. Hoffmann, K. Aehlig, and M. Hofmann. "Multivariate Amortized Resource Analysis". In: *ACM Trans. Program. Lang. Syst.* 34.3 (2012), 14:1–14:62.

[Hal+05] V. Halava, T. Harju, M. Hirvensalo, and J. Karhumäki. *Skolem's Problem – On the Border Between Decidability and Undecidability*. Tech. rep. 683. Turku Center for Computer Science, 2005.

[Har+19] M. Hark, B. L. Kaminski, J. Giesl, and J.-P. Katoen. "Aiming Low Is Harder - Inductive Proof Rules for Lower Bounds on Weakest Preexpectations in Probabilistic Program Verification". In: *CoRR* abs/1904.01117 (2019).

[Har+20] M. Hark, B. L. Kaminski, J. Giesl, and J.-P. Katoen. "Aiming Low is Harder: Induction for Lower Bounds in Probabilistic Program Verification". In: *Proc. ACM Program. Lang.* 4.POPL (2020), 37:1–37:28.

[HBK20] A. Humenberger, N. Bjørner, and L. Kovács. "Algebra-Based Loop Synthesis". In: *Proc. IFM*. Lecture Notes in Computer Science Vol. 12546. 2020, pp. 440–459.

[HDW17] J. Hoffmann, A. Das, and S.-C. Weng. "Towards Automatic Resource Bound Analysis for OCaml". In: *Proc. POPL*. 2017, pp. 359–373.

[Heh11] E. C. R. Hehner. "A Probability Perspective". In: *Formal Aspects Comput.* 23.4 (2011), pp. 391–419.

[Hen+18] J. Hensel, J. Giesl, F. Frohn, and T. Ströder. "Termination and Complexity Analysis for Programs With Bitvector Arithmetic by Symbolic Execution". In: *J. Log. Algebraic Methods Program.* 97 (2018), pp. 105–130.

[Her71] G. T. Herman. "Strong Computability and Variants of the Uniform Halting Problem". In: *Math. Log. Quart.* 17.1 (1971), pp. 115–131.

[HFC18] M. Huang, H. Fu, and K. Chatterjee. "New Approaches for Almost-Sure Termination of Probabilistic Programs". In: *Proc. APLAS*. Lecture Notes in Computer Science Vol. 11275. 2018, pp. 181–201.

[HFG20] M. Hark, F. Frohn, and J. Giesl. "Polynomial Loops: Beyond Termination". In: *Proc. LPAR*. EPiC Series in Computing Vol. 73. 2020, pp. 279–297.

[Hin+16] W. Hino, H. Kobayashi, I. Hasuo, and B. Jacobs. "Healthiness from Duality". In: *Proc. LICS*. 2016, pp. 682–691.

[HJK18] A. Humenberger, M. Jaroschek, and L. Kovács. "Invariant Generation for Multi-Path Loops With Polynomial Assignments". In: *Proc. VMCAI*. Lecture Notes in Computer Science Vol. 10747. 2018, pp. 226–246.

[HK21] A. Humenberger and L. Kovács. "Algebra-Based Synthesis of Loops and Their Invariants (Invited Paper)". In: *Proc. VMCAI*. Lecture Notes in Computer Science Vol. 12597. 2021, pp. 17–28.

[HLM07] A. J. Hetzel, J. S. Liew, and K. E. Morrison. "The Probability That a Matrix of Integers is Diagonalizable". In: *Am. Math. Mon.* 114.6 (2007), pp. 491–499.

[Hoa69] C. A. R. Hoare. "An Axiomatic Basis for Computer Programming". In: *Commun. ACM* 12.10 (1969), pp. 576–580.

[HOW19] M. Hosseini, J. Ouaknine, and J. Worrell. "Termination of Linear Loops Over the Integers". In: *Proc. ICALP*. LIPIcs Vol. 132. 2019, 118:1–118:13.

[HS15] J. Hoffmann and Z. Shao. "Type-Based Amortized Resource Analysis With Integers and Arrays". In: *J. Funct. Program.* 25 (2015).

[Hua+19] M. Huang, H. Fu, K. Chatterjee, and A. K. Goharshady. "Modular Verification for Almost-Sure Termination of Probabilistic Programs". In: *Proc. ACM Program. Lang.* 3.OOPSLA (2019), 129:1–129:29.

[Jan+16] N. Jansen, C. Dehnert, B. L. Kaminski, J.-P. Katoen, and L. Westhofen. "Bounded Model Checking for Probabilistic Programs". In: *Proc. ATVA*. Lecture Notes in Computer Science Vol. 9938. 2016, pp. 68–85.

[JGP00] J. Jachymski, L. Gajek, and P. Pokarowski. "The Tarski–Kantorovitch Principle and the Theory of Iterated Function Systems". In: *Bulletin of the Australian Mathematical Society* 61.2 (2000), pp. 247–261.

[JM09] B. Jeannet and A. Miné. "Apron: A Library of Numerical Abstract Domains for Static Analysis". In: *CAV*. Lecture Notes in Computer Science Vol. 5643. 2009, pp. 661–667.

[Joh+13] F. Johansson et al. *MpMath: A Python Library for Arbitrary-Precision Floating-Point Arithmetic (Version 0.18)*. 2013. http://mpmath.org/.

[Jon90] C. Jones. "Probabilistic Non–Determinism". PhD Thesis. University of Edinburgh, UK, 1990.

[Kal02] O. Kallenberg. *Foundations of Modern Probability*. Probability and its Applications. Springer, 2002.

[Kam+16] B. L. Kaminski, J.-P. Katoen, C. Matheja, and F. Olmedo. "Weakest Precondition Reasoning for Expected Run-Times of Probabilistic Programs". In: *Proc. ESOP*. Lecture Notes in Computer Science Vol. 9632. 2016, pp. 364–389.

[Kam+18] B. L. Kaminski, J.-P. Katoen, C. Matheja, and F. Olmedo. "Weakest Precondition Reasoning for Expected Runtimes of Randomized Algorithms". In: *J. ACM* 65.5 (2018), 30:1–30:68.

[Kam19] B. L. Kaminski. "Advanced Weakest Precondition Calculi for Probabilistic Programs". PhD Thesis. RWTH Aachen University, Germany, 2019.

[Kan60] L. V. Kantorovich. "Mathematical Methods of Organizing and Planning Production". In: *Manage. Sci.* 6.4 (1960), pp. 366–422. In Russian, Leningrad University, 1939.

[Kar84] N. Karmarkar. "A New Polynomial-Time Algorithm for Linear Programming". In: *Proc. STOC*. 1984, pp. 302–311.

[Kat+10] J.-P. Katoen, A. McIver, L. Meinicke, and C. C. Morgan. "Linear-Invariant Generation for Probabilistic Programs: - Automated Support for Proof-Based Methods". In: *Proc. SAS*. Lecture Notes in Computer Science Vol. 6337. 2010, pp. 390–406.

[KB79] R. Kannan and A. Bachem. "Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix". In: *SIAM J. Comput.* 8.4 (1979), pp. 499–507.

[Kei15] K. Keimel. “Healthiness Conditions for Predicate Transformers”. In: Electronic Notes in Theoretical Computer Science Vol. 319 (2015), pp. 255–270.

[Ken+20] G. Kenison, R. Lipton, J. Ouaknine, and J. Worrell. “On the Skolem Problem and Prime Powers”. In: *Proc. ISSAC*. 2020, pp. 289–296.

[Kha80] L. Khachiyan. “Polynomial Algorithms in Linear Programming”. In: *USSR Comput. Math. Math. Phys.* 20.1 (1980), pp. 53–72.

[Kin+18] Z. Kincaid, J. Cyphert, J. Breck, and T. W. Reps. “Non-Linear Reasoning for Invariant Synthesis”. In: *Proc. ACM Program. Lang.* 2.POPL (2018), 54:1–54:33.

[Kin+19] Z. Kincaid, J. Breck, J. Cyphert, and T. W. Reps. “Closed Forms for Numerical Loops”. In: *Proc. ACM Program. Lang.* 3.POPL (2019), 55:1–55:29.

[KK17] B. L. Kaminski and J.-P. Katoen. “A Weakest Pre-Expectation Semantics for Mixed-Sign Expectations”. In: *Proc. LICS*. 2017, pp. 1–12.

[KKM19] B. L. Kaminski, J.-P. Katoen, and C. Matheja. “On the Hardness of Analyzing Probabilistic Programs”. In: *Acta Informatica* 56.3 (2019), pp. 255–285.

[KKM20] B. L. Kaminski, J.-P. Katoen, and C. Matheja. “Expected Runtime Analyis by Program Verification”. In: *Foundations of Probabilistic Programming*. 2020, pp. 185–220.

[Kle52] S. C. Kleene. *Introduction to Metamathematics*. North Holland, 1952.

[KLG20] N. Kobayashi, U. D. Lago, and C. Grellois. “On the Termination Problem for Probabilistic Higher-Order Recursive Programs”. In: *Log. Methods Comput. Sci.* 16.4 (2020).

[Kna28] B. Knaster. “Un Theoreme sur les Functions d’Ensembles”. In: *Ann. Soc. Polon. Math.* 6 (1928), pp. 133–134. with Alfred Tarski.

[Knu92] D. E. Knuth. “Two Notes on Notation”. In: *Am. Math. Monthly* 99.5 (1992), pp. 403–422.

[KoAT] F. Meyer, T. Küspert, M. Hark, J. Giesl, and P. Giesl. *KoAT for Probabilistic Systems*. 2019 – 2021. The source code is available at https://github.com/aprove-developers/KoAT2-Releases/tree/probabilistic.

[Kov08] L. Kovács. “Reasoning Algebraically About P-Solvable Loops”. In: *Proc. TACAS*. Lecture Notes in Computer Science Vol. 4963. 2008, pp. 249–264.

[Koz81] D. Kozen. “Semantics of Probabilistic Programs”. In: *J. Comput. Syst. Sci.* 22.3 (1981), pp. 328–350.

[Koz85] D. Kozen. “A Probabilistic PDL”. In: *J. Comput. Syst. Sci.* 30.2 (1985), pp. 162–178.

[KP11] M. Kauers and P. Paule. *The Concrete Tetrahedron – Symbolic Sums, Recurrence Equations, Generating Functions, Asymptotic Estimates*. Springer, 2011.

[KP97] L. Khachiyan and L. Porkolab. “Computing Integral Points in Convex Semi-Algebraic Sets”. In: *Proc. FOCS*. 1997, pp. 162–171.

[Küs19] T. Küspert. “Computing Expected Runtimes for Sub-Classes of Probabilistic Programs”. Bachelor’s Thesis. RWTH Aachen University, Germany, 2019.

[Lan09] E. Landau. *Handbuch der Lehre von der Verteilung der Primzahlen. 2 / von Edmund Landau.* Teubner, 1909.

[Lar+13] D. Larraz, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio. “Proving Termination of Imperative Programs using Max-SMT”. In: *Proc. FMCAD*. 2013, pp. 218–225.

[Lar+14] D. Larraz, K. Nimkar, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio. “Proving Non-Termination Using Max-SMT”. In: *Proc. CAV*. Lecture Notes in Computer Science Vol. 8559. 2014, pp. 779–796.

[LD60] A. H. Land and A. G. Doig. “An Automatic Method of Solving Discrete Programming Problems”. In: *Econometrica* 28.3 (1960), pp. 497–520.

[Len83a] A. K. Lenstra. “Factoring Polynominals Over Algebraic Number Fields”. In: *Proc. EUROCAL*. Lecture Notes in Computer Science Vol. 162. 1983, pp. 245–254.

[Len83b] H. W. Lenstra Jr. “Integer Programming With a Fixed Number of Variables”. In: *Math. Oper. Res.* 8.4 (1983), pp. 538–548.

[Len92] H. W. Lenstra Jr. “Algorithms in Algebraic Number Theory”. In: *Bull. Amer. Math. Soc. (N.S.)* 26.2 (1992), pp. 211–244.

[LFR21] U. D. Lago, C. Faggian, and S. R. D. Rocca. “Intersection Types and (Positive) Almost-Sure Termination”. In: *Proc. ACM Program. Lang.* 5.POPL (2021), pp. 1–32.

[LH15] J. Leike and M. Heizmann. “Ranking Templates for Linear Loops”. In: *Log. Methods Comput. Sci.* 11.1 (2015).

[LH18] J. Leike and M. Heizmann. "Geometric Nontermination Arguments". In: *Proc. TACAS*. Lecture Notes in Computer Science Vol. 10806. 2018, pp. 266–283.

[Li14] Y. Li. "A Recursive Decision Method for Termination of Linear Programs". In: *Proc. SNC*. 2014, pp. 97–106.

[Li16] Y. Li. "Termination of Single-Path Polynomial Loop Programs". In: *Proc. ICTAC*. Lecture Notes in Computer Science Vol. 9965. 2016, pp. 33–50.

[Li17a] Y. Li. "Termination of Semi-Algebraic Loop Programs". In: *Proc. SETTA*. Lecture Notes in Computer Science Vol. 10606. 2017, pp. 131–146.

[Li17b] Y. Li. "Witness to Non-Termination of Linear Programs". In: *Theor. Comput. Sci.* 681 (2017), pp. 75–100.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovasz. "Factoring Polynomials With Rational Coefficients". In: *Math. Ann.* 261 (1982), pp. 515–534.

[LNS82] J.-L. Lassez, V. L. Nguyen, and L. Sonenberg. "Fixed Point Theorems and Semantics: A Folk Tale". In: *Inf. Process. Lett.* 14.3 (1982), pp. 112–116.

[Mat70] J. V. Matijasevič. "The Diophantineness of Enumerable Sets". In: *Dokl. Akad. Nauk SSSR* 191 (1970), pp. 279–282.

[MB08] L. M. de Moura and N. Bjørner. "Z3: An Efficient SMT Solver". In: *TACAS*. Lecture Notes in Computer Science Vol. 4963. 2008, pp. 337–340.

[McI+18] A. McIver, C. Morgan, B. L. Kaminski, and J.-P. Katoen. "A New Proof Rule for Almost-Sure Termination". In: *Proc. ACM Program. Lang.* 2.POPL (2018), 33:1–33:28.

[Meu+17] A. Meurer, C. P. Smith, M. Paprocki, O. Certík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, S. Roucka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. M. Scopatz. "SymPy: Symbolic Computing in Python". In: *PeerJ Computer Science* 3.e103 (2017).

[MHG21a] F. Meyer, M. Hark, and J. Giesl. 2021. KoAT for Probabilistic Transition Systems: Web interface, binary, Docker image, and examples available at the web site https://aprove-developers.github.io/ExpectedUpperBounds/.

[MHG21b] F. Meyer, M. Hark, and J. Giesl. "Inferring Expected Runtimes of Probabilistic Integer Programs Using Expected Sizes". In: *Proc. TACAS (1)*. Lecture Notes in Computer Science Vol. 12651. 2021, pp. 250–269.

[MM05] A. McIver and C. Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer, 2005.

[MM99] C. Morgan and A. McIver. "An Expectation-Transformer Model for Probabilistic Temporal Logic". In: *Log. J. IGPL* 7.6 (1999), pp. 779–804.

[MMS96] C. Morgan, A. McIver, and K. Seidel. "Probabilistic Predicate Transformers". In: *ACM Trans. Program. Lang. Syst.* 18.3 (1996), pp. 325–353.

[Mon05] D. Monniaux. "Abstract Interpretation of Programs as Markov Decision Processes". In: *Sci. Comput. Program.* 58.1-2 (2005), pp. 179–205.

[Moo+21] M. Moosbrugger, E. Bartocci, J.-P. Katoen, and L. Kovács. "Automated Termination Analysis of Polynomial Probabilistic Programs". In: *Proc. ESOP*. Lecture Notes in Computer Science Vol. 12648. 2021, pp. 491–518.

[Mor96] C. Morgan. "Proof Rules for Probabilistic Loops". In: *Proc. BCS–FACS 7th Refinement Workshop*. Workshops in Computing. 1996.

[MU05] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[NCH18] V. C. Ngo, Q. Carbonneaux, and J. Hoffmann. "Bounded Expectations: Resource Analysis for Probabilistic Programs". In: *Proc. PLDI*. 2018, pp. 496–512. Implementation and artifact available at the web site https://channgo2203.github.io/research/.

[NOW20] E. Neumann, J. Ouaknine, and J. Worrell. "On Ranking Function Synthesis and Termination for Polynomial Programs". In: *Proc. CONCUR*. LIPIcs Vol. 171. 2020, 15:1–15:15.

[OBP16a] S. de Oliveira, S. Bensalem, and V. Prevosto. "Polynomial Invariants by Linear Algebra". In: *Proc. ATVA*. Lecture Notes in Computer Science Vol. 9938. 2016, pp. 479–494.

[OBP16b] S. de Oliveira, S. Bensalem, and V. Prevosto. "Polynomial Invariants by Linear Algebra". In: *CoRR* abs/1611.07726 (2016).

[OBP17] S. de Oliveira, S. Bensalem, and V. Prevosto. "Synthesizing Invariants by Solving Solvable Loops". In: *Proc. ATVA*. Lecture Notes in Computer Science Vol. 10482. 2017, pp. 327–343.

[Olm+16] F. Olmedo, B. L. Kaminski, J.-P. Katoen, and C. Matheja. "Reasoning about Recursive Probabilistic Programs". In: *Proc. LICS*. 2016, pp. 672–681.

[Olm+18] F. Olmedo, F. Gretz, N. Jansen, B. L. Kaminski, J.-P. Katoen, and A. McIver. "Conditioning in Probabilistic Programming". In: *ACM Trans. Program. Lang. Syst.* 40.1 (2018), 4:1–4:50.

[OPW15] J. Ouaknine, J. a. S. Pinto, and J. Worrell. "On Termination of Integer Linear Loops". In: *Proc. SODA*. 2015, pp. 957–969.

[ORY09] S. Orange, G. Renault, and K. Yokoyama. "Computation Schemes for Splitting Fields of Polynomials". In: *Proc. ISSAC*. 2009, pp. 279–286.

[OW14a] J. Ouaknine and J. Worrell. "Positivity Problems for Low-Order Linear Recurrence Sequences". In: *Proc. SODA*. 2014, pp. 366–379.

[OW14b] J. Ouaknine and J. Worrell. "On the Positivity Problem for Simple Linear Recurrence Sequences," in: *Proc. ICALP (2)*. Lecture Notes in Computer Science Vol. 8573. 2014, pp. 318–329.

[OW14c] J. Ouaknine and J. Worrell. "Ultimate Positivity is Decidable for Simple Linear Recurrence Sequences". In: *Proc. ICALP (2)*. Lecture Notes in Computer Science Vol. 8573. 2014, pp. 330–341.

[OW15] J. Ouaknine and J. Worrell. "On Linear Recurrence Sequences and Loop Termination". In: *SIGLOG News* 2.2 (2015), pp. 4–13.

[Par69] D. Park. "Fixpoint Induction and Proofs of Program Properties". In: *Machine Intelligence* 5 (1969).

[PDM17] A. D. Pia, S. S. Dey, and M. Molinaro. "Mixed-Integer Quadratic Programming is in NP". In: *Math. Program.* 162.1-2 (2017), pp. 225–240.

[Pér+20] V. Pérez, M. Klemen, P. López-García, J. F. Morales, and M. V. Hermenegildo. "Cost Analysis of Smart Contracts Via Parametric Resource Analysis". In: *Proc. SAS*. Lecture Notes in Computer Science Vol. 12389. 2020, pp. 7–31.

[Plo76] G. D. Plotkin. "A Powerdomain Construction". In: *SIAM J. Comput.* 5.3 (1976), pp. 452–487.

[Pól30] G. Pólya. "Eine Wahrscheinlichkeitsaufgabe in der Kundenwerbung". In: *Zeitschrift für Angewandte Mathematik und Mechanik* 10.1 (1930), pp. 96–97.

[PR04] A. Podelski and A. Rybalchenko. "A Complete Method for the Synthesis of Linear Ranking Functions". In: *Proc. VMCAI*. Lecture Notes in Computer Science Vol. 2937. 2004, pp. 239–251.

[Put94] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[RaML] RaML (Resource Aware ML): Web interface available at the web site https://www.raml.co/interface/.

[Ren92] J. Renegar. "On the Computational Complexity and Geometry of the First-Order Theory of the Reals, Part I: Introduction. Preliminaries. The Geometry of Semi-Algebraic Sets. The Decision Problem for the Existential Theory of the Reals." In: *J. Symb. Comput.* 13.3 (1992), pp. 255–300.

[RK04] E. Rodríguez-Carbonell and D. Kapur. "Automatic Generation of Polynomial Loop Invariants: Algebraic Foundation". In: *Proc. ISSAC*. 2004, pp. 266–273.

[RK07] E. Rodríguez-Carbonell and D. Kapur. "Generating all Polynomial Invariants in Simple Loops". In: *J. Symb. Comput.* 42.4 (2007), pp. 443–476.

[Rob49] J. Robinson. "Definability and Decision Problems in Arithmetic". In: *J. of Symb. Log.* 14.2 (1949), pp. 98–114.

[Roc18] D. S. Roche. "What Can (and Can't) we Do With Sparse Polynomials?" In: *Proc. ISSAC*. 2018, pp. 25–30.

[Rou62] E. Rouché. "Mémoire sur la Série de Langrange". In: *J. Éc. polytech.* 22 (1862), pp. 193–224.

[RV96] J.-L. Roch and G. Villard. "Fast Parallel Computation of the Jordan Normal Form of Matrices". In: *Parallel Process. Lett.* 06.02 (1996), pp. 203–212.

[Sch09] M. Schaefer. "Complexity of Some Geometric and Topological Problems". In: *Proc. GD*. Lecture Notes in Computer Science Vol. 5849. 2009, pp. 334–344.

[SchSt17] M. Schaefer and D. Stefankovic. "Fixed Points, Nash Equilibria, and the Existential Theory of the Reals". In: *Theory Comput. Syst.* 60.2 (2017), pp. 172–193.

[Sid19] S. V. Sidorov. "On the Similarity of Certain Integer Matrices With Single Eigenvalue Over the Ring of Integers". In: *Math. Notes* 105.5 (2019), pp. 756–762.

[SL96] A. Storjohann and G. Labahn. "Asymptotically Fast Computation of Hermite Normal Forms of Integer Matrices". In: *Proc. ISSAC*. 1996, pp. 259–266.

[Spi64] F. Spitzer. *Principles of Random Walk*. Springer, 1964.

[SSH17] A. Srikanth, B. Sahin, and W. R. Harris. "Complexity Verification Using Guided Theorem Enumeration". In: *Proc. POPL*. 2017, pp. 639–652.

[SSM04] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. "Constraint-Based Linear-Relations Analysis". In: *Proc. SAS*. Vol. 3148. Lecture Notes in Computer Science. 2004, pp. 53–68.

[SZV17] M. Sinn, F. Zuleger, and H. Veith. "Complexity and Resource Bound Analysis of Imperative Programs Using Difference Constraints". In: *J. Autom. Reason.* 59.1 (2017), pp. 3–45.

[Tar55] A. Tarski. "A Lattice-Theoretical Fixpoint Theorem and its Applications". In: *Pac. J. Math.* 5.2 (1955), pp. 285–309.

[Tar98] A. Tarski. "A Decision Method for Elementary Algebra and Geometry". In: *Quantifier Elimination and Cylindrical Algebraic Decomposition*. 1998, pp. 24–84. Originally appeared in 1951, University of California Press, Berkeley and Los Angeles.

[Tiw04] A. Tiwari. "Termination of Linear Programs". In: *Proc. CAV*. Lecture Notes in Computer Science Vol. 3114. 2004, pp. 70–82.

[TPDB] *TPDB (Termination Problems Data Base)*. URL: http://termination-portal.org/wiki/TPDB.

[Tur37] A. M. Turing. "On Computable Numbers, With an Application to the Entscheidungsproblem". In: *Proc. London Math. Soc.* s2-42.1 (1937), pp. 230–265.

[Var85] M. Y. Vardi. "Automatic Verification of Probabilistic Concurrent Finite-State Programs". In: *Proc. FOCS*. 1985, pp. 327–338.

[Wan+19] P. Wang, H. Fu, A. K. Goharshady, K. Chatterjee, X. Qin, and W. Shi. "Cost Analysis of Nondeterministic Probabilistic Programs". In: *Proc. PLDI*. 2019, pp. 204–220.

[WHR21] D. Wang, J. Hoffmann, and T. Reps. "Central Moment Analysis for Cost Accumulators in Probabilistic Programs". In: *Proc. PLDI*. 2021, pp. 559–573.

[Wik21] Wikipedia. *List of software bugs — Wikipedia, The Free Encyclopedia*. 2021. URL: http://en.wikipedia.org/w/index.php?title=List%5C%20of%5C%20software%5C%20bugs&oldid=1004770905 (visited on 02/05/2021).

[WKH20] D. Wang, D. M. Kahn, and J. Hoffmann. "Raising Expectations: Automating Expected Cost Analysis With Types". In: *Proc. ACM Program. Lang.* 4.ICFP (2020).

[Wu+10] B. Wu, L. Shen, Z. Bi, and Z. Zeng. “Termination of Loop Programs With Polynomial Guards”. In: *Proc. ICCSA (4)*. Lecture Notes in Computer Science Vol. 6019. 2010, pp. 482–496.

[Xia+11] B. Xia, L. Yang, N. Zhan, and Z. Zhang. “Symbolic Decision Procedure for Termination of Linear Programs”. In: *Formal Aspects Comput.* 23.2 (2011), pp. 171–190.

[XL13] M. Xu and Z.-B. Li. “Symbolic Termination Analysis of Solvable Loops”. In: *J. Symb. Comput.* 50 (2013), pp. 28–49.

[XZ10] B. Xia and Z. Zhang. “Termination of Linear Programs With Nonlinear Constraints”. In: *J. Symb. Comput.* 45.11 (2010), pp. 1234–1249.

[Zul19] F. Zuleger. “The Polynomial Complexity of Vector Addition Systems With States”. In: *CoRR* abs/1907.01076 (2019).

Appendix A

Details on Probability Theory

This section is devoted to a more detailed introduction of the concepts from measure and probability theory that we use in Part III and the related proofs. For further details we refer to [Bau11a; Bau11b; GS01; Kal02]. Even if we state the proofs of some of the results here, these results are common knowledge in the related fields (see, e.g., [Bau11a; Bau11b; GS01; Kal02]).

A.1 σ-Fields

When setting up a probability space over some sample space Ω, which can be any set, we have to distinguish the sets whose probabilities we want to be able to measure. The collection of these measurable sets is called a σ-field as defined in Def. 14.4.3.

In the setting of program verification, we have seen that Ω is the set of all runs. We regard σ-fields $\mathcal{F} \subseteq \mathfrak{P}(\Omega)$ of the form $\langle \mathcal{E} \rangle_\sigma$, where $\mathcal{E}$ is a collection of cylinder sets. (More precisely, we regard fields $\mathcal{F}_n \subseteq \mathfrak{P}(\Omega)$, where $\mathcal{F}_n$ is the smallest σ-field containing all cylinder sets of order n or smaller). In our setting, a set of runs $\mathcal{E}$ is a *cylinder set* of order n if all runs in $\mathcal{E}$ have the same $n+1$ first configurations, and all the following configurations can be arbitrary (see Def. 14.4.4).

Let $(\mathcal{F}_i)_{i \in I}$ be a family of σ-fields of Ω. Then $\bigcap_{i \in I} \mathcal{F}_i$ is also a σ-field of Ω. This leads to the formal definition of "generated" σ-algebras. For any set $\mathcal{E}$ of subsets of Ω, let $\langle \mathcal{E} \rangle_\sigma \subseteq \mathcal{F}$ consist of all elements that are contained in all σ-fields that are supersets of $\mathcal{E}$. The mapping from $\mathcal{E}$ to $\langle \mathcal{E} \rangle_\sigma$ is also called *σ-operator*.

Definition A.1.1 (Generating σ-Fields)**.** Let $\mathcal{E} \subseteq \mathfrak{P}(\Omega)$. Then the smallest

σ-field of Ω containing $\mathcal{E}$ is

$$\langle \mathcal{E} \rangle_\sigma = \bigcap_{\mathcal{E} \subseteq \mathcal{F},\ \mathcal{F}\ \sigma\text{-field of } \Omega} \mathcal{F}.$$

It turns out that there is a special case in which the generated σ-field is easy to describe, namely in the case where a countable covering of the space Ω is given. This special type of a σ-field can be used to describe the elements of the σ-fields $\mathcal{F}_n^{\text{loop}}$ (see Def. 14.4.16) and $\mathcal{G}_n^{\text{loop}}$ (see Def. 14.4.22).

Lemma A.1.2 (Generating σ-Fields for Covering of Ω)**.** *If* $\Omega = \biguplus_{i \in \mathbb{N}} A_i$ *for a sequence* $A_i \in \mathfrak{P}(\Omega)$ *and* $\mathcal{H} = \langle \{A_i \mid i \in \mathbb{N}\} \rangle_\sigma$ *then*

$$\mathcal{H} = \underbrace{\left\{ \biguplus_{i \in J} A_i \;\middle|\; J \subseteq \mathbb{N} \right\}}_{=\mathcal{G}}.$$

Proof. Showing that $\mathcal{G}$ is a σ-field is enough to prove the desired result: it contains all the sets A_i and every σ-algebra containing all the A_i has to contain all their countable unions, i.e., $\mathcal{G}$:

» $\Omega \in \mathcal{G}$ by choosing $J = \mathbb{N}$.

» Let $J \subseteq \mathbb{N}$. Then $\Omega \setminus \left(\biguplus_{i \in J} A_i \right) = \biguplus_{i \in \mathbb{N} \setminus J} A_i \in \mathcal{G}$.

» Let $J_n \subseteq \mathbb{N}$. Then $\bigcup_{n \in \mathbb{N}} \biguplus_{i \in J_n} A_i = \biguplus_{i \in \bigcup_{n \in \mathbb{N}} J_n} A_i \in \mathcal{G}$.

□

A.2 Probability Spaces

Intuitively, a measurable space $(\Omega, \mathcal{F})$ (see Def. 14.4.3) provides the structure for defining a *measure*. Probability spaces are measurable spaces $(\Omega, \mathcal{F})$ attached with a so-called probability measure $\mathbb{P}$ (see Def. 14.4.6) where the measure of the sample space is 1, i.e., $\mathbb{P}(\Omega) = 1$. Here, the intuition for a probability measure $\mathbb{P}$ is that for any set $A \in \mathcal{F}$, $\mathbb{P}(A)$ is the probability that an element chosen from Ω is contained in A.

In this thesis we consider properties that hold almost-surely (see Def. 15.2.2), for example almost-sure termination or almost-sure convergence. If a property α holds almost-surely it does not need to hold for all $\vartheta \in \Omega$. However, the measure $\mathbb{P}$ cannot

distinguish $A_\alpha = \Omega$ and $A_\alpha \neq \Omega$ if $\mathbb{P}(A_\alpha) = 1$, so in the sense of $\mathbb{P}$, almost-surely holding properties can be considered as holding globally. Regarding almost-sure properties, we state the famous lemma of Borel-Cantelli (see, e.g., [GS01]).

Lemma A.2.1 (Lemma of Borel-Cantelli)**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $A_i \in \mathcal{F}$ such that $\sum_{i\in\mathbb{N}} \mathbb{P}(A_i) < \infty$. Then*

$$\mathbb{P}\left(\bigcap_{n\in\mathbb{N}} \bigcup_{i\geq n} A_i\right) = 0$$

A.3 Integrals of Arbitrary Measures

We now introduce a notion of an integral with respect to an arbitrary measure. Therefore, we fix a measurable space $(\Omega, \mathcal{F})$ and a measure m. The objective is to define a "mean" of a measurable function f. The basic idea is to partition the image of f into sets A_i on which f has a constant value α_i. Then we compute the weighted average of the α_i, where the weights are the measures of the A_i. This definition is fine if f takes only finitely many values (these functions are called *elementary*). If f takes infinitely (countably or even uncountably) many values, we have to approximate f step by step by such functions with finite image. This yields a limit process. In this thesis we only consider the cases where $m = \mathbb{P}$ is a probability measure or a probability submeasure (i.e., $\mathbb{P}(\Omega) \leq 1$).

Definition A.3.1 (Elementary Function, [Bau11a, Def. 10.1])**.** An *elementary function* is a non-negative measurable function $f : \Omega \longrightarrow \overline{\mathbb{R}}_{\geq 0}$ taking only finitely many finite values, i.e., there exist $A_1, \ldots, A_n \in \mathcal{F}$ and $\alpha_1, \ldots, \alpha_n \geq 0$ such that

$$f = \sum_{i=1}^{n} \alpha_i \cdot [A_i].$$

Given an elementary function, the decomposition into a linear combination of indicator functions is not unique. But to define an integral we have to guarantee that its value does not depend on the chosen decomposition.

Lemma A.3.2 (Decomposition of Elementary Functions [Bau11a, Lemma 10.2])**.** *Let $f = \sum_{i=1}^{n} \alpha_i \cdot [A_i] = \sum_{j=1}^{n} \beta_j \cdot [B_j]$ be an elementary function, where $\alpha_i, \beta_j \geq 0$, $A_i, B_j \in \mathcal{F}$ for all i and j. Then*

$$\sum_{i=1}^{n} \alpha_i \cdot m(A_i) = \sum_{j=1}^{n} \beta_j \cdot m(B_j).$$

Hence, we can now define the integral of an elementary function.

Definition A.3.3 (Integral of Elementary Function, [Bau11a, Def. 10.3])**.** Let $f = \sum_{i=1}^{n} \alpha_i \cdot [A_i]$ be an elementary function. Then we define its integral w.r.t. m:

$$\int_\Omega f \, dm = \int f \, dm = \sum_{i=1}^{n} \alpha_i \cdot m(A_i).$$

The well-definedness is justified by Lemma A.3.2 as it shows the independence of the chosen decomposition of the elementary function.

However, the measurable functions we use are not elementary. They take arbitrary (countably or even uncountably) many values. So we have to generalize Def. A.3.3. It can be shown that any non-negative measurable function is the limit of a monotonically increasing sequence of elementary functions.

Theorem A.3.4 (Limit of Elementary Functions, [Bau11a, Cor. 11.2., Thm. 11.6])**.** *Let $f : \Omega \longrightarrow \overline{\mathbb{R}}_{\geq 0}$ be a non-negative measurable function. Then there exists a monotonic sequence $f_0 \leq f_1 \leq \ldots$ of elementary functions such that*

$$f = \sup_{n \in \mathbb{N}} f_n = \lim_{n \longrightarrow \omega} f_n.$$

Furthermore, for any two such sequences $(f_n)_{n\in\mathbb{N}}$ and $(g_n)_{n\in\mathbb{N}}$ we have

$$\sup_{n\in\mathbb{N}} \int f_n \, dm = \sup_{n\in\mathbb{N}} \int g_n \, dm.$$

This theorem justifies the following definition of an integral for an arbitrary non-negative function.

Definition A.3.5 (Integral of Arbitrary Functions, [Bau11a, Def. 11.3])**.** Let $f : \Omega \longrightarrow \overline{\mathbb{R}}_{\geq 0}$ be a non-negative measurable function and $(f_n)_{n\in\mathbb{N}}$ a monotonic sequence of elementary functions such that $f = \sup_{n\in\mathbb{N}} f_n$. Then we define the integral of f w.r.t. m by

$$\int_\Omega f \, dm = \int f \, dm = \sup_{n\in\mathbb{N}} \int f_n \, dm.$$

Before we state the properties of the integral used in this thesis we define the integral on a measurable subset of Ω.

Definition A.3.6 (Integral on Measurable Subset, [Bau11a, Def. 12.4])**.** Let $f : \Omega \longrightarrow \overline{\mathbb{R}}_{\geq 0}$ be a non-negative measurable function and $A \in \mathcal{F}$. Then $f \cdot [A]$ is non-negative and measurable, and we define

$$\int_A f \, dm = \int f \cdot [A] \, dm.$$

With this definition of an integral, a very special property holds for monotonically increasing sequences of non-negative functions: taking the limit (it always exists due to monotonicity) and the integral can be intertwined.

Theorem A.3.7 (Monotone Convergence Theorem, [Bau11a, Thm. 11.4])**.** *Let $(f_n)_{n\in\mathbb{N}}$ be a monotonic sequence of non-negative measurable functions, i.e., $f_n : \Omega \longrightarrow \overline{\mathbb{R}}_{\geq 0}$ is measurable and $f_0 \leq f_1 \leq \ldots$. Then $\sup_{n\in\mathbb{N}} f_n = \lim_{n\longrightarrow\omega} f_n$ is measurable and*

$$\lim_{n\longrightarrow\omega} \int f_n \, dm = \int \lim_{n\longrightarrow\omega} f_n \, dm.$$

The integral now satisfies certain properties.

Lemma A.3.8 (Properties of the Integral,[Bau11a, Eq. (11.5), (11.6), (11.7), (11.8), (12.6), (12.8)])**.** *Let $a, b \geq 0$, $f, g : \Omega \longrightarrow \overline{\mathbb{R}}_{\geq 0}$ be measurable functions, and $A, A_i \in \mathcal{F}$. Then*

(i) **Linearity:** $\int_A a \cdot f + b \cdot g \, dm = a \cdot \int_A f \, dm + b \cdot \int_A g \, dm.$

(ii) **Measure:** $\int_A 1 \, dm = m(A).$

(iii) **Additivity:** $\int_{\biguplus_{i\in\mathbb{N}} A_i} f \, dm = \sum_{i\in\mathbb{N}} \int_{A_i} f \, dm.$

(iv) **Monotonicity:** $f \leq g \implies \int_A f \, dm \leq \int_A g \, dm.$

A.4 Random Variables

A measurable map $X : \Omega \longrightarrow \overline{\mathbb{R}}_{\geq 0}$ is a *random variable* (see Def. 14.4.11), i.e., X maps elements of Ω to $\overline{\mathbb{R}}_{\geq 0}$. If $\mathbb{P}$ is a probability measure for Ω (i.e., for $A \subseteq \Omega$, $\mathbb{P}(A)$ is the probability that an element chosen from Ω is contained in A), one obtains a corresponding probability measure $\mathbb{P}^X$ for $\overline{\mathbb{R}}_{\geq 0}$. For $A' \in \mathcal{B}(\overline{\mathbb{R}}_{\geq 0})$, $\mathbb{P}\left(X^{-1}\left(A'\right)\right)$ is the probability that an element chosen from Ω is mapped by X to an element contained in A'. In other words, instead of regarding the probabilities for choosing elements from A, one now regards the probabilities for the values of the random variable X.

It is $\mathbb{E}(X) = \int_\Omega X d\mathbb{P}$ and thus, the expected value satisfies the properties in Lemma A.3.8 . If the image of X is countable, we have the following alternative characterization of the expected value (see Def. 14.4.11).

Lemma A.4.1 (Expected Value as Sum)**.** *If X is a discrete random variable we have $\mathbb{E}(X) = \sum_{r \in \overline{\mathbb{R}}_{\geq 0}} r \cdot \mathbb{P}(X = r)$. Note that this series has only countably many non-zero non-negative addends. Hence, it either converges or it diverges to infinity.*

Proof. Let $X(\Omega) = \{r_1, r_2, \dots\}$. Then $\Omega = \biguplus_{i \in \mathbb{N}} X^{-1}(\{r_i\})$. Hence,

$$
\begin{aligned}
& \mathbb{E}(X) \\
= & \int X d\mathbb{P} \\
= & \int_{\biguplus_{i \in \mathbb{N}} X^{-1}(\{r_i\})} X \, d\mathbb{P} \\
= & \sum_{i \in \mathbb{N}} \int_{X^{-1}(\{r_i\})} X \, d\mathbb{P} && \text{(by Lemma A.3.8 (Additivity))} \\
= & \sum_{i \in \mathbb{N}} \int_{X^{-1}(\{r_i\})} r_i \, d\mathbb{P}X && (X \text{ is constant on } X^{-1}(\{r_i\})) \\
= & \sum_{i \in \mathbb{N}} r_i \cdot \int_{X^{-1}(\{r_i\})} 1 \, d\mathbb{P} && \text{(by Lemma A.3.8 (Linearity))} \\
= & \sum_{i \in \mathbb{N}} r_i \cdot \mathbb{P}\left(X^{-1}(\{r_i\})\right) && \text{(by Lemma A.3.8 (Measure))} \\
= & \sum_{i \in \mathbb{N}} r_i \cdot \mathbb{P}(X = r_i) \\
= & \sum_{r \in \overline{\mathbb{R}}_{\geq 0}} r \cdot \mathbb{P}(X = r).
\end{aligned}
$$

□

A.5 Uniform Integrability

Given any stochastic process (i.e., a sequence of random variables, $X_n : \Omega \longrightarrow \overline{\mathbb{R}}_{\geq 0}$ on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$) that has an almost-surely existing limit the question arises whether we can construct the expectation of the limit as the limit of the expectations of the X_n. However, this is false in general, a counterexample is given in Counterex. 15.3.2. Therefore, we distinguish stochastic processes with this special property and call it *uniform integrability (u.i.)*, see Def. 15.3.1.

To check for uniform integrability of certain stochastic processes is one of the main purposes of Ch. 15. There are two sufficient criteria which we list below. The first one is the "Monotone Convergence Theorem for Random Variables", a corollary of Thm. A.3.7. It states that a sequence of monotonically increasing random variables is always uniformly integrable.

Corollary A.5.1 (Monotone Convergence Theorem for Random Variables, [Bau11a, Thm. 11.4])**.** *Let $(X_n)_{n\in\mathbb{N}}$ be a monotonic sequence of non-negative random variables, i.e., $X_n : \Omega \longrightarrow \overline{\mathbb{R}}_{\geq 0}$ is measurable and $X_0 \leq X_1 \leq \ldots$. Then $\lim_{n\longrightarrow\omega} X_n : \Omega \longrightarrow \overline{\mathbb{R}}_{\geq 0}$ is measurable and*

$$\lim_{n\longrightarrow\omega} \mathbb{E}(X_n) = \mathbb{E}\left(\lim_{n\longrightarrow\omega} X_n\right).$$

The second sufficient criterion states that if the sequence is bounded by an integrable random variable M, uniform integrability is given as well.

Lemma A.5.2 (Bounded Stochastic Processes are U.I., [GS01, Thm. 7.10.4])**.** *Let $(X_n)_{n\in\mathbb{N}}$ be a sequence of random variables and M a non-negative random variable on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with $X_n \leq M$ for all $n \in \mathbb{N}$. If $\mathbb{E}(M) < \infty$, $(X_n)_{n\in\mathbb{N}}$ is uniformly integrable.*

A.6 Conditional Expected Values

We introduced the notion of the conditional expected value w.r.t. a sub-σ-field on a fixed probability space $(\Omega, \mathcal{F}, \mathbb{P})$ in Def. 14.4.24. The idea is that given a random variable X and a subfield $\mathcal{G} \subseteq \mathcal{F}$ we would like to approximate X by another $\mathcal{G}$-measurable random variable w.r.t. expectation. Intuitively, this means that we want to construct a (possibly infinite) non-negative linear combination of the functions $[G], G \in \mathcal{G}$, in such a way that restricted to a set $G \in \mathcal{G}$, the random variable X and this linear combination have the same average value w.r.t. $\mathbb{P}$. We now recapitulate some results (see, e.g, [Bau11b, Sec. 15]) needed in the proofs of this thesis.

If the sub-σ-field has the special structure described in Lemma A.1.2, the property just stated only needs to be checked on the generators.

Lemma A.6.1 (Conditional Expected Value)**.** *Let X be a random variable. If $\Omega = \biguplus_{i\in\mathbb{N}} A_i$ for a sequence $A_i \in \mathcal{F}$ and $\mathcal{G} = \langle \{A_i \mid i \in \mathbb{N}\} \rangle_\sigma$, a $\mathcal{G}$-measurable function Y is a conditional expected value of X w.r.t. $\mathcal{G}$ iff*

$$\mathbb{E}(X \cdot [A_i]) = \mathbb{E}(Y \cdot [A_i]), \text{ for all } i \in \mathbb{N}.$$

Proof. By definition it is left to show that $\mathbb{E}(X \cdot [A_i]) = \mathbb{E}(Y \cdot [A_i])$ for all $i \in \mathbb{N}$ implies $\mathbb{E}(X \cdot [G]) = \mathbb{E}(Y \cdot [G])$ for any $G \in \mathcal{G}$. But due to Lemma A.1.2 it is enough to show this for any disjoint union $\biguplus_{i \in J} A_i$. We have $\left[\biguplus_{i \in J} A_i\right] = \sum_{i \in J} [A_i]$ and this series always converges pointwise as at most one of the addends is non-zero for any $\alpha \in \Omega$. Hence we have

$$\begin{aligned}
&\mathbb{E}\left(X \cdot \left[\biguplus_{i \in J} A_i\right]\right) \\
&= \mathbb{E}\left(X \cdot \sum_{i \in J} [A_i]\right) \\
&= \sum_{i \in J} \mathbb{E}(X \cdot [A_i]) && \text{(by Lemma A.3.8)} \\
&= \sum_{i \in J} \mathbb{E}(Y \cdot [A_i]) \\
&= \mathbb{E}\left(Y \cdot \left[\biguplus_{i \in J} A_i\right]\right).
\end{aligned}$$

□

Furthermore, in this special case, we can indeed construct the conditional expectation as a non-negative linear combination of the functions $[G]$.

Lemma A.6.2 (Computation of Conditional Expected Value)**.** *Let X be a random variable. If $\Omega = \biguplus_{i \in \mathbb{N}} A_i$ for a sequence $A_i \in \mathcal{F}$ and $\mathcal{G} = \langle \{A_i \mid i \in \mathbb{N}\} \rangle_\sigma$, we have*

$$\mathbb{E}(X \mid \mathcal{G}) = \sum_{\substack{i \in \mathbb{N}, \\ \mathbb{P}(A_i) > 0}} [A_i] \cdot \frac{\mathbb{E}(X \cdot [A_i])}{\mathbb{P}(A_i)},$$

where we use the convention $c \cdot \infty = \infty$ if $c > 0$ and $0 \cdot \infty = 0$.

Proof. Let $Y = \sum_{i \in \mathbb{N}, \mathbb{P}(A_i) > 0} [A_i] \cdot \frac{\mathbb{E}(X \cdot [A_i])}{\mathbb{P}(A_i)}$. Then Y is obviously $\mathcal{G}$-measurable. By Lemma A.6.1, it is left to show that $\mathbb{E}(X \cdot [A_i]) = \mathbb{E}(Y \cdot [A_i])$ for all $i \in \mathbb{N}$ with

$\mathbb{P}(A_i) > 0$ since otherwise both sides are 0. We have

$$\begin{aligned}
&\mathbb{E}(Y \cdot [A_i]) \\
&= \mathbb{E}\left([A_i] \cdot \frac{\mathbb{E}(X \cdot [A_i])}{\mathbb{P}(A_i)}\right) && \text{(as the } A_i \text{ are pairwise disjoint)} \\
&= \mathbb{E}([A_i]) \cdot \frac{\mathbb{E}(X \cdot [A_i])}{\mathbb{P}(A_i)} \\
&= \mathbb{P}(A_i) \cdot \frac{\mathbb{E}(X \cdot [A_i])}{\mathbb{P}(A_i)} && \text{(as } \mathbb{P}(A_i) > 0) \\
&= \mathbb{E}(X \cdot [A_i]).
\end{aligned}$$

This concludes the proof. □

So, in this special case a conditional expectation always exists. Note that a conditional expectation is almost surely unique (see, e.g., [Bau11b; GS01]). In this thesis, whenever we talk about the conditional expected value we are in the situation of Lemma A.6.2 (see Thm. 14.4.25, 15.3.10, and 15.7.8). This justifies why we defined "the conditional expectation" and did not discuss its existence. Moreover, we can choose the value of the conditional expectation on the sets A_i with $\mathbb{P}(A_i) = 0$ arbitrarily. For example, in the proofs of Thm. 14.4.25 and 15.7.8 (see App. B.21 and B.33) we choose its value on the A_i with $\mathbb{P}(A_i) = 0$ in a way which is in some sense consistent with its value on the A_i with $\mathbb{P}(A_i) > 0$. To over-approximate a conditional expectation we can use the following corollary from Lemma A.6.2.

Corollary A.6.3 (Over-Approximating Conditional Expected Value)**.** *Let X be a random variable. If $\Omega = \biguplus_{i \in \mathbb{N}} A_i$ for a sequence $A_i \in \mathcal{F}$ and $\mathcal{G} = \langle \{A_i \mid i \in \mathbb{N}\} \rangle_\sigma$, a $\mathcal{G}$-measurable function Y satisfies $\mathbb{E}(X \mid \mathcal{G}) \leq Y$ iff*

$$\mathbb{E}(X \cdot [A_i]) \leq \mathbb{E}(Y \cdot [A_i]), \text{ for all } i \in \mathbb{N}.$$

Proof. Since Y is $\mathcal{G}$-measurable and the conditional expectation is almost-surely unique, we have

$$Y = \sum_{\substack{i \in \mathbb{N}, \\ \mathbb{P}(A_i) > 0}} [A_i] \cdot \frac{\mathbb{E}(Y \cdot [A_i])}{\mathbb{P}(A_i)}$$

by Lemma A.6.2. Again, by Lemma A.6.2 the result follows. □

Recall that $\mathbb{E}(X \mid \mathcal{G})$ is a random variable that is like X, but for those elements that are not distinguishable in the sub-σ-field $\mathcal{G}$, it "distributes the value of X equally". This statement is formulated by the following lemma.

Lemma A.6.4 (Expected Value Does Not Change When Regarding Conditional Expected Values, [Bau11b, Eq. (15.6)]). *Let X be a random variable on $(\Omega, \mathcal{F}, \mathbb{P})$ and let $\mathcal{G}$ be a sub-σ-field of $\mathcal{F}$. Then*

$$\mathbb{E}(X) = \mathbb{E}(\mathbb{E}(X \mid \mathcal{G}))$$

Proof.

$$\begin{aligned} & \mathbb{E}(\mathbb{E}(X \mid \mathcal{G})) \\ = \ & \mathbb{E}(\mathbb{E}(X \mid \mathcal{G}) \cdot [\Omega]) && \text{(as } [\Omega] \equiv 1) \\ = \ & \mathbb{E}(X \cdot [\Omega]) && \text{(by Def. 14.4.24, as } \Omega \in \mathcal{G}) \\ = \ & \mathbb{E}(X) && \text{(as } [\Omega] \equiv 1) \end{aligned}$$

□

The following theorem shows (i) that linear operations carry over to conditional expected values w.r.t. sub-σ-fields, (ii) that every random variable approximates itself if it is already measurable w.r.t. the sub-σ-field $\mathcal{G}$, and (iii) it allows to simplify multiplications with $\mathcal{G}$-measurable random variables.

Theorem A.6.5 (Properties of Conditional Expected Value, [Bau11b, Eq. (15.6), (15.10), (15.21)]). *Let X, Y be random variables on $(\Omega, \mathcal{F}, \mathbb{P})$, let $\mathcal{G}$ be a sub-σ-field of $\mathcal{F}$, and let $a, b \in \mathbb{R}_{\geq 0}$. Then the following properties hold.*

(i) $\mathbb{E}(a \cdot X + b \cdot Y \mid \mathcal{G}) = a \cdot \mathbb{E}(X \mid \mathcal{G}) + b \cdot \mathbb{E}(Y \mid \mathcal{G})$.

(ii) If X is itself $\mathcal{G}$-measurable, we have $\mathbb{E}(X \mid \mathcal{G}) = X$.

(iii) If X is $\mathcal{G}$-measurable, we have $\mathbb{E}(X \cdot Y \mid \mathcal{G}) = X \cdot \mathbb{E}(Y \mid \mathcal{G})$.

A.7 Martingale Theory

Martingales (Def. 14.4.29) are a powerful concept from probability theory which we use in Part III for proving many results. Here, we state two famous theorems from martingale theory which we need in the proofs for Sect. 15.4 and 16.1.

Theorem A.7.1 (Martingale Convergence Theorem [Bau11b, Thm. 19.1 and 19.3]). *Let $(X_n)_{n\in\mathbb{N}}$ be a submartingale on a probability space $(\Omega, \mathfrak{F}, \mathbb{P})$ w.r.t. a filtration $(\mathfrak{F}_n)_{n\in\mathbb{N}}$. If there is a constant $c \geq 0$ such that $X_n \leq c$ for every $n \in \mathbb{N}$, there exists a random variable X_ω such that $\mathbb{P}(\{\vartheta \in \Omega \mid \lim_{n\to\omega} X_n(\vartheta) = X_\omega(\vartheta)\}) = 1$, i.e., $(X_n)_{n\in\mathbb{N}}$ converges almost-surely to X_ω. Furthermore, $(X_n)_{n\in\mathbb{N}}$ is u.i., i.e., $\lim_{n\to\omega} \mathbb{E}(X_n) = \mathbb{E}(X_\omega)$. The following hold:*

» *If* $(X_n)_{n\in\mathbb{N}}$ *is a submartingale, we have* $\mathbb{E}(X_n) \leq \mathbb{E}(X_\omega)$ *for all* $n \in \mathbb{N}$.

» *If* $(X_n)_{n\in\mathbb{N}}$ *is a martingale, we have* $\mathbb{E}(X_n) = \mathbb{E}(X_\omega)$ *for all* $n \in \mathbb{N}$.

For the following theorem, recall that we only consider non-negative random variables.

Theorem A.7.2 (Supermartingale Convergence Theorem [Bau11b, Cor. 19.2])**.** *Let* $(X_n)_{n\in\mathbb{N}}$ *be a supermartingale on a probability space* $(\Omega, \mathfrak{F}, \mathbb{P})$ *w.r.t. a filtration* $(\mathfrak{F}_n)_{n\in\mathbb{N}}$. *There exists a random variable* X_ω *such that* $\mathbb{P}(\{\vartheta \in \Omega \mid \lim_{n\to\omega} X_n(\vartheta) = X_\omega(\vartheta)\}) = 1$, *i.e.,* $(X_n)_{n\in\mathbb{N}}$ *converges almost-surely to* X_ω *and we have* $\mathbb{E}(X_n) \geq \mathbb{E}(X_\omega)$ *for all* $n \in \mathbb{N}$.

Appendix B

Missing Proofs

B.1 Proof of Thm. 5.1.2

Proof. We prove both statements separately.

(i) Due to weak non-linearity, we have $u_i = m_i \cdot x_i + pol_i$ with $x_i \notin \mathcal{V}(pol_i)$ for all $1 \leq i \leq d$. Then

$$u_i(\vec{u}) = m_i \cdot (m_i \cdot x_i + pol_i) + pol_i(\vec{u}) = m_i^2 \cdot x_i + m_i \cdot pol_i + pol_i(\vec{u}).$$

Assume that $x_i \in \mathcal{V}(pol_i(\vec{u}))$. Since $x_i \notin \mathcal{V}(pol_i)$ by weak non-linearity, there must be an $x_j \in \mathcal{V}(pol_i)$ with $x_j \neq x_i$ and $x_i \in \mathcal{V}(u_j)$. By definition, $x_j \neq x_i$ and $x_i \in \mathcal{V}(u_j)$ imply $x_j >_{\vec{u}} x_i$. But $x_j \in \mathcal{V}(pol_i)$ also implies $x_i >_{\vec{u}} x_j$, which would violate well-foundedness of $>_{\vec{u}}$, i.e., it would contradict the triangularity of $(\varphi, \vec{u})$.

Hence, m_i^2 is the coefficient of x_i in $u_i(\vec{u})$. Since $m_i^2 \geq 0$, this proves that $(\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$ is non-negative.

Note that $x_i >_{\vec{u}(\vec{u})} x_j$ implies $x_j \in \mathcal{V}(pol_i)$ (in this case we also have $x_i >_{\vec{u}} x_j$) or it implies that there is an $x_k \in \mathcal{V}(pol_i)$ with $x_j \in \mathcal{V}(u_k)$ (in this case we have $x_i >_{\vec{u}} x_k$ and $x_k \geq_{\vec{u}} x_j$). So in both cases, $x_i >_{\vec{u}(\vec{u})} x_j$ implies $x_i >_{\vec{u}} x_j$. Thus, we obtain $>_{\vec{u}(\vec{u})} \subseteq >_{\vec{u}}$. As $>_{\vec{u}}$ is well founded, this means that $>_{\vec{u}(\vec{u})}$ is well founded, too. Hence, $(\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$ is triangular.

(ii) Now we prove that $(\varphi, \vec{u})$ does not terminate on $\vec{e} \in \mathcal{S}^d$ iff $(\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u}))$ does not terminate on $\vec{e}$.

$$
\begin{aligned}
& (\varphi, \vec{u}) \text{ does not terminate on } \vec{e} \\
\iff & \forall n \in \mathbb{N}.\ \varphi(\vec{u}^n(\vec{e})) && \text{(by Def. 4.1.3)} \\
\iff & \forall n \in \mathbb{N}.\ \varphi(\vec{u}^{2\cdot n}(\vec{e})) \wedge \varphi(\vec{u}^{2\cdot n+1}(\vec{e})) \\
\iff & \forall n \in \mathbb{N}.\ \varphi(\vec{u}^{2\cdot n}(\vec{e})) \wedge \varphi(\vec{u})(\vec{u}^{2\cdot n}(\vec{e})) \\
\iff & \forall n \in \mathbb{N}.\ (\varphi \wedge \varphi(\vec{u}))\,(\vec{u}(\vec{u}))^n(\vec{e}) \\
\iff & (\varphi \wedge \varphi(\vec{u}), \vec{u}(\vec{u})) \text{ does not terminate on } \vec{e}
\end{aligned}
$$

This proves the second statement.

□

B.2 Proof of Lemma 6.1.4

Proof. Recall that for $f, g : \mathbb{N} \longrightarrow \mathbb{R}$, $f(n) \in o(g(n))$ means

$$\forall m > 0.\ \exists n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ |f(n)| < m \cdot |g(n)|.$$

First consider the case $b_2 > b_1$. We have $b_2^n = b_1^n \cdot \left(\frac{b_2}{b_1}\right)^n$, where $\frac{b_2}{b_1} > 1$. As we clearly have $n^{a_1} \in o\left(\left(\frac{b_2}{b_1}\right)^n\right)$, we obtain $n^{a_1} \cdot b_1^n \in o\left(\left(\frac{b_2}{b_1}\right)^n \cdot b_1^n\right) = o(b_2^n) \subseteq o(n^{a_2} \cdot b_2^n)$, i.e., $n^{a_1} \cdot b_1^n \in o(n^{a_2} \cdot b_2^n)$.

Now consider the case $b_2 = b_1$ and $a_2 > a_1$. Then $n^{a_1} \cdot b_1^n \in o(n^{a_2} \cdot b_2^n)$ trivially holds. □

B.3 Proof of Equation (6.1.6)

Proof. If $p(\vec{e}) = 0$, then $k = 0$ by Def. 6.1.5 and hence $o(p(\vec{e})) = o(k \cdot n^a \cdot b^n) = o(0)$. Otherwise, $p(\vec{e})$ has the form

$$k \cdot n^a \cdot b^n + \textstyle\sum_{i=1}^m k_i \cdot n^{a_i} \cdot b_i^n$$

for $k \neq 0$ and $m \geq 0$. We have $k_i = k_i^{(b_i, a_i)} \in \text{coefs}\left(p(\vec{e})\right)$ and hence $(b, a) >_{lex} (b_i, a_i)$ for all $1 \leq i \leq m$. Thus, Lemma 6.1.4 implies $n^{a_i} \cdot b_i^n \in o(n^a \cdot b^n)$ and we get

$$o(p(\vec{e})) = o\left(k \cdot n^a \cdot b^n + \textstyle\sum_{i=1}^m k_i \cdot n^{a_i} \cdot b_i^n\right) = o(n^a \cdot b^n) = o(k \cdot n^a \cdot b^n).$$

□

B.4 Proof of Equation (6.1.7)

Proof. If $k = 0$, the claim is trivial, so assume $k \neq 0$, i.e., $p(\vec{e}) = k \cdot b^n \cdot n^a + p'$ for some $p' \in \mathbb{NPE}_{\mathcal{S}}$. By Lemma 6.1.4 we have

$$\begin{aligned} & p' \in o(k \cdot b^n \cdot n^a) \\ \iff\ & \forall m > 0.\, \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, |p'| < m \cdot |k \cdot b^n \cdot n^a| \\ \implies\ & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, |p'| < |k \cdot b^n \cdot n^a|. \end{aligned}$$

Assume $k > 0$. Then

$$\begin{aligned} & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, |p'| < |k \cdot b^n \cdot n^a| \\ \implies\ & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, -p' < |k \cdot b^n \cdot n^a| \\ \iff\ & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, -p' < k \cdot b^n \cdot n^a \\ \iff\ & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, 0 < k \cdot b^n \cdot n^a + p' \\ \iff\ & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, 0 < p(\vec{e}) \\ \iff\ & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, \operatorname{sign}\left(p(\vec{e})\right) = \operatorname{sign}(k). \end{aligned}$$

If $k < 0$,

$$\begin{aligned} & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, |p'| < |k \cdot b^n \cdot n^a| \\ \implies\ & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, p' < |k \cdot b^n \cdot n^a| \\ \iff\ & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, p' < -k \cdot b^n \cdot n^a \\ \iff\ & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, k \cdot b^n \cdot n^a + p' < 0 \\ \iff\ & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, p(\vec{e}) < 0 \\ \iff\ & \exists n_0 \in \mathbb{N}.\, \forall n \in \mathbb{N}_{>n_0}.\, \operatorname{sign}\left(p(\vec{e})\right) = \operatorname{sign}(k). \end{aligned}$$

□

B.5 Proof of Thm. 6.1.9

Proof. We have to prove

$$(6.1.10) \iff \exists \vec{x} \in \mathcal{S}^d.\ \operatorname{red}(\xi) \tag{B.5.1}$$

where $\operatorname{red}(\xi)$ results from replacing each atom $p \triangleright 0$ in ξ by $\operatorname{red}(p \triangleright 0)$. Since each $\operatorname{red}(p \triangleright 0)$ can be computed in polynomial time due to Lemma 6.1.7, the computation of the formula "$\exists \vec{x} \in \mathcal{S}^d.\ \operatorname{red}(\xi)$" clearly works in polynomial time.

To prove (B.5.1), we introduce the notion of a *fundamental set*. Let $p_1 \rhd_1 0, \dots, p_k \rhd_k 0$ denote the atoms in ξ. We call a subset $I \subseteq \{1, \dots, k\}$ *fundamental* if $\bigwedge_{i\in I} p_i \rhd_i 0 \implies \xi$. Recall that w.l.o.g., we can assume that ξ does not contain any Boolean connectives except $\wedge$ and $\vee$. Thus, whenever $\xi \neq \text{false}$, the formula ξ must have fundamental sets. Clearly, we have

$$\exists \vec{x} \in \mathcal{S}^d.\ \mathrm{red}(\xi) \iff \exists \text{ fundamental set } I.\ \bigwedge_{i\in I} \mathrm{red}(p_i \rhd_i 0) \text{ is valid.}$$

Thus, to prove (B.5.1), it suffices to show the following:

$$(6.1.10) \iff \exists \text{ fundamental set } I.\ \bigwedge_{i\in I} \mathrm{red}(p_i \rhd_i 0) \text{ is valid.} \tag{B.5.2}$$

For the " $\Longleftarrow$ "-direction of (B.5.2), assume that there is such a fundamental set, i.e.,

$$\bigwedge_{i\in I} \mathrm{red}(p_i \rhd_i 0)$$

is valid. Then as in the proof of Lemma 6.1.7, we obtain that for each $i \in I$, there is an $n_i \in \mathbb{N}$ such that

$$\exists \vec{x} \in \mathcal{S}^d.\ \forall n \in \mathbb{N}_{>n_i}.\ p_i \rhd_i 0.$$

As I is finite, $n_{\max} = \max\{n_i \mid i \in I\}$ exists. Hence, we get

$$\exists \vec{x} \in \mathcal{S}^d.\ \forall n \in \mathbb{N}_{>n_{max}}.\ \bigwedge_{i\in I} p_i \rhd_i 0.$$

Since I is fundamental, this implies (6.1.10).

For the " $\Longrightarrow$ "-direction, assume (6.1.10). Then there is an $\vec{e} \in \mathcal{S}^d$ and an $n_0 \in \mathbb{N}$ such that for each $n \in \mathbb{N}_{>n_0}$, there is a fundamental set I_n such that $\bigwedge_{i\in I_n} p_i(\vec{e}) \rhd_i 0$ holds. As there are only finitely many fundamental sets, there is some fundamental set I that occurs infinitely often in $(I_n)_{n\in\mathbb{N}_{>n_0}}$. Hence we get

$$\exists n_0 \in \mathbb{N}.\ \exists^\infty n \in \mathbb{N}_{>n_0}.\ \bigwedge_{i\in I} p_i(\vec{e}) \rhd_i 0. \tag{B.5.3}$$

By definition of poly-exponential expressions, each $p_i(\vec{e})$ is weakly monotonic in n for large enough n. Thus, (B.5.3) implies

$$\exists n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ \bigwedge_{i\in I} p_i(\vec{e}) \rhd_i 0.$$

As $\vec{e} \in \mathcal{S}^d$, this implies that there is a fundamental set I such that $\bigwedge_{i\in I} \mathrm{red}(p_i \rhd_i 0)$ holds. □

B.6 Proof of Thm. 6.1.10

Proof. By Cor. 5.1.3, termination of *twn*-loops is reducible to termination of *tnn*-loops. Given a *tnn*-loop $(\varphi, \vec{u})$, we obtain $\vec{q}_{norm} \in \left(\mathbb{NPE}_{\mathcal{S}}[\vec{x}]\right)^d$ such that $(\varphi, \vec{u})$ is (eventually) non-terminating iff (6.1.5) holds, where φ is a propositional formula over the atoms $\left\{ \alpha \geq 0, \alpha > 0 \;\middle|\; \alpha \in \mathbb{R}_{\mathbb{A}}[\vec{x}] \right\}$. Hence, $\varphi(\vec{q}_{norm})$ is a propositional formula over the atoms $\left\{ p \triangleright 0 \;\middle|\; p \in \mathbb{NPE}_{\mathcal{S}}[\vec{x}], \triangleright \in \{\geq, >\} \right\}$. Thus, by Thm. 6.1.9, validity of (6.1.5) resp. (6.1.10) is reducible to $\mathrm{Th}_{\exists}(\mathcal{S}, \mathbb{R}_{\mathbb{A}})$. □

B.7 Proof of Cor. 6.1.11

Proof. By Cor. 5.1.3, termination of *twn*-loops is reducible to termination of *tnn*-loops. By Thm. 6.1.10, termination of *tnn*-loops is reducible to invalidity of a formula $\chi \in \mathrm{Th}_{\exists}(\mathcal{S}, \mathbb{R}_{\mathbb{A}})$. If $\mathcal{S} = \mathbb{R}_{\mathbb{A}}$ validity of χ is decidable, and if $\mathcal{S} = \mathbb{Z}$ or $\mathcal{S} = \mathbb{Q}$ validity of χ is semi-decidable [Coh69; Tar98]. But χ is valid iff the loop is non-terminating. Hence, non-termination is decidable for $\mathcal{S} = \mathbb{R}_{\mathbb{A}}$ and semi-decidable if $\mathcal{S} = \mathbb{Z}$ or $\mathcal{S} = \mathbb{Q}$. The claim (ii) for $\mathcal{S} = \mathbb{R}_{\mathbb{A}}$ follows since *deciding* non-termination is equivalent to deciding termination. Finally, (i) and the claim (ii) for $\mathcal{S} = \mathbb{R}$ follow due to elementary equivalence of $\mathbb{R}_{\mathbb{A}}$ and $\mathbb{R}$. □

B.8 Proof of Lemma 6.1.12

Proof. We have:

$$
\begin{aligned}
& \vec{e} \text{ witnesses eventual non-termination of } (\varphi, \vec{u}) \\
\iff & \exists n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ \left(\varphi(\vec{q}_{norm})\right)(\vec{e}) && \text{(by (6.1.5))} \\
\iff & \exists n_0 \in \mathbb{N}.\ \forall n \in \mathbb{N}_{>n_0}.\ \xi(\vec{e}) \\
\iff & \mathrm{red}(\xi)(\vec{e}) && \text{(as in the proof of Thm. 6.1.9)}
\end{aligned}
$$

□

B.9 Proof of Lemma 8.1.5

Proof. Let $rt_{(\varphi,\vec{u})}(\vec{e}) = n_{\vec{e}}$ and $rt_{(\varphi_{\mathrm{ch}},\vec{u}_{\mathrm{ch}})}(\vec{e}) = n'_{\vec{e}}$. Then we have

$$
\begin{aligned}
& n'_{\vec{e}} = \min \left\{ n \in \mathbb{N} \;\middle|\; \neg(\varphi_{\mathrm{ch}}(\vec{u}^{\,n}_{\mathrm{ch}}(\vec{e}))) \right\} && \text{(by definition of } rt_{(\varphi_{\mathrm{ch}},\vec{u}_{\mathrm{ch}})}\text{)} \\
\iff & \forall n < n'_{\vec{e}}.\ \varphi_{\mathrm{ch}}(\vec{u}^{\,n}_{\mathrm{ch}}(\vec{e})) \wedge \neg\varphi_{\mathrm{ch}}(\vec{u}^{\,n'_{\vec{e}}}_{\mathrm{ch}}(\vec{e}))
\end{aligned}
$$

$$\iff \forall n < n'_{\vec{e}}.\ \left(\varphi(\vec{u}^{2\cdot n}(\vec{e})) \wedge \varphi(\vec{u}(\vec{u}^{2\cdot n}(\vec{e})))\right) \wedge \neg\left(\varphi(\vec{u}^{2\cdot n'_{\vec{e}}}(\vec{e})) \wedge \varphi(\vec{u}(\vec{u}^{2\cdot n'_{\vec{e}}}(\vec{e})))\right)$$

(by definition of φ_{ch} and $\vec{u}_{\text{ch}}$)

$$\iff \forall n < n'_{\vec{e}}.\ \left(\varphi(\vec{u}^{2\cdot n}(\vec{e})) \wedge \varphi(\vec{u}^{2\cdot n+1}(\vec{e}))\right) \wedge \neg\left(\varphi(\vec{u}^{2\cdot n'_{\vec{e}}}(\vec{e})) \wedge \varphi(\vec{u}^{2\cdot n'_{\vec{e}}+1}(\vec{e}))\right)$$

$$\iff \forall m < 2\cdot n'_{\vec{e}}.\ \varphi(\vec{u}^{m}(\vec{e})) \wedge \left(\neg\varphi(\vec{u}^{2\cdot n'_{\vec{e}}}(\vec{e})) \vee \neg\varphi(\vec{u}^{2\cdot n'_{\vec{e}}+1}(\vec{e}))\right)$$

$$\iff 2\cdot n'_{\vec{e}} \leq rt_{(\varphi,\vec{u})}(\vec{e}) \leq 2\cdot n'_{\vec{e}} + 1.$$

□

B.10 Proof of Lemma 9.1.6

Proof. Let $(\varphi, \vec{u})$ be a loop. Since $id^{-1}_{S[\vec{x}]} = id_{S[\vec{x}]}$, we obtain $Tr_{id_{S[\vec{x}]}}(\varphi, \vec{u}) = (\varphi', \vec{u}\,')$ with

$$\begin{aligned} \varphi' &= id^{-1}_{S[\vec{x}]}(\varphi) = \varphi \\ \vec{u}\,' &= (id^{-1}_{S[\vec{x}]} \circ \tilde{u} \circ id_{S[\vec{x}]})(\vec{x}) = \tilde{u}(\vec{x}) = \vec{u} \end{aligned}$$

Now we take $\eta_1, \eta_2 \in \text{Aut}_S\left(S[\vec{x}]\right)$. Note that $(\eta_1 \circ \eta_2)^{-1} = \eta_2^{-1} \circ \eta_1^{-1}$. Let $Tr_{\eta_1\circ\eta_2}(\varphi, \vec{u}) = (\varphi', \vec{u}\,')$, $Tr_{\eta_1}(\varphi, \vec{u}) = (\varphi'', \vec{u}\,'')$, and $Tr_{\eta_2}(\varphi'', \vec{u}\,'') = (\varphi''', \vec{u}\,''')$. We have

$$\begin{aligned} \varphi' &= (\eta_2^{-1} \circ \eta_1^{-1})(\varphi) \\ \varphi'' &= \eta_1^{-1}(\varphi) \\ \varphi''' &= \eta_2^{-1}\left(\varphi''\right) \\ &= \eta_2^{-1}\left(\eta_1^{-1}(\varphi)\right) \\ &= (\eta_2^{-1} \circ \eta_1^{-1})(\varphi) \\ &= \varphi' \end{aligned}$$

Moreover, we have

$$\begin{aligned}
\vec{u}' &= \left(\eta_2^{-1}\circ\eta_1^{-1}\circ\tilde{u}\circ\eta_1\circ\eta_2\right)(\vec{x}) \\
&= (\eta_2(\vec{x}))\,(\eta_1(\vec{x}))\,(\vec{u})\,(\eta_1^{-1}(\vec{x}))\,(\eta_2^{-1}(\vec{x})) \\
\vec{u}'' &= \left(\eta_1^{-1}\circ\tilde{u}\circ\eta_1\right)(\vec{x}) \\
&= (\eta_1(\vec{x}))\,(\vec{u})\,(\eta_1^{-1}(\vec{x})) \\
\vec{u}''' &= \left(\eta_2^{-1}\circ\widetilde{\vec{u}''}\circ\eta_2\right)(\vec{x}) \\
&= (\eta_2(\vec{x}))\,(\eta_1(\vec{x}))\,(\vec{u})\,(\eta_1^{-1}(\vec{x}))\,(\eta_2^{-1}(\vec{x})) \\
&= \vec{u}'
\end{aligned}$$

□

B.11 Proof of Lemma 9.1.7

Proof. Let $\vec{e} \in S^d$ and $n \in \mathbb{N}$. We have

$$\begin{aligned}
& \varphi'((\vec{u}')^n((\eta(\vec{x}))\,(\vec{e}))) \\
\iff & \eta^{-1}(\varphi)\ ((\vec{u}')^n((\eta(\vec{x}))\,(\vec{e}))) \\
\iff & \varphi\,[\vec{x}/\eta^{-1}(\vec{x})]\ \underbrace{[\vec{x}/\vec{u}']}_{n \text{ times}}[\vec{x}/\eta(\vec{x})]\,[\vec{x}/\vec{e}] \\
\iff & \varphi\,[\vec{x}/\eta^{-1}(\vec{x})]\,[\vec{x}/\eta(\vec{x})]\ \underbrace{[\vec{x}/\vec{u}]}_{n \text{ times}}\ [\vec{x}/\eta^{-1}(\vec{x})]\,[\vec{x}/\eta(\vec{x})]\,[\vec{x}/\vec{e}] \\
\iff & \varphi\ \underbrace{[\vec{x}/\vec{u}]}_{n \text{ times}}\ [\vec{x}/\vec{e}] \\
\iff & \varphi(\vec{u}^n(\vec{e}))
\end{aligned}$$

□

B.12 Proof of Thm. 9.1.10

Proof. In Cor. 9.1.8, we have seen that if $\vec{e}$ is a witness for (eventual) non-termination of $(\varphi, \vec{u})$, $\hat{\eta}(\vec{e})$ witnesses (eventual) non-termination of $Tr_\eta(\varphi, \vec{u})$. Now let $\vec{e}'$ be a witness for (eventual) non-termination of $Tr_\eta(\varphi, \vec{u})$. Then by Cor. 9.1.8, $\widehat{\eta^{-1}}(\vec{e}')$ witnesses (eventual) non-termination of $Tr_{\eta^{-1}}(Tr_\eta(\varphi, \vec{u})) \overset{\text{Lemma 9.1.6}}{=} Tr_{\eta\circ\eta^{-1}}(\varphi, \vec{u}) = (\varphi, \vec{u})$.

Hence, $\hat{\eta}$ maps witnesses for (eventual) non-termination of $(\varphi, \vec{u})$ to witnesses for (eventual) non-termination of $Tr_\eta(\varphi, \vec{u})$ and $\widehat{\eta^{-1}}$ maps witnesses for (eventual) non-termination of $Tr_\eta(\varphi, \vec{u})$ to witnesses for (eventual) non-termination of $(\varphi, \vec{u})$. These two mappings are inverse to each other: For $\vec{e}\,' \in S^d$ we have

$$
\begin{aligned}
&\hat{\eta}(\widehat{\eta^{-1}}(\vec{e}\,')) \\
&= \hat{\eta}((\eta^{-1}(\vec{x}))(\vec{e}\,')) && \text{(by definition of } \widehat{\eta^{-1}}) \\
&= (\eta(\vec{x}))((\eta^{-1}(\vec{x}))(\vec{e}\,')) && \text{(by definition of } \hat{\eta}) \\
&= \eta(\vec{x})[\vec{x}/\eta^{-1}(\vec{x})][\vec{x}/\vec{e}\,'] \\
&= \vec{e}\,'
\end{aligned}
$$

$$
\begin{aligned}
&\widehat{\eta^{-1}}(\hat{\eta}(\vec{e}\,')) \\
&= \widehat{\eta^{-1}}((\eta(\vec{x}))(\vec{e}\,')) && \text{(by definition of } \hat{\eta}) \\
&= (\eta^{-1}(\vec{x}))((\eta(\vec{x}))(\vec{e}\,')) && \text{(by definition of } \widehat{\eta^{-1}}) \\
&= \eta^{-1}(\vec{x})[\vec{x}/\eta(\vec{x})][\vec{x}/\vec{e}\,'] \\
&= \vec{e}\,'.
\end{aligned}
$$

Hence, $\hat{\eta}$ is indeed a bijection with inverse mapping $\widehat{\eta^{-1}}$. □

B.13 Proof of Lemma 9.1.12

Proof. Let $\vec{e}\,' \in \hat{\eta}(F)$. Then $\vec{e}\,' = \hat{\eta}(\vec{e})$ for some $\vec{e} \in F$. As F is $\vec{u}$-invariant, we have $\vec{u}(\vec{e}) \in F$. We obtain

$$
\begin{aligned}
\vec{u}\,'(\vec{e}\,') &= (\eta(\vec{x}))\,(\vec{u})\,(\eta^{-1}(x))\,(\vec{e}\,') \\
&= (\eta(\vec{x}))\,(\vec{u})\,(\eta^{-1}(x))\,(\eta(\vec{x}))\,(\vec{e}) \\
&= (\eta(\vec{x}))\,(\vec{u})\,(\vec{e}) \\
&= \hat{\eta}(\vec{u}(\vec{e})) \in \hat{\eta}(F).
\end{aligned}
$$

□

B.14 Proof of Lemma 9.1.14

Proof. Let F be characterized by $\psi_F \in \mathrm{Th}_\exists(S, \mathbb{R}_\mathbb{A})$. Consider the following formula $\psi \in \mathrm{Th}_\exists(S, \mathbb{R}_\mathbb{A})$:

$$\exists \vec{y} \in \mathbb{R}_\mathbb{A}^d.\ \psi_F(\vec{y}) \wedge \vec{x} = \left(\eta(\vec{x})\right)(\vec{y}).$$

Then $\psi(\vec{e})$ holds for a point $\vec{e} \in \mathbb{R}_\mathbb{A}^d$ iff $\vec{e} = \hat{\eta}(\vec{u})$ for some $\vec{u} \in \mathbb{R}_\mathbb{A}^d$ where $\psi_F(\vec{u})$ holds, i.e., where $\vec{u} \in F$. □

B.15 Proof of Thm. 9.1.16

Proof. As $(\varphi, \vec{u})$ is solvable, there is a partitioning $\mathcal{J} = \{J_1, \dots, J_k\}$ as in Def. 4.1.6, i.e., $\{1, \dots, d\} = \biguplus_{i=1}^k J_i$ and for all $1 \leq i \leq k$ we have $\vec{u}_{J_i} = A_i \cdot \vec{x}_{J_i} + \vec{p}_i$, where $\vec{p}_i \in (S[\vec{x}_{J_{i+1}}, \dots, \vec{x}_{J_k}])^{d_i}$. W.l.o.g., assume that $\vec{x}$ is ordered according to $\mathcal{J}$, i.e., if $x_{i_1} \in J_{j_1}$ and $x_{i_2} \in J_{j_2}$ for $j_1 < j_2$, then $i_1 < i_2$.

For each A_i, let $Q_i = T_i \cdot A_i \cdot T_i^{-1}$ be its Jordan normal form, where T_i is the corresponding transformation matrix. Since A_i has only real eigenvectors, this means that the entries of Q_i, T_i, and T_i^{-1} are real algebraic numbers. Let η be the endomorphism defined by $\eta(\vec{x}_{J_i}) = T_i \cdot \vec{x}_{J_i}$. This means that η is induced by the block diagonal matrix $Diag(T_1, T_2, \dots, T_k)$. Then η is an automorphism and its inverse satisfies $\eta^{-1}(\vec{x}_{J_i}) = T_i^{-1} \cdot \vec{x}_{J_i}$. Furthermore, the degree of η is obviously 1. Moreover, η and η^{-1} are compatible with the partition, i.e., the images of the variables in $\vec{x}_{J_i}$ under η and η^{-1} are polynomials only using the variables $\vec{x}_{J_i}$. For each $1 \leq i \leq k$ we have:

$$\begin{aligned}
&(\eta^{-1} \circ \tilde{u} \circ \eta)(\vec{x}_{J_i}) \\
&= \eta(\vec{x}_{J_i})[\vec{x}/\vec{u}][\vec{x}/\eta^{-1}(\vec{x})] \\
&= \left(T_i \cdot \vec{x}_{J_i}\right)[\vec{x}/\vec{u}][\vec{x}/\eta^{-1}(\vec{x})] \\
&= (T_i \cdot \vec{u}_{J_i})[\vec{x}/\eta^{-1}(\vec{x})] \\
&= \left(T_i \cdot \left(A_i \cdot \vec{x}_{J_i} + \vec{p}_i\right)\right)[\vec{x}/\eta^{-1}(\vec{x})] \\
&= \left(T_i \cdot A_i \cdot \vec{x}_{J_i} + T_i \cdot \vec{p}_i\right)[\vec{x}/\eta^{-1}(\vec{x})] \\
&= \left(T_i \cdot A_i \cdot \vec{x}_{J_i}\right)[\vec{x}/\eta^{-1}(\vec{x})] + \left(T_i \cdot \vec{p}_i\right)[\vec{x}/\eta^{-1}(\vec{x})] \\
&= \left(T_i \cdot A_i \cdot T_i^{-1} \cdot \vec{x}_{J_i}\right) + \left(T_i \cdot \vec{p}_i\right)[\vec{x}/\eta^{-1}(\vec{x})] \\
&= \left(Q_i \cdot \vec{x}_{J_i}\right) + \left(T_i \cdot \vec{p}_i\right)[\vec{x}/\eta^{-1}(\vec{x})]
\end{aligned}$$

We have $T_i \cdot \vec{p}_i \in \mathcal{S}[\vec{x}_{J_{i+1}}, \dots, \vec{x}_{J_k}]^{d_i}$. Therefore, we have $\left(T_i \cdot \vec{p}_i\right)[\vec{x}/\eta^{-1}(\vec{x})] \in \mathcal{S}[\vec{x}_{J_{i+1}}, \dots, \vec{x}_{J_k}]^{d_i}$ as well as η^{-1} is compatible with the partitioning. This implies that $Tr_\eta(\varphi, \vec{u})$ is weakly non-linear. Since we assumed that $\vec{x}$ is ordered w.r.t. the partitioning and each Q_i is triangular, $Tr_\eta(\varphi, \vec{u})$ is triangular, too. Thus, $Tr_\eta(\varphi, \vec{u})$ is in *twn*-form. □

B.16 Proof of Lemma 9.2.2

Proof. Let $\delta = \deg(\eta)$. Note that for any $s \in \mathbb{N}$, there is only a finite number of monomials over $\vec{x}$ of degree s. (More precisely, the number of monomials of exactly degree s is $\binom{d+s-1}{s}$.) Hence, for any $1 \leq i \leq d$ we can construct the following term that stands for $\eta^{-1}(x_i)$:

$$\sum_{m \text{ is a monomial of (at most) degree } \delta^{d-1}} a_{i,m} \cdot m$$

Here, the monomials m contain the variables $\vec{x}$ and the $a_{i,m}$ are variables that stand for the unknown coefficients of the polynomial $\eta^{-1}(x_i)$.

Hence, for any $1 \leq i \leq d$ we now build a formula $\rho_{r,i}$ which stands for the requirement "$\left(\eta \circ \eta^{-1}\right)(x_i) = \left(\eta^{-1}(x_i)\right)(\eta(\vec{x})) = x_i$" (i.e., that η^{-1} is a right inverse of η):

$$\rho_{r,i} : \sum_{m \text{ is a monomial of (at most) degree } \delta^{d-1}} a_{i,m} \cdot \eta(m) = x_i$$

Similarly, for any $1 \leq i \leq d$ we construct a formula $\rho_{l,i}$ which stands for the requirement "$\left(\eta^{-1} \circ \eta\right)(x_i) = (\eta(x_i))\left(\eta^{-1}(\vec{x})\right) = x_i$" (i.e., that η^{-1} is a left inverse of η):

$$\rho_{l,i} : \eta(x_i) \left(\sum_{m \text{ is a monomial of (at most) degree } \delta^{d-1}} a_{i,m} \cdot m \right) = x_i$$

Thus, the formula

$$\forall \vec{x} \in \mathbb{R}_{\mathbb{A}}^d. \quad \bigwedge_{i=1}^{d} \rho_{r,i} \wedge \bigwedge_{i=1}^{d} \rho_{l,i} \tag{B.16.1}$$

is valid iff η has an inverse of degree at most δ^{d-1}. By Thm. 9.2.1, this is equivalent to the question whether η has an inverse, i.e., whether η is an automorphism. Unfortunately, (B.16.1) $\notin \mathrm{Th}_\exists(\mathbb{R}_{\mathbb{A}})$. However, $\bigwedge_{i=1}^{d} \rho_{r,i} \wedge \bigwedge_{i=1}^{d} \rho_{l,i}$ has to hold for all $\vec{x} \in \mathbb{R}_{\mathbb{A}}^d$. So, we can reduce this formula to a system of equations: one simply has to check whether

there is an instantiation of the unknown coefficients $a_{i,m}$ such that all monomials in $\rho_{r,i}$ and $\rho_{l,i}$ except x_i get the coefficient 0 and the monomial x_i gets the coefficient 1. When building the conjunction of these equations and existentially quantifying the unknown coefficients $a_{i,m}$, one indeed obtains a formula from $\mathrm{Th}_\exists(\mathbb{R}_\mathbb{A})$. □

B.17 Proof of Thm. 9.2.3

Proof. For every $1 \leq i \leq d$, let

$$\eta(x_i) = \sum_{m \text{ is a monomial of (at most) degree } \delta} b_{i,m} \cdot m,$$

where the $b_{i,m}$ are variables that stand for unknown coefficients. By Lemma 9.2.2 there is a $\mathrm{Th}_\exists(\mathbb{R}_\mathbb{A})$-formula that contains both $b_{i,m}$ and the variables $u_{i,m}$ (for the coefficients of η^{-1}) which expresses that η is an automorphism.

Furthermore, using these coefficients we can construct a formula from $\mathrm{Th}_\exists(\mathbb{R}_\mathbb{A})$ which expresses that the update $\vec{u}' = (u'_1, \dots, u'_d) = (\eta^{-1} \circ \tilde{u} \circ \eta)(\vec{x})$ is a *twn*-loop: Note that we have $\deg(\vec{u}') = \deg((\eta^{-1} \circ \tilde{u} \circ \eta)(\vec{x})) \leq \deg(\eta^{-1}) \cdot \deg(\tilde{u}) \cdot \deg(\eta) \leq \delta^{d-1} \cdot \deg(\tilde{u}) \cdot \delta$. So there is a bound on the degree of the polynomials occurring in the transformed loop $Tr_\eta(\varphi, \vec{u})$. Hence, for every $1 \leq i \leq d$, let

$$u'_i = \sum_{m \text{ is a monomial of (at most) degree } \delta^{d-1} \cdot \deg(\tilde{u}) \cdot \delta} c_{i,m} \cdot m,$$

where the variables $c_{i,m}$ stand again for unknown coefficients. Now we can build a $\mathrm{Th}_\exists(\mathbb{R}_\mathbb{A})$-formula which is valid iff $\vec{u}'$ is in *twn*-form by requiring that certain coefficients $c_{i,m}$ are zero. Moreover, we can construct a $\mathrm{Th}_\exists(\mathbb{R}_\mathbb{A})$-formula which is valid iff $\vec{u}' = (\eta^{-1} \circ \tilde{u} \circ \eta)(\vec{x})$. This proves the theorem. □

B.18 Proof of Lemma 10.3.5

Proof. In the following, let $\vec{u} = A \cdot \vec{x}$.

$\Longrightarrow$: Let $\succ_{\vec{u}}$ be a total ordering and λ the unique eigenvalue of A, i.e., the diagonal of A only contains λ. We now prove that the matrix $A - \lambda \cdot I_{d \times d}$ has rank at least $d - 1$, i.e., its kernel, which is the eigenspace of A w.r.t. λ, has dimension at most 1. As λ is an eigenvalue of A, its eigenspace then must be exactly one-dimensional.

Since A is triangular so is $A - \lambda \cdot I_{d \times d}$. As $\succ_{\vec{u}}$ is a total ordering, the super-diagonal of $A - \lambda \cdot I_{d \times d}$ contains only non-zero values, whereas its diagonal contains only

zeros. Thus, by deleting the first row and the last column of this matrix we obtain a triangular $(d-1)\times(d-1)$ submatrix D whose diagonal is the super-diagonal of $A-\lambda\cdot I_{d\times d}$, i.e., it contains only non-zero values. Thus, the product of the diagonal entries of D is non-zero, i.e., $\det(D)\neq 0$. But D is a submatrix of $A-\lambda\cdot I_{d\times d}$, i.e., $A-\lambda\cdot I_{d\times d}$ has a non-zero $(d-1)\times(d-1)$ minor. Hence, $\operatorname{rank}(A-\lambda\cdot I_{d\times d})\geq(d-1)$.

$\Longleftarrow$: Let us assume that $\succ_{\vec{u}}$ is not a total ordering. In this case, the super-diagonal of $A-\lambda\cdot I_{d\times d}$ contains a zero and its diagonal contains only zeros. Due to triangularity, its rank can be at most $d-2$. Thus, A has at least two linear independent eigenvectors. □

B.19 Proof of Lemma 10.4.6

Proof. By definition, $\mathbb{Q}(\lambda_1,\dots,\lambda_k)$ is the splitting field of the minimal polynomial $\mu_A(y)$ of A. Since A is uniform, its minimal polynomial μ_A and characteristic polynomial $\chi_A(y)=\prod_{i=1}^{k}(y-\lambda_i)^{\nu_i}$ coincide.

We now prove the following: For all $1\leq i\leq k$, the degree of the field extension $\mathbb{Q}(\lambda_1,\dots,\lambda_i)/\mathbb{Q}(\lambda_1,\dots,\lambda_{i-1})$, i.e., the dimension of $\mathbb{Q}(\lambda_1,\dots,\lambda_i)$ as a vector space over $\mathbb{Q}(\lambda_1,\dots,\lambda_{i-1})$, is bounded from above by $\left(d-\sum_{j=1}^{i-1}\nu_j\right)$. Here, $\mathbb{Q}(\lambda_1,\dots,\lambda_{i-1})=\mathbb{Q}$ for $i=1$.

So let $1\leq i\leq k$. In the polynomial ring $\mathbb{Q}(\lambda_1,\dots,\lambda_{i-1})[y]$ we can factor $\mu_A(y)=\chi_A(y)=\prod_{j=1}^{i-1}(y-\lambda_j)^{\nu_j}\cdot f(y)$ for some $f(y)\in\mathbb{Q}(\lambda_1,\dots,\lambda_{i-1})[y]$. Since $\mu_A(\lambda_i)=0$ and the eigenvalues are pairwise distinct, we must have $f(\lambda_i)=0$. Note that

$$\deg(f)=\deg(\mu_A)-\sum_{j=1}^{i-1}\nu_j=d-\sum_{j=1}^{i-1}\nu_j.$$

The minimal polynomial $\mu_{\lambda_i}\in\mathbb{Q}(\lambda_1,\dots,\lambda_{i-1})[y]$ of λ_i divides f since $f(\lambda_i)=0$. Thus,

$$\begin{aligned}&\dim_{\mathbb{Q}(\lambda_1,\dots,\lambda_{i-1})}\left(\mathbb{Q}(\lambda_1,\dots,\lambda_i)\right)\\&=\dim_{\mathbb{Q}(\lambda_1,\dots,\lambda_{i-1})}\left(\mathbb{Q}(\lambda_1,\dots,\lambda_{i-1})[y]/\left\langle\mu_{\lambda_i}\right\rangle\right)\\&=\deg(\mu_i)\leq\deg(f)=d-\sum_{j=1}^{i-1}\nu_j,\end{aligned}$$

where $\dim_{\mathbb{Q}(\lambda_1,\dots,\lambda_{i-1})}$ denotes the dimension as a vector space over $\mathbb{Q}(\lambda_1,\dots,\lambda_{i-1})$.

Since the degree of the field extension $\mathbb{Q}(\lambda_1,\dots,\lambda_k)/\mathbb{Q}$, i.e., the $\mathbb{Q}$-dimension of $\mathbb{Q}(\lambda_1,\dots,\lambda_k)$, is the product of the degrees of the field extensions $\mathbb{Q}(\lambda_1,\dots,\lambda_i)/\mathbb{Q}(\lambda_1,\dots,\lambda_{i-1})$, we have

$$\dim_{\mathbb{Q}}\left(\mathbb{Q}(\lambda_1,\dots,\lambda_k)\right)\leq\prod_{i=1}^{k}\left(d-\sum_{j=1}^{i-1}\nu_j\right)\in\mathcal{O}(d^k).$$

This proves the claim. □

B.20 Proof of Lemma 14.4.23

Proof. We have to prove that $X_n^{f,I}$ is $\mathcal{G}_n^{\text{loop}}$-measurable, i.e., $\left(X_n^{f,I}\right)^{-1}(B) \in \mathcal{G}_n^{\text{loop}}$ for any $B \in \mathfrak{B}$.

Consider any run $\vartheta \in \Omega^{\text{loop}}$. $X_n^{f,I}(\vartheta)$ just depends on $\vartheta[0], \dots, \vartheta[n+1]$, i.e., $X_n^{f,I}$ is constant on the cylinder set $\text{Cyl}(\vartheta[0] \cdots \vartheta[n+1])$. Therefore, it is constant on the generators of $\mathcal{G}_n^{\text{loop}}$, i.e., on every $\text{Cyl}(\pi)$ with $|\pi| = n+2$. Since there are only countably many of these generators and these are pairwise disjoint, we are in the situation of Lemma A.1.2. Hence, $\left(X_n^{f,I}\right)^{-1}(B)$ is a countable union of generators of $\mathcal{G}_n^{\text{loop}}$ for any $B \in \mathfrak{B}$, i.e., $\left(X_n^{f,I}\right)^{-1}(B) \in \mathcal{G}_n^{\text{loop}}$. □

B.21 Proof of Thm. 14.4.25

Proof. Let $s \in \Sigma$ and $\mathfrak{s} \in \text{Scheds}$. Due to Lemma 14.4.23, $X_n^{f,\text{wp}\Phi_f(I)}$ is $\mathcal{G}_n^{\text{loop}}$-measurable. By Lemma A.6.2, it is enough to show that for any $\pi = s_0 \cdots s_{n+1} \in \Sigma^{n+2}$ with $\mathbb{P}_{s,\mathfrak{s}}^{\text{loop}}(\text{Cyl}(\pi)) > 0$ we have

$$\begin{aligned}&\int_{\text{Cyl}(\pi)} X_{n+1}^{f,I} \, d\left(\mathbb{P}_{s,\mathfrak{s}}^{\text{loop}}\right)\\ &= \mathbb{P}_{s,\mathfrak{s}}^{\text{loop}}(\text{Cyl}(\pi)) \cdot \left([\neg\varphi](s_{n+1}) \cdot f(s_{n+1}) + [\varphi](s_{n+1}) \cdot \sum_{s_{n+2} \in \Sigma} \mu_{s_{n+1},\mathfrak{s}}^{\mathbf{P}}(s_{n+2}) \cdot I(s_{n+2}) \right).\end{aligned} \tag{B.21.1}$$

Then we get that $\mathbb{E}_{s,\mathfrak{s}}^{\text{loop}}\left(X_{n+1}^{f,I} \,\middle|\, \mathcal{G}_n^{\text{loop}} \right)$ takes the value

$$[\neg\varphi](s_{n+1}) \cdot f(s_{n+1}) + [\varphi](s_{n+1}) \cdot \sum_{s_{n+2} \in \Sigma} \mu_{s_{n+1},\mathfrak{s}}^{\mathbf{P}}(s_{n+2}) \cdot I(s_{n+2})$$

on $\text{Cyl}(\pi)$ by Lemma A.6.2. If $s_{n+1} \in \Sigma_{\neg\varphi}$, then s_{n+1} is the state on which the loop has terminated for any run with prefix π due to the definition of $\mathbb{P}_{s,\mathfrak{s}}^{\text{loop}}$. Note that if $\mathbb{P}_{s,\mathfrak{s}}^{\text{loop}}(\text{Cyl}(\pi)) = 0$, we can choose the value that $\mathbb{E}_{s,\mathfrak{s}}^{\text{loop}}\left(X_{n+1}^{f,I} \,\middle|\, \mathcal{G}_n^{\text{loop}} \right)$ takes on this set

arbitrarily. Thus, here we set it as follows:

$$\begin{cases} \sum_{s_{n+2}\in\Sigma} \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}}(s_{n+2}) \cdot I(s_{n+2}), & \text{if } \pi \in \Sigma_\varphi^{n+2} \\ f(s_i), & \text{if } s_0,\dots,s_{i-1} \in \Sigma_\varphi \wedge s_i \in \Sigma_{\neg\varphi}. \end{cases}$$

By taking the infimum over all schedulers we then have

$$\begin{aligned}
& [\mathrm{Cyl}(\pi)] \cdot \inf_{\mathfrak{s}\in\mathsf{Scheds}} \mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^{f,I}_{n+1} \,\middle|\, \mathcal{G}^{\mathrm{loop}}_n\right) \\
= & \inf_{\mathfrak{s}\in\mathsf{Scheds}} [\mathrm{Cyl}(\pi)] \cdot \mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^{f,I}_{n+1} \,\middle|\, \mathcal{G}^{\mathrm{loop}}_n\right) \\
= & [\mathrm{Cyl}(\pi)] \cdot \begin{cases} \inf_{\mathfrak{s}\in\mathsf{Scheds}} \sum_{s_{n+2}\in\Sigma} \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}}(s_{n+2}) \cdot I(s_{n+2}), & \text{if } \pi \in \Sigma_\varphi^{n+2} \\ f(s_i), & \text{if } s_0,\dots,s_{i-1} \in \Sigma_\varphi \wedge s_i \in \Sigma_{\neg\varphi} \end{cases} \\
= & [\mathrm{Cyl}(\pi)] \cdot \begin{cases} \mathrm{wp}\,[\![\mathbf{P}]\!](I)(s_{n+1}), & \text{if } \pi \in \Sigma_\varphi^{n+2} \\ f(s_i), & \text{if } s_0,\dots,s_{i-1} \in \Sigma_\varphi \wedge s_i \in \Sigma_{\neg\varphi} \end{cases} \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(by Def. 14.2.10 and 14.2.19)} \\
= & [\mathrm{Cyl}(\pi)] \cdot \begin{cases} \mathrm{wp}\,\Phi_f(I)(s_{n+1}), & \text{if } \pi \in \Sigma_\varphi^{n+2} \\ f(s_i), & \text{if } s_0,\dots,s_{i-1} \in \Sigma_\varphi \wedge s_i \in \Sigma_{\neg\varphi} \end{cases} \\
= & [\mathrm{Cyl}(\pi)] \cdot X_n^{f,\mathrm{wp}\,\Phi_f(I)}
\end{aligned}$$

As $\pi \in \Sigma^{n+2}$ was chosen arbitrarily, this proves the result.

We now show (B.21.1). If $\pi \notin s\Sigma^{n+1}$, the set $\mathrm{Cyl}(\pi)$ is a null set. So assume $s_0 \cdots s_{n+1} = \pi \in s\Sigma^{n+2}$ and $\mathrm{Cyl}(\pi)$ is not a null set, i.e., $\mathfrak{p}^{\mathrm{loop}}_{s,\mathfrak{s}}(\pi) > 0$. Note that if $s_i \in \Sigma_{\neg\varphi}$ for some $i \le n$, then $s_{n+1} \in \Sigma_{\neg\varphi}$ as otherwise $\mathfrak{p}^{\mathrm{loop}}_{s,\mathfrak{s}}(\pi) = 0$. Hence, we assume $\pi = s_0 \cdots s_{n+1} \in \Sigma_\varphi^{n+1}\Sigma$. We use a case analysis to prove the desired result.

(i) $s_{n+1} \in \Sigma_{\neg\varphi}$, i.e., $T^\varphi(\pi') = n+1$ for all $\pi' \in \mathrm{Cyl}(\pi)$

$$\begin{aligned}
& \int_{\mathrm{Cyl}(\pi)} X^{f,I}_{n+1}\, d\left(\mathbb{P}^{\mathrm{loop}}_{s,\mathfrak{s}}\right) \\
= & \int_{\mathrm{Cyl}(\pi)} f(s_{n+1})\, d\left(\mathbb{P}^{\mathrm{loop}}_{s,\mathfrak{s}}\right) \\
= & \int_{\mathrm{Cyl}(\pi)} [\neg\varphi](s_{n+1}) \cdot f(s_{n+1}) + [\varphi](s_{n+1}) \cdot \sum_{s_{n+2}\in\Sigma} \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}}(s_{n+2}) \cdot I(s_{n+2})\, d\left(\mathbb{P}^{\mathrm{loop}}_{s,\mathfrak{s}}\right)
\end{aligned}$$

$$= \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}(\text{Cyl}(\pi)) \left([\neg\varphi](s_{n+1}) \cdot f(s_{n+1}) + [\varphi](s_{n+1}) \cdot \sum_{s_{n+2}\in\Sigma} \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}}(s_{n+2}) \cdot I(s_{n+2}) \right)$$

(by Lemma A.3.8)

(ii) $s_{n+1} \in \Sigma_\varphi$, i.e., $T^\varphi(\pi') > n+1$ for all $\pi' \in \text{Cyl}(\pi)$

$$\begin{aligned}
&\int_{\text{Cyl}(\pi)} X^{f,I}_{n+1} \, d\left(\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\right) \\
&= \int_{\biguplus_{s_{n+2}\in\Sigma} \text{Cyl}(\pi s_{n+2})} X^{f,I}_{n+1} \, d\left(\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\right) \\
&= \sum_{s_{n+2}\in\Sigma} \int_{\text{Cyl}(\pi s_{n+2})} X^{f,I}_{n+1} \, d\left(\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\right) \\
&= \sum_{s_{n+2}\in\Sigma} \int_{\text{Cyl}(\pi s_{n+2})} I(s_{n+2}) \, d\left(\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\right) \\
&= \sum_{s_{n+2}\in\Sigma} \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}(\text{Cyl}(\pi s_{n+2})) \cdot I(s_{n+2}) && \text{(by Lemma A.3.8)} \\
&= \sum_{s_{n+2}\in\Sigma} \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}(\text{Cyl}(\pi)) \cdot \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}}(s_{n+2}) \cdot I(s_{n+2}) && \text{(by Lemma 14.4.9)} \\
&= \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}(\text{Cyl}(\pi)) \cdot \sum_{s_{n+2}\in\Sigma} \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}}(s_{n+2}) \cdot I(s_{n+2}) \\
&= \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}(\text{Cyl}(\pi)) \cdot \left([\neg\varphi](s_{n+1}) \cdot f(s_{n+1}) + [\varphi](s_{n+1}) \cdot \sum_{s_{n+2}\in\Sigma} \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}'}(s_{n+2}) \cdot I(s_{n+2}) \right)
\end{aligned}$$

□

B.22 Proof of Cor. 14.4.27

Proof. Fix a state $s \in \Sigma$ and a scheduler $\mathfrak{s} \in \text{Scheds}$. We prove the result by induction.

Induction base $n = 0$:

$$\begin{aligned}
&\mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(X^{f,I}_0\right) \\
&= [\neg\varphi](s) \cdot f(s) + [\varphi](s) \cdot \sum_{s'\in\Sigma} \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\left(\text{Cyl}\left(ss'\right)\right) \cdot I(s')
\end{aligned}$$

$$
\begin{aligned}
&= [\neg\varphi](s) \cdot f(s) + [\varphi](s) \cdot \sum_{s' \in \Sigma} \mu^{\mathbf{P}}_{\mathfrak{s},s}\left(s'\right) \cdot I(s') && \text{(by Lemma 14.4.9)} \\
&\geq [\neg\varphi](s) \cdot f(s) + [\varphi](s) \cdot \underbrace{\inf_{\mathfrak{s}' \in \mathsf{Scheds}} \sum_{s' \in \Sigma} \mu^{\mathbf{P}}_{\mathfrak{s}',s}\left(s'\right) \cdot I(s')}_{= \mathrm{wp}[\![\mathbf{P}]\!](I)(s)} \\
&= {}^{\mathrm{wp}}\Phi_f(I)(s).
\end{aligned}
$$

Induction Step: $n \longrightarrow n+1$**:** Assume the result holds for a fixed $n \in \mathbb{N}$.

$$
\begin{aligned}
&\mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^{f,I}_{n+1}\right) \\
&= \mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(\mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^{f,I}_{n+1} \,\middle|\, \mathcal{G}^{\mathrm{loop}}_{n}\right)\right) && \text{(by Lemma A.6.4)} \\
&\geq \mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^{f,{}^{\mathrm{wp}}\Phi_f(I)}_{n}\right) && \text{(by Thm. 14.4.25)} \\
&\geq {}^{\mathrm{wp}}\Phi^{n+1}_f({}^{\mathrm{wp}}\Phi_f(I))(s) && \text{(by I.H.)} \\
&= {}^{\mathrm{wp}}\Phi^{n+2}_f(I)(s).
\end{aligned}
$$

□

B.23 Proof of Lemma 15.2.3

Proof. Let $\vartheta \in \Omega^{\mathrm{loop}}$. Then we have

$$
\begin{aligned}
&\lim_{n\longrightarrow\omega} X^{f,I}_n(\vartheta) \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right](\vartheta) \\
&= \lim_{n\longrightarrow\omega} X^{f,I}_{n+T^{\varphi}(\vartheta)}(\vartheta) \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right](\vartheta) \\
&= \lim_{n\longrightarrow\omega} f(\vartheta[T^{\varphi}(\vartheta)]) \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right](\vartheta) \\
&= f(\vartheta[T^{\varphi}(\vartheta)]) \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right](\vartheta) \\
&= X^{f}_{T^{\varphi}(\vartheta)}(\vartheta).
\end{aligned}
$$

If the loop is AST, we have $\mathbb{P}^{\mathrm{loop}}_{\mathfrak{s},s}\left(X^{f,I}_n \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right] = X^{f,I}_n\right) = 1$ for any $s \in \Sigma$ and $\mathfrak{s} \in \mathsf{Scheds}$. So by the result just shown, $\mathbf{X}^{f,I}$ converges pointwise to $X^{f}_{T^{\varphi}}$ on a set with probability one. Thus, $\mathbf{X}^{f,I}$ converges almost-surely to $X^{f}_{T^{\varphi}}$. □

B.24 Proof of Thm. 15.2.4

Proof. Let $s \in \Sigma$ and $\mathfrak{s} \in \mathrm{Scheds}$. Consider the expectation $I = 0$ and the Iverson bracket $\left[(T^{\varphi})^{-1}(\mathbb{N})\right]$. By definition, $X_n^{f,I} \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right] \leq X_{n+1}^{f,I} \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right]$ for all $n \in \mathbb{N}$. To apply the Monotone Convergence Theorem Cor. A.5.1, we calculate the expected value of $X_n^{f,I} \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right]$. Note that $X_n^{f,I}$ is zero on the set $T^{-1}(\{\omega\})$. Hence,

$$
\begin{aligned}
& {}^{\mathrm{wp}}\Phi_f^{n+1}(s) \\
\leq\ & \mathbb{E}^{\mathrm{loop}}_{\mathfrak{s},s}\left(X_n^{f,I}\right) && \text{(by Cor. 14.4.27)} \\
=\ & \mathbb{E}^{\mathrm{loop}}_{\mathfrak{s},s}\left(X_n^{f,I} \cdot \left[T^{-1}(\mathbb{N}) \uplus T^{-1}(\{\omega\})\right]\right) \\
=\ & \mathbb{E}^{\mathrm{loop}}_{\mathfrak{s},s}\left(X_n^{f,I} \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right]\right) + \mathbb{E}^{\mathrm{loop}}_{\mathfrak{s},s}\left(\underbrace{X_n^{f,I} \cdot \left[T^{-1}(\{\omega\})\right]}_{=0}\right) \\
=\ & \mathbb{E}^{\mathrm{loop}}_{\mathfrak{s},s}\left(X_n^{f,I} \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right]\right).
\end{aligned}
$$

Hence we have

$$
\begin{aligned}
& \mathbb{E}^{\mathrm{loop}}_{\mathfrak{s},s}\left(X_T^f\right) \\
=\ & \mathbb{E}^{\mathrm{loop}}_{\mathfrak{s},s}\left(\lim_{n\longrightarrow\omega} X_n^{f,I} \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right]\right) && \text{(by Lemma 15.2.3)} \\
=\ & \lim_{n\longrightarrow\omega} \mathbb{E}^{\mathrm{loop}}_{\mathfrak{s},s}\left(X_n^{f,I} \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right]\right) && \text{(by Cor. A.5.1)} \\
\geq\ & \lim_{n\longrightarrow\omega} {}^{\mathrm{wp}}\Phi_f^{n+1}(I)(s) \\
=\ & \lim_{n\longrightarrow\omega} {}^{\mathrm{wp}}\Phi_f^{n+1}(0)(s) \\
=\ & \mathrm{lfp}\, {}^{\mathrm{wp}}\Phi_f(s).
\end{aligned}
$$

In case of awp we get the same result with flipped inequations by Cor. 14.4.28. □

B.25 Proof of Cor. 15.3.3

Proof. By Def. 15.3.1, $\mathbf{X}^{f,I}$ is uniformly integrable w.r.t. $\mathbb{P}_s^{\mathrm{loop}}$ for every $s \in \Sigma$ if and only if for every $s \in \Sigma$ the equation

$$
\lim_{n\longrightarrow\omega} \mathbb{E}^{\mathrm{loop}}_s\left(X_n^{f,I}\right) = \mathbb{E}^{\mathrm{loop}}_s\left(\lim_{n\longrightarrow\omega} X_n^{f,I}\right) \overset{\text{Lemma 15.2.3}}{=} \mathbb{E}^{\mathrm{loop}}_s\left(X_{T^{\varphi}}^f\right)
$$

holds.

Note that for purely probabilistic loops wp and awp coincide, i.e., equality holds in Thm. 15.2.4 and Cor. 14.4.27 (resp. Cor. 14.4.28). In this case, Thm. 15.2.4 and AST yield $\mathbb{E}_s^{\text{loop}}\left(X^f_{T^\varphi}\right) = (\text{lfp}\,{}^{\text{wp}}\Phi_f)(s)$. So $\mathbf{X}^{f,I}$ is uniformly integrable iff $\mathbb{E}_s^{\text{loop}}\left(X_n^{f,I}\right)$ converges to $(\text{lfp}\,{}^{\text{wp}}\Phi_f)(s)$. By Cor. 14.4.27, this is equivalent to the requirement that ${}^{\text{wp}}\Phi_f^n(I)$ converges to $\text{lfp}\,{}^{\text{wp}}\Phi_f$. This is the definition of uniform integrability of I for f, see Def. 15.1.9. □

B.26 Proof of Lemma 15.3.4

Proof. Let $n \in \mathbb{N}$ and $s \in \Sigma$. We have already proved in Lemma 14.4.23 that $X_n^{f,I}$ is $\mathcal{G}_n^{\text{loop}}$-measurable. First of all, by Cor. 14.4.27 and 14.4.28 we have

$$\mathbb{E}_s^{\text{loop}}\left(X_n^{f,I}\right) = {}^{\text{wp}}\Phi_f^{n+1}(I)(s) < \infty$$

for purely probabilistic loops. Secondly, by Thm. 14.4.25 and ${}^{\text{wp}}\Phi_f(I) \geq I$ we have

$$\mathbb{E}_s^{\text{loop}}\left(X_{n+1}^{f,I} \,\middle|\, \mathcal{G}_n^{\text{loop}}\right) = X_n^{f,{}^{\text{wp}}\Phi_f(I)} \geq X_n^{f,I}.$$

By Def. 14.4.29, this proves that $\mathbf{X}^{f,I}$ is a submartingale w.r.t. $(\mathcal{G}_n^{\text{loop}})_{n\in\mathbb{N}}$. □

B.27 Proof of Thm. 15.3.10

Proof. First of all, $X_n^{0,\Delta I}$ is $\mathcal{G}_n^{\text{loop}}$-measurable as seen in Lemma 14.4.23. For any $\vartheta \in \Omega$ we have

$$\left|X_{n+1}^{f,I} - X_n^{f,I}\right|(\vartheta) = \begin{cases} 0, & \text{if } T^\varphi(\vartheta) \leq n \\ \left|f(\vartheta[n+1]) - I(\vartheta[n+1])\right|, & \text{if } T^\varphi(\vartheta) = n+1 \\ \left|I(\vartheta[n+2]) - I(\vartheta[n+1])\right|, & \text{if } T^\varphi(\vartheta) > n+1 \end{cases}$$

$$= \begin{cases} 0, & \text{if } T^\varphi(\vartheta) \leq n+1 \\ \left|I(\vartheta[n+2]) - I(\vartheta[n+1])\right|, & \text{if } T^\varphi(\vartheta) > n+1 \end{cases}$$

as I harmonizes with f.

Fix $s \in \Sigma$. To show the result we prove

$$[\text{Cyl}(\pi)] \cdot \mathbb{E}_s^{\text{loop}}\left(\left|X_{n+1}^{f,I} - X_n^{f,I}\right| \cdot [\text{Cyl}(\pi)]\right) = [\text{Cyl}(\pi)] \cdot \mathbb{P}_s^{\text{loop}}(\text{Cyl}(\pi)) \cdot X_n^{0,\Delta I} \quad \text{(B.27.1)}$$

for any $\pi \in \Sigma^{n+2}$. Lemma A.6.2 then implies the desired result. Here, both sides are functions of type $\Omega^{\text{loop}} \to \overline{\mathbb{R}}_{\geq 0}$. Note that both sides of this equality are constant 0 if $\mathbb{P}_s^{\text{loop}}(\text{Cyl}(\pi)) = 0$, so in this case the equality holds trivially.

Take any $\pi \in \Sigma^{n+2}$ admissible for s, i.e., $\mathbb{P}_s^{\text{loop}}(\text{Cyl}(\pi)) > 0$. Furthermore, as both random variables $\left|X_{n+1}^{f,I} - X_n^{f,I}\right|$ and $X_n^{0,\Delta I}$ are constant zero if all runs in $\text{Cyl}(\pi)$ have a looping time $\leq n+1$, (B.27.1) holds trivially in this case as well. So, assume $\pi = s_0 \cdots s_{n+1} \in s\Sigma_{\varphi}^{n+1}$. We have

$$
\begin{aligned}
&\mathbb{E}_s^{\text{loop}}\left(\left|X_{n+1}^{f,I} - X_n^{f,I}\right| \cdot [\text{Cyl}(\pi)]\right) \\
&= \int_{\text{Cyl}(\pi)} \left|X_{n+1}^{f,I} - X_n^{f,I}\right| \, d\left(\mathbb{P}_s^{\text{loop}}\right) \\
&= \int_{\biguplus_{s_{n+2}\in\Sigma} \text{Cyl}(\pi s_{n+2})} \left|X_{n+1}^{f,I} - X_n^{f,I}\right| \, d\left(\mathbb{P}_s^{\text{loop}}\right) \\
&= \sum_{s_{n+2}\in\Sigma} \int_{\text{Cyl}(\pi s_{n+2})} \left|X_{n+1}^{f,I} - X_n^{f,I}\right| \, d\left(\mathbb{P}_s^{\text{loop}}\right) \\
&= \sum_{s_{n+2}\in\Sigma} \int_{\text{Cyl}(\pi s_{n+2})} |I(s_{n+2}) - I(s_{n+1})| \, d\left(\mathbb{P}_s^{\text{loop}}\right) \\
&= \sum_{s_{n+2}\in\Sigma} \mathbb{P}_s^{\text{loop}}(\text{Cyl}(\pi s_{n+2})) \cdot |I(s_{n+2}) - I(s_{n+1})| && \text{(by Lemma A.3.8)} \\
&= \sum_{s_{n+2}\in\Sigma} \mathbb{P}_s^{\text{loop}}(\text{Cyl}(\pi)) \cdot \mu_{s_{n+1}}^{\mathbf{P}}(s_{n+2}) \cdot |I(s_{n+2}) - I(s_{n+1})| && \text{(by Lemma 14.4.9)} \\
&= \mathbb{P}_s^{\text{loop}}(\text{Cyl}(\pi)) \cdot \sum_{s_{n+2}\in\Sigma} \mu_{s_{n+1}}^{\mathbf{P}}(s_{n+2}) \cdot |I(s_{n+2}) - I(s_{n+1})| \\
&= \mathbb{P}_s^{\text{loop}}(\text{Cyl}(\pi)) \cdot [\varphi](s_{n+1}) \cdot \sum_{s_{n+2}\in\Sigma} \mu_{s_{n+1}}^{\mathbf{P}}(s_{n+2}) \cdot |I(s_{n+2}) - I(s_{n+1})| \\
&= \mathbb{P}_s^{\text{loop}}(\text{Cyl}(\pi)) \cdot [\varphi](s_{n+1}) \cdot \text{wp}\,[\![\mathbf{P}]\!]\,(|I - I(s_{n+1})|)(s_{n+1}) \\
&= \mathbb{P}_s^{\text{loop}}(\text{Cyl}(\pi)) \cdot \Delta I(s_{n+1}) \\
&= \mathbb{P}_s^{\text{loop}}(\text{Cyl}(\pi)) \cdot X_n^{0,\Delta I}.
\end{aligned}
$$

This proves (B.27.1). □

B.28 Proof of Thm. 15.3.11

The following auxiliary lemma is needed for the proof of Thm. 15.3.11.

Lemma B.28.1 (Sufficient Condition for Uniform Integrability for a Fixed State)**.** *Let $I \ll \infty$ be a conditionally difference bounded expectation that harmonizes with $f \ll \infty$, ${}^{\mathrm{wp}}\Phi_f(I) \ll \infty$ and $s \in \Sigma$. Let the expected looping time of the underlying loop of the form in Fig. 14.16 be finite for s, i.e., $\mathbb{E}_s^{loop}(T^\varphi) < \infty$ and ${}^{\mathrm{wp}}\Phi_f(I)(s) < \infty$. Then ${}^{\mathrm{wp}}\Phi_f^n(I)(s) < \infty$ for all $n \in \mathbb{N}$ and*

$$\lim_{n\to\omega} {}^{\mathrm{wp}}\Phi_f^n(I)(s) = \mathrm{lfp}\, {}^{\mathrm{wp}}\Phi_f(s).$$

Proof. We present a proof based on the proof of [GS01, Thm. 12.5.(9)]. Consider the process $\mathbf{X}^{f,I}$. As I harmonizes with f, Thm. 15.3.10 implies that if I is conditionally difference bounded by the constant $c \geq 0$, we have $\mathbb{E}_s^{\mathrm{loop}}\left(\left|X_{n+1}^{f,I} - X_n^{f,I}\right| \,\middle|\, \mathcal{G}_n^{\mathrm{loop}}\right) \leq c$ for all $n \in \mathbb{N}$. Now we show that $\mathbf{X}^{f,I}$ is uniformly integrable w.r.t. $\mathbb{P}_s^{\mathrm{loop}}$.

Note that by Def. 14.4.21, we have $\mathbf{X}^{f,I}_{\wedge T^\varphi} = \mathbf{X}^{f,I}$: Let $n \in \mathbb{N}$ and $\vartheta \in \Omega$. If $T^\varphi(\vartheta) \leq n$, we have $X^{f,I}_{n\wedge T^\varphi(\vartheta)}(\vartheta) = X^{f,I}_{T^\varphi(\vartheta)}(\vartheta) = f(\vartheta[T^\varphi(\vartheta)]) = X_n^{f,I}(\vartheta)$. Nonetheless, if $T^\varphi(\vartheta) > n$, we have $X^{f,I}_{n\wedge T^\varphi(\vartheta)}(\vartheta) = X_n^{f,I}(\vartheta)$.

For any $n \in \mathbb{N}$ and any run $\vartheta \in \Omega^{\mathrm{loop}}$ we have

$$\begin{aligned}
X_n^{f,I}(\vartheta) &= \left|X_n^{f,I}(\vartheta)\right| = \left|X^{f,I}_{n\wedge T^\varphi}(\vartheta)\right| = \left|X^{f,I}_{\min(n,T^\varphi(\vartheta))}(\vartheta)\right| \\
&= \left|X_0^{f,I}(\vartheta) + \sum_{k=0}^{\min(n,T^\varphi(\vartheta))-1} \left(X_{k+1}^{f,I}(\vartheta) - X_k^{f,I}(\vartheta)\right)\right| \\
&\leq \underbrace{\left|X_0^{f,I}(\vartheta)\right|}_{=X_0^{f,I}(\vartheta)} + \sum_{k=0}^{\min(n,T^\varphi(\vartheta))-1} \left|X_{k+1}^{f,I}(\vartheta) - X_k^{f,I}(\vartheta)\right| \\
&\leq X_0^{f,I}(\vartheta) + \sum_{k=0}^{T^\varphi(\vartheta)-1} \left|X_{k+1}^{f,I}(\vartheta) - X_k^{f,I}(\vartheta)\right| \\
&= \underbrace{X_0^{f,I}(\vartheta) + \sum_{k\in\mathbb{N}} \left|X_{k+1}^{f,I}(\vartheta) - X_k^{f,I}(\vartheta)\right| \cdot \left[\{T^\varphi \geq k+1\}\right](\vartheta)}_{=W^{f,I}(\vartheta)}.
\end{aligned}$$

We show that the expected value of $W^{f,I}$ is finite.

$$
\begin{aligned}
&\mathbb{E}_s^{\mathrm{loop}}\left(W^{f,I}\right)\\
&= \mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I} + \sum_{k\in\mathbb{N}}\left|\left(X_k^{f,I} - X_{k+1}^{f,I}\right)\cdot\left[\{T^{\varphi} \geq k+1\}\right]\right|\right)\\
&= \mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I}\right) + \sum_{k\in\mathbb{N}}\mathbb{E}_s^{\mathrm{loop}}\left(\left|\left(X_k^{f,I} - X_{k+1}^{f,I}\right)\cdot\left[\{T^{\varphi} \geq k+1\}\right]\right|\right)\\
&= \mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I}\right) + \sum_{k\in\mathbb{N}}\mathbb{E}_s^{\mathrm{loop}}\left(\mathbb{E}_s^{\mathrm{loop}}\left(\left|\left(X_k^{f,I} - X_{k+1}^{f,I}\right)\cdot\left[\{T^{\varphi} \geq k+1\}\right]\right| \,\middle|\, \mathcal{G}_k^{\mathrm{loop}}\right)\right)\\
&\qquad\qquad \text{(by Lemma A.6.4)}\\
&= \mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I}\right) + \sum_{k\in\mathbb{N}}\mathbb{E}_s^{\mathrm{loop}}\left(\mathbb{E}_s^{\mathrm{loop}}\left(\left|\left(X_k^{f,I} - X_{k+1}^{f,I}\right)\right| \,\middle|\, \mathcal{G}_k^{\mathrm{loop}}\right)\cdot\left[\{T^{\varphi} \geq k+1\}\right]\right)\\
&\qquad\qquad \text{(by Thm. A.6.5 (iii))}\\
&\leq \mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I}\right) + \sum_{k\in\mathbb{N}}\mathbb{E}_s^{\mathrm{loop}}\left(c\cdot\left[\{T^{\varphi} \geq k+1\}\right]\right)\\
&= \mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I}\right) + \sum_{k\in\mathbb{N}} c\cdot\mathbb{E}_s^{\mathrm{loop}}\left(\left[\{T^{\varphi} \geq k+1\}\right]\right) \qquad \text{(by Lemma A.3.8)}\\
&= \mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I}\right) + \sum_{k\in\mathbb{N}} c\cdot\mathbb{P}_s^{\mathrm{loop}}\left(T^{\varphi} \geq k+1\right)\\
&= \mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I}\right) + c\cdot\sum_{k\in\mathbb{N}}\mathbb{P}_s^{\mathrm{loop}}\left(T^{\varphi} \geq k+1\right)\\
&= \mathbb{E}_s^{\mathrm{loop}}\left(X_0^{f,I}\right) + c\cdot\sum_{k\in\mathbb{N}} k\cdot\mathbb{P}_s^{\mathrm{loop}}\left(T^{\varphi} = k\right)\\
&= {}^{\mathrm{wp}}\Phi_f(I)(s) + c\cdot\mathbb{E}_s^{\mathrm{loop}}\left(T^{\varphi}\right) < \infty\\
&\qquad\qquad \text{(by Cor. 14.4.27 and 14.4.28 and as } \mathbb{E}_s^{\mathrm{loop}}\left(T^{\varphi}\right) < \infty\text{)}
\end{aligned}
$$

By Lemma A.5.2 uniform integrability of $\mathbf{X}^{f,I}$ w.r.t. $\mathbb{P}_s^{\mathrm{loop}}$ follows. Therefore, we have that $\lim_{n\to\omega} {}^{\mathrm{wp}}\Phi_f^n(I)(s) = \mathrm{lfp}\,{}^{\mathrm{wp}}\Phi_f(s)$ by the proof of Cor. 15.3.3. Furthermore, we can extract the following for any $n \in \mathbb{N}$:

$$
\begin{aligned}
&{}^{\mathrm{wp}}\Phi_f^{n+1}(I)(s)\\
&= \mathbb{E}_s^{\mathrm{loop}}\left(X_n^{f,I}\right) \qquad \text{(by Cor. 14.4.27 and 14.4.28)}
\end{aligned}
$$

$$\leq \mathbb{E}_s^{\text{loop}}\left(W^{f,I}\right) \qquad \text{(by monotonicity of the expectation)}$$
$$\leq {}^{\text{wp}}\Phi_f(I)(s) + c \cdot \mathbb{E}_s^{\text{loop}}\left(T^{\varphi}\right) < \infty \qquad \text{(by the calculation above)}$$

So we have just shown that ${}^{\text{wp}}\Phi_f^n(I)(s) < \infty$ for any $n \in \mathbb{N}$. □

We note that by applying Lemma B.28.1 to every state $s \in \Sigma$ we also have the following result.

Corollary B.28.2 (Sufficient Condition for Uniform Integrability)**.** *Let $I \ll \infty$ be a conditionally difference bounded expectation that harmonizes with $f \ll \infty$, ${}^{\text{wp}}\Phi_f(I) \ll \infty$, and let the expected looping time of the underlying loop of the form in Fig. 14.16 be finite for every $s \in \Sigma$, where $\mathbf{P}$ is AST. Then I is u.i. for f and $\Phi_f^n(I) \ll \infty$ for any $n \in \mathbb{N}$.*

We now present a proof for Thm. 15.3.11 in analogy to the Optional Stopping Theorem (Thm. 15.3.5).

Proof of Thm. 15.3.11. That the subinvariant I is a lower bound iff it is uniformly integrable for f is exactly Thm. 15.1.8.

Recall that $\mathbf{X}^{f,I}_{\wedge T^{\varphi}} = \mathbf{X}^{f,I}$ holds (see the proof of Cor. B.28.2). Secondly, in any of the three cases (i) to (iii), we have ${}^{\text{wp}}\Phi_f^n(I) \ll \infty$ for any $n \in \mathbb{N}$: in (i) it is a precondition, in (ii) it holds due to Cor. B.28.2, and in (iii) the boundedness of f and I implies that ${}^{\text{wp}}\Phi_f^n(I)$ is bounded as well. So in particular, it is finite. Therefore, in any of the three cases $\mathbf{X}^{f,I}$ is a submartingale by Lemma 15.3.4 as I is a subinvariant.

Furthermore, in any of the three cases (i) to (iii) the underlying loop is almost-surely terminating. Hence by Lemma 15.2.3 we have

$$\lim_{n \longrightarrow \omega} X^{f,I}_{n \wedge T^{\varphi}} = \lim_{n \longrightarrow \omega} X^{f,I}_n = X^f_{T^{\varphi}},$$

almost-surely for every $s \in \Sigma$. So if we can prove for all of the three cases (i) to (iii)

that $\mathbf{X}^{f,I} = \mathbf{X}^{f,I}_{\wedge T^\varphi}$ is uniformly integrable w.r.t. $\mathbb{P}^{\text{loop}}_s$ for all $s \in \Sigma$, we have for all $s \in \Sigma$:

$$
\begin{aligned}
& I(s) \\
\leq\ & {}^{\text{wp}}\Phi_f(I)(s) && \text{(as } I \text{ is a subinvariant)} \\
=\ & \mathbb{E}^{\text{loop}}_s\left(X^{f,I}_0\right) && \text{(by Cor. 14.4.27 and 14.4.28)} \\
\leq\ & \lim_{n\to\omega} \mathbb{E}^{\text{loop}}_s\left(X^{f,I}_n\right) && \text{(as } \mathbf{X}^{f,I} \text{ is a submartingale)} \\
=\ & \mathbb{E}^{\text{loop}}_s\left(\lim_{n\to\omega} X^{f,I}_n\right) && \text{(by the uniform integrability of } \mathbf{X}^{f,I}\text{)} \\
=\ & \mathbb{E}^{\text{loop}}_s\left(X^{f}_{T^\varphi}\right) && \text{(by Lemma 15.2.3)} \\
=\ & (\operatorname{lfp}{}^{\text{wp}}\Phi_f)(s) && \text{(by Thm. 15.2.4 and wp = awp)}
\end{aligned}
$$

i.e., $I \preceq \operatorname{lfp}{}^{\text{wp}}\Phi_f = \text{wp}\,[\![\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}]\!]\,(f)$ as desired.

We now use the Optional Stopping Theorem (Thm. 15.3.5) and Cor. B.28.2 to prove the uniform integrability w.r.t. $\mathbb{P}^{\text{loop}}_s$ for all $s \in \Sigma$.

(i) Let $s \in \Sigma$. Then there is an $N(s) \in \mathbb{N}$ with $\mathbb{P}^{\text{loop}}_s(T^\varphi \preceq N(s))$, i.e., the looping time is almost-surely bounded for any $s \in \Sigma$. So by Thm. 15.3.5 (i), $\mathbf{X}^{f,I}$ is uniformly integrable w.r.t. $\mathbb{P}^{\text{loop}}_s$ for all $s \in \Sigma$.

(ii) Due to Cor. B.28.2, I is uniformly integrable for f. Hence $\mathbf{X}^{f,I}$ is uniformly integrable w.r.t. $\mathbb{P}^{\text{loop}}_s$ for all $s \in \Sigma$ by Cor. 15.3.3. As $\mathbf{X}^{f,I}$ is a submartingale, the result follows from Thm. 15.3.5.

(iii) If f and I are bounded, so is the process $\mathbf{X}^{f,I} = \mathbf{X}^{f,I}_{\wedge T^\varphi}$. By Thm. 15.3.5 (iii), $\mathbf{X}^{f,I}$ is uniformly integrable w.r.t. $\mathbb{P}^{\text{loop}}_s$ for all $s \in \Sigma$.

□

B.29 Proof of Thm. 15.4.1

Proof. Let $f, I \in \mathbb{F}$ be **bounded** such that $I \preceq {}^{\text{wp}}\Phi_f(I)$, i.e., I is a subinvariant and assume there is some $c \geq 0$ with $f, I \preceq c$. Then the process $\mathbf{X}^{f,I}$ satisfies $X^{f,I}_n \preceq c$ for every $n \in \mathbb{N}$. As I is a subinvariant, we have by Lemma 15.3.4 that $\mathbf{X}^{f,I}$ is a submartingale. By Thm. A.7.1, there exists a random variable $X^{f,I}_\omega$ such that $X^{f,I}_n$ converges to $X^{f,I}_\omega$ almost-surely. By Lemma 15.2.3, we get that for any run $\vartheta \in \Omega^{\text{loop}}$

with $T^{\varphi}(\vartheta) \in \mathbb{N}$ we must have $X_{\omega}^{f,I}(\vartheta) = X_{T^{\varphi}}^{f}(\vartheta)$, i.e., $X_{\omega}^{f,I} \cdot \left[(T^{\varphi})^{-1}(\mathbb{N})\right] = X_{T^{\varphi}}^{f}$. We conclude for an arbitrary initial state $s \in \Sigma$:

$$
\begin{aligned}
I(s) &\leq {}^{\mathsf{wp}}\Phi_f(I)(s) = \mathbb{E}_s^{\mathsf{loop}}\left(X_0^{f,I}\right) && \text{(by Cor. 14.4.27 and 14.4.28)}\\
&\leq \mathbb{E}_s^{\mathsf{loop}}\left(X_{\omega}^{f,I}\right) && \text{(by Thm. A.7.1)}\\
&= \mathbb{E}_s^{\mathsf{loop}}\left(\left(\left[(T^{\varphi})^{-1}(\mathbb{N})\right] + \left[(T^{\varphi})^{-1}(\{\omega\})\right]\right) \cdot X_{\omega}^{f,I}\right)\\
&= \mathbb{E}_s^{\mathsf{loop}}\left(\left[(T^{\varphi})^{-1}(\mathbb{N})\right] \cdot X_{\omega}^{f,I}\right) + \mathbb{E}_s^{\mathsf{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right] \cdot X_{\omega}^{f,I}\right)\\
&= \mathbb{E}_s^{\mathsf{loop}}\left(X_{T^{\varphi}}^{f}\right) + \mathbb{E}_s^{\mathsf{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right] \cdot X_{\omega}^{f,I}\right)\\
&= \mathsf{lfp}\, {}^{\mathsf{wp}}\Phi_f(s) + \mathbb{E}_s^{\mathsf{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right] \cdot X_{\omega}^{f,I}\right). && \text{(by Thm. 15.2.4)}
\end{aligned}
$$

Consequently,

$$
I(s) \leq \mathsf{lfp}\, {}^{\mathsf{wp}}\Phi_f(s) + \mathbb{E}_s^{\mathsf{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right] \cdot X_{\omega}^{f,I}\right). \tag{B.29.1}
$$

If I is an invariant, i.e., a *fixed* point of ${}^{\mathsf{wp}}\Phi_f$, the process $\mathbf{X}^{f,I}$ is a *martingale* by Cor. 14.4.27 and 14.4.28. By Thm. 14.4.25, we have for an arbitrary initial state s

$$
\mathbb{E}_s^{\mathsf{loop}}\left(X_{n+1}^{f,I} \,\middle|\, \mathcal{G}_n^{\mathsf{loop}}\right) = X_n^{f,{}^{\mathsf{wp}}\Phi_f(I)} = X_n^{f,I}.
$$

Hence, in this case = instead of ≤ holds in (B.29.1) by Thm. 14.4.25 and A.7.1. We now discuss the results of Thm. 14.4.1. First of all, $T(s) = \mathsf{wp}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](1)(s) = \mathbb{P}_s^{\mathsf{loop}}\left((T^{\varphi})^{-1}(\mathbb{N})\right)$ by using $X^{1,T} = \left[(T^{\varphi})^{-1}(\mathbb{N})\right]$ and Thm. 15.2.4.

(i) Assume $I = [G]$ for some predicate G, i.e., $I(s) \in \{0,1\}$. W.l.o.g. let $I(s) = 1$ as the claim holds trivially if $I(s) = 0$. Then

$$
\mathbb{E}_s^{\mathsf{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right] \cdot \underbrace{X_{\omega}^{f,I}}_{\leq 1}\right) \leq \mathbb{E}_s^{\mathsf{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right]\right) = \mathbb{P}_s^{\mathsf{loop}}\left(T^{\varphi}\right)^{-1}(\{\omega\})\big).
$$

By (B.29.1) we get

$$\begin{aligned} & I(s) \cdot T(s) \\ = \; & T(s) \\ = \; & \mathbb{P}_s^{\text{loop}}\left((T^{\varphi})^{-1}(\mathbb{N})\right) \\ = \; & 1 - \mathbb{P}_s^{\text{loop}}\left(T^{\varphi}\right)^{-1}(\{\omega\})\big) \\ = \; & I(s) - \mathbb{P}_s^{\text{loop}}\left(T^{\varphi}\right)^{-1}(\{\omega\})\big) \\ \leq \; & \text{lfp}^{\,\text{wp}}\Phi_f(s) \end{aligned}$$

so we have

$$I \cdot T \preceq \text{wp}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](f).$$

(ii) Assume that for some predicate G we have $[G] \preceq T$. Again, w.l.o.g. let $[G](s) = 1$. But then we must have $1 \leq T(s) = \mathbb{P}_s^{\text{loop}}\left((T^{\varphi})^{-1}(\mathbb{N})\right) \leq 1$, i.e., $\mathbb{P}_s^{\text{loop}}\left((T^{\varphi})^{-1}(\mathbb{N})\right) = 1$. So, $\mathbb{E}_s^{\text{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right] \cdot X_{\omega}^{f,I}\right) = 0$ and by (B.29.1) we have $[G](s) \cdot I(s) = I(s) \leq \text{wp}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](f)(s)$, i.e.,

$$[G] \cdot I \preceq \text{wp}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](f).$$

(iii) Assume there is some $\varepsilon > 0$ with $\varepsilon \cdot I \preceq T$. By definition, $T = \text{lfp}^{\,\text{wp}}\Phi_1$. By (B.29.1), we have

$$\begin{aligned} & T(s) \\ = \; & \text{lfp}^{\,\text{wp}}\Phi_1(s) + \mathbb{E}_s^{\text{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right] \cdot X_{\omega}^{1,T}\right) \\ = \; & T(s) + \mathbb{E}_s^{\text{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right] \cdot X_{\omega}^{1,T}\right), \end{aligned}$$

i.e., $\mathbb{E}_s^{\text{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right] \cdot X_{\omega}^{1,T}\right) = 0$. By definition, $\frac{T}{\varepsilon}$ is a fixed point of ${}^{\text{wp}}\Phi_{\frac{1}{\varepsilon}}$. Thus,

$$\begin{aligned} & \mathbb{E}_s^{\text{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right] \cdot X_{\omega}^{f,I}\right) \\ \leq \; & \mathbb{E}_s^{\text{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right] \cdot X_{\omega}^{\frac{1}{\varepsilon},\frac{T}{\varepsilon}}\right) && (f \text{ is irrelevant}) \\ = \; & \frac{1}{\varepsilon} \cdot \mathbb{E}_s^{\text{loop}}\left(\left[(T^{\varphi})^{-1}(\{\omega\})\right] \cdot X_{\omega}^{1,T}\right) = \frac{1}{\varepsilon} \cdot 0 = 0 \end{aligned}$$

So by (B.29.1), we can conclude that $I(s) \leq \text{lfp}^{\,\text{wp}}\Phi_f(s)$ for any state s, i.e.,

$$I \preceq \text{wp}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](f).$$

□

B.30 Proof of Thm. 15.6.1

Proof. We use the connection between wp and ert in Thm. 14.3.12: For any purely probabilistic pGCL-program $\mathbf{P}$ we have

$$\operatorname{ert}\,[\![\mathbf{P}]\!]\,(t) = \operatorname{wp}\,[\![\mathbf{P}]\!]\,(t) + \operatorname{ert}\,[\![\mathbf{P}]\!]\,(0)\,.$$

Our goal is to show ${}^{\text{ert}}\Phi_t^n(I) = {}^{\text{wp}}\Phi_t^n(I) + {}^{\text{ert}}\Phi_0^n(0)$ for all $n \in \mathbb{N}$. We use induction on n to prove this result. In the base, case we have $n = 0$. Here, we obtain

$$\begin{aligned}
&{}^{\text{ert}}\Phi_t^0(I)\\
&= I\\
&= I + 0\\
&= {}^{\text{wp}}\Phi_t^0(I) + {}^{\text{ert}}\Phi_0^0(0).
\end{aligned}$$

In the induction step, we use the induction hypothesis ${}^{\text{ert}}\Phi_t^n(I) = {}^{\text{wp}}\Phi_t^n(I) + {}^{\text{ert}}\Phi_0^n(0)$. Then we have

$$\begin{aligned}
&{}^{\text{ert}}\Phi_t^{n+1}(I)\\
&= {}^{\text{ert}}\Phi_t({}^{\text{ert}}\Phi_t^n(I))\\
&= 1 + [\neg\varphi]\cdot t + [\varphi]\cdot \operatorname{ert}\,[\![\mathbf{P}]\!]\,\left({}^{\text{ert}}\Phi_t^n(I)\right)\\
&= 1 + [\neg\varphi]\cdot t + [\varphi]\cdot \operatorname{wp}\,[\![\mathbf{P}]\!]\,\left({}^{\text{ert}}\Phi_t^n(I)\right) + [\varphi]\cdot \operatorname{ert}\,[\![\mathbf{P}]\!]\,(0) && \text{(by Thm. 14.3.12)}\\
&= 1 + [\neg\varphi]\cdot t + [\varphi]\cdot \operatorname{wp}\,[\![\mathbf{P}]\!]\,\left({}^{\text{wp}}\Phi_t^n(I) + {}^{\text{ert}}\Phi_0^n(0)\right) + [\varphi]\cdot \operatorname{ert}\,[\![\mathbf{P}]\!]\,(0) && \text{(by I.H.)}\\
&= 1 + [\neg\varphi]\cdot t + [\varphi]\cdot \operatorname{wp}\,[\![\mathbf{P}]\!]\,\left({}^{\text{wp}}\Phi_t^n(I)\right) + [\varphi]\cdot \operatorname{wp}\,[\![\mathbf{P}]\!]\,\left({}^{\text{ert}}\Phi_0^n(0)\right) + [\varphi]\cdot \operatorname{ert}\,[\![\mathbf{P}]\!]\,(0)\\
&\qquad\qquad \text{(by Thm. 14.3.4 since } \mathbf{P} \text{ is purely probabilistic)}\\
&= 1 + {}^{\text{wp}}\Phi_t\left({}^{\text{wp}}\Phi_t^n(I)\right) + [\varphi]\cdot \operatorname{wp}\,[\![\mathbf{P}]\!]\,\left({}^{\text{ert}}\Phi_0^n(0)\right) + [\varphi]\cdot \operatorname{ert}\,[\![\mathbf{P}]\!]\,(0)\\
&= 1 + {}^{\text{wp}}\Phi_t^{n+1}(I) + [\varphi]\cdot \operatorname{ert}\,[\![\mathbf{P}]\!]\,\left({}^{\text{ert}}\Phi_0^n(0)\right) && \text{(by Thm. 14.3.12)}\\
&= {}^{\text{wp}}\Phi_t^{n+1}(I) + 1 + [\neg\varphi]\cdot 0 + [\varphi]\cdot \operatorname{ert}\,[\![\mathbf{P}]\!]\,\left({}^{\text{ert}}\Phi_0^n(0)\right)\\
&= {}^{\text{wp}}\Phi_t^{n+1}(I) + {}^{\text{ert}}\Phi_0^{n+1}(0).
\end{aligned}$$

So ${}^{\text{ert}}\Phi_t^n(I) = {}^{\text{wp}}\Phi_t^n(I) + {}^{\text{ert}}\Phi_0^n(0)$ holds for all $n \in \mathbb{N}$.

Now let $s \in \Sigma$. Then one of the following two cases occurs.

(i) $\operatorname{ert}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](0)(s) = \infty$

In this case, we have by Thm. 14.3.12 $\operatorname{ert}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](t)(s) = \infty \geq I(s)$.

(ii) $\operatorname{ert}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](0)(s) < \infty$

First of all, in Thm. 15.3.11 (ii) we have seen that if I harmonizes with t, I is conditionally difference bounded, ${}^{\mathrm{wp}}\Phi_t(I) \ll \infty$, and for every $s' \in \Sigma$ it is $\mathbb{E}^{\mathrm{loop}}_{s'}(T^{\varphi}) < \infty$, we have $\lim_{n\to\omega} {}^{\mathrm{wp}}\Phi^n_t(I) = \operatorname{lfp} {}^{\mathrm{wp}}\Phi_t$. However, in Thm. 15.3.11 (ii) we need that the expected looping time is finite for *every* initial state $s' \in \Sigma$. As we cannot ensure this condition, we use Lemma B.28.1, a specialized result used in the proof of Thm. 15.3.11 which is indeed dependent on the initial state $s \in \Sigma$.

Furthermore, the expected runtime of the program with initial state $s \in \Sigma$ is finite. So, the expected looping time of the program has to be finite as well, i.e., $\operatorname{ert}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](0)(s) < \infty$ implies $\mathbb{E}^{\mathrm{loop}}_{s}(T^{\varphi}) < \infty$.

Hence, I harmonizes with t, I is conditionally difference bounded, ${}^{\mathrm{wp}}\Phi_t(I) \ll \infty$, and $\mathbb{E}^{\mathrm{loop}}_{s}(T^{\varphi}) < \infty$. Thus, we can apply Lemma B.28.1 and get

$$\lim_{n\to\omega} {}^{\mathrm{wp}}\Phi^n_t(I)(s) = \operatorname{lfp} {}^{\mathrm{wp}}\Phi_t(s) = \mathrm{wp}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](t)(s).$$

Hence we have

$$\begin{aligned}
&\lim_{n\to\omega} {}^{\mathrm{ert}}\Phi^n_t(I)(s)\\
&= \lim_{n\to\omega} {}^{\mathrm{wp}}\Phi^n_t(I)(s) + {}^{\mathrm{ert}}\Phi^n_0(0)(s) && \text{(by calculation above)}\\
&= \mathrm{wp}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](t)(s) + \operatorname{ert}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](0)(s)\\
&= \operatorname{ert}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](t)(s). && \text{(by Thm. 14.3.12)}
\end{aligned}$$

But the sequence $\left({}^{\mathrm{ert}}\Phi^n_t(I)\right)_{n\in\mathbb{N}}$ is monotonically increasing as I is an ert-subinvariant. Hence,

$$I(s) \leq \lim_{n\to\omega} {}^{\mathrm{ert}}\Phi^n_t(I)(s) = \operatorname{ert}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](t)(s).$$

Combining these results, we get $I \preceq \operatorname{ert}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](t)$. □

B.31 Proof of Thm. 15.7.3

We adapt the proof of Thm. 15.3.11. Again, we first show that in (ii), it is enough to have ${}^{\mathrm{awp}}\Phi_f(I) \ll \infty$ for showing that $\mathbf{X}^{f,I}$ is a submartingale.

Lemma B.31.1. *Let $I \ll \infty$ be a worst-case conditionally difference bounded expectation that harmonizes with $f \ll \infty$, ${}^{\mathrm{awp}}\Phi_f(I) \ll \infty$. Let $s \in \Sigma$. Let the expected looping time of the loop be finite for every scheduler $\mathfrak{s} \in \mathsf{Scheds}$, i.e., $\mathbb{E}^{loop}_{s,\mathfrak{s}}(T^\varphi) < \infty$. Then $\mathbb{E}^{loop}_{s,\mathfrak{s}}\left(X^{f,I}_n\right) < \infty$ for all $n \in \mathbb{N}$ and $\mathbf{X}^{f,I}$ is uniformly integrable w.r.t. $\mathbb{P}^{loop}_{s,\mathfrak{s}}$ for all $\mathfrak{s} \in \mathsf{Scheds}$.*

Proof. We adapt the proof of Lemma B.28.1. Fix a scheduler $\mathfrak{s} \in \mathsf{Scheds}$. Consider the process $\mathbf{X}^{f,I}$ as studied in Sect. 15.2. As I harmonizes with f, we have seen in Thm. 15.7.8 that if I is worst-case conditionally difference bounded by the constant $c \geq 0$, then $\mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(\left|X^{f,I}_{n+1} - X^{f,I}_n\right| \,\middle|\, \mathcal{G}^{\mathrm{loop}}_n\right) \leq c$, where $\mathcal{G}^{\mathrm{loop}}_n$ belongs to the filtration defined in Def. 14.4.22. Now we show that $\mathbf{X}^{f,I}$ is uniformly integrable w.r.t. $\mathbb{P}^{\mathrm{loop}}_{s,\mathfrak{s}}$.

Note that by definition of $\mathbf{X}^{f,I}$ (Def. 14.4.21), we have $\mathbf{X}^{f,I}_{\wedge T^\varphi} = \mathbf{X}^{f,I}$ (see the proof of Lemma B.28.1). Similar to the proof of Lemma B.28.1, we have for any $n \in \mathbb{N}$, and any run $\vartheta \in \Omega^{\mathrm{loop}}$

$$X^{f,I}_n(\vartheta) \leq \underbrace{X^{f,I}_0(\vartheta) + \sum_{k\in\mathbb{N}} \left|X^{f,I}_{k+1}(\vartheta) - X^{f,I}_k(\vartheta)\right| \cdot \left[\{T^\varphi \geq k+1\}\right](\vartheta)}_{=W^{f,I}(\vartheta)}.$$

We show that the expectation of $W^{f,I}$ is finite.

$$\begin{aligned}
&\mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(W^{f,I}\right)\\
&= \mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^{f,I}_0 + \sum_{k\in\mathbb{N}} \left|\left(X^{f,I}_k - X^{f,I}_{k+1}\right) \cdot \left[\{T^\varphi \geq k+1\}\right]\right|\right)\\
&= \mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^{f,I}_0\right) + \sum_{k\in\mathbb{N}} \mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(\left|\left(X^{f,I}_k - X^{f,I}_{k+1}\right) \cdot \left[\{T^\varphi \geq k+1\}\right]\right|\right)\\
&= \mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^{f,I}_0\right) + \sum_{k\in\mathbb{N}} \mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(\mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(\left|\left(X^{f,I}_k - X^{f,I}_{k+1}\right) \cdot \left[\{T^\varphi \geq k+1\}\right]\right| \,\middle|\, \mathcal{G}^{\mathrm{loop}}_k\right)\right)\\
&\qquad\qquad \text{(by Lemma A.6.4)}
\end{aligned}$$

$$= \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(X_0^{f,I}\right) + \sum_{k\in\mathbb{N}} \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(\mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(\left|\left(X_k^{f,I} - X_{k+1}^{f,I}\right)\right| \,\middle|\, \mathcal{G}_k^{\text{loop}}\right) \cdot \left[\{T^\varphi \geq k+1\}\right]\right)$$

(by Thm. A.6.5 (iii))

$$\leq \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(X_0^{f,I}\right) + \sum_{k\in\mathbb{N}} \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(c \cdot \left[\{T^\varphi \geq k+1\}\right]\right)$$

$$= \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(X_0^{f,I}\right) + \sum_{k\in\mathbb{N}} c \cdot \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(\left[\{T^\varphi \geq k+1\}\right]\right)$$

$$= \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(X_0^{f,I}\right) + \sum_{k\in\mathbb{N}} c \cdot \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\left(T^\varphi \geq k+1\right)$$

$$= \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(X_0^{f,I}\right) + c \cdot \sum_{k\in\mathbb{N}} \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\left(T^\varphi \geq k+1\right)$$

$$= \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(X_0^{f,I}\right) + c \cdot \sum_{k\in\mathbb{N}} k \cdot \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\left(T^\varphi = k\right)$$

$$\leq {}^{\text{awp}}\Phi_f(I)(s) + c \cdot \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(T^\varphi\right) < \infty\,. \qquad \text{(by Cor. 14.4.28 and as } \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(T^\varphi\right) < \infty\text{)}$$

We can extract the following for any $n \in \mathbb{N}$:

$$\begin{aligned}
&\mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(X_n^{f,I}\right) \\
&\leq \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(W^{f,I}\right) && \text{(by monotonicity of the expectation)} \\
&\leq {}^{\text{awp}}\Phi_f(I)(s) + c \cdot \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(T^\varphi\right) < \infty\,. && \text{(by the calculation above)}
\end{aligned}$$

So we have just shown that $\mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(X_n^{f,I}\right) < \infty$ for any $n \in \mathbb{N}$ and any $\mathfrak{s} \in \text{Scheds}$ and that $\mathbf{X}^{f,I}$ is uniformly integrable by Lemma A.5.2 w.r.t. $\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}$. □

Of course, the generalization of Cor. B.28.2 also holds.

Corollary B.31.2 (Sufficient Condition for Uniform Integrability in Case of Non-Determinism)**.** *Let $I \ll \infty$ be a worst-case conditionally difference bounded expectation that harmonizes with $f \ll \infty$, ${}^{\text{awp}}\Phi_f(I) \ll \infty$, and let the expected looping time of the underlying loop of the form in Fig. 14.16 be finite for every $s \in \Sigma$ and $\mathfrak{s} \in \text{Scheds}$, where $\mathbf{P}$ is AST. Then $\mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(X_n^{f,I}\right) < \infty$ for all $n \in \mathbb{N}$, $s \in \Sigma$, and $\mathfrak{s} \in \text{Scheds}$ and $\mathbf{X}^{f,I}$ is uniformly integrable w.r.t. $\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}$ for all $s \in \Sigma$ and $\mathfrak{s} \in \text{Scheds}$.*

Now we prove Thm. 15.7.3.

Proof of Thm. 15.7.3. Recall that $\mathbf{X}^{f,I}_{\wedge T^\varphi} = \mathbf{X}^{f,I}$ holds. Secondly, in any of the three cases, $\mathbf{X}^{f,I}$ is a submartingale as I is a wp-subinvariant by Lemmas 15.7.4 and B.31.1.

Furthermore, in any of the three cases (i) to (iii), the loop in Fig. 14.16 is almost-surely terminating. Hence by Lemma 15.2.3 we have

$$\lim_{n\to\omega} X^{f,I}_{n\wedge T^\varphi} = \lim_{n\to\omega} X^{f,I}_n = X^f_{T^\varphi},$$

almost-surely for every $s \in \Sigma$ and $\mathfrak{s} \in \mathsf{Scheds}$. So if we can prove for all of the three cases (i) to (iii) that $\mathbf{X}^{f,I} = \mathbf{X}^{f,I}_{\wedge T^\varphi}$ is uniformly integrable w.r.t. $\mathbb{P}^{\mathrm{loop}}_{s,\mathfrak{s}}$ for any $s \in \Sigma$ and $\mathfrak{s} \in \mathsf{Scheds}$, we have independent of $s \in \Sigma$ and $\mathfrak{s} \in \mathsf{Scheds}$

$$\mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^{f,I}_0\right) \leq \mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}\left(X^f_{T^\varphi}\right)$$

as $\mathbf{X}^{f,I}$ is a submartingale. Thus we have by Conj. 15.7.2

$$\begin{aligned}
I(s) &\leq {}^{\mathrm{wp}}\Phi_f(I)(s) && \text{(as } I \text{ is a wp-subinvariant)}\\
&= \inf_{\mathfrak{s}'\in\mathsf{Scheds}} \mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}'}\left(X^{f,I}_0\right) && \text{(by Conj. 15.7.2 (i))}\\
&\leq \inf_{\mathfrak{s}'\in\mathsf{Scheds}} \mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}'}\left(X^f_{T^\varphi}\right)\\
&= \mathrm{wp}\left[\!\left[\textbf{while}\,(\varphi)\,\textbf{do}\,\{\,\mathbf{P}\,\}\right]\!\right](f)(s) && \text{(by Conj. 15.7.2 (ii))}
\end{aligned}$$

as desired.

We now use the Optional Stopping Theorem (Thm. 15.3.5) and Cor. B.31.2 to prove uniform integrability.

(i) Let $s \in \Sigma$ and $\mathfrak{s} \in \mathsf{Scheds}$. Then there is an $N(s,\mathfrak{s}) \in \mathbb{N}$ with $\mathbb{P}^{\mathrm{loop}}_{s,\mathfrak{s}}(T^\varphi \leq N(s,\mathfrak{s}))$, i.e., the looping time of Fig. 14.16 is almost-surely bounded for any $s \in \Sigma$. By Thm. 15.3.5 (i), $\mathbf{X}^{f,I}$ is uniformly integrable w.r.t. $\mathbb{P}^{\mathrm{loop}}_{s,\mathfrak{s}}$ for any $s \in \Sigma$ and $\mathfrak{s} \in \mathsf{Scheds}$.

(ii) Uniform integrability w.r.t. $\mathbb{P}^{\mathrm{loop}}_{s,\mathfrak{s}}$ for any $s \in \Sigma$ and $\mathfrak{s} \in \mathsf{Scheds}$ of $\mathbf{X}^{f,I}$ follows directly from Cor. B.31.2.

(iii) If f and I are bounded, so is the process $\mathbf{X}^{f,I} = \mathbf{X}^{f,I}_{\wedge T^\varphi}$. By Thm. 15.3.5 (iii), $\mathbf{X}^{f,I}$ is uniformly integrable w.r.t. $\mathbb{P}^{\mathrm{loop}}_{s,\mathfrak{s}}$ for any $s \in \Sigma$ and $\mathfrak{s} \in \mathsf{Scheds}$.

□

B.32 Proof of Lemma 15.7.4

Proof. This proof is similar to the proof of Lemma 15.3.4.

Let $n \in \mathbb{N}$, $s \in \Sigma$, and $\mathfrak{s} \in \mathrm{Scheds}$. We have already proved in Lemma 14.4.23 that $X_n^{f,I}$ is $\mathcal{G}_n^{\mathrm{loop}}$-measurable. First of all, by Cor. 14.4.28 we have $\mathbb{E}_{s,\mathfrak{s}}^{\mathrm{loop}}\left(X_n^{f,I}\right) \leq {}^{\mathrm{awp}}\Phi_f^{n+1}(I)(s) < \infty$. Secondly, by Thm. 14.4.25 and ${}^{\mathrm{wp}}\Phi_f(I) \succeq I$ we have

$$\begin{aligned}
&\mathbb{E}_{s,\mathfrak{s}}^{\mathrm{loop}}\left(X_{n+1}^{f,I} \,\middle|\, \mathcal{G}_n^{\mathrm{loop}}\right) \\
&\geq \inf_{\mathfrak{s}' \in \mathrm{Scheds}} \mathbb{E}_{s,\mathfrak{s}'}^{\mathrm{loop}}\left(X_{n+1}^{f,I} \,\middle|\, \mathcal{G}_n^{\mathrm{loop}}\right) \\
&= X_n^{f,{}^{\mathrm{wp}}\Phi_f(I)} && \text{(by Thm. 14.4.25)} \\
&\geq X_n^{f,I}.
\end{aligned}$$

By Def. 14.4.29, this proves that $\mathbf{X}^{f,I}$ is a submartingale with respect to $(\mathcal{G}_n^{\mathrm{loop}})_{n\in\mathbb{N}}$. □

B.33 Proof of Thm. 15.7.8

Proof. The proof of this result is similar to the proof of Thm. 15.3.10.

First of all, $X_n^{0,\Delta I}$ is $\mathcal{G}_n^{\mathrm{loop}}$-measurable as seen in Lemma 14.4.23. As seen in the proof of Thm. 15.3.10, for any $\vartheta \in \Omega^{\mathrm{loop}}$ we have

$$\left|X_{n+1}^{f,I} - X_n^{f,I}\right|(\vartheta) = \begin{cases} 0, & \text{if } T^{\varphi}(\vartheta) \leq n+1 \\ \left|I(\vartheta[n+2]) - I(\vartheta[n+1])\right|, & \text{if } T^{\varphi}(\vartheta) > n+1 \end{cases}$$

as I harmonizes with f.

Fix $s \in \Sigma$ and $\mathfrak{s} \in \mathrm{Scheds}$. To show the result we prove

$$\begin{aligned}
&\mathbb{E}_{s,\mathfrak{s}}^{\mathrm{loop}}\left(\left|X_{n+1}^{f,I} - X_n^{f,I}\right| \cdot [\mathrm{Cyl}(\pi)]\right) \\
&= \mathbb{P}_{s,\mathfrak{s}}^{\mathrm{loop}}(\mathrm{Cyl}(\pi)) \cdot [\varphi](s_{n+1}) \cdot \sum_{s_{n+2}\in\Sigma} \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}}(s_{n+2}) \cdot |I(s_{n+2}) - I(s_{n+1})| \qquad \text{(B.33.1)}
\end{aligned}$$

for any $\pi \in \Sigma^{n+2}$ with $\mathbb{P}_{s,\mathfrak{s}}^{\mathrm{loop}}(\mathrm{Cyl}(\pi)) > 0$ and use Lemma A.6.2 to obtain the desired result. Moreover, similar as in the proof of Thm. 14.4.25 in App. B.21, we choose the conditional expectation of $\left|X_{n+1}^{f,I} - X_n^{f,I}\right|$ to take the following value on $[\mathrm{Cyl}(\pi)]$ for $\pi = s_0 \cdots s_{n+1} \in \Sigma^{n+2}$ if $\mathbb{P}_{s,\mathfrak{s}}^{\mathrm{loop}}(\mathrm{Cyl}(\pi)) = 0$:

$$\begin{cases} \sum_{s_{n+2}\in\Sigma} \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}}(s_{n+2}) \cdot |I(s_{n+2}) - I(s_{n+1})|, & \text{if } \pi \in \Sigma_\varphi^{n+2} \\ 0, & \text{otherwise.} \end{cases}$$

Take any admissible $\pi \in \Sigma^{n+2}$, i.e., $\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}(\text{Cyl}(\pi)) > 0$. Furthermore, as $\left|X^{f,I}_{n+1} - X^{f,I}_n\right|$ is zero on $\text{Cyl}(\pi)$ if $s_i \in \Sigma_{\neg\varphi}$ for some $i \leq n+1$ and thus $s_{n+1} \in \Sigma_{\neg\varphi}$ due to admissibility, (B.33.1) holds trivially. So, assume $\pi = s_0 \cdots s_{n+1} \in s\Sigma_\varphi^{n+1}$.

$$\begin{aligned}
&\mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(\left|X^{f,I}_{n+1} - X^{f,I}_n\right| \cdot [\text{Cyl}(\pi)]\right) \\
&= \int_{\text{Cyl}(\pi)} \left|X^{f,I}_{n+1} - X^{f,I}_n\right| \, d\left(\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\right) \\
&= \int_{\biguplus_{s_{n+2}\in\Sigma} \text{Cyl}(\pi s_{n+2})} \left|X^{f,I}_{n+1} - X^{f,I}_n\right| \, d\left(\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\right) \\
&= \sum_{s_{n+2}\in\Sigma} \int_{\text{Cyl}(\pi s_{n+2})} \left|X^{f,I}_{n+1} - X^{f,I}_n\right| \, d\left(\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\right) \\
&= \sum_{s_{n+2}\in\Sigma} \int_{\text{Cyl}(\pi s_{n+2})} |I(s_{n+2}) - I(s_{n+1})| \, d\left(\mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}\right) \\
&= \sum_{s_{n+2}\in\Sigma} \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}(\text{Cyl}(\pi s_{n+2})) \cdot |I(s_{n+2}) - I(s_{n+1})| && \text{(by Lemma A.3.8)} \\
&= \sum_{s_{n+2}\in\Sigma} \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}(\text{Cyl}(\pi)) \cdot \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}}(s_{n+2}) \cdot |I(s_{n+2}) - I(s_{n+1})| && \text{(by Lemma 14.4.9)} \\
&= \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}(\text{Cyl}(\pi)) \cdot \sum_{s_{n+2}\in\Sigma} \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}}(s_{n+2}) \cdot |I(s_{n+2}) - I(s_{n+1})| \\
&= \mathbb{P}^{\text{loop}}_{s,\mathfrak{s}}(\text{Cyl}(\pi)) \cdot [\varphi](s_{n+1}) \cdot \sum_{s_{n+2}\in\Sigma} \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}}(s_{n+2}) \cdot |I(s_{n+2}) - I(s_{n+1})|
\end{aligned}$$

So by Lemma A.6.2, we have for all $\pi \in \Sigma^{n+2}$

$$\begin{aligned}
&[\text{Cyl}(\pi)] \cdot \sup_{\mathfrak{s}\in\text{Scheds}} \mathbb{E}^{\text{loop}}_{s,\mathfrak{s}}\left(\left|X^{f,I}_{n+1} - X^{f,I}_n\right| \,\middle|\, \mathcal{G}^{\text{loop}}_n\right) \\
&= [\text{Cyl}(\pi)] \cdot \begin{cases} \sup_{\mathfrak{s}\in\text{Scheds}} \sum_{s_{n+2}\in\Sigma} \mu^{\mathbf{P}}_{s_{n+1},\mathfrak{s}}(s_{n+2}) \cdot |I(s_{n+2}) - I(s_{n+1})|, & \text{if } \pi \in \Sigma_\varphi^{n+2} \\ 0, & \text{otherwise} \end{cases}
\end{aligned}$$

$$= [\mathrm{Cyl}(\pi)] \cdot \begin{cases} \text{awp } [\![\mathbf{P}]\!] \left(|I - I(s_{n+1})|\right)(s_{n+1}), & \text{if } \pi \in \Sigma_\varphi^{n+2} \\ 0, & \text{otherwise} \end{cases}$$

$$= [\mathrm{Cyl}(\pi)] \cdot \begin{cases} \Delta I(s_{n+1}), & \text{if } \pi \in \Sigma_\varphi^{n+2} \\ 0, & \text{otherwise} \end{cases}$$

$$= [\mathrm{Cyl}(\pi)] \cdot X_n^{0,\Delta I}.$$

□

B.34 Proof of Thm. 15.7.10

Proof. The proof of this result is analogous to the proof of Thm. 15.6.1.

We use the connection between awp and ert in Cor. 15.7.9: For any probabilistic program $\mathbf{P}$ we have

$$\text{ert } [\![\mathbf{P}]\!]\ (t) = \text{awp } [\![\mathbf{P}]\!]\ (t) + \text{ert } [\![\mathbf{P}]\!]\ (0)\,.$$

Our goal is to show ${}^{\text{ert}}\Phi_t^n(I) \preceq {}^{\text{awp}}\Phi_t^n(I) + {}^{\text{ert}}\Phi_0^n(0)$ for all $n \in \mathbb{N}$. We use induction on n to prove this result. In the base case, we have $n = 0$. Here, we obtain

$$\begin{aligned} & {}^{\text{ert}}\Phi_t^0(I) \\ &= I \\ &= I + 0 \\ &= {}^{\text{awp}}\Phi_t^0(I) + {}^{\text{ert}}\Phi_0^0(0). \end{aligned}$$

In the induction step, we use the induction hypothesis ${}^{\text{ert}}\Phi_t^n(I) \preceq {}^{\text{awp}}\Phi_t^n(I) + {}^{\text{ert}}\Phi_0^n(0)$. Then we have

$$\begin{aligned} & {}^{\text{ert}}\Phi_t^{n+1}(I) \\ &= {}^{\text{ert}}\Phi_t({}^{\text{ert}}\Phi_t^n(I)) \\ &= 1 + [\neg\varphi] \cdot t + [\varphi] \cdot \text{ert } [\![\mathbf{P}]\!] \left({}^{\text{ert}}\Phi_t^n(I)\right) \\ &= 1 + [\neg\varphi] \cdot t + [\varphi] \cdot \text{awp } [\![\mathbf{P}]\!] \left({}^{\text{ert}}\Phi_t^n(I)\right) + [\varphi] \cdot \text{ert } [\![\mathbf{P}]\!]\ (0) && \text{(by Cor. 15.7.9)} \\ &\preceq 1 + [\neg\varphi] \cdot t + [\varphi] \cdot \text{awp } [\![\mathbf{P}]\!] \left({}^{\text{awp}}\Phi_t^n(I) + {}^{\text{ert}}\Phi_0^n(0)\right) + [\varphi] \cdot \text{ert } [\![\mathbf{P}]\!]\ (0) && \text{(by I.H.)} \\ &\preceq 1 + [\neg\varphi] \cdot t + [\varphi] \cdot \text{awp } [\![\mathbf{P}]\!] \left({}^{\text{awp}}\Phi_t^n(I)\right) + [\varphi] \cdot \text{awp } [\![\mathbf{P}]\!] \left({}^{\text{ert}}\Phi_0^n(0)\right) + [\varphi] \cdot \text{ert } [\![\mathbf{P}]\!]\ (0) \\ & && \text{(by Thm. 14.3.4 for awp)} \end{aligned}$$

$$
\begin{aligned}
&= 1 + {}^{\mathrm{awp}}\Phi_t^{n+1}(I) + [\varphi] \cdot \mathrm{awp}\ [\![\mathbf{P}]\!]\ \left({}^{\mathrm{ert}}\Phi_0^n(0)\right) + [\varphi] \cdot \mathrm{ert}\ [\![\mathbf{P}]\!]\ (0) \\
&= 1 + {}^{\mathrm{awp}}\Phi_t\left({}^{\mathrm{awp}}\Phi_t^n(I)\right) + [\varphi] \cdot \mathrm{ert}\ [\![\mathbf{P}]\!]\ \left({}^{\mathrm{ert}}\Phi_0^n(0)\right) && \text{(by Cor. 15.7.9)} \\
&= {}^{\mathrm{awp}}\Phi_t^{n+1}(I) + 1 + [\neg\varphi] \cdot 0 + [\varphi] \cdot \mathrm{ert}\ [\![\mathbf{P}]\!]\ \left({}^{\mathrm{ert}}\Phi_0^n(0)\right) \\
&= {}^{\mathrm{awp}}\Phi_t^{n+1}(I) + {}^{\mathrm{ert}}\Phi_0^{n+1}(0).
\end{aligned}
$$

So ${}^{\mathrm{ert}}\Phi_t^n(I) \preceq {}^{\mathrm{awp}}\Phi_t^n(I) + {}^{\mathrm{ert}}\Phi_0^n(0)$ holds for all $n \in \mathbb{N}$.

Now let $s \in \Sigma$. Then one of the following two cases occurs.

(i) $\mathrm{ert}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (0)\,(s) = \infty$

In this case, we have $\mathrm{ert}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (t)\,(s) = \infty \geq I(s)$ by Cor. 15.7.9.

(ii) $\mathrm{ert}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (0)\,(s) < \infty$

First of all, assuming Conj. 15.7.2 (i) and (ii), we have seen in Thm. 15.7.3 (ii) that if I is worst-case conditionally difference bounded, ${}^{\mathrm{awp}}\Phi_t(I) \ll \infty$, and for every $s' \in \Sigma$ and $\mathfrak{s} \in \mathrm{Scheds}$ we have $\mathbb{E}^{\mathrm{loop}}_{s',\mathfrak{s}}(T^\varphi) < \infty$, $\lim_{n\to\omega} {}^{\mathrm{wp}}\Phi_t^n(I) = \mathrm{lfp}\,{}^{\mathrm{wp}}\Phi_t$ holds. However, in Thm. 15.7.3 (ii) we need that the expected looping time is finite for *every* initial state $s' \in \Sigma$ and every scheduler $\mathfrak{s} \in \mathrm{Scheds}$. As we cannot ensure this condition, we use Lemma B.31.1, a specialized result used in the proof of Thm. 15.7.3 which is indeed dependent on the initial state $s \in \Sigma$.

Furthermore, the expected runtime of the program with initial state $s \in \Sigma$ is finite for every $\mathfrak{s} \in \mathrm{Scheds}$. So, the expected looping time has to be finite as well, i.e., $\mathrm{ert}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (0)\,(s) < \infty$ implies $\mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}(T^\varphi) < \infty$ for all $\mathfrak{s} \in \mathrm{Scheds}$. Hence, I harmonizes with t, I is worst-case conditionally difference bounded,${}^{\mathrm{awp}}\Phi_t(I) \ll \infty$, and $\mathbb{E}^{\mathrm{loop}}_{s,\mathfrak{s}}(T^\varphi) < \infty$ for all $\mathfrak{s} \in \mathrm{Scheds}$.

Thus, we can apply Lemma B.28.1 and get

$$\lim_{n\to\omega} {}^{\mathrm{wp}}\Phi_t^n(I)(s) = \mathrm{lfp}\,{}^{\mathrm{wp}}\Phi_t(s) = \mathrm{wp}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (t)\,(s).$$

As I is a wp-subinvariant it is also an awp-subinvariant and therefore an ert-subinvariant by Cor. 15.7.9. So, ${}^{\mathrm{wp}}\Phi_t^n(I)(s)$ is a monotonic sequence and thus

$$I \preceq \mathrm{wp}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (t)\,(s) \preceq \mathrm{awp}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (t)\,(s).$$

By Thm. 15.1.5, this means that we must have

$$\lim_{n\to\omega} {}^{\mathrm{awp}}\Phi_t^n(I)(s) = \mathrm{lfp}\,{}^{\mathrm{awp}}\Phi_t(s) = \mathrm{awp}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (t)\,(s).$$

We conclude

$$\begin{aligned}
&\lim_{n\to\omega} {}^{\mathrm{ert}}\Phi_t^n(I)(s)\\
\leq\ &\lim_{n\to\omega} {}^{\mathrm{awp}}\Phi_t^n(I)(s) + {}^{\mathrm{ert}}\Phi_0^n(0)(s) && \text{(by calculation above)}\\
=\ &\mathrm{awp}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (t)\,(s) + \mathrm{ert}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (0)\,(s)\\
=\ &\mathrm{ert}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (t)\,(s).
\end{aligned}$$

But the sequence ${}^{\mathrm{ert}}\Phi_t^n(I)_{n\in\mathbb{N}}$ is monotonically increasing as I is also an ert-subinvariant. Hence,

$$I(s) \leq \lim_{n\to\omega} {}^{\mathrm{ert}}\Phi_t^n(I)(s) = \mathrm{ert}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (t)\,(s).$$

Combining these results, we get $I \preceq \mathrm{ert}\ [\![\mathbf{while}\ (\varphi)\ \mathbf{do}\ \{\ \mathbf{P}\ \}]\!]\ (t)$. □

B.35 Proof of Cor. 16.1.5

Proof. Any admissible run of the loop starting in $\vec{x}_0$ stays in $\vec{x}_0$ forever, i.e., there is a unique admissible run whose looping time is 0. Thus,

$$\mathrm{explt}\ [\![\mathbf{L}]\!]\left(\vec{x}_0\right) = \mathbb{E}^{\mathbf{L}}_{\vec{x}_0}\left(T^{\mathbf{L}}\right) = \sum_{j\in\mathbb{N}} j\cdot\mathbb{P}^{\mathbf{L}}_{\vec{x}_0}\left(T^{\mathbf{L}} = j\right) = 0\cdot\mathbb{P}^{\mathbf{L}}_{\vec{x}_0}\left(T^{\mathbf{L}} = 0\right) = 0.$$

□

B.36 Proof of Thm. 16.2.1

Proof. The characteristic function $\Phi_{\langle\mathbf{L}\rangle}$ is continuous (and thus monotonic) by Cor. 16.1.8. Hence, one can show that if $f \succeq \Phi_{\langle\mathbf{L}\rangle}(f)$ (i.e., f is a superinvariant), then $f \succeq \mathrm{lfp}\,\Phi_{\langle\mathbf{L}\rangle}$ (see Park induction as described in Thm. 15.1.1). By Cor. 16.1.8, this means that $f(\vec{x}_0) \geq \mathrm{lfp}\,\Phi_{\langle\mathbf{L}\rangle}(\vec{x}_0) = \mathrm{explt}\ [\![\mathbf{L}]\!]\left(\vec{x}_0\right)$ for all $\vec{x}_0 \in \mathbb{Z}^d$. Hence, to prove Thm. 16.2.1, it suffices to show $f \succeq \Phi_{\langle\mathbf{L}\rangle}(f)$ for the function $f : \mathbb{Z}^d \to \overline{\mathbb{R}_{\geq 0}}$ with $f(\vec{x}) = \frac{1}{p'}$ if $\vec{a}\bullet\vec{x} - b > 0$ and $f(\vec{x}) = 0$ if $\vec{a}\bullet\vec{x} - b \leq 0$. For $\vec{x}$ with $\vec{a}\bullet\vec{x} - b \leq 0$, we have

$\Phi_{\langle \mathbf{L} \rangle}(f)(\vec{x}) = f(\vec{x})$. If $\vec{a} \bullet \vec{x} - b > 0$, we get

$$\begin{aligned}
&\Phi_{\langle \mathbf{L} \rangle}(f)(\vec{x}) \\
&= \sum_{1 \le j \le n} p_{\vec{c}_j}(\vec{x}) \cdot f(\vec{x} + \vec{c}_j) + p'(\vec{x}) \cdot f(\vec{w}) + 1 \\
&\le \sum_{1 \le j \le n} p_{\vec{c}_j}(\vec{x}) \cdot \frac{1}{p'} + p'(\vec{x}) \cdot 0 + 1 \\
&= \frac{1}{p'} \cdot \sum_{1 \le j \le n} p_{\vec{c}_j}(\vec{x}) + 1 \\
&= \frac{1 - p'(\vec{x})}{p'} + 1 \\
&\le \frac{1 - p'}{p'} + 1 \\
&= \frac{1}{p'} \\
&= f(\vec{x})
\end{aligned}$$

□

B.37 Proof of Thm. 16.3.4

Proof. We prove Thm. 16.3.4 by proving each part separately.

(i) We have

$$a \bullet \vec{x} - b > 0 \iff rdw_{\mathbf{L}}\left(\vec{x}\right) > 0.$$

Since the condition of $\mathbf{L}$ is $a \bullet \vec{x} - b > 0$ and the condition of $\mathbf{L}^{rdw}$ is $x > 0$, the result follows.

(ii) If $a \bullet \vec{x}_0 - b \le 0$, i.e. $rdw_{\mathbf{L}}\left(\vec{x}_0\right) \le 0$, then

$$\mathbb{P}^{\mathbf{L}}_{\vec{x}_0}\left(T^{\mathbf{L}} = 0\right) = 1 = \mathbb{P}^{\mathbf{L}^{rdw}}_{rdw_{\mathbf{L}}(\vec{x}_0)}\left(T^{\mathbf{L}^{rdw}} = 0\right)$$

by (i) and thus, the result follows.

So let $a \bullet \vec{x}_0 - b > 0$ and thus $x_0 = rdw_{\mathbf{L}}\left(\vec{x}_0\right) > 0$. Let $\pi = y_0 \dots y_j \in \mathbb{Z}^{j+1}$ such that $y_0, \dots, y_{j-1} > 0$ and $y_j \le 0$. First note that we have

$$(rdw_{\mathbf{L}})^{-1}(\mathrm{Cyl}\left(y_0 \cdots y_j\right)) = \biguplus_{\substack{\vec{z}_0, \dots, \vec{z}_j \in \mathbb{Z}^d, \\ rdw_{\mathbf{L}}(\vec{z}_0) = y_0, \dots, rdw_{\mathbf{L}}(\vec{z}_j) = y_j}} \mathrm{Cyl}\left(\vec{z}_0 \cdots \vec{z}_j\right). \tag{B.37.1}$$

Here, $\mathrm{Cyl}\left(y_0 \cdots y_j\right) \subseteq \mathbb{Z}^\omega$ and $\mathrm{Cyl}\left(\vec{z}_0 \cdots \vec{z}_j\right) \subseteq (\mathbb{Z}^d)^\omega$.

Note that both sides of the equation (B.37.1) can be empty, i.e., there might not be any $\vec{z}_u$ with $rdw_{\mathbf{L}}\left(\vec{z}_u\right) = y_u$ for some $1 \leq u \leq j$. We prove that

$$\mathbb{P}_{x_0}^{\mathbf{L}^{rdw}}\left(\mathrm{Cyl}\left(y_0 \cdots y_j\right)\right) = \mathbb{P}_{\vec{x}_0}^{\mathbf{L}}\left(\biguplus_{\substack{\vec{z}_0,\ldots,\vec{z}_j \in \mathbb{Z}^d,\\ rdw_{\mathbf{L}}(\vec{z}_0)=y_0,\ldots,rdw_{\mathbf{L}}(\vec{z}_j)=y_j}} \mathrm{Cyl}\left(\vec{z}_0 \cdots \vec{z}_j\right)\right). \quad \text{(B.37.2)}$$

The left-hand side is 0 whenever $y_0 \neq x_0$ and otherwise we have

$$\begin{aligned}
&\mathbb{P}_{x_0}^{\mathbf{L}^{rdw}}\left(\mathrm{Cyl}\left(y_0 \cdots y_j\right)\right)\\
&= \prod_{1\leq u\leq j}\left(p_{y_u - y_{u-1}}^{rdw} + \left[y_u = rdw_{\mathbf{L}}\left(\vec{w}\right)\right]\cdot p'\right)\\
&= \prod_{1\leq u\leq j}\left(p_{y_u - y_{u-1}}^{rdw} + \left[y_u = w_{\mathbf{L}}\right]\cdot p'\right)\\
&= \prod_{1\leq u\leq j}\left(\sum_{\substack{-k_{\mathbf{L}}\leq t\leq m_{\mathbf{L}},\\ t = y_u - y_{u-1}}} p_t^{rdw} + \left[y_u = w_{\mathbf{L}}\right]\cdot p'\right).
\end{aligned}$$

For the right-hand side of (B.37.2) recall that $rdw_{\mathbf{L}}\left(\vec{x}_0\right) = x_0$ and that we only regard prefixes $\vec{z}_0$ where $rdw_{\mathbf{L}}\left(\vec{z}_0\right) = y_0$. So if $y_0 \neq x_0$, all of these tuples $\vec{z}_0$ are different from $\vec{x}_0$. Hence, then the right-hand of (B.37.2) side is also 0. Otherwise, we have

$$\begin{aligned}
&\mathbb{P}_{\vec{x}_0}^{\mathbf{L}}\left(\biguplus_{\substack{\vec{z}_0,\ldots,\vec{z}_j \in \mathbb{Z}^d,\\ rdw_{\mathbf{L}}(\vec{z}_0)=y_0,\ldots,rdw_{\mathbf{L}}(\vec{z}_j)=y_j}} \mathrm{Cyl}\left(\vec{z}_0 \cdots \vec{z}_j\right)\right)\\
&= \mathbb{P}_{\vec{x}_0}^{\mathbf{L}}\left(\biguplus_{\substack{\vec{z}_1,\ldots,\vec{z}_j \in \mathbb{Z}^d,\\ rdw_{\mathbf{L}}(\vec{z}_1)=y_1,\ldots,rdw_{\mathbf{L}}(\vec{z}_j)=y_j}} \mathrm{Cyl}\left(\vec{x}_0, \vec{z}_1, \ldots, \vec{z}_j\right)\right)\\
&= \sum_{\substack{\vec{z}_1,\ldots,\vec{z}_j \in \mathbb{Z}^d,\\ rdw_{\mathbf{L}}(\vec{z}_1)=y_1,\ldots,rdw_{\mathbf{L}}(\vec{z}_j)=y_j}} \mathbb{P}_{\vec{x}_0}^{\mathbf{L}}\left(\mathrm{Cyl}\left(\vec{x}_0, \vec{z}_1, \ldots, \vec{z}_j\right)\right)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{\substack{\vec{z}_1,\ldots,\vec{z}_j \in \mathbb{Z}^d,\\ rdw_{\mathbf{L}}(\vec{z}_1)=y_1,\ldots,rdw_{\mathbf{L}}(\vec{z}_j)=y_j}} \left(p_{\vec{z}_1-\vec{x}_0} + \left[\vec{z}_1 = \vec{w}\right]\cdot p'\right) \\
&\quad \cdot \prod_{2\le u\le j} \left(p_{\vec{z}_u-\vec{z}_{u-1}} + \left[\vec{z}_u = \vec{w}\right]\cdot p'\right) \\
&= \sum_{\substack{\vec{c}_{v_1},\ldots,\vec{c}_{v_j}\in\{\vec{c}_1,\ldots,\vec{c}_n\},\\ rdw_{\mathbf{L}}(\vec{x}_0+\vec{c}_{v_1})=y_1,\\ rdw_{\mathbf{L}}(\vec{x}_0+\vec{c}_{v_1}+\vec{c}_{v_2})=y_2,\\ \vdots\\ rdw_{\mathbf{L}}\left(\vec{x}_0+\vec{c}_{v_1}+\ldots+\vec{c}_{v_j}\right)=y_j}} \left(p_{\vec{c}_{v_1}} + [y_1 = w_{\mathbf{L}}]\cdot p'\right)\cdot\ldots\cdot\left(p_{\vec{c}_{v_j}} + \left[y_j = w_{\mathbf{L}}\right]\cdot p'\right) \\
&= \sum_{\substack{\vec{c}_{v_1},\ldots,\vec{c}_{v_j}\in\{\vec{c}_1,\ldots,\vec{c}_n\},\\ y_1-y_0=\vec{a}\bullet\vec{c}_{v_1},\\ \vdots\\ y_j-y_{j-1}=\vec{a}\bullet\vec{c}_{v_j}}} \left(p_{\vec{c}_{v_1}} + [y_1 = w_{\mathbf{L}}]\cdot p'\right)\cdot\ldots\cdot\left(p_{\vec{c}_{v_j}} + \left[y_j = w_{\mathbf{L}}\right]\cdot p'\right) \qquad (\dagger) \\
&= \left(\sum_{\substack{\vec{c}\in\{\vec{c}_1,\ldots,\vec{c}_n\},\\ y_1-y_0=\vec{a}\bullet\vec{c}}} p_{\vec{c}} + [y_1 = w_{\mathbf{L}}]\cdot p'\right)\cdot\ldots\cdot\left(\sum_{\substack{\vec{c}\in\{\vec{c}_1,\ldots,\vec{c}_n\},\\ y_j-y_{j-1}=\vec{a}\bullet\vec{c}}} p_{\vec{c}} + \left[y_j = w_{\mathbf{L}}\right]\cdot p'\right) \\
&= \prod_{1\le u\le j}\left(\sum_{\substack{1\le t\le n,\\ \vec{a}\bullet\vec{c}_t=y_u-y_{u-1}}} p_{\vec{c}_t} + [y_u = w_{\mathbf{L}}]\cdot p'\right) \\
&= \prod_{1\le u\le j}\left(\sum_{\substack{-k_{\mathbf{L}}\le t\le m_{\mathbf{L}},\\ t=y_u-y_{u-1}}} p_t^{rdw} + [y_u = w_{\mathbf{L}}]\cdot p'\right) \\
&= \mathbb{P}_{x_0}^{\mathbf{L}^{rdw}}\left(\mathrm{Cyl}\left(y_0\cdots y_j\right)\right).
\end{aligned}
$$

For Equation (†), note that $rdw_{\mathbf{L}}\left(\vec{x}_0 + \vec{c}_{v_1}\right) = \vec{a}\bullet(\vec{x}_0+\vec{c}_{v_1})-b = \vec{a}\bullet\vec{x}_0+\vec{a}\bullet\vec{c}_{v_1}-b = rdw_{\mathbf{L}}\left(\vec{x}_0\right) + \vec{a}\bullet\vec{c}_{v_1} = y_0 + \vec{a}\bullet\vec{c}_{v_1}$. Hence, $rdw_{\mathbf{L}}\left(\vec{x}_0 + \vec{c}_{v_1}\right) = y_1$ means that $y_1 - y_0 = \vec{a}\bullet\vec{c}_{v_1}$. Similarly, $rdw_{\mathbf{L}}(\vec{x}_0 + \vec{c}_{v_1} + \vec{c}_{v_2}) = y_0 + \vec{a}\bullet\vec{c}_{v_1} + \vec{a}\bullet\vec{c}_{v_2} = y_1 + \vec{a}\bullet\vec{c}_{v_2}$. So $rdw_{\mathbf{L}}\left(\vec{x}_0 + \vec{c}_{v_1} + \vec{c}_{v_2}\right) = y_2$ means that $y_2 - y_1 = \vec{a}\bullet\vec{c}_{v_2}$, etc.

For any $j \in \mathbb{N}$ and any $\vec{x}_0 \in \mathbb{Z}^d$ we then obtain the following.

$$\mathbb{P}_{\mathbf{L}}^{\vec{x}_0}\left(T^{\mathbf{L}} = j\right)$$

$$
\begin{aligned}
&= \mathbb{P}^{\vec{x}_0}_{\mathbf{L}}\left(T^{\mathbf{L}^{rdw}} \circ rdw_{\mathbf{L}} = j\right) && \text{(by (i))}\\
&= \mathbb{P}^{\vec{x}_0}_{\mathbf{L}}\left((rdw_{\mathbf{L}})^{-1}(T^{\mathbf{L}^{rdw}})^{-1}(\{j\})\right)\\
&= \mathbb{P}^{\vec{x}_0}_{\mathbf{L}}\left((rdw_{\mathbf{L}})^{-1}\left(\biguplus_{\substack{y_0,\ldots,y_{j-1}\in\mathbb{Z}_{>0},\\ y_j\in\mathbb{Z}_{\leq 0}}} \mathrm{Cyl}\left(y_0\cdots y_j\right)\right)\right)\\
&= \mathbb{P}^{\vec{x}_0}_{\mathbf{L}}\left(\biguplus_{\substack{y_0,\ldots,y_{j-1}\in\mathbb{Z}_{>0},\\ y_j\in\mathbb{Z}_{\leq 0}}} (rdw_{\mathbf{L}})^{-1}\left(\mathrm{Cyl}\left(y_0\cdots y_j\right)\right)\right)\\
&= \sum_{\substack{y_0,\ldots,y_{j-1}\in\mathbb{Z}_{>0},\\ y_j\in\mathbb{Z}_{\leq 0}}} \mathbb{P}^{\mathbf{L}}_{\vec{x}_0}\left((rdw_{\mathbf{L}})^{-1}\left(\mathrm{Cyl}\left(y_0\cdots y_j\right)\right)\right)\\
&= \sum_{\substack{y_0,\ldots,y_{j-1}\in\mathbb{Z}_{>0},\\ y_j\in\mathbb{Z}_{\leq 0}}} \mathbb{P}^{\mathbf{L}^{rdw}}_{rdw_{\mathbf{L}}(\vec{x}_0)}\left(\mathrm{Cyl}\left(y_0\cdots y_j\right)\right) && \text{(by (B.37.2))}\\
&= \mathbb{P}^{\mathbf{L}^{rdw}}_{rdw_{\mathbf{L}}(\vec{x}_0)}\left(\biguplus_{\substack{y_0,\ldots,y_{j-1}\in\mathbb{Z}_{>0},\\ y_j\in\mathbb{Z}_{\leq 0}}} \mathrm{Cyl}\left(y_0\cdots y_j\right)\right)\\
&= \mathbb{P}^{\mathbf{L}^{rdw}}_{rdw_{\mathbf{L}}(\vec{x}_0)}\left(T^{\mathbf{L}^{rdw}} = j\right).
\end{aligned}
$$

As the above equality holds for every $j \in \mathbb{N}$ it also holds for $j = \omega$.

□

B.38 Proof of Cor. 16.3.9

Proof. If $\mathbf{L}$ has direct termination (i.e., $p' \neq 0$), $\mathbf{L}$ and $\mathbf{L}^{rdw}$ are PAST by Thm. 16.2.1. Otherwise, by Thm. 16.3.4 we can reduce termination of $\mathbf{L}$ to termination of $\mathbf{L}^{rdw}$ on inputs which are in the image of $rdw_{\mathbf{L}}$. Thus, by Thm. 16.3.4, if $\mathbf{L}^{rdw}$ is (P)AST, then $\mathbf{L}$ is (P)AST.

For the other direction, note that the image of $rdw_{\mathbf{L}}$ is not necessarily $\mathbb{Z}$. However, the termination behavior of $\mathbf{L}^{rdw}$ is the same for all $x > 0$. Hence, to show that if $\mathbf{L}$

is (P)AST, then $\mathbf{L}^{rdw}$ is (P)AST, we prove that $rdw_{\mathbf{L}}$'s image also includes positive values. To see this, note that $\vec{a} \neq \vec{0}$ implies $\vec{a} \bullet \vec{a} > 0$. Hence, for any natural number $u > \frac{b}{\vec{a} \bullet \vec{a}}$ we obtain $rdw_{\mathbf{L}}\left(u \cdot \vec{a}\right) = u \cdot \vec{a} \bullet \vec{a} - b > \frac{b}{\vec{a} \bullet \vec{a}} \cdot \vec{a} \bullet \vec{a} - b = 0$.

□

B.39 Proof of Thm. 16.3.10

Proof. We now show that for *cpa*-loops $\mathbf{L}$ without direct termination, one can not only decide termination, but also directly obtain asymptotically exact bounds on their expected looping time. More precisely, we show that $\mathsf{explt}\ [\![\mathbf{L}]\!]\left(\vec{x}_0\right)$ is asymptotically linear whenever $\mathbf{L}$ is PAST (and we even provide actual upper and lower bounds). To prove this result, we have used *Wald's Lemma* from probability theory (see, e.g., [GS01]) in [GGH19b]. Here, we present a different proof based on induction.

By Thm. 16.3.4, it is enough to prove the result for $\mathbf{L}^{rdw}$, i.e.,

$$-\frac{1}{\mu_{\mathbf{L}^{rdw}}} \cdot x_0 \leq \mathsf{explt}\ \left[\!\left[\mathbf{L}^{rdw}\right]\!\right](x_0) \leq -\frac{1}{\mu_{\mathbf{L}^{rdw}}} \cdot x_0 + \frac{1-k_{\mathbf{L}}}{\mu_{\mathbf{L}^{rdw}}}$$

whenever $x_0 \in \mathbb{Z}_{>0}$. For this, assume that in the body of $\mathbf{L}^{rdw}$ the variable x is updated to $x + j$ with probability p_j for $-k_{\mathbf{L}} = -k \leq j \leq m = m_{\mathbf{L}}$.

Consider $f = [x > 0] \cdot \frac{x+k-1}{-\mu_{\mathbf{L}^{rdw}}}$. For $x \leq 0$ we have $\Phi_{\langle \mathbf{L}^{rdw} \rangle}(f)(x) = 0 = f(x)$. For $x > 0$ we have

$$\begin{aligned}
&\Phi_{\langle \mathbf{L}^{rdw} \rangle}(f)(x) \\
&= 1 + \sum_{j=-k}^{m} p_j \cdot f(x+j) \\
&= 1 + \sum_{j=-k}^{m} p_j \cdot [x+j > 0] \cdot \frac{x+j+k-1}{-\mu_{\mathbf{L}^{rdw}}} \\
&\leq 1 + \sum_{j=-k}^{m} p_j \cdot \frac{x+j+k-1}{-\mu_{\mathbf{L}^{rdw}}} && \text{(as } x+j+k-1 \geq 0\text{)} \\
&= 1 + \sum_{j=-k}^{m} p_j \cdot \frac{x+k-1}{-\mu_{\mathbf{L}^{rdw}}} + \sum_{j=-k}^{m} p_j \cdot \frac{j}{-\mu_{\mathbf{L}^{rdw}}} \\
&= 1 + \frac{x+k-1}{-\mu_{\mathbf{L}^{rdw}}} + \frac{\mu_{\mathbf{L}^{rdw}}}{-\mu_{\mathbf{L}^{rdw}}} \\
&= 1 + \frac{x+k-1}{-\mu_{\mathbf{L}^{rdw}}} - 1
\end{aligned}$$

$$= \frac{x+k-1}{-\mu_{\mathbf{L}^{rdw}}}$$
$$= f(x)$$

By Cor. 16.1.8 and Thm. 15.1.1, the claim follows.

Now consider the lower bound. Unfortunately, we have only considered the expected runtime via the ert-transformer in Thm. 15.6.1. However, by using the results of [Wan+19] one can derive a similar rule for the characteristic looping time function (the reason is that [Wan+19] allows for costs on the different program statements, so that we can give cost 1 to the body of $\mathbf{L}^{rdw}$ and cost 0 to evaluating the condition, while the ert-transformer has a "fixed" cost model which differs slightly from the one for looping times).

Take $g = [x > 0] \cdot \frac{x}{-\mu_{\mathbf{L}^{rdw}}}$. To prove that g is a lower bound on explt $[\![\mathbf{L}^{rdw}]\!]$, we have to show that

(i) g is a $\Phi_{\langle \mathbf{L}^{rdw} \rangle}$-subinvariant, and

(ii) $\sum_{j=-k}^{m} p_j \cdot |g(x) - g(x+j)| \leq c$ for some constant $c \geq 0$ (this is the equivalent condition to "c.d.b").

Since the body of $\mathbf{L}^{rdw}$ is loop-free, there is no finiteness condition as in Thm. 15.6.1. Moreover, as we do not allow for different "postruntimes" there is no harmonization condition needed.

Note that (ii) is clear since x changes at most by a constant in $[-k, m]$ in every iteration. Thus, g is expected to change at most by $\frac{\max(m,k)}{-\mu_{\mathbf{L}^{rdw}}}$ in every iteration of the loop (more precisely, g is even difference bounded, since $|g(x) - g(x+j)| \leq \frac{\max(m,k)}{-\mu_{\mathbf{L}^{rdw}}}$, see Ex. 15.3.14).

We are left to show that g is a $\Phi_{\langle \mathbf{L}^{rdw} \rangle}$-subinvariant. For $x \leq 0$ this is clear as $\Phi_{\langle \mathbf{L}^{rdw} \rangle}(g)(x) = 0 = g(x)$. For $x > 0$ we have

$$\begin{aligned}
&\Phi_{\langle \mathbf{L}^{rdw} \rangle}(g)(x) \\
&= 1 + \sum_{j=-k}^{m} p_j \cdot g(x+j) \\
&= 1 + \sum_{j=-k}^{m} p_j \cdot [x+j > 0] \cdot \frac{x+j}{-\mu_{\mathbf{L}^{rdw}}} \\
&\geq 1 + \sum_{j=-k}^{m} p_j \cdot \frac{x+j}{-\mu_{\mathbf{L}^{rdw}}} && \text{(we add only non-positive addends)}
\end{aligned}$$

$$
\begin{aligned}
&= 1 + \sum_{j=-k}^{m} p_j \cdot \frac{x}{-\mu_{\mathrm{L}} rdw} + \sum_{j=-k}^{m} p_j \cdot \frac{j}{-\mu_{\mathrm{L}} rdw} \\
&= 1 + \frac{x}{-\mu_{\mathrm{L}} rdw} + \frac{\mu_{\mathrm{L}} rdw}{-\mu_{\mathrm{L}} rdw} \\
&= 1 + \frac{x}{-\mu_{\mathrm{L}} rdw} - 1 \\
&= \frac{x}{-\mu_{\mathrm{L}} rdw} \\
&= g(x).
\end{aligned}
$$

□

B.40 Proof of Lemma 16.4.7

Proof. We use a consequence of Rouché's Theorem [Rou62] for polynomials: Given a univariate polynomial $\sum_{j=0}^{v} a_j \cdot x^j$, a number $t \in \mathbb{R}_{>0}$, and an index $u \in \mathbb{N}_{\leq v}$ such that

$$
|a_u| \cdot t^u > \sum_{j=0,\ j \neq u}^{v} |a_j| \cdot t^j, \tag{B.40.1}
$$

then the polynomial has exactly u (possibly complex) roots (counted with multiplicity) of absolute value less than t.

We now apply Rouché's Theorem to the characteristic polynomial and proceed by case analysis. First, we consider the case where $p' > 0$. Here, we choose $t = 1$ and $u = k$. Then (B.40.1) becomes

$$
|p_0 - 1| > \sum_{j=-k,\ j \neq 0}^{m} |p_j|.
$$

As $|p_0 - 1| = 1 - p_0$ and $|p_j| = p_j$ for all j, this is equivalent to

$$
1 > \sum_{j=-k}^{m} p_j = 1 - p'
$$

which is true since $p' > 0$. So by Rouché's Theorem, the characteristic polynomial χ_{L} has k roots λ with $|\lambda| < 1$.

However, we would like to conclude that there are no more than k roots λ with $|\lambda| \leq 1$. Thus, we still need to show that χ_L has no root λ with $|\lambda| = 1$. Clearly,

$$\begin{aligned} & 0 = \chi_L(\lambda) \\ \iff & 0 = \sum_{j=-k}^{m} p_j \cdot \lambda^{k+j} - \lambda^k \\ \iff & \lambda^k = \sum_{j=-k}^{m} p_j \cdot \lambda^{k+j} \end{aligned}$$

If $|\lambda| = 1$ holds, we have

$$1 = |1| = \left|\lambda^k\right| \leq \sum_{j=-k}^{m} |p_j| \cdot |\lambda|^j = \sum_{j=-k}^{m} p_j = 1 - p'$$

by using $|p_j| = p_j$. However, this implies $p' = 0$. Therefore, there cannot be a root of absolute value 1 whenever $p' > 0$.

Now we consider the case where $p' = 0$ and thus $\sum_{j=-k}^{m} p_j = 1$. Our goal is to show that for all small enough $\varepsilon > 0$, the inequality (B.40.1) holds if we set $t = 1 + \varepsilon$ and $u = k$. This gives a result on the number of roots with absolute value less than $1 + \varepsilon$. If $\varepsilon \longrightarrow 0$, this yields the number of roots of absolute value at most 1. Using $t = 1 + \varepsilon$ and $u = k$, (B.40.1) becomes

$$\begin{aligned} & |p_0 - 1| \cdot t^k > \sum_{j=-k,\, j\neq 0}^{m} |p_j| \cdot t^{j+k} \\ \iff & (1 - p_0) \cdot (1 + \varepsilon)^k > \sum_{j=-k,\, j\neq 0}^{m} p_j \cdot (1 + \varepsilon)^{j+k} \\ \iff & (1 - p_0) + k \cdot (1 - p_0) \cdot \varepsilon + \mathcal{O}(\varepsilon^2) > \left(\sum_{j=-k,\, j\neq 0}^{m} p_j \right) + \left(\sum_{j=-k,\, j\neq 0}^{m} p_j \cdot (j + k) \cdot \varepsilon \right) + \mathcal{O}(\varepsilon^2) \\ & \qquad\qquad (\text{as } (1 + \varepsilon)^j = 1 + j \cdot \varepsilon + \mathcal{O}(\varepsilon^2) \text{ for any } j \geq 0) \\ \iff & k \cdot (1 - p_0) \cdot \varepsilon + \mathcal{O}(\varepsilon^2) > \left(\sum_{j=-k,\, j\neq 0}^{m} p_j \cdot (j + k) \cdot \varepsilon \right) + \mathcal{O}(\varepsilon^2) \qquad (\text{as } \textstyle\sum_{j=-k}^{m} p_j = 1) \\ \iff & k \cdot (1 - p_0) + \mathcal{O}(\varepsilon) > \left(\sum_{j=-k,\, j\neq 0}^{m} p_j \cdot (j + k) \right) + \mathcal{O}(\varepsilon) \qquad (\text{as } \varepsilon > 0) \end{aligned}$$

$$\begin{aligned}
&\iff k \cdot (1 - p_0) > \left(\sum_{j=-k,\ j\neq 0}^{m} p_j \cdot (j+k) \right) + \mathcal{O}(\varepsilon) \\
&\iff k > \left(\sum_{j=-k}^{m} p_j \cdot (j+k) \right) + \mathcal{O}(\varepsilon) \\
&\iff k > \left(\sum_{j=-k}^{m} p_j \cdot j \right) + k + \mathcal{O}(\varepsilon) \\
&\iff 0 > \mu_{\mathbf{L}} + \mathcal{O}(\varepsilon)
\end{aligned}$$

As $\mu_{\mathbf{L}} < 0$ as $\mathbf{L}$ is PAST (see Cor. 16.3.6), this is true for all sufficiently small ε. Hence, there are exactly k roots of absolute value less than $1 + \varepsilon$, where ε is sufficiently small. Since this holds for all such ε, there are exactly k roots of absolute value ≤ 1. □

B.41 Proof of Lemma 16.4.9

Proof. To encode the requirement on the $a_{j,u}$, we modify (16.4.3) into a new constraint (B.41.1) which ensures $a_{j,u} = 0$ whenever $|\lambda_j| > 1$. More precisely, this new constraint (B.41.1) is a recurrence equation such that the characteristic polynomial χ_0 of its homogeneous part has all the roots of $\chi_{\mathbf{L}}$ except those whose absolute value is greater than 1, i.e., $\chi_0(\lambda) = \prod_{j=1,\ |\lambda_j|\leq 1}^{c} (\lambda - \lambda_j)^{\nu_j}$. Thus, we can define the coefficients $q_j \in \mathbb{C}$ by

$$\chi_0(\lambda) = \prod_{j=1,\ |\lambda_j|\leq 1}^{c} (\lambda - \lambda_j)^{\nu_j} = \lambda^k - \sum_{j=-k}^{-1} q_j \cdot \lambda^{k+j}.$$

Note that the degree of the polynomial χ_0 is indeed k because by Lemma 16.4.7 we have $\sum_{j=1,\ |\lambda_j|\leq 1}^{c} \nu_j = k$.

Moreover, the constant add-on of the new recurrence equation is constructed in such a way that the particular solutions C_{const} resp. $C_{lin} \cdot x$ of (16.4.3) as defined in Def. 16.4.2 are also solutions of the inhomogeneous recurrence equation. Thus, let $D_{const} = C_{const} \cdot \left(1 - \sum_{j=-k}^{-1} q_j\right)$ and $D_{lin} = -C_{lin} \cdot \sum_{j=-k}^{-1} j \cdot q_j$. Instead of (16.4.3), we now consider the constraint

$$f(x) = \sum_{j=-k}^{-1} q_j \cdot f(x+j) + D \quad \text{for all } x > 0, \tag{B.41.1}$$

where we choose $D = D_{const}$ if $p' > 0$ and $D = D_{lin}$ if $p' = 0$. We show the following two claims:

(i) There is exactly one function $f : \mathbb{Z} \longrightarrow \mathbb{C}$ which satisfies (16.4.2) and (B.41.1).

(ii) A function $f : \mathbb{Z} \longrightarrow \mathbb{C}$ satisfies (B.41.1) iff f has the form (16.4.6) where $a_{j,u} = 0$ whenever $|\lambda_j| > 1$ (and thus it satisfies (16.4.3)).

These two claims imply the statement of the lemma. To see this, note that by (i) there exists a function which satisfies (16.4.2) and (B.41.1) and by (ii) this function also satisfies (16.4.3) and it has $a_{j,u} = 0$ whenever $|\lambda_j| > 1$. This function is unique because if there were two different functions f_1 and f_2 that satisfy (16.4.2) and (16.4.3) and have $a_{j,u} = 0$ whenever $|\lambda_j| > 1$, by (ii) these two functions would also both satisfy (B.41.1). But this would be a contradiction to the uniqueness stated in (i).

We now prove the claims (i) and (ii). For (i), note that the recurrence equation (B.41.1) is formulated in such a way that $f(x)$ only depends on the values of f on the smaller values $x-1, \dots, x-k$ (i.e., it is a recurrence of order k). By the constraint (16.4.2), the initial value of f on negative values is uniquely determined (i.e., $f(0) = f(-1) = \dots = f(-k+1) = 0$). Hence, by induction on x one can easily prove that there is a unique function $f : \mathbb{Z} \longrightarrow \mathbb{C}$ that satisfies both (16.4.2) and (B.41.1).

For the claim (ii), we only have to show that C_{const} is a solution of the inhomogeneous recurrence equation (B.41.1) if $p' > 0$ and $C_{lin} \cdot x$ is a solution of (B.41.1) if $p' = 0$. Once this is shown, it is clear that all solutions of (B.41.1) result from adding the particular solution C_{const} resp. $C_{lin} \cdot x$ of the inhomogeneous equation to a solution of the homogeneous variant of (B.41.1) (where D is replaced by 0). Any solution of this homogeneous variant can be represented as a linear combination of the solutions $\lambda_j^x \cdot x^u$ where $|\lambda_j| \leq 1$ and $u \in \{0, \dots, v_j - 1\}$. These are linearly independent solutions of the homogeneous variant of (B.41.1) since χ_0 is the corresponding characteristic polynomial (see Thm. 16.4.5). Thus, the solutions of (B.41.1) are all functions of the form (16.4.6) with $a_{j,u} = 0$ whenever $|\lambda_j| > 1$, which proves (ii).

It remains to show that C_{const} resp. $C_{lin} \cdot x$ are particular solutions of the inhomogeneous recurrence equation (B.41.1). If $p' > 0$, the definition of D_{const} indeed implies $C_{const} = C_{const} \cdot \sum_{j=-k}^{-1} q_j + D_{const}$. If $p' = 0$, we have to show

$$C_{lin} \cdot x = C_{lin} \cdot \sum_{j=-k}^{-1} q_j \cdot (x+j) + D_{lin}. \tag{B.41.2}$$

Since 1 is a root of χ_L (i.e., one of the λ_j with $|\lambda_j| \leq 1$ is $\lambda_j = 1$), 1 is also a root of χ_0. So we have $0 = \chi_0(1) = 1 - \sum_{j=-k}^{-1} q_j$, which implies $\sum_{j=-k}^{-1} q_j = 1$. So (B.41.2) is equivalent to

$$C_{lin} \cdot x = C_{lin} \cdot x + C_{lin} \cdot \sum_{j=-k}^{-1} j \cdot q_j - C_{lin} \cdot \sum_{j=-k}^{-1} j \cdot q_j$$

which obviously holds. □

B.42 Proof of Thm. 16.4.10

Proof. By Cor. 16.1.8, the expected looping time explt $[\![\mathbf{L}]\!]$ is the least fixed point of the characteristic function $\Phi_{\langle \mathbf{L} \rangle}$, i.e., the smallest function $f(x) : \mathbb{Z} \to \overline{\mathbb{R}_{\geq 0}}$ which satisfies (16.4.1), or equivalently, the smallest function which satisfies (16.4.2) and (16.4.3).

As f satisfies (16.4.3), it is a function of the form (16.4.6) by Thm. 16.4.5, i.e., there exist coefficients $a_{j,u} \in \mathbb{C}$ such that for all $x > -k$ we have

$$f(x) = C_{\mathbf{L}}(x) + \sum_{j=1}^{c} \sum_{u=0}^{\nu_j - 1} a_{j,u} \cdot \lambda_j^x \cdot x^u.$$

If we had $a_{j,u} \neq 0$ for a coefficient where $|\lambda_j| > 1$, $f(x)$ would not be bounded by a constant (if $p' > 0$) resp. by a linear function (if $p' = 0$). Thus, this would contradict Thm. 16.2.1 (if $p' > 0$) resp. Thm. 16.3.10 (if $p' = 0$).

By Lemma 16.4.9, there is a unique function $f : \mathbb{Z} \to \mathbb{C}$ which satisfies both (16.4.2) and (16.4.3) and has $a_{j,u} = 0$ whenever $|\lambda_j| > 1$. So this function must be the expected looping time (and hence, it maps any integer to a non-negative real number). Due to (16.4.2) the function must satisfy $f(x) = 0$ for all $x \leq 0$. Therefore, it must satisfy the linear equations (16.4.8). Nonetheless, the linear equations (16.4.8) cannot have more than one solution because otherwise this would yield two different functions that satisfy both (16.4.2) and (16.4.3) and have $a_{j,u} = 0$ whenever $|\lambda_j| > 1$, in contradiction to Lemma 16.4.9.

If $k = 0$, then $p' > 0$ as $\mathbf{L}$ is PAST. Lemma 16.4.7 implies that $\chi_{\mathbf{L}}$ has no root with $|\lambda| \leq 1$ and thus, $\text{explt } [\![\mathbf{L}]\!](x) = C_{const} + \sum_{j=1,|\lambda_j|\leq 1}^{c} \ldots = C_{const}$ for $x > 0$. □

B.43 Proof of Cor. 16.4.12

Proof. If $\mathbf{L}$ is trivial, its expected looping time is obvious. Otherwise, by Cor. 16.3.9 one can decide if $\mathbf{L}$ is PAST and in that case, $\mathbf{L}^{rdw}$ is PAST as well. For any *cpa*-loop $\mathbf{L}$, we have $\text{explt } [\![\mathbf{L}]\!](\vec{x}) = \text{explt } [\![\mathbf{L}^{rdw}]\!](rdw_{\mathbf{L}}(\vec{x}))$ due to Thm. 16.3.4. As $\text{explt } [\![\mathbf{L}^{rdw}]\!](rdw_{\mathbf{L}}(\vec{x}))$ can be computed exactly by Thm. 16.4.10, this also holds for $\text{explt } [\![\mathbf{L}]\!](\vec{x})$. □

B.44 Proof of Thm. 17.2.10

Proof. Let $(\mathcal{RB}, \mathcal{SB})$ be a (non-probabilistic) bound pair. Furthermore, let $g \in \mathcal{GT}$ be some general transition, let $s_0 \in \Sigma$, and let $\mathfrak{s}$ be a scheduler. Finally, let $\vartheta = (\ell_0, t_0, s_0)(\ell_1, t_1, s_1) \cdots \in \mathsf{Conf}^\omega$ be an admissible run under the chosen scheduler $\mathfrak{s}$ and the chosen initial state s_0. In particular, this means that $t_0 = t_{in}$. We have

$$
\begin{aligned}
& |s_0|\,(\mathcal{RB}_{\mathbb{E}}(g)) \\
= \; & \sum_{t \in g} |s_0|\,(\mathcal{RB}(t)) \\
\geq \; & \sum_{t \in g} \sup\Big\{ \big|\{ i \mid t_i' = t \}\big| \;\Big|\; f' = (_, t_0', _) \cdots (_, t_n', _) \wedge \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f') > 0 \Big\} \\
\geq \; & |\{ i \in \mathbb{N} \mid t_i \in g \}| \qquad \text{(since every prefix of } \vartheta \text{ is admissible)} \\
= \; & \mathcal{R}(g)(\vartheta).
\end{aligned}
$$

Non-admissible runs do not influence the expected value $\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{R}(g))$ since the set of all non-admissible runs has probability zero: if a run is non-admissible, by definition it has a non-admissible prefix. There are only countably many such non-admissible prefixes f. By the construction of $\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}$ in Sect. 17.1, we have $\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathrm{Cyl}(f)) = \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) = 0$ and therefore, the countable union $\bigcup_{f \in \mathsf{Conf}^+,\, f \text{ is non-admissible}} \mathrm{Cyl}(f)$ has probability zero. But this set is precisely the set of non-admissible runs. Note that $|s_0|\,(\mathcal{RB}_{\mathbb{E}}(g)) \in \overline{\mathbb{R}}_{\geq 0}$ can be viewed as a constant function, i.e., its expected value is again $|s_0|\,(\mathcal{RB}_{\mathbb{E}}(g))$. As ϑ is an arbitrary admissible run, we get by monotonicity of the expected value operator

$$
|s_0|\,(\mathcal{RB}_{\mathbb{E}}(g)) = \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(|s_0|\,(\mathcal{RB}_{\mathbb{E}}(g))) \geq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{R}(g)).
$$

Now, let in addition $\alpha = (g, \ell, x) \in \mathcal{GRV}$ and let s_0, $\mathfrak{s}$, and ϑ be as above. We have

$$
\begin{aligned}
& |s_0|\,(\mathcal{SB}_{\mathbb{E}}(\alpha)) \\
= \; & \sum_{t = (_,_,_,_,\ell) \in g} |s_0|\,(\mathcal{SB}(t, x)) \\
\geq \; & \max\Big\{ |s_0|\,(\mathcal{SB}(t, x)) \mid t = (_, _, _, _, \ell) \in g \Big\} \\
\geq \; & \max\Big\{ \sup\Big\{ |s_i'(x)| \;\Big|\; f' = (_, _, _) \cdots (_, t_i', s_i') \wedge \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f') > 0 \Big\} \;\Big|\; t_i' = (_, _, _, _, \ell) \in g \Big\} \\
\geq \; & \sup\{ |s_i(x)| \mid i \in \mathbb{N} \wedge t_i = (_, _, _, _, \ell) \in g \} \quad \text{(since every prefix of } \vartheta \text{ is admissible)} \\
= \; & \mathcal{S}(\alpha)(\vartheta).
\end{aligned}
$$

As ϑ was an arbitrary admissible run, similar as above we have

$$
|s_0|\,(\mathcal{SB}_{\mathbb{E}}(\alpha)) = \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(|s_0|\,(\mathcal{SB}_{\mathbb{E}}(\alpha))) \geq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{S}(\alpha)).
$$

□

B.45 Proof of Thm. 17.3.8

At the end of this section, we will prove (a refinement of) Thm. 17.3.8. However, the proof of this theorem is very intricate and needs some preparation involving the theory of so-called ranking supermartingales (see Def. B.45.11).

Note that our definition of PLRFs differs from the standard definition of probabilistic ranking functions since we regard distinguished subsets $\mathcal{GT}_{>} \subseteq \mathcal{GT}_{\text{ni}} \subseteq \mathcal{GT}$. For that reason, we have to prove that a stochastic process defined according to our PLRFs is indeed a ranking supermartingale (this will be shown in Cor. B.45.15). Then we can use existing results on ranking supermartingales (Thm. B.45.16) to show that our PLRFs yield bounds on a stopping time defined according to $\mathcal{GT}_{>}$ (Def. B.45.10). However, in the parts of a run belonging to executions of general transitions from $\mathcal{GT}_{\text{ni}}$, we have to take into account that there can occur transitions from $\mathcal{GT}_{\text{ni}} \setminus \mathcal{GT}_{>}$ before the first general transition from $\mathcal{GT}_{>}$ is executed. Thus, we also have to show that a stochastic process which takes into account the part of the run before the first application of a transition from $\mathcal{GT}_{>}$ corresponds to a martingale (this is shown implicitly in Lemma B.45.20).

Let $\mathbf{T}$ be a PITS and $\mathfrak{r}$ be a PLRF for $\mathcal{GT}_{>} \subseteq \mathcal{GT}_{\text{ni}}$ as in Def. 17.3.1. We define functions $I_0, I_1, \ldots : \mathsf{Conf}^{\omega} \to \mathfrak{P}(\mathbb{N})$ that map a run to the set of indices that correspond to the evaluation steps upon entering $\mathcal{GT}_{\text{ni}}$ for the $(i+1)$-th time. These functions are used to define random variables $R_{i,j} : \mathsf{Conf}^{\omega} \to \mathbb{R}_{\geq 0}$ where $R_{i,j}$ maps a run to $\mathfrak{r}$ evaluated in the configuration before the $(j+1)$-th execution of $g \in \mathcal{GT}_{>}$ upon entering $\mathcal{GT}_{\text{ni}}$ for the $(i+1)$-th time. We start by defining $I_i(\vartheta)$, which is the set of indices of the run ϑ belonging to the $(i+1)$-th time in $\mathcal{GT}_{\text{ni}}$.

Definition B.45.1 (Index Sets for $\mathcal{GT}_{\text{ni}}$). $I_0(\vartheta)$ for a run $\vartheta = (\ell_0, t_0, s_0)(\ell_1, t_1, s_1)\cdots$ is the smallest set of natural numbers such that:

(i) $\inf\{n \in \mathbb{N} \mid t_n \in \bigcup \mathcal{GT}_{\text{ni}}\} \in I_0(\vartheta)$

(ii) $n \in I_0(\vartheta) \wedge t_{n+1} \in \bigcup \mathcal{GT}_{\text{ni}} \implies n+1 \in I_0(\vartheta)$

Moreover, $I_{i+1}(\vartheta)$ for $i \in \mathbb{N}$ is the smallest set of natural numbers such that:

(i) $\inf\left\{n \in \mathbb{N} \setminus \bigcup_{k=0}^{i} I_k(\vartheta) \;\middle|\; t_n \in \bigcup \mathcal{GT}_{\text{ni}}\right\} \in I_{i+1}(\vartheta)$

(ii) $n \in I_{i+1}(\vartheta) \wedge t_{n+1} \in \bigcup \mathcal{GT}_{\text{ni}} \implies n+1 \in I_{i+1}(\vartheta)$

Here, if the infimum in the respective first cases does not exist in $\mathbb{N}$ (i.e., it is ω) we define $I_0 = \emptyset$ (or $I_{i+1} = \emptyset$, respectively).

Furthermore, we define mappings $I^{>}_{i,j} : \mathsf{Conf}^\omega \to \mathbb{N}$ for $i, j \in \mathbb{N}$ such that for a fixed $\vartheta = (\ell_0, t_0, s_0)(\ell_1, t_1, s_1)\cdots \in \mathsf{Conf}^\omega$ the value $I^{>}_{i,j}(\vartheta)$ is the index where some transition from $\mathcal{GT}_{>}$ is evaluated for the $(j+1)$-th time upon entering $\mathcal{GT}_{\mathrm{ni}}$ for the $(i+1)$-th time, i.e., $I^{>}_{i,j}(\vartheta)$ is the $(j+1)$-th smallest element of $I_i(\vartheta) \cap \{k \in \mathbb{N} \mid t_k \in \bigcup \mathcal{GT}_{>}\}$.

Definition B.45.2 (Indices for $\mathcal{GT}_{>}$)**.** Let $i, j \in \mathbb{N}$. The mappings $I^{>}_{i,j} : \mathsf{Conf}^\omega \to \mathbb{N}$ are defined as follows:

$$I^{>}_{i,j}(\vartheta) = \begin{cases} \inf\left(\mathrm{cuts}^j\left(I_i(\vartheta) \cap \{k \in \mathbb{N} \mid t_k \in \bigcup \mathcal{GT}_{>}\}\right)\right), \\ \qquad\qquad \text{if } \mathrm{cuts}^j\left(I_i(\vartheta) \cap \{k \in \mathbb{N} \mid t_k \in \bigcup \mathcal{GT}_{>}\}\right) \neq \emptyset \\ 0, \qquad\qquad \text{otherwise} \end{cases}$$

This means that if the desired configuration does not exist, the index defaults to 0. Here, $\mathrm{cuts}(X) = X \setminus \{\inf X\}$ removes the set's smallest element (as $X \subseteq \mathbb{N}$ we have $\inf X \in X$ iff $X \neq \emptyset$), i.e., the j-th composition $\mathrm{cuts}^j(X)$ is the set X without the first j smallest element(s).

For $f \in \mathsf{Conf}^+$ and $j \in \mathbb{N}$ we define:

$$I^{>}_{i,j}(f) = \begin{cases} m, & \text{if for all } \vartheta \in \mathrm{Cyl}(f) \text{ we have } I^{>}_{i,j}(\vartheta) = m \\ \bot, & \text{otherwise.} \end{cases}$$

So, the mapping $I^{>}_{i,j}$ is defined on a finite path f whenever f already determines the value of $I^{>}_{i,j}$ uniquely for every run with prefix f.

Definition B.45.3 (Stochastic Processes $(R_{i,j})_{i,j\in\mathbb{N}}$)**.** For $i, j \in \mathbb{N}$ we define the map $R_{i,j} : \mathsf{Conf}^\omega \to \mathbb{R}_{\geq 0}$ on a run $\vartheta = (\ell_0, t_0, s_0)(\ell_1, t_1, s_1)\cdots \in \mathsf{Conf}^\omega$ by

$$R_{i,j}(\vartheta) = \begin{cases} \max\left\{ s_{I^{>}_{i,j}(\vartheta)-1}\left(\mathfrak{r}(\ell_{I^{>}_{i,j}(\vartheta)-1})\right), 0 \right\}, & \text{if } I^{>}_{i,j}(\vartheta) > 0 \\ 0, & \text{otherwise} \end{cases}$$

For $f \in \mathsf{Conf}^+$ and $i, j \in \mathbb{N}$ we define:

$$R_{i,j}(f) = \begin{cases} r, & \text{if for all } \vartheta \in \mathrm{Cyl}(f) \text{ we have } I^{>}_{i,j}(\vartheta) > 0 \wedge R_{i,j}(\vartheta) = r \\ \bot, & \text{otherwise} \end{cases}$$

Hence, $R_{i,j}(f) \in \mathbb{R}_{\geq 0}$ if f is long enough to determine the value of $R_{i,j}$ uniquely. Moreover, for any run ϑ, the value $R_{i,j}(\vartheta)$ is the value that $\mathfrak{r}$ takes *before* executing $\mathcal{GT}_>$ for the $(j+1)$-th time within the indices $I_i(\vartheta)$ or zero. We now prove this "already determined" behavior.

Lemma B.45.4 ($R_{i,j}$ is Already Determined by Long Enough Prefixes)**.** *Let $\vartheta \in \mathsf{Conf}^\omega$. If $R_{i,j}(\vartheta) = r > 0$ for some $i, j \in \mathbb{N}$, there is a prefix $f \in \mathsf{Conf}^+$ such that for all $0 \leq k \leq j$*

$$R_{i,k}(f) = R_{i,k}(\vartheta).$$

Proof. Let $\vartheta = c_0 \cdots$ be a run with $R_{i,j}(\vartheta) = r > 0$. Then $0 < I^{>}_{i,j}(\vartheta)$ by Def. B.45.3. We choose $f = c_0 \cdots c_{I^{>}_{i,j}(\vartheta)}$. Trivially, we then have $I^{>}_{i,k}(\vartheta) = I^{>}_{i,k}(f)$ for each $0 \leq k \leq j$. Therefore, by Def. B.45.3 we get $R_{i,k}(f) = R_{i,k}(\vartheta)$ for every $0 \leq k \leq j$. □

Lemma B.45.4 eases our reasoning. First of all, it is clear that $I^{>}_{i,j}$ is $\mathcal{F}^{\mathrm{pt}}$-measurable: If a run has a *positive* value under $I^{>}_{i,j}$, this value only depends on a prefix of the run. Moreover, since we know that the value of some run $\vartheta \in \mathsf{Conf}^\omega$ under $R_{i,j}$ is either zero or already determined by some prefix of ϑ, this directly implies that $R_{i,j}$ is $\mathcal{F}^{\mathrm{pt}}$-measurable.

In the next step, we construct a suitable filtration $(\mathcal{F}^{\mathrm{pt}}_{i,j})_{j \in \mathbb{N}}$ of $\mathcal{F}^{\mathrm{pt}}$ such that the sequence $(R_{i,j})_{j \in \mathbb{N}}$ becomes a stochastic process adapted to this filtration for every fixed $i \in \mathbb{N}$. Intuitively every $\mathcal{F}^{\mathrm{pt}}_{i,j}$ should contain all sets associated with a finite path f ending in a configuration with index $I^{>}_{i,j}$ such that $R_{i,j}$ evaluates to a value $r \in \mathbb{R}_{\geq 0}$. Formally, these sets are defined as follows:

$$S_{i,j,f}(r) = \mathrm{Cyl}(f) \cap R^{-1}_{i,j}(\{r\}) \cap \left(I^{>}_{i,j}\right)^{-1}(\{|f|\}), \tag{B.45.1}$$

where $|f|$ denotes the length of the finite path f. Here, we also allow $f = \varepsilon$, where $|\varepsilon| = 0$ and $\mathrm{Cyl}(\varepsilon) = \mathsf{Conf}^\omega$. However, this implies $S_{i,j,\varepsilon}(r) = \emptyset$ whenever $r > 0$. Still, we need it for technical reasons in Lemmas B.45.7 and B.45.8.

Again, for any $n \in \mathbb{N}$, we denote the n-th configuration in the run ϑ by $\vartheta[n]$. So, a run $\vartheta \in \mathsf{Conf}^\omega$ satisfies $\vartheta \in S_{i,j,f}(r)$ whenever

» the finite path f is a prefix of ϑ,

» $R_{i,j}(\vartheta) = r$, and

» $f = \vartheta[0]\vartheta[1]\cdots\vartheta[I^{>}_{i,j}(\vartheta) - 1]$.

In the case of $r = 0$, we get the runs ϑ with prefix f where $R_{i,j}$ evaluates to 0 due to the corresponding configuration. However, if for some run ϑ we have $I^{>}_{i,j}(\vartheta) = 0$, also $R_{i,j}(\vartheta) = 0$ but in general there is *no* prefix f' of ϑ with $R_{i,j}(f') = 0$, i.e., those runs are not covered by the just defined sets. As a result of this observation only sets $S_{i,j,f}(r)$ with $r > 0$ are regarded. We will consider the remaining runs afterwards. However, a run ϑ which has f as a prefix and satisfies $R_{i,j}(\vartheta) > 0$ is not necessarily contained in $S_{i,j,f}(R_{i,j}(\vartheta))$. The reason is that f does *not* necessarily determine the value of $I^{>}_{i,j}(\vartheta)$. To deal with this situation in the following reasoning, we define two subsets of configurations, which will increase readability of our reasoning later on.

$$\mathsf{Conf}_{\mathcal{GT}_>} = \{c = (_, t, _) \in \mathsf{Conf} \mid t \in \bigcup \mathcal{GT}_>\},$$
$$\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} = \{c = (_, t, _) \in \mathsf{Conf} \mid t \in \bigcup \mathcal{GT}_{\mathrm{ni}}\}.$$

We observe the following lemma.

Lemma B.45.5 (Observations on $S_{i,j,f}$)**.** *Let $f \in \mathsf{Conf}^+$ with last configuration $(\ell_f, _, s_f)$, $r > 0$, and $i, j \in \mathbb{N}$. If $S_{i,j,f}(r) \neq \emptyset$ then*

(i) $r = s_f\left(\mathfrak{r}(\ell_f)\right)$.

(ii)
$$S_{i,j,f}(r) = \biguplus_{c \in \mathsf{Conf}_{\mathcal{GT}_>}} \mathrm{Cyl}(f\,c).$$

Proof. We prove both statements separately.

(i) Let $\vartheta \in S_{i,j,f}(r)$. Then $I^{>}_{i,j}(\vartheta) = |f|$. By (B.45.1) we have $r = R_{i,j}(\vartheta)$ and this value is uniquely determined by the configuration $\vartheta[|f| - 1]$. But this is the last configuration of f, so, $r = s_f\left(\mathfrak{r}(\ell_f)\right)$.

(ii) Let $\vartheta \in S_{i,j,f}(r)$. By (B.45.1), $\vartheta[|f|] = (_, t, _)$ such that $t \in \bigcup \mathcal{GT}_>$, i.e., $\vartheta[|f|] \in \mathsf{Conf}_{\mathcal{GT}_>}$ since $I^{>}_{i,j}(\vartheta) = |f|$.

For the other direction, since $S_{i,j,f}(r) \neq \emptyset$ this means that for every run with prefix f we have $I^{>}_{i,j-1}(f) = I^{>}_{i,j-1}(\vartheta)$. However, f does not determine the value $I^{>}_{i,j}$, i.e., $I^{>}_{i,j}(f) = \bot$. Thus, if $\vartheta \in \mathrm{Cyl}(f\,c)$ for some $c \in \mathsf{Conf}_{\mathcal{GT}_>}$, then $I^{>}_{i,j}(\vartheta) = |f|$, i.e., $\vartheta \in S_{i,j,f}(r)$.

□

Nevertheless, as already mentioned, not all runs ϑ with $R_{i,j}(\vartheta) = 0$ can be merged into a single set: in this way we ignore the information that $R_{i,j'}(\vartheta) \neq 0$ is still possible for some $j' < j$. Therefore, we decompose these runs into more fine-grained sets $S^0_{i,j,j',f}(r)$. For $j' \in \mathbb{N}$ with $j' < j$ we define

$$S^0_{i,j,j',f}(r) = \begin{cases} \emptyset, \text{ if } j' > 0 \wedge r = 0 \\ \mathrm{Cyl}(f) \cap \left(\bigcap_{k=j'+1}^{j} R^{-1}_{i,k}(\{0\}) \right) \cap R^{-1}_{i,j'}(\{r\}) \cap \left(I^{>}_{i,j'} \right)^{-1} (\{|f| - 1\}), \text{ otherwise} \end{cases}$$

Intuitively, this set contains all runs on which $R_{i,j}$ evaluates to zero but either a previous value at time j' (given by $R_{i,j'}$) evaluates to $r > 0$ (if $j' > 0$, we must have $r > 0$), or all previous values evaluate to 0 (if $j' = 0$ and $r = 0$).

The above defined sets can now be used as generating sets for a sequence of σ-fields $(\mathcal{F}^{\mathrm{pt}}_{i,j})_{j \in \mathbb{N}}$ which we will prove to be a filtration in Lemma B.45.8.

Definition B.45.6 (Filtration for $R_{i,j}$)**.** For $i, j \in \mathbb{N}$, let $\mathcal{F}^{\mathrm{pt}}_{i,j}$ be the smallest σ-field containing

$$\begin{cases} \left\{ S_{i,j,f}(r) \mid f \in \mathsf{Conf}^+ \vee f = \varepsilon, r \geq 0 \right\}, & \text{if } j = 0 \\ \left\{ S_{i,j,f}(r) \mid f \in \mathsf{Conf}^+, r > 0 \right\} \cup \left\{ S^0_{i,j,j',f}(r) \mid f \in \mathsf{Conf}^+, j' < j, r \geq 0 \right\}, & \text{if } j > 0. \end{cases} \tag{B.45.2}$$

We denote this generating system of $\mathcal{F}^{\mathrm{pt}}_{i,j}$ by $\mathcal{E}^{\mathrm{pt}}_{i,j}$.

A σ-field which is generated by a countable partition of the sample space is easier to handle, e.g., when it comes to computing conditional expected values (see Lemmas A.6.1 and A.6.2 and Cor. A.6.3). Hence, we prove that our just defined sequence of σ-fields has this property w.r.t. the respective generating system $\mathcal{E}^{\mathrm{pt}}_{i,j}$.

Lemma B.45.7 ($\mathcal{F}^{\mathrm{pt}}_{i,j}$ is Generated by a Countable Partition of Conf^ω)**.** *For every $i, j \in \mathbb{N}$ the generating system $\mathcal{E}^{pt}_{i,j}$ of $\mathcal{F}^{pt}_{i,j}$ is a countable partition of Conf^ω. Here, we only take non-empty sets into account.*

Proof. Let $i, j \in \mathbb{N}$ be fixed. By definition, the sets in $\mathcal{E}^{\mathrm{pt}}_{i,j}$ are pairwise disjoint. Moreover, using Lemma B.45.4, and the countability of Conf^+ we already have that the functions $R_{i,j'}$ only take countably many values, for any $j' \in \mathbb{N}$. So, $\mathcal{E}^{\mathrm{pt}}_{i,j}$ contains only countably many non-empty sets.

Let $\vartheta \in \mathsf{Conf}^\omega$. Whenever $R_{i,j}(\vartheta) > 0$, Lemma B.45.4 ensures that $\vartheta \in S_{i,j,f}(R_{i,j}(\vartheta))$ for some prefix f of ϑ.

Now assume $R_{i,j}(\vartheta) = 0$. If $j = 0$, then $\vartheta \in S_{i,j,f}(0) = S_{i,0,f}(0)$ for $f = \vartheta[0]\cdots\vartheta[I_{i,0}(\vartheta) - 1]$ if $I_{i,0}(\vartheta) > 0$ and $\vartheta \in S_{i,j,\varepsilon}(0) = S_{i,0,\varepsilon}(0)$ if $I_{i,0}(\vartheta) = 0$. So, assume $j > 0$. Whenever $R_{i,j'}(\vartheta) > 0$ for some $0 \leq j' < j$, then again Lemma B.45.4 ensures that $\vartheta \in S^0_{i,j,j',f}(R_{i,j'}(\vartheta))$ for some prefix f of ϑ. So, the only remaining case is $R_{i,j'}(\vartheta) = 0$ for all $0 \leq j' < j$. Then, $\vartheta \in S^0_{i,j,0,f}(0)$ for $f = \vartheta[0]\cdots\vartheta[I_{i,0}(\vartheta)]$. So, $\mathcal{E}^{\mathrm{pt}}_{i,j}$ is indeed a partition of Conf^ω. □

Note that Lemma B.45.7 yields that all elements of $\mathcal{F}^{\mathrm{pt}}_{i,j}$ are just countable disjoint unions of the just described generators (see Lemma A.1.2).

We now prove that $(\mathcal{F}^{\mathrm{pt}}_{i,j})_{j\in\mathbb{N}}$ indeed forms a filtration of $\mathcal{F}^{\mathrm{pt}}$.

Lemma B.45.8 ($(\mathcal{F}^{\mathrm{pt}}_{i,j})_{j\in\mathbb{N}}$ is a Filtration of $\mathcal{F}^{\mathrm{pt}}$). *For all $i, j \in \mathbb{N}$ we have $\mathcal{F}^{pt}_{i,j} \subseteq \mathcal{F}^{pt}_{i,j+1}$ and $\mathcal{F}^{pt}_{i,j} \subseteq \mathcal{F}^{pt}$, i.e., $(\mathcal{F}^{pt}_{i,j})_{j\in\mathbb{N}}$ is a filtration of $\mathcal{F}^{pt}$.*

Proof. First of all, we have seen that for any $i, j \in \mathbb{N}$ both $R_{i,j}$ and $I^{>}_{i,j}$ are $\mathcal{F}^{\mathrm{pt}}$-measurable. So by definition we have $S_{i,j,f}(r) \in \mathcal{F}^{\mathrm{pt}}$ and $S^0_{i,j,j',f}(r) \in \mathcal{F}^{\mathrm{pt}}$, i.e., $\mathcal{F}^{\mathrm{pt}}_{i,j} \subseteq \mathcal{F}^{\mathrm{pt}}$.

Now we show that for any $f \in \mathsf{Conf}^+$ and $j \in \mathbb{N}$ such that $S_{i,j,f}(r) \neq \emptyset$ for some $r > 0$ we have $S_{i,j,f}(r) \in \mathcal{F}^{\mathrm{pt}}_{i,j+1}$. By Lemma B.45.5 (ii) we have

$$S_{i,j,f}(r) = \biguplus_{c\in\mathsf{Conf}_{\mathcal{GT}_>}} \mathrm{Cyl}(fc).$$

We partition $S_{i,j,f}(r)$ into $S_{i,j,f}(r) = A \uplus B$ such that $R_{i,j+1}(\vartheta_a) > 0$ for every $\vartheta_a \in A$ and $R_{i,j+1}(\vartheta_b) = 0$ for every $\vartheta_b \in B$. Then we have

$$A = \bigcup_{f'\in\mathsf{Conf}^*} \bigcup_{c\in\mathsf{Conf}_{\mathcal{GT}_>}} \bigcup_{r'>0} S_{i,j+1,fcf'}(r') \in \mathcal{F}^{\mathrm{pt}}_{i,j+1},$$

where the innermost union leads to only countably many non-empty sets and

$$B = \bigcup_{c\in\mathsf{Conf}_{\mathcal{GT}_>}} S^0_{i,j+1,j,fc}(r) \in \mathcal{F}^{\mathrm{pt}}_{i,j+1}.$$

So we conclude $S_{i,j,f}(r) = A \uplus B \in \mathcal{F}^{\mathrm{pt}}_{i,j+1}$.

Now we consider the sets $S^0_{i,j,j',f}(r)$. We fix $j, j' \in \mathbb{N}$, $f \in \mathsf{Conf}^+$, and $r \in \mathbb{R}_{\geq 0}$. Again, for every run ϑ we either have $R_{i,j+1}(\vartheta) > 0$ or $R_{i,j+1}(\vartheta) = 0$. Consider the partition $S^0_{i,j,j',f}(r) = D \uplus E$ where $R_{i,j+1}(\vartheta_d) > 0$ for every $\vartheta_d \in D$ and $R_{i,j+1}(\vartheta_e) = 0$ for

every $\vartheta_e \in E$. Since $R_{i,j+1}(\vartheta_d) > 0$ for runs $\vartheta_d \in D$, by Lemma B.45.4 there is a finite prefix of ϑ_d that is long enough to determine the value of $R_{i,k}(\vartheta_d)$ for any $k \leq j+1$. Moreover, by definition of $S^0_{i,j,j',f}(r)$ this prefix has f as a prefix itself and it has to ensure that $R_{i,k}$ evaluates to zero for $j'+1 \leq k \leq j$. Using all these requirements, we can write D as follows.

$$D = \bigcup_{\substack{f' \in \mathsf{Conf}^* \\ R_{i,k}(ff')=0, j'<k\leq j}} \bigcup_{r'>0} S_{i,j+1,ff'}(r'),$$

i.e., $D \in \mathcal{F}^{\mathrm{pt}}_{i,j+1}$. Nonetheless, we know that for all runs $\vartheta_e \in E$ we have $R_{i,j+1}(\vartheta_e) = 0$ and therefore

$$E = S^0_{i,j+1,j',f}(r) \in \mathcal{F}^{\mathrm{pt}}_{i,j+1}.$$

Finally $S^0_{i,j,j',f} = D \uplus E \in \mathcal{F}^{\mathrm{pt}}_{i,j+1}$.

There still remains the case of showing that $S_{i,0,f}(0) \in \mathcal{F}^{\mathrm{pt}}_{i,1}$ for $f \in \mathsf{Conf}^* = \mathsf{Conf}^+ \cup \{\varepsilon\}$ since this case is not yet covered. However, the reasoning required for this case is similar to the reasoning above: We partition $S_{i,0,f}(0)$ such that on the first part $R_{i,1}$ takes strictly positive values and on the other part $R_{i,1}$ is constant zero.

$$S_{i,0,f}(0) = \bigcup_{\substack{f' \in \mathsf{Conf}^* \\ R_{i,0}(ff')=0}} \bigcup_{r'>0} S_{i,1,ff'}(r') \cup \bigcup_{c \in \mathsf{Conf}} S^0_{i,1,0,fc}(0) \in \mathcal{F}^{\mathrm{pt}}_{i,1}.$$

Therefore, $\mathcal{F}^{\mathrm{pt}}_{i,j} \subseteq \mathcal{F}^{\mathrm{pt}}_{i,j+1} \subseteq \mathcal{F}^{\mathrm{pt}}$ which concludes the proof. □

For our filtration to be useful in combination with the stochastic process $(R_{i,j})_{j\in\mathbb{N}}$, the process has to be adapted to $(\mathcal{F}^{\mathrm{pt}}_{i,j})_{j\in\mathbb{N}}$.

Lemma B.45.9 ($(R_{i,j})_{j\in\mathbb{N}}$ is Adapted to $(\mathcal{F}^{\mathrm{pt}}_{i,j})_{j\in\mathbb{N}}$)**.** *For every $i, j \in \mathbb{N}$ the function $R_{i,j}$ is measurable w.r.t. the σ–field $\mathcal{F}^{pt}_{i,j}$.*

Proof. For the proof we fix $i, j \in \mathbb{N}$. We have already seen that $R_{i,j}$ takes only countably many values. The measurability is shown by proving $\left\{ \vartheta \in \mathsf{Conf}^\omega \mid R_{i,j}(\vartheta) = r \right\} \in \mathcal{F}^{\mathrm{pt}}_{i,j}$ for every $r \geq 0$. We begin with the case $r > 0$:

$$\left\{ \vartheta \in \mathsf{Conf}^\omega \mid R_{i,j}(\vartheta) = r \right\} = \bigcup_{f \in \mathsf{Conf}^+} S_{i,j,f}(r) \in \mathcal{F}^{\mathrm{pt}}_{i,j}$$

Now using the above statement we can also show the remaining case $R_{i,j}^{-1}(\{0\}) \in \mathcal{F}_{i,j}^{\mathrm{pt}}$.

$$\begin{aligned} R_{i,j}^{-1}(\{0\}) &= \left\{ \vartheta \in \mathsf{Conf}^{\omega} \;\middle|\; R_{i,j}(\vartheta) = 0 \right\} \\ &= \mathsf{Conf}^{\omega} \setminus \bigcup_{r' \in \mathbb{R}_{>0}} \underbrace{R_{i,j}^{-1}(\{r'\})}_{\in \mathcal{F}_{i,j}^{\mathrm{pt}}} \in \mathcal{F}_{i,j}^{\mathrm{pt}}. \end{aligned}$$

Hence every $R_{i,j}$ is measurable w.r.t. $\mathcal{F}_{i,j}^{\mathrm{pt}}$, i.e., the process $(R_{i,j})_{j\in\mathbb{N}}$ is adapted to $(\mathcal{F}_{i,j}^{\mathrm{pt}})_{j\in\mathbb{N}}$. □

The point of interest in the above defined stochastic process $(R_{i,j})_{j\in\mathbb{N}}$ is the first index where it takes the value zero. At this point, no more transition $t \in \bigcup \mathcal{GT}_{>}$ can be executed by the properties of $\mathfrak{r}$ (see Def. 17.3.1). To introduce this "point of interest" more formally, we use a stopping time (see Def. 14.4.17).

Definition B.45.10 (Stopping Time for $(R_{i,j})_{j\in\mathbb{N}}$)**.** For every $i \in \mathbb{N}$, let $T_i' : \mathsf{Conf}^{\omega} \longrightarrow \overline{\mathbb{N}}$ be defined by $T_i'(\vartheta) = \inf \left\{ j \in \mathbb{N} \;\middle|\; R_{i,j}(\vartheta) = 0 \right\}$.

Remember $\inf \emptyset = \omega$, i.e., $T_i'(\vartheta) = \omega$ if $R_{i,j}(\vartheta) > 0$ for all $j \in \mathbb{N}$. Due to the adaptedness of $(R_{i,j})_{j\in\mathbb{N}}$ to the filtration $(\mathcal{F}_{i,j}^{\mathrm{pt}})_{j\in\mathbb{N}}$, T_i' is indeed a stopping time for $(\mathcal{F}_{i,j}^{\mathrm{pt}})_{j\in\mathbb{N}}$.

The next lemma states that as long as our process has not terminated it is expected to decrease by at least 1. This allows us to prove that $(R_{i,j})_{j\in\mathbb{N}}$ forms a ranking supermartingale for T_i'. So let us first define this notion which refines the concept of supermartingales (see Def. 14.4.29) w.r.t. a fixed stopping time (the conditional expectation decreases as long as the process has not stopped w.r.t this stopping time).

Definition B.45.11 (Ranking Supermartingale [ACN18])**.** Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and T a stopping time for the filtration $(\mathcal{F}_j)_{j\in\mathbb{N}}$. A stochastic process $(X_j)_{j\in\mathbb{N}}$ is a (strict) *ranking supermartingale for* T iff for all $j \in \mathbb{N}$

(i) X_j is adapted to $\mathcal{F}_j$

(ii) We have

$$\mathbb{E}\Big(X_{j+1} \cdot \left[T^{-1}\left(\mathbb{N}_{>j}\right) \right] \;\Big|\; \mathcal{F}_j \Big) \leq (X_j - 1) \cdot \left[T^{-1}\left(\mathbb{N}_{>j}\right) \right]$$

Note that by Def. 14.4.11, a random variable takes only non-negative values, i.e., $X_j : \Omega \longrightarrow \overline{\mathbb{R}}_{\geq 0}$ for every $j \in \mathbb{N}$ in Def. B.45.11.

We now have to prove that the process $(R_{i,j})_{j\in\mathbb{N}}$ indeed satisfies the (second) ranking condition of Def. B.45.11. However, the proof of Lemma B.45.12 is very intricate. To increase readability we formulate it as a corollary and split its proof into several lemmas (Lemmas B.45.12 to B.45.14) which we prove separately. Combining these lemmas then yields the complete proof for the statement.

Lemma B.45.12 (Ranking Condition for $(R_{i,j})_{j\in\mathbb{N}}$)**.** *Let s_0 be an arbitrary initial state, $\mathfrak{s} \in \mathsf{Scheds}$ an arbitrary scheduler, and $i, j \in \mathbb{N}$. Then on the set $\left(T_i'\right)^{-1}\left(\mathbb{N}_{>j}\right)$ we have:*

$$\mathbb{E}^{pt}_{s_0,\mathfrak{s}}\left(R_{i,j+1} \,\middle|\, \mathcal{F}^{pt}_{i,j}\right) \leq R_{i,j} - 1$$

Proof. The inequation given above is equivalent to:

$$\left[T_i' > j\right] \cdot \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(R_{i,j+1} \,\middle|\, \mathcal{F}^{\mathrm{pt}}_{i,j}\right) \leq \left[T_i' > j\right] \cdot (R_{i,j} - 1),$$

where $T_i' > j$ stands for the set $\left(T_i'\right)^{-1}\left(\mathbb{N}_{>j}\right)$.

Since $\left(T_i'\right)^{-1}\left(\mathbb{N}_{>j}\right) \in \mathcal{F}^{\mathrm{pt}}_{i,j}$ (i.e., $\left[T_i' > j\right]$ is $\mathcal{F}^{\mathrm{pt}}_{i,j}$-measurable), by Thm. A.6.5 (iii), this inequation is itself equivalent to

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[T_i' > j\right] \cdot R_{i,j+1} \,\middle|\, \mathcal{F}^{\mathrm{pt}}_{i,j}\right) \leq \left[T_i' > j\right] \cdot (R_{i,j} - 1). \tag{B.45.3}$$

First of all, due to the properties Boundedness (a), Boundedness (b), and Decrease of $\mathfrak{r}$ in Def. 17.3.1, we have $R_{i,j}(\vartheta) \geq 1$ for any admissible run $\vartheta \in \left(T_i'\right)^{-1}\left(\mathbb{N}_{>j}\right)$.

By Cor. A.6.3, it suffices to only consider the generators $\mathcal{E}^{\mathrm{pt}}_{i,j}$ of $\mathcal{F}^{\mathrm{pt}}_{i,j}$ to prove this statement. Note that on the generators of the form $S^0_{i,j,j',f}(r)$, by Def. B.45.10 we have $T_i' \leq j$. So in this case, both sides of (B.45.3) become zero and the inequation holds trivially. Similarly, if $r = j = 0$, then $T_i' = 0 = j$ on the sets $S_{i,0,f}(0)$, i.e., both sides of (B.45.3) become zero, too. Therefore, we do not need to consider those generators here. So, we can assume $r > 0$ for the generators $S_{i,j,f}(r)$. Moreover, if a generator is in fact a null set, the left-hand side becomes zero and we can omit it as well. Furthermore, for any admissible run ϑ with $\vartheta \in S_{i,j,f}(r)$ we have by the property Non-Increase of $\mathfrak{r}$ that $\vartheta \in \left(T_i'\right)^{-1}\left(\mathbb{N}_{>j}\right)$. So by Cor. A.6.3, it is sufficient to prove

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[S_{i,j,f}(r)\right] \cdot \left[T_i' > j\right] \cdot R_{i,j+1}\right) \leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[S_{i,j,f}(r)\right] \cdot \left[T_i' > j\right] \cdot (R_{i,j} - 1)\right)$$

whenever $\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S_{i,j,f}(r)) > 0$. Moreover, as discussed above we have

$$\left[T_i' > j\right] \cdot \left[S_{i,j,f}(r)\right] = \left[S_{i,j,f}(r)\right] \qquad \text{almost-surely.}$$

This equation only holds for admissible runs and hence, it only holds almost-surely. So the above inequation is equivalent to

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[S_{i,j,f}(r)\right]\cdot R_{i,j+1}\right) \leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[S_{i,j,f}(r)\right]\cdot (R_{i,j}-1)\right) = \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S_{i,j,f}(r))\cdot(r-1)$$

since $R_{i,j}$ is constant with value r on $S_{i,j,f}(r)$. Note that this means, as $R_{i,j} \geq 1$ on the set $S_{i,j,f}(r) \subseteq \left(T'_i\right)^{-1}\left(\mathbb{N}_{>j}\right)$, that $\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S_{i,j,f}(r)) > 0$ already implies $r \geq 1$. That this inequation holds is exactly the statement of Lemma B.45.13. □

Lemma B.45.13. *Let s_0 be an arbitrary initial state, $\mathfrak{s} \in \mathsf{Scheds}$ an arbitrary scheduler, and $i, j \in \mathbb{N}$. Let $f \in \mathsf{Conf}^+$ and $r \geq 1$ such that $\mathbb{P}^{pt}_{s_0,\mathfrak{s}}\left(S_{i,j,f}(r)\right) > 0$. Then we have:*

$$\mathbb{E}^{pt}_{s_0,\mathfrak{s}}\left(\left[S_{i,j,f}(r)\right]\cdot R_{i,j+1}\right) \leq \mathbb{P}^{pt}_{s_0,\mathfrak{s}}\left(S_{i,j,f}(r)\right)\cdot(r-1)$$

Proof. Let $f \in \mathsf{Conf}^+$ and $r \geq 1$ such that $\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S_{i,j,f}(r)\right) > 0$. We have

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[S_{i,j,f}(r)\right]\cdot R_{i,j+1}\right)\\
&= \sum_{r'\in\mathbb{R}_{\geq 0}} r'\cdot \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[S_{i,j,f}(r)\right]\cdot R_{i,j+1} = r'\right) \qquad \text{(by Def. 14.4.11)}
\end{aligned}$$

Note that the set of non-negative real numbers $\mathbb{R}_{\geq 0}$ is uncountable but we have seen that all except countably many addends in the above sum are zero. This justifies why we can use the sum for the expected value and can avoid the integral.

We are interested in the value of $R_{i,j+1}$. However, in a run $\vartheta \in S_{i,j,f}(r)$ the position $I^{>}_{i,j+1}(\vartheta)$ might occur arbitrarily far after $I^{>}_{i,j}(\vartheta)$ (if it occurs at all), while in the meantime transitions from $\mathcal{GT}_{\mathrm{ni}} \setminus \mathcal{GT}_{>}$ are evaluated. To simplify the analysis, our goal is to ignore all runs where this next evaluation happens after more than $m \geq 1$ steps. To achieve this, we consider the sets A_m:

$$A_m = S_{i,j,f}(r) \cap (I^{>}_{i,j+1})^{-1}([I^{>}_{i,j}(f)+1, I^{>}_{i,j}(f)+m]).$$

So, A_m contains all runs $\vartheta \in S_{i,j,f}(r)$ such that the configuration with index $I^{>}_{i,j+1}(\vartheta)$ occurs within the next $m \geq 1$ evaluation steps after $I^{>}_{i,j}(\vartheta)$. By definition, we have $A_m \subseteq A_{m+1}$ for all $m \geq 1$. Therefore, $A_\omega = \lim_{m\to\omega} A_m = \bigcup_{m\geq 1} A_m = S_{i,j,f}(r) \cap (I^{>}_{i,j+1})^{-1}\left(\mathbb{N}_{\geq(I^{>}_{i,j}(f)+1)}\right)$. Since $I^{>}_{i,j+1}(\vartheta) \neq 0$ already implies $I^{>}_{i,j+1}(\vartheta) > I^{>}_{i,j}(\vartheta)$, we observe

$$\vartheta \in S_{i,j,f}(r) \setminus A_\omega \implies I^{>}_{i,j+1}(\vartheta) = 0.$$

But $I^{>}_{i,j+1}(\vartheta) = 0$ implies $R_{i,j+1}(\vartheta) = 0$, so we have

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[S_{i,j,f}(r)\right]\cdot R_{i,j+1}\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left([A_\omega] + \left[S_{i,j,f}(r)\setminus A_\omega\right]\right)\cdot R_{i,j+1}\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left([A_\omega]\cdot R_{i,j+1}\right) + \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\Big(\underbrace{\left[S_{i,j,f}(r)\setminus A_\omega\right]\cdot R_{i,j+1}}_{=0}\Big)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left([A_\omega]\cdot R_{i,j+1}\right).
\end{aligned}$$

By Cor. A.5.1 we can then conclude

$$\begin{aligned}
&\lim_{m\longrightarrow\omega}\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left([A_m]\cdot R_{i,j+1}\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left([A_\omega]\cdot R_{i,j+1}\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[S_{i,j,f}(r)\right]\cdot R_{i,j+1}\right).
\end{aligned}\tag{B.45.4}$$

By Lemma B.45.14 we have for all $m \geq 1$

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}([A_m]\cdot R_{i,j+1}) \leq \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S_{i,j,f}(r)\right)\cdot(r-1).$$

So using (B.45.4), we have therefore proven

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[S_{i,j,f}(r)\right]\cdot R_{i,j+1}\right) \leq \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S_{i,j,f}(r))\cdot(r-1)$$

which concludes the proof. □

Lemma B.45.14. *Let s_0 be an arbitrary initial state, $\mathfrak{s} \in \mathsf{Scheds}$ an arbitrary scheduler, and $i, j \in \mathbb{N}$. Let $f \in \mathsf{Conf}^+$, $m \in \mathbb{N}_{\geq 1}$, and $r \geq 1$ such that $\mathbb{P}^{pt}_{s_0,\mathfrak{s}}\left(S_{i,j,f}(r)\right) > 0$ and let $A_m = S_{i,j,f}(r) \cap (I^{>}_{i,j+1})^{-1}([I^{>}_{i,j}(f)+1, I^{>}_{i,j}(f)+m])$. Then we have:*

$$\mathbb{E}^{pt}_{s_0,\mathfrak{s}}([A_m]\cdot R_{i,j+1}) \leq \mathbb{P}^{pt}_{s_0,\mathfrak{s}}\left(S_{i,j,f}(r)\right)\cdot(r-1).\tag{B.45.5}$$

Proof. W.l.o.g. let $\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(A_m) > 0$ as otherwise the statement is clear.

When we reason about measurable sets, the easiest way is to write them as a disjoint union of cylinder sets (if possible). For A_m this is indeed possible: we know that the runs must have the prefix f, right after f there must be the position of $I^{>}_{i,j}$ and then within at most m steps we must reach the position of $I^{>}_{i,j+1}$. In other words

$$A_m = \biguplus_{1\leq m_1\leq m}\ \biguplus_{\substack{c_0,c_{m_1}\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m_1-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ f'=f\,c_0\,c_1\cdots c_{m_1}}} \mathrm{Cyl}\left(f'\right)$$

Using this decomposition we then get

$$
\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}([A_m]\cdot R_{i,j+1}) \\
&= \sum_{m_1=1}^{m} \sum_{\substack{c_0,c_{m_1}\in \mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\dots,c_{m_1-1}\in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus \mathsf{Conf}_{\mathcal{GT}_>},\\ f'=f\,c_0\,c_1\cdots c_{m_1}}} R_{i,j+1}(f')\cdot p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f') \qquad (\dagger)\\
&= \sum_{m_1=1}^{m-1} \sum_{\substack{c_0,c_{m_1}\in \mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\dots,c_{m_1-1}\in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus \mathsf{Conf}_{\mathcal{GT}_>},\\ f'=f\,c_0\,c_1\cdots c_{m_1}}} R_{i,j+1}(f')\cdot p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f') \\
&\quad + \sum_{\substack{c_0,c_m\in \mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\dots,c_{m-1}\in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus \mathsf{Conf}_{\mathcal{GT}_>},\\ f'=f\,c_0\,c_1\cdots c_{m}}} R_{i,j+1}(f')\cdot p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f')
\end{aligned}
$$

Let $c_k = (\ell'_k, t'_k, s'_k)$ for every $k \in \{0, \dots, m\}$.

Note that in the following sums, all addends including a term of the form $s'\left(\mathfrak{r}(\ell')\right)$ are non-negative. The reason is as follows: Let f' be an *admissible finite path* such that $f' = f\,c_0\,c_1\cdots c_{m_1}$ with $c_0, c_{m_1} \in \mathsf{Conf}_{\mathcal{GT}_>}$, $c_1, \dots, c_{m_1-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}$, and $c_k = (\ell'_k, t'_k, s'_k)$. Then, any admissible path ending in one of the $c_0, \dots, c_{m_1-1}$ is continued by the scheduler $\mathfrak{s}$ with some transition $t \in \bigcup \mathcal{GT}_{\mathrm{ni}}$. Moreover, from any of the $c_0, \dots, c_{m_1-1}$ we can reach the configuration c_{m_1} with a positive probability. By the properties Boundedness (b) and Decrease, we must have $s'_{m_1-1}\left(\mathfrak{r}(\ell'_{m_1-1})\right) \geq 0$ (since c_{m_1-1} is continued with a transition from $\bigcup \mathcal{GT}_>$ by the scheduler $\mathfrak{s}$). Then, by the properties Boundedness (a) and Non-Increase, we must have $s'_k\left(\mathfrak{r}(\ell'_k)\right) \geq 0$ for $0 \leq k \leq m_1 - 2$, which can be proven by induction. Consequently, whenever for some $0 \leq k \leq m_1 - 2$ we have $\mu^{\mathrm{pt}}_{c_k,\mathfrak{s}}(\ell', t', s') > 0$, then $t' \in \bigcup \mathcal{GT}_{\mathrm{ni}}$ due to the scheduler $\mathfrak{s}$ and hence, $s'\left(\mathfrak{r}(\ell')\right) \geq 0$, again by the property Boundedness (a).

Let us now consider the very last addend:

$$
\sum_{\substack{c_0,c_m\in \mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\dots,c_{m-1}\in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus \mathsf{Conf}_{\mathcal{GT}_>},\\ f'=f\,c_0\,c_1\cdots c_{m}}} R_{i,j+1}(f')\cdot p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f').
$$

We have

$$\begin{aligned}
&\sum_{\substack{c_0,c_m\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ f'=f\,c_0\,c_1\cdots c_m}} R_{i,j+1}(f')\cdot \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f')\\
&= \sum_{\substack{c_0,c_m\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ f'=f\,c_0\,c_1\cdots c_m}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right)\cdot \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f') \qquad \text{(by the definition of } R_{i,j+1})\\
&= \sum_{\substack{c_0,c_m\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>}}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right)\cdot \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\,c_0\,c_1\cdots c_{m-1})\cdot \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(c_m)
\end{aligned}$$

Now the idea is to propagate the value of $\mathfrak{r}$ back to the last configuration of f.

$$\begin{aligned}
&\sum_{\substack{c_0,c_m\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>}}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right)\cdot \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\,c_0\,c_1\cdots c_{m-1})\cdot \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(c_m)\\
&= \sum_{\substack{c_0,c_m\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>}}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right)\cdot \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\,c_0\,c_1\cdots c_{m-2})\cdot\\
&\qquad \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1})\cdot \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(c_m)\\
&= \sum_{\substack{c_0\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-2}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>}}} \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\,c_0\,c_1\cdots c_{m-2})\cdot\\
&\qquad \sum_{\substack{c_m\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_{m-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>}}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right)\cdot \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1})\cdot \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(c_m)\\
&= \sum_{\substack{c_0\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-2}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>}}} \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\,c_0\,c_1\cdots c_{m-2})\cdot\\
&\qquad \sum_{c_{m-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right)\cdot \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1})\cdot\left(\sum_{c_m\in\mathsf{Conf}_{\mathcal{GT}_>}} \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(c_m)\right)\\
&= \sum_{\substack{c_0\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-2}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>}}} \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\,c_0\,c_1\cdots c_{m-2})\cdot \sum_{\substack{c_{m-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(\mathsf{Conf}_{\mathcal{GT}_>})>0}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right)\cdot \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1})
\end{aligned}$$

The very last step holds since from a configuration we either end with probability 1 in $\mathsf{Conf}_{\mathcal{GT}_>}$ or we end there with probability 0 due to the definition of the scheduler. Here, we stored in the summation index-set the fact that we of course only take those configurations into account from which we can continue to a configuration in $\mathsf{Conf}_{\mathcal{GT}_>}$ with a positive probability (which by definition directly implies that we do so with probability 1 as explained before).

In the sum above, we have $t'_i \in \bigcup \mathcal{GT}_{\mathrm{ni}}$ for every addend. Using these observations combined with the linearity of the expected value operator as well as the property Non-Increase of PLRFs one obtains:

$$
\begin{aligned}
&\sum_{\substack{c_0 \in \mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0 c_1 \cdots c_{m-2}) \cdot \sum_{\substack{c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>},\\ \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(\mathsf{Conf}_{\mathcal{GT}_>})>0}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right) \cdot \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1}) \\
\leq &\sum_{\substack{c_0 \in \mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>},\\ \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(\diamond \mathsf{Conf}_{\mathcal{GT}_>})>0}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0 c_1 \cdots c_{m-2}) \cdot \sum_{c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right) \cdot \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1}) \\
\leq &\sum_{\substack{c_0 \in \mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>},\\ \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>})>0,\\ \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(\diamond \mathsf{Conf}_{\mathcal{GT}_>})>0}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0 c_1 \cdots c_{m-2}) \cdot s'_{m-2}\left(\mathfrak{r}(\ell'_{m-2})\right) \qquad \text{(by Non-Increase)}
\end{aligned}
$$

Again, we stored in the summation index-set the fact that we only take those configurations into account from which we can continue to a configuration in $\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}$ with a probability > 0 (i.e., we continue to $\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}$ with probability 1) and that we only consider those $c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}$ from which we can reach (after possibly several steps) $\mathsf{Conf}_{\mathcal{GT}_>}$ with a positive probability, which we denote by $\mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}\left(\diamond \mathsf{Conf}_{\mathcal{GT}_>}\right) > 0$ by a slight abuse of notation.

Now let us have a look at the addend for $m_1 = m-1$ of (†). Similar to the reasoning for the addend with $m_1 = m$ we have

$$
\begin{aligned}
&\sum_{\substack{c_0, c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>},\\ f' = f\, c_0 c_1 \cdots c_{m-1}}} R_{i,j+1}(f') \cdot p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f') \\
= &\sum_{\substack{c_0, c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_>},\\ c_1,\ldots,c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}}} s'_{m-2}\left(\mathfrak{r}(\ell'_{m-2})\right) \cdot p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0 c_1 \cdots c_{m-2}) \cdot \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1})
\end{aligned}
$$

$$\leq \sum_{\substack{c_0 \in \mathsf{Conf}_{\mathcal{GT}_>}, \\ c_1,\ldots,c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}, \\ \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(\mathsf{Conf}_{\mathcal{GT}_>})>0}} s'_{m-2}\left(\mathfrak{r}(\ell'_{m-2})\right) \cdot \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0\, c_1 \cdots c_{m-2})$$

So, when we combine these two steps we get the following for the sum in (†).

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}([A_m] \cdot R_{i,j+1}) \\
&= \sum_{m_1=1}^{m} \sum_{\substack{c_0, c_{m_1} \in \mathsf{Conf}_{\mathcal{GT}_>}, \\ c_1,\ldots,c_{m_1-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}, \\ f' = f\, c_0\, c_1 \cdots c_{m_1}}} R_{i,j+1}(f') \cdot \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f') \\
&\leq \sum_{m_1=1}^{m-2} \sum_{\substack{c_0, c_{m_1} \in \mathsf{Conf}_{\mathcal{GT}_>}, \\ c_1,\ldots,c_{m_1-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}, \\ f' = f\, c_0\, c_1 \cdots c_{m_1}}} R_{i,j+1}(f') \cdot \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f') \\
&\quad + \sum_{\substack{c_0 \in \mathsf{Conf}_{\mathcal{GT}_>}, \\ c_1,\ldots,c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}, \\ \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(\diamond\mathsf{Conf}_{\mathcal{GT}_>})>0}} s'_{m-2}\left(\mathfrak{r}(\ell'_{m-2})\right) \cdot \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0\, c_1 \cdots c_{m-2})
\end{aligned}$$

Proceeding in this way, i.e., iteratively propagating the point where to evaluate $\mathfrak{r}$ back by using the property Non-Increase of $\mathfrak{r}$, we gradually over-approximate this sum by

$$\sum_{c_0 \in \mathsf{Conf}_{\mathcal{GT}_>}} \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0) \cdot s'_0\left(\mathfrak{r}(\ell'_0)\right).$$

Finally, we can proceed with the crucial step in this part of the proof. To do so, let us assume that the last configuration in f is $c_f = (\ell_f, t_f, s_f)$.

$$\begin{aligned}
&\sum_{c_0 \in \mathsf{Conf}_{\mathcal{GT}_>}} \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0) \cdot s'_0\left(\mathfrak{r}(\ell'_0)\right) \\
&= \sum_{c_0 \in \mathsf{Conf}_{\mathcal{GT}_>}} \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) \cdot \mu^{\mathrm{pt}}_{c_f,\mathfrak{s}}(c_0) \cdot s'_0\left(\mathfrak{r}(\ell'_0)\right) \\
&= \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) \cdot \sum_{c_0 \in \mathsf{Conf}_{\mathcal{GT}_>}} \mu^{\mathrm{pt}}_{c_f,\mathfrak{s}}(c_0) \cdot s'_0\left(\mathfrak{r}(\ell'_0)\right)
\end{aligned}$$

$$\begin{aligned}
&\leq \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) \cdot (s_f\left(\mathfrak{r}(\ell_f)\right) - 1) && \text{(by Decrease)}\\
&= \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) \cdot (r-1) && \text{(by Lemma B.45.5)}\\
&= \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S_{i,j,f}(r)) \cdot (r-1).
\end{aligned}$$

Let us elaborate on the last equation. We assumed that $\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S_{i,j,f}(r)) > 0$. Moreover, we have seen in Lemma B.45.5 that

$$S_{i,j,f}(r) = \biguplus_{c_0 \in \mathsf{Conf}_{\mathcal{GT}_>}} \mathrm{Cyl}(f\,c)$$

By definition of the scheduler $\mathfrak{s}$, this means that it chooses to continue f with a general transition from $\mathcal{GT}_>$ (all other runs are not admissible). But this then means that

$$\begin{aligned}
&\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S_{i,j,f}(r))\\
&= \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\biguplus_{c_0 \in \mathsf{Conf}_{\mathcal{GT}_>}} \mathrm{Cyl}(f\,c)\right)\\
&= \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\biguplus_{c_0 \in \mathsf{Conf}} \mathrm{Cyl}(f\,c)\right)\\
&= \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Cyl}(f)\right)\\
&= \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f).
\end{aligned}$$

So, we have shown (B.45.5). □

Corollary B.45.15. *Let* $(\mathsf{Conf}^\omega, \mathcal{F}^{pt}, \mathbb{P}^{pt}_{s_0,\mathfrak{s}})$ *be the probability space of a PITS* $\mathbf{T}$ *for some initial state* $s_0 \in \Sigma$ *and scheduler* $\mathfrak{s} \in \mathsf{Scheds}$. *Let* $i \in \mathbb{N}$ *and* $(\mathcal{F}^{pt}_{i,j})_{j\in\mathbb{N}}$ *be the filtration from Def. B.45.6. Then,* $(R_{i,j})_{j\in\mathbb{N}}$ *is a ranking supermartingale for* T'_i.

Proof. First of all, we have seen in Lemma B.45.9 that $(R_{i,j})_{j\in\mathbb{N}}$ is adapted to $(\mathcal{F}^{\mathrm{pt}}_{i,j})_{j\in\mathbb{N}}$. Secondly, by definition, $R_{i,j}$ takes only non-negative real values. Thirdly, we have proven in Lemma B.45.12 that the second condition required in Def. B.45.11 is satisfied. So $(R_{i,j})_{j\in\mathbb{N}}$ is indeed a ranking supermartingale. □

Theorem B.45.16 (Ranking Supermartingale Bounds Stopping Time [ACN18, Thm. 7.2])**.** *Let* $(\Omega, \mathcal{F}, \mathbb{P})$ *be a probability space with a filtration* $(\mathcal{F}_j)_{j\in\mathbb{N}}$ *and* T *a stopping time. If* $(X_j)_{j\in\mathbb{N}}$ *is a ranking supermartingale for* T, *then*

$$\mathbb{E}(T) \leq \mathbb{E}(X_0).$$

Applying Thm. B.45.16 to the process $(R_{i,j})_{j\in\mathbb{N}}$ yields the following result.

Corollary B.45.17. *Let s_0 be an arbitrary initial state, $\mathfrak{s} \in \mathsf{Scheds}$ an arbitrary scheduler, and $i \in \mathbb{N}$. Then*

$$\mathbb{E}^{pt}_{s_0,\mathfrak{s}}(T'_i) \leq \mathbb{E}^{pt}_{s_0,\mathfrak{s}}(R_{i,0}).$$

However, so far we have only considered the case of $\mathcal{GT}_{>}$ transitions. While this is a crucial ingredient of the proof of Thm. 17.3.8, we also need to take the position into account when $\mathcal{GT}_{\mathrm{ni}}$ is entered for the first time. So we define a random variable similar to $R_{i,0}$, which we call E_i because it refers to the $(i+1)$-th entry into $\mathcal{GT}_{\mathrm{ni}}$. We first define the index I^e_i, i.e., the index of the $(i+1)$-th entry into $\mathcal{GT}_{\mathrm{ni}}$.

Definition B.45.18 (Index for Entry Into $\mathcal{GT}_{\mathrm{ni}}$)**.** For any $i \in \mathbb{N}$ we define the mapping $I^e_i : \mathsf{Conf}^\omega \to \mathbb{N}$ as follows:

$$I^e_i(\vartheta) = \begin{cases} \inf\left(I_i(\vartheta)\right), & \text{if } I_i(\vartheta) \neq \emptyset \\ 0, & \text{otherwise.} \end{cases}$$

Again, this means that if the desired configuration does not exist, the index defaults to 0. For $f \in \mathsf{Conf}^+$ we define:

$$I^e_i(f) = \begin{cases} m, & \text{if for all } \vartheta \in \mathrm{Cyl}(f) \text{ we have } I^e_i(\vartheta) = m \\ \bot, & \text{otherwise.} \end{cases}$$

In analogy to the process $R_{i,j}$ we now define the random variable E_i based on I^e_i.

Definition B.45.19 (Stochastic Process $(E_i)_{i\in\mathbb{N}}$)**.** For any $i \in \mathbb{N}$ we define the map $E_i : \mathsf{Conf}^\omega \to \mathbb{R}_{\geq 0}$ as follows. Let $\vartheta = (\ell_0, t_0, s_0)(\ell_1, t_1, s_1)\cdots \in \mathsf{Conf}^\omega$.

$$E_i(\vartheta) = \begin{cases} \max\left\{ s_{I^e_i(\vartheta)-1}\left(\mathfrak{r}(\ell_{I^e_i(\vartheta)-1})\right), 0 \right\}, & \text{if } I^e_i(\vartheta) > 0 \\ 0, & \text{otherwise} \end{cases}$$

For $f \in \mathsf{Conf}^+$ and $i \in \mathbb{N}$ we define:

$$E_i(f) = \begin{cases} r, & \text{if for all } \vartheta \in \mathrm{Cyl}(f) \text{ we have } I^e_i(\vartheta) > 0 \wedge E_i(\vartheta) = r \\ \bot, & \text{otherwise} \end{cases}$$

Clearly, E_i takes only non-negative values. We aim to prove Lemma B.45.20 for a yet to define σ-field $\mathcal{G}_i$. Here, the idea is the following: as already mentioned, so far we have only considered the first occurrence of a general transition from $\mathcal{GT}_>$ in a part of the run belonging to $\mathcal{GT}_{\mathrm{ni}}$. However, before the first such occurrence there can occur general transitions from $\mathcal{GT}_{\mathrm{ni}} \setminus \mathcal{GT}_>$ which is handled by the random variable E_i. Then Lemma B.45.20 implies that $\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(E_i)$ is *at least* $\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(R_{i,0}\right)$. This will be crucial for the proof of Thm. 17.3.8. Nevertheless, while proving Lemma B.45.20 is cumbersome, it is still easier than proving the statement $\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(R_{i,0}\right) \leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(E_i)$ for the expected values directly, as with the conditional expectation we only have to take certain parts of a run into account, whereas for the general expected value we would have to take the whole run into account.

Lemma B.45.20 (Relating $R_{i,0}$ and E_i)**.** *Let s_0 be an arbitrary initial state, $\mathfrak{s} \in \mathsf{Scheds}$ an arbitrary scheduler, and $i \in \mathbb{N}$. Then*

$$\mathbb{E}^{pt}_{s_0,\mathfrak{s}}\left(R_{i,0} \mid \mathcal{G}_i\right) \leq E_i.$$

The construction of $\mathcal{G}_i$ is similar to the construction of $\mathcal{F}^{\mathrm{pt}}_{i,0}$, except that we concentrate on E_i instead of $R_{i,0}$. So, in analogy to $S_{i,0,f}(r)$ we define the set

$$S^e_{i,f}(r) = \mathrm{Cyl}(f) \cap E_i^{-1}(\{r\}) \cap \left(I^e_i\right)^{-1}(\{|f|\}),$$

where $i \in \mathbb{N}$, $f \in \mathsf{Conf}^+$ or $f = \varepsilon$, and $r \geq 0$. Clearly, there are only countably many non-empty $S^e_{i,f}(r)$. By definition, these sets are also pairwise disjoint. Moreover, they form a partition of Conf^ω. We make the following observation similar to Lemma B.45.5.

Corollary B.45.21 (Observations on $S^e_{i,f}$)**.** *Let $f \in \mathsf{Conf}^+$ with last configuration $(\ell_f, _, s_f)$, $r > 0$, and $i, j \in \mathbb{N}$. If $S^e_{i,f}(r) \neq \emptyset$ then*

(i) $r = s_f\left(\mathfrak{r}(\ell_f)\right)$.

(ii)

$$S^e_{i,f}(r) = \biguplus_{c \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}} \mathrm{Cyl}(fc).$$

We then give the following definition.

Definition B.45.22 (σ-Field for Entries)**.** For $i \in \mathbb{N}$ we define

$$\mathcal{G}_i = \left\langle \left\{ S^e_{i,f}(r) \;\middle|\; f \in \mathsf{Conf}^+ \vee f = \varepsilon, r \geq 0 \right\} \right\rangle_\sigma .$$

By Cor. B.45.21 (ii), we directly obtain the following corollary.

Corollary B.45.23. *For every $i \in \mathbb{N}$ the σ-field $\mathcal{G}_i$ is a sub-σ-field of $\mathcal{F}^{pt}$.*

We can now prove Lemma B.45.20.

Proof of Lemma B.45.20. Let $f \in \mathsf{Conf}^* = \mathsf{Conf}^+ \cup \{\varepsilon\}$ and $r \geq 0$ such that $\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S^e_{i,f}(r)\right) > 0$. Due to Cor. A.6.3, it is enough to prove

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(R_{i,0} \cdot \left[S^e_{i,f}(r)\right]\right) \leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(E_i \cdot \left[S^e_{i,f}(r)\right]\right). \tag{B.45.6}$$

By definition of $R_{i,0}$, for any admissible run $\vartheta \in \mathsf{Conf}^\omega$ we have $R_{i,0}(\vartheta) = 0$ whenever $I^{>}_{i,0}(\vartheta) = 0$ since $0 \notin I_i(\vartheta)$. So to prove (B.45.6), it is enough to consider the runs in $S^e_{i,f}(r) \cap I^{-1}_{i,0}(\mathbb{N} \setminus \{0\})$ (where $I^{-1}_{i,0}$ abbreviates $(I^{>}_{i,0})^{-1}$), i.e., we have

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(R_{i,0} \cdot \left[S^e_{i,f}(r)\right]\right) = \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(R_{i,0} \cdot \left[S^e_{i,f}(r) \cap I^{-1}_{i,0}(\mathbb{N} \setminus \{0\})\right]\right).$$

By definition of $S^e_{i,f}(r)$ and the simple fact that $I^e_i \leq I^{>}_{i,0}$ we have $S^e_{i,f}(r) \cap I^{-1}_{i,0}(\mathbb{N} \setminus \{0\}) = S^e_{i,f}(r) \cap I^{-1}_{i,0}\left(\mathbb{N}_{\geq |f|}\right)$.

So we consider the sets $S^e_{i,f}(r) \cap I^{-1}_{i,0}([|f|, |f| + m])$ for $m \in \mathbb{N}$ which satisfy

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(R_{i,0} \cdot \left[S^e_{i,f}(r) \cap I^{-1}_{i,0}([|f|, |f| + m])\right]\right) \leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(E_i \cdot \left[S^e_{i,f}(r)\right]\right), \tag{B.45.7}$$

as we will prove in Lemma B.45.24. In contrast to Lemma B.45.14, here, we include the case $m = 0$ since $I^e_i(\vartheta) = I^{>}_{i,0}(\vartheta)$ is indeed possible.

Since $S^e_{i,f}(r) \cap I^{-1}_{i,0}([|f|,|f|+m]) \subseteq S^e_{i,f}(r) \cap I^{-1}_{i,0}([|f|,|f|+m+1])$, we then conclude

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(R_{i,0}\cdot\left[S^e_{i,f}(r)\right]\right)\\
&=\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(R_{i,0}\cdot\left[S^e_{i,f}(r)\cap I^{-1}_{i,0}(\mathbb{N}\setminus\{0\})\right]\right)\\
&=\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\lim_{m\to\omega}R_{i,0}\cdot\left[S^e_{i,f}(r)\cap I^{-1}_{i,0}([|f|,|f|+m])\right]\right)\\
&=\lim_{m\to\omega}\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(R_{i,0}\cdot\left[S^e_{i,f}(r)\cap I^{-1}_{i,0}([|f|,|f|+m])\right]\right) \qquad \text{(by Cor. A.5.1)}\\
&\leq\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(E_i\cdot\left[S^e_{i,f}(r)\right]\right).
\end{aligned}$$

This proves (B.45.6). □

Lemma B.45.24. *Let s_0 be an arbitrary initial state, $\mathfrak{s}\in\mathsf{Scheds}$ an arbitrary scheduler, and $i\in\mathbb{N}$. Let $f\in\mathsf{Conf}^*$ and $r\geq 0$ with $\mathbb{P}^{pt}_{s_0,\mathfrak{s}}\left(S^e_{i,f}(r)\right)>0$. For all $m\in\mathbb{N}$ we have*

$$\mathbb{E}^{pt}_{s_0,\mathfrak{s}}\left(R_{i,0}\cdot\left[S^e_{i,f}(r)\cap I^{-1}_{i,0}([|f|,|f|+m])\right]\right)\leq\mathbb{E}^{pt}_{s_0,\mathfrak{s}}\left(E_i\cdot\left[S^e_{i,f}(r)\right]\right).$$

Proof. This proof is very similar to the proof of Lemma B.45.14. However, it is not exactly the same, so we state it here for the sake of completeness. W.l.o.g. let $\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S^e_{i,f}(r)\cap I^{-1}_{i,0}([|f|,|f|+m])\right)>0$ as otherwise the statement is clear.

We have

$$S^e_{i,f}(r)\cap I^{-1}_{i,0}([|f|,|f|+m])=\biguplus_{0\leq m_1\leq m}\ \biguplus_{\substack{c_{m_1}\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_0,c_1,\ldots,c_{m_1-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ f'=fc_0c_1\cdots c_{m_1}}}\mathrm{Cyl}\left(f'\right)$$

Using this decomposition we then get

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[S^e_{i,f}(r)\cap I^{-1}_{i,0}([|f|,|f|+m])\right]\cdot R_{i,0}\right)\\
&=\sum_{m_1=0}^{m}\ \sum_{\substack{c_{m_1}\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_0,c_1,\ldots,c_{m_1-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ f'=fc_0c_1\cdots c_{m_1}}} R_{i,0}(f')\cdot\mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f') \qquad (\dagger)\\
&=\sum_{m_1=0}^{m-1}\ \sum_{\substack{c_{m_1}\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_0,c_1,\ldots,c_{m_1-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ f'=fc_0c_1\cdots c_{m_1}}} R_{i,0}(f')\cdot\mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f')
\end{aligned}$$

$$+ \sum_{\substack{c_m \in \mathsf{Conf}_{\mathcal{GT}_>}, \\ c_0, c_1, \ldots, c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}, \\ f' = f c_0 c_1 \cdots c_m}} R_{i,0}(f') \cdot \mathfrak{p}^{\mathrm{pt}}_{s_0, \mathfrak{s}}(f')$$

Let $c_k = (\ell'_k, t'_k, s'_k)$ for every $k \in \{0, \ldots, m\}$.

Again, in the following sums, all addends including a term of the form $s'\left(\mathfrak{r}(\ell')\right)$ are non-negative. The reason is as follows: Let f' be an *admissible finite path* such that $f' = f c_0 c_1 \cdots c_{m_1}$ with $c_{m_1} \in \mathsf{Conf}_{\mathcal{GT}_>}$, $c_0, \ldots, c_{m_1-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}$, and $c_k = (\ell'_k, t'_k, s'_k)$. Then, any admissible path ending in one of the $c_0, \ldots, c_{m_1-1}$ is continued by the scheduler $\mathfrak{s}$ with some transition $t \in \bigcup \mathcal{GT}_{\mathrm{ni}}$. Moreover, from any of the $c_0, \ldots, c_{m_1-1}$ we can reach the configuration c_{m_1} with a positive probability. By the properties Boundedness (b) and Decrease, we must have $s'_{m_1-1}\left(\mathfrak{r}(\ell'_{m_1-1})\right) \geq 0$ (since c_{m_1-1} is continued with a transition from $\bigcup \mathcal{GT}_>$ by the scheduler $\mathfrak{s}$). Then, by the properties Boundedness (a) and Non-Increase, we must have $s'_k\left(\mathfrak{r}(\ell'_k)\right) \geq 0$ for all $0 \leq k \leq m_1 - 2$, which can be proven by induction. Consequently, whenever for some $0 \leq k \leq m_1 - 2$ we have $\mu^{\mathrm{pt}}_{c_k, \mathfrak{s}}(\ell', t', s') > 0$, then $t' \in \bigcup \mathcal{GT}_{\mathrm{ni}}$ due to the scheduler $\mathfrak{s}$ and hence, $s'\left(\mathfrak{r}(\ell')\right) \geq 0$, again by the property Boundedness (a).

Let us now consider the very last addend:

$$\sum_{\substack{c_m \in \mathsf{Conf}_{\mathcal{GT}_>}, \\ c_0, c_1, \ldots, c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}, \\ f' = f c_0 c_1 \cdots c_m}} R_{i,0}(f') \cdot \mathfrak{p}^{\mathrm{pt}}_{s_0, \mathfrak{s}}(f').$$

Then we have

$$\begin{aligned}
&\sum_{\substack{c_m \in \mathsf{Conf}_{\mathcal{GT}_>}, \\ c_0, c_1, \ldots, c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}, \\ f' = f c_0 c_1 \cdots c_m}} R_{i,0}(f') \cdot \mathfrak{p}^{\mathrm{pt}}_{s_0, \mathfrak{s}}(f') \\
= &\sum_{\substack{c_m \in \mathsf{Conf}_{\mathcal{GT}_>}, \\ c_0, c_1, \ldots, c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}, \\ f' = f c_0 c_1 \cdots c_m}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right) \cdot \mathfrak{p}^{\mathrm{pt}}_{s_0, \mathfrak{s}}(f') \qquad \text{(by the definition of } R_{i,0}) \\
= &\sum_{\substack{c_m \in \mathsf{Conf}_{\mathcal{GT}_>}, \\ c_0, c_1, \ldots, c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right) \cdot \mathfrak{p}^{\mathrm{pt}}_{s_0, \mathfrak{s}}(f c_0 c_1 \cdots c_{m-1}) \cdot \mu^{\mathrm{pt}}_{c_{m-1}, \mathfrak{s}}(c_m)
\end{aligned}$$

Now the idea is to propagate the value of $\mathfrak{r}$ back to the last configuration of f.

$$\begin{aligned}
&\sum_{\substack{c_m \in \mathsf{Conf}_{\mathcal{GT}_>},\\ c_0,c_1,\ldots,c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right) \cdot p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0 c_1 \cdots c_{m-1}) \cdot \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(c_m) \\
= &\sum_{\substack{c_m \in \mathsf{Conf}_{\mathcal{GT}_>},\\ c_0,c_1,\ldots,c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right) \cdot p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0 c_1 \cdots c_{m-2}) \cdot \\
&\qquad \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1}) \cdot \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(c_m) \\
= &\sum_{c_0,c_1,\ldots,c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0 c_1 \cdots c_{m-2}) \cdot \\
&\qquad \sum_{\substack{c_m \in \mathsf{Conf}_{\mathcal{GT}_>},\\ c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right) \cdot \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1}) \cdot \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(c_m) \\
= &\sum_{c_0,c_1,\ldots,c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0 c_1 \cdots c_{m-2}) \cdot \\
&\qquad \sum_{c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right) \cdot \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1}) \cdot \left(\sum_{c_m \in \mathsf{Conf}_{\mathcal{GT}_>}} \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(c_m) \right) \\
= &\sum_{c_0,c_1,\ldots,c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0 c_1 \cdots c_{m-2}) \cdot \sum_{\substack{c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>},\\ \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(\mathsf{Conf}_{\mathcal{GT}_>})>0}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right) \cdot \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1})
\end{aligned}$$

Again, the very last step holds since from a configuration we either end with probability 1 in $\mathsf{Conf}_{\mathcal{GT}_>}$ or we end there with probability 0 due to the definition of the scheduler. Here, we stored in the summation index-set the fact that we of course only take those configurations into account from which we can continue to a configuration in $\mathsf{Conf}_{\mathcal{GT}_>}$ with a positive probability (which by definition directly implies that we do so with probability 1 as explained before). We then obtain:

$$\begin{aligned}
&\sum_{c_0,c_1,\ldots,c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0 c_1 \cdots c_{m-2}) \cdot \sum_{\substack{c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>},\\ \mu^{\mathrm{pt}}_{c_{m-1},\mathfrak{s}}(\mathsf{Conf}_{\mathcal{GT}_>})>0}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right) \cdot \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1}) \\
\leq &\sum_{c_0,c_1,\ldots,c_{m-2} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0 c_1 \cdots c_{m-2}) \cdot \sum_{\substack{c_{m-1} \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}} \setminus \mathsf{Conf}_{\mathcal{GT}_>},\\ \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(\diamond \mathsf{Conf}_{\mathcal{GT}_>})>0}} s'_{m-1}\left(\mathfrak{r}(\ell'_{m-1})\right) \cdot \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1})
\end{aligned}$$

$$\leq \sum_{\substack{c_0,c_1,\ldots,c_{m-2}\in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>})>0,\\ \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(\diamond\mathsf{Conf}_{\mathcal{GT}_>})>0}} \mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\,c_0\,c_1\cdots c_{m-2})\cdot s'_{m-2}\left(\mathfrak{r}(\ell'_{m-2})\right) \qquad \text{(by Non-Increase of } \mathfrak{r})$$

Again, we stored in the summation index-set the fact that we only take those configurations into account from which we can continue in one step to a configuration in $\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>}$ with a positive probability and from which we can reach $\mathsf{Conf}_{\mathcal{GT}_>}$ with a positive probability.

Now let us have a look at the addend for $m_1 = m-1$ of (†). Similar to the reasoning for the addend with $m_1 = m$ we have

$$\begin{aligned}
&\sum_{\substack{c_{m-1}\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_0,c_1,\ldots,c_{m-2}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ f'=f\,c_0\,c_1\cdots c_{m-1}}} R_{i,0}(f')\cdot\mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f')\\
&= \sum_{\substack{c_{m-1}\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_0,c_1,\ldots,c_{m-2}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>}}} s'_{m-2}\left(\mathfrak{r}(\ell'_{m-2})\right)\cdot\mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\,c_0\,c_1\cdots c_{m-2})\cdot\mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(c_{m-1})\\
&\leq \sum_{\substack{c_0,c_1,\ldots,c_{m-2}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(\mathsf{Conf}_{\mathcal{GT}_>})>0}} s'_{m-2}\left(\mathfrak{r}(\ell'_{m-2})\right)\cdot\mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\,c_0\,c_1\cdots c_{m-2})
\end{aligned}$$

So, when we combine these two steps we get the following for the sum in (†).

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[S^e_{i,f}(r)\cap I^{-1}_{i,0}\left([|f|,|f|+m]\right)\right]\cdot R_{i,0}\right)\\
&= \sum_{m_1=1}^{m}\sum_{\substack{c_{m_1}\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_0,c_1,\ldots,c_{m_1-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ f'=f\,c_0\,c_1\cdots c_{m_1}}} R_{i,0}(f')\cdot\mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f')\\
&\leq \sum_{m_1=1}^{m-2}\sum_{\substack{c_{m_1}\in\mathsf{Conf}_{\mathcal{GT}_>},\\ c_0,c_1,\ldots,c_{m_1-1}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ f'=f\,c_0\,c_1\cdots c_{m_1}}} R_{i,0}(f')\cdot\mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f')\\
&\qquad + \sum_{\substack{c_0,c_1,\ldots,c_{m-2}\in\mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}\setminus\mathsf{Conf}_{\mathcal{GT}_>},\\ \mu^{\mathrm{pt}}_{c_{m-2},\mathfrak{s}}(\diamond\mathsf{Conf}_{\mathcal{GT}_>})>0}} s'_{m-2}\left(\mathfrak{r}(\ell'_{m-2})\right)\cdot\mathfrak{p}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\,c_0\,c_1\cdots c_{m-2})
\end{aligned}$$

Proceeding in this way, i.e., iteratively propagating the point where to evaluate $\mathfrak{r}$ back by using the property Non-Increase of $\mathfrak{r}$, we gradually over-approximate this sum by

$$\sum_{c_0 \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0) \cdot s_0'\left(\mathfrak{r}(\ell_0')\right).$$

Here, each addend in the sum is non-negative since $\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S^e_{i,f}(r) \cap I^{-1}_{i,0}\left([|f|, |f| + m]\right)\right) > 0$, i.e., we can reach $\mathsf{Conf}_{\mathcal{GT}_>}$ from f with a positive probability. Finally, we can proceed with the crucial step in this part of the proof. To do so, let us assume that the last configuration in f is $c_f = (\ell_f, t_f, s_f)$.

$$\begin{aligned}
&\sum_{c_0 \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f\, c_0) \cdot s_0'\left(\mathfrak{r}(\ell_0')\right) \\
&= \sum_{c_0 \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) \cdot \mu^{\mathrm{pt}}_{c_f,\mathfrak{s}}(c_0) \cdot s_0'\left(\mathfrak{r}(\ell_0')\right) \\
&= p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) \cdot \sum_{c_0 \in \mathsf{Conf}_{\mathcal{GT}_{\mathrm{ni}}}} \mu^{\mathrm{pt}}_{c_f,\mathfrak{s}}(c_0) \cdot s_0'\left(\mathfrak{r}(\ell_0')\right) \\
&\leq p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) \cdot \left(s_f\left(\mathfrak{r}(\ell_f)\right)\right) && \text{(by property Non-Increase of } \mathfrak{r}) \\
&= p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) \cdot r && \text{(by Cor. B.45.21)} \\
&= \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S^e_{i,f}(r)) \cdot r && \text{(see proof of Lemma B.45.14)} \\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(E_i \cdot \left[S^e_{i,f}(r)\right]\right).
\end{aligned}$$

This proves the statement of the lemma. □

Finally, we have all information at hand to prove Thm. 17.3.8. This proof combines the results we have obtained on the connection between $R_{i,0}$ and T_i' (Cor. B.45.17) and the just proved connection between $R_{i,0}$ and E_i (Lemma B.45.20).

In fact, we even prove the soundness of a refinement of Thm. 17.3.8 which allows us to use *expected* instead of *non-probabilistic* runtime bounds on the entry transitions of $\mathcal{GT}_{\mathrm{ni}}$ in special cases. If $\mathfrak{r}(\ell)$ is *constant* then its value is *independent* of the values of the program variables at the location ℓ. In this special case, for every $h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)$ instead of the non-probabilistic runtime bounds $\mathcal{RB}(t)$ for $t \in h$, we can use the *expected* runtime bound $\mathcal{RB}_{\mathbb{E}}(h)$. To this end we consider a partition $\{\mathcal{EL}_{\mathrm{c},\mathcal{GT}_{\mathrm{ni}},\mathfrak{r}}, \mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}},\mathfrak{r}}\}$ of the entry locations $\mathcal{EL}_{\mathcal{GT}_{\mathrm{ni}}}$.

Definition B.45.25 (Constant & Non-Constant Entry Locations)**.** Let $\mathfrak{r}$ be a PLRF and $\mathcal{GT}_{\text{ni}} \subseteq \mathcal{GT}$. Then we define the set of constant (resp. non-constant) entry locations $\mathcal{EL}_{\text{c},\mathcal{GT}_{\text{ni}},\mathfrak{r}}$ (resp. $\mathcal{EL}_{\text{nc},\mathcal{GT}_{\text{ni}},\mathfrak{r}}$) by

$$\mathcal{EL}_{\text{c},\mathcal{GT}_{\text{ni}},\mathfrak{r}} = \left\{ \ell \;\middle|\; \ell \in \mathcal{EL}_{\mathcal{GT}_{\text{ni}}}, \mathfrak{r}(\ell) \in \mathbb{R} \right\} \quad \text{and} \tag{B.45.8}$$

$$\mathcal{EL}_{\text{nc},\mathcal{GT}_{\text{ni}},\mathfrak{r}} = \left\{ \ell \;\middle|\; \ell \in \mathcal{EL}_{\mathcal{GT}_{\text{ni}}}, \mathfrak{r}(\ell) \notin \mathbb{R} \right\}. \tag{B.45.9}$$

If the ranking function $\mathfrak{r}$ is clear from the context we omit it in the index.

This special treatment of locations ℓ where $\mathfrak{r}(\ell)$ is constant can allow the propagation of already computed expected runtime bounds. For example, let $\mathcal{GT}_{\text{ni}} = \mathcal{GT}_{>} = \{g\}$ where $g = \{t\}$ with $t = (\ell, _, _, _, \ell')$ and $\ell \neq \ell'$. Then a trivial ranking function $\mathfrak{r}$ with $\mathfrak{r}(\ell) = 1$, and $\mathfrak{r}(\ell') = 0$ allows us to compute an expected time bound for g by adding the expected time bounds of $\mathcal{GT}_{\text{ni}}$'s entry transitions.

Nevertheless, constant ranking functions are only of limited use. For example, if g has the same start and target location one can*not* decrease g with a constant ranking function. Similarly, if one uses constant ranking functions for the transitions g and g', where g is an entry transition w.r.t. $\mathcal{GT}_{\text{ni}}{}' = \{g'\}$ and g' is an entry transition w.r.t. $\mathcal{GT}_{\text{ni}} = \{g\}$, then this propagation of expected bounds cannot turn infinite expected runtime bounds for g and g' into finite ones.

Theorem B.45.26 (Expected Time Bounds)**.** *Let $(\mathcal{RB}_{\mathbb{E}}, \mathcal{SB}_{\mathbb{E}})$ be an expected bound pair, $\mathcal{RB}$ a (non-probabilistic) runtime bound, and $\mathfrak{r}$ a PLRF for some $\mathcal{GT}_{>} \subseteq \mathcal{GT}_{\text{ni}} \subseteq \mathcal{GT}$. Then $\mathcal{RB}'_{\mathbb{E}} : \mathcal{GT} \to \mathcal{B}$ is an expected runtime bound where*

$$\mathcal{RB}'_{\mathbb{E}}(g) = \begin{cases} \mathcal{RB}_{\mathbb{E}}(g), & \text{if } g \notin \mathcal{GT}_{>} \\ \displaystyle\sum_{\substack{\ell \in \mathcal{EL}_{\text{nc},\mathcal{GT}_{\text{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\text{ni}}}(\ell)}} \left(\sum_{t=(_,_,_,_,\ell) \in h} \mathcal{RB}(t) \right) \cdot \left(\mathcal{SB}_{\mathbb{E}}(h, \ell, \cdot)\left(\lceil \mathfrak{r}(\ell) \rceil\right)\right) & \\ \qquad + \displaystyle\sum_{\substack{\ell \in \mathcal{EL}_{\text{c},\mathcal{GT}_{\text{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\text{ni}}}(\ell)}} \mathcal{RB}_{\mathbb{E}}(h) \cdot \lceil \mathfrak{r}(\ell) \rceil, & \text{if } g \in \mathcal{GT}_{>} \end{cases}$$

Proof. Let $\mathcal{SB}_{\mathbb{E}}$, $\mathcal{RB}$, $\mathcal{RB}_{\mathbb{E}}$, and $\mathfrak{r}$ be according to the theorem. For $g \notin \mathcal{GT}_{>}$ we have $\mathcal{RB}'_{\mathbb{E}}(g) = \mathcal{RB}_{\mathbb{E}}(g)$ and since $\mathcal{RB}_{\mathbb{E}}$ is required to be a valid expected time bound, $\mathcal{RB}'_{\mathbb{E}}$ itself is a valid expected time bound for $g \notin \mathcal{GT}_{>}$. So, assume $g \in \mathcal{GT}_{>}$ and fix some $s_0 \in \Sigma$.

Moreover, consider

$$n = |s_0| \left(\sum_{\substack{\ell \in \mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}}},\\ h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} \sum_{t=(_,_,_,_,\ell) \in h} \mathcal{RB}(t) \right) \in \overline{\mathbb{R}}_{\geq 0}.$$

If $n = \infty$, then, as the sum has only finitely many addends, there has to be some $\ell \in \mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}}}$, some $h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)$, and some $t' = (_,_,_,_,\ell) \in h$ with $\mathcal{RB}(t') = \infty$. If $|s_0| \left(\mathcal{RB}'_{\mathbb{E}}(g)\right) = \infty$, we infer a valid expected time bound. Thus, let us consider the case where $|s_0| \left(\mathcal{RB}'_{\mathbb{E}}(g)\right) < \infty$. This is only possible if $|s_0| \left(\mathcal{SB}_{\mathbb{E}}(h,\ell,\cdot)(\lceil \mathfrak{r}(\ell) \rceil)\right) = 0$. Since $\mathfrak{r}(\ell)$ is a non-constant linear polynomial with real coefficients in the program variables (as $\ell \in \mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}}}$), by definition of $\lceil \cdot \rceil$ we can only have $|s_0| \left(\mathcal{SB}_{\mathbb{E}}(h,\ell,\cdot)(\lceil \mathfrak{r}(\ell) \rceil)\right) = 0$ if $\mathfrak{r}(\ell) = \sum_{x \in \mathcal{PV}} r_x \cdot x$ for real numbers $r_x \in \mathbb{R}$ for $x \in \mathcal{PV}$ (i.e., the constant coefficient is zero) such that $r_x \neq 0$ implies $|s_0| \left(\mathcal{SB}_{\mathbb{E}}(h,\ell,x)\right) = 0$. By definition of $\mathcal{SB}_{\mathbb{E}}$ and the random variable $\mathcal{S}(h,\ell,x)$, this means that in any admissible run for all $x \in \mathcal{PV}$ with $r_x \neq 0$, the value of x is zero after entering ℓ with the general transition h where the system was started in the initial state s_0. Thus, in any admissible run the following holds: if a part belonging to $\mathcal{GT}_{\mathrm{ni}}$ is entered via the location ℓ by h, then since $\mathfrak{r}(\ell)$ evaluates to zero in this configuration, no transition from $\mathcal{GT}_{>}$ occurs in this part (as to execute such a general transition, $\mathfrak{r}$ must evaluate to a positive value, but only non-increasing transitions precede the one from $\mathcal{GT}_{>}$). Thus, if $|s_0| \left(\mathcal{RB}'_{\mathbb{E}}(g)\right) < \infty$ but $n = \infty$, the probability is zero that $\mathcal{GT}_{\mathrm{ni}}$ is entered infinitely often and a general transition from $\mathcal{GT}_{>}$ is executed in each of these infinite parts belonging to $\mathcal{GT}_{\mathrm{ni}}$. Similarly, we can assume $|s_0| \left(\sum_{\ell \in \mathcal{EL}_{\mathrm{c},\mathcal{GT}_{\mathrm{ni}}}} \sum_{h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)} \mathcal{RB}_{\mathbb{E}}(h) \right) < \infty$. Combining these two assumptions implies that the set of all runs entering $\mathcal{GT}_{\mathrm{ni}}$ *infinitely* often such that the value of E_i (i.e., the value of the ranking function when entering $\mathcal{GT}_{\mathrm{ni}}$) is positive for infinitely many i has probability 0. Therefore, we do not consider those runs in the following, i.e., in the following for an admissible run there exist only finitely many i such that E_i is positive. Note that if $E_i = 0$ for some visit to $\mathcal{GT}_{\mathrm{ni}}$, in this visit of $\mathcal{GT}_{\mathrm{ni}}$ *no* general transition from $\mathcal{GT}_{>}$ can be executed.

Now we show that for every $g \in \mathcal{GT}_{>}$ we have

$$\begin{aligned} & \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{R}(g)) \\ \leq\ & |s_0| \left(\sum_{\substack{\ell \in \mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}}},\\ h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} \left(\sum_{t=(_,_,_,_,\ell) \in h} \mathcal{RB}(t) \right) \cdot \left(\mathcal{SB}_{\mathbb{E}}(h,\ell,\cdot)(\lceil \mathfrak{r}(\ell) \rceil)\right) \right) \end{aligned}$$

$$+ \sum_{\substack{\ell \in \mathcal{EL}_{\mathrm{c},\mathcal{GT}_{\mathrm{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} \mathcal{RB}_{\mathbb{E}}(h) \cdot \lceil \mathfrak{r}(\ell) \rceil \Bigg) \tag{B.45.10}$$

To ease notation, we define $\mathcal{T}^{in}_{\mathcal{GT}_{\mathrm{ni}}} = \bigcup_{\ell \in \mathcal{EL}_{\mathcal{GT}_{\mathrm{ni}}}} \bigcup_{h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)} h \subseteq \mathcal{T}$. To prove (B.45.10), we partition the set Conf^ω such that for each set in the partition we have all information at hand to prove the desired statement. The crucial information is in which of the visits of $\mathcal{GT}_{\mathrm{ni}}$ the value of E_i was positive, in which of these visits at least one transition from $\mathcal{GT}_{>}$ was used, and which transition was used to enter $\mathcal{GT}_{\mathrm{ni}}$ in this case. We store this information in sets of the form $B_{N,M,(t_i)_{i\in N}}$, where $M \subseteq N \subseteq \mathbb{N}$ are *finite*, and $t_i \in \mathcal{T}^{in}_{\mathcal{GT}_{\mathrm{ni}}}$ for all $i \in N$. If $N = \emptyset$, then $M = \emptyset$ and by default $(t_i)_{i\in N} = ()$. In this case, we define

$$B_{N,M,(t_i)_{i\in N}} = B_{\emptyset,\emptyset,()} = \{\vartheta \in \mathsf{Conf}^\omega \mid E_i(\vartheta) = 0 \text{ for all } i \in \mathbb{N}\}.$$

For example, this set contains all admissible runs that never visit $\mathcal{GT}_{\mathrm{ni}}$.

For $N \neq \emptyset$ where $M \subseteq N \subseteq \mathbb{N}$ and $t_i \in \mathcal{T}^{in}_{\mathcal{GT}_{\mathrm{ni}}}$ for $i \in N$, we define

$$\begin{aligned} B_{N,M,(t_i)_{i\in N}} = \Big\{ \vartheta \in \mathsf{Conf}^\omega \;\Big|\; & E_i(\vartheta) > 0 \wedge \vartheta[I^e_i(\vartheta) - 1] = (_, t_i, _) \text{ for } i \in N \\ & \wedge E_i(\vartheta) = 0 \text{ for } i \notin N \\ & \wedge I^{>}_{i,0}(\vartheta) > 0 \iff i \in M \Big\}, \end{aligned}$$

i.e., this set consists of all admissible runs where the visits to $\mathcal{GT}_{\mathrm{ni}}$ where E_i evaluates to a positive value are exactly the ones with $i \in N$, the transition used directly *before* the $(i+1)$-th visit is t_i, and the indices in M are exactly those visits in which $\mathcal{GT}_{>}$ is executed at least once. In total, we have *countably* many sets $B_{N,M,(t_i)_{i\in N}}$ since N and M are from the countable set of *finite* subsets of $\mathbb{N}$, and $(t_i)_{i\in N}$, when interpreted as a finite sequence, is from the countable set $\mathcal{T}^*$. We have:

(i) $B_{N,M,(t_i)_{i\in N}}$ is measurable for all $M \subseteq N \subseteq \mathbb{N}$ and $t_i \in \mathcal{T}^{in}_{\mathcal{GT}_{\mathrm{ni}}}$ for all $i \in N$.

(ii) The sets $B_{N,M,(t_i)_{i\in N}}$ are pairwise disjoint.

(iii)
$$\biguplus_{\substack{M \subseteq N \subseteq \mathbb{N}, \\ |N| \in \mathbb{N}, \\ t_i \in \mathcal{T}^{in}_{\mathcal{GT}_{\mathrm{ni}}} \text{ for all } i \in N}} B_{N,M,(t_i)_{i\in N}} = \mathsf{Conf}^\omega.$$

Here, (iii) holds only almost-surely, i.e., there are runs which are not contained in any of the $B_{N,M,(t_i)_{i\in N}}$ but these runs are *not* admissible under the assumption that E_i only evaluates to a positive value finitely often.

Let us fix some $N, M, (t_i)_{i\in N}$. Moreover, we assume $t_i = (_,_,_,_,\ell_i)$ with $\ell_i \in \mathcal{EL}_{\mathcal{GT}_{\mathrm{ni}}}$ and let g_i be the unique general transition with $t_i \in g_i$. We then have

$$
\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[B_{N,M,(t_i)_{i\in N}}\right]\cdot\sum_{i\in N}E_i\right)\\
&=\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[B_{N,M,(t_i)_{i\in N}}\right]\cdot\sum_{i\in N}\max\left\{s_{I_i^e-1}\left(\mathfrak{r}(\ell_{I_i^e-1})\right),0\right\}\right) && \text{(by Def. B.45.19)}\\
&\le\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[B_{N,M,(t_i)_{i\in N}}\right]\cdot\sum_{i\in N}\left|s_{I_i^e-1}\right|\left(\lceil\mathfrak{r}(\ell_i)\rceil\right)\right)\\
&\le\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[B_{N,M,(t_i)_{i\in N}}\right]\cdot\sum_{i\in N}\mathcal{S}(g_i,\ell_i,\cdot)\left(\lceil\mathfrak{r}(\ell_i)\rceil\right)\right)\\
&=\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[B_{N,M,(t_i)_{i\in N}}\right]\cdot\sum_{\substack{\ell\in\mathcal{EL}_{\mathcal{GT}_{\mathrm{ni}}},\\ h\in\mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}}\left(\sum_{t=(_,_,_,_,\ell)\in h}C_{(t_i)_{i\in N}}(t)\right)\cdot\mathcal{S}(h,\ell,\cdot)\left(\lceil\mathfrak{r}(\ell)\rceil\right)\right)
\end{aligned}
$$

where $C_{(t_i)_{i\in N}}(t)$ denotes the number of occurrences of transition t in $(t_i)_{i\in N}$. We continue by splitting the sum:

$$
\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[B_{N,M,(t_i)_{i\in N}}\right]\cdot\sum_{\substack{\ell\in\mathcal{EL}_{\mathcal{GT}_{\mathrm{ni}}},\\ h\in\mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}}\left(\sum_{t=(_,_,_,_,\ell)\in h}C_{(t_i)_{i\in N}}(t)\right)\cdot\mathcal{S}(h,\ell,\cdot)\left(\lceil\mathfrak{r}(\ell)\rceil\right)\right)\\
&=\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[B_{N,M,(t_i)_{i\in N}}\right]\cdot\left(\sum_{\substack{\ell\in\mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}}},\\ h\in\mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}}\left(\sum_{t=(_,_,_,_,\ell)\in h}C_{(t_i)_{i\in N}}(t)\right)\cdot\mathcal{S}(h,\ell,\cdot)\left(\lceil\mathfrak{r}(\ell)\rceil\right)\right)\right.
\end{aligned}
$$

$$
\begin{aligned}
&+ \sum_{\substack{\ell \in \mathcal{EL}_{\mathrm{c},\mathcal{GT}_{\mathrm{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} \left(\sum_{t=(_,_,_,_,\ell)\in h} C_{(t_i)_{i\in N}}(t) \right) \cdot |s_0| \left(\mathcal{S}(h,\ell,\cdot)\left(\lceil \mathfrak{r}(\ell) \rceil\right)\right) \Bigg) \Bigg) \\
\leq\ & \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \Bigg(\left[B_{N,M,(t_i)_{i\in N}} \right] \cdot \Bigg(\sum_{\substack{\ell \in \mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} \left(\sum_{t=(_,_,_,_,\ell)\in h} |s_0| \left(\mathcal{RB}(t) \right) \right) \cdot \mathcal{S}(h,\ell,\cdot)\left(\lceil \mathfrak{r}(\ell) \rceil\right) \\
&+ \sum_{\substack{\ell \in \mathcal{EL}_{\mathrm{c},\mathcal{GT}_{\mathrm{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} \mathcal{R}(h) \cdot \left(\lceil \mathfrak{r}(\ell) \rceil\right) \Bigg) \Bigg) \\
=\ & \sum_{\substack{\ell \in \mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} \left(\sum_{t=(_,_,_,_,\ell)\in h} |s_0| \left(\mathcal{RB}(t) \right) \right) \cdot \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \left(\left[B_{N,M,(t_i)_{i\in N}} \right] \cdot \mathcal{S}(h,\ell,\cdot)\left(\lceil \mathfrak{r}(\ell) \rceil\right) \right) \\
&+ \sum_{\substack{\ell \in \mathcal{EL}_{\mathrm{c},\mathcal{GT}_{\mathrm{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} \lceil \mathfrak{r}(\ell) \rceil \cdot \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \left(\left[B_{N,M,(t_i)_{i\in N}} \right] \cdot \mathcal{R}(h) \right) \qquad (\ddagger)
\end{aligned}
$$

Here, in the last step we have used that $\lceil \mathfrak{r}(\ell) \rceil$ is constant in the second part of the sum. Note that the penultimate inequation holds since $\vartheta \in B_{N,M,(t_i)_{i\in N}}$ implies that the number of positions at which a transition $(_,_,_,_,\ell) \in h$ appears in ϑ, and therefore also in the tuple $(t_i)_{i\in N}$, is bounded by the minimum of $\sum_{t=(_,_,_,_,\ell)}(|s_0|\left(\mathcal{RB}(t)\right))$ and $\mathcal{R}(h)(\vartheta)$. Hence, it is bounded by both values.

Recall the definition of T'_i from Def. B.45.10:

$$T'_i(\vartheta) = \inf\{j \in \mathbb{N} \mid R_{i,j}(\vartheta) = 0\}$$

By definition we have for every $\vartheta \in \mathsf{Conf}^\omega$

$$I^{>}_{i,0}(\vartheta) = 0 \implies R_{i,0}(\vartheta) = 0 \implies T'_i(\vartheta) = 0.$$

Moreover, the properties Boundedness (a), Boundedness (b), Non-Increase, and Decrease of PLRFs together imply that the set of runs where a transition $g \in \mathcal{GT}_{>}$ is executed when $\mathfrak{r}$ is already 0 or less is a null set. So, for all $i \in M$ and all admissible runs $\vartheta \in B_{N,M,(t_i)_{i\in N}}$ we have

$$R_{i,0}(\vartheta) > 0.$$

Therefore, $T'_i(\vartheta) \geq \left| I_i(\vartheta) \cap \left\{ m \in \mathbb{N} \;\middle|\; \iota'_m \in \bigcup \mathcal{GT}_> \right\} \right|$ for any admissible run $\vartheta = (\ell_0, \iota'_0, s_0) \cdots \in B_{N,M,(t_i)_{i\in N}}$. Then for $g \in \mathcal{GT}_>$ we have

$$\begin{aligned}
&\left[B_{N,M,(t_i)_{i\in N}}\right](\vartheta) \cdot \mathcal{R}(g)(\vartheta) \\
&= \left[B_{N,M,(t_i)_{i\in N}}\right](\vartheta) \cdot \left| \biguplus_{i\in M} \left(I_i(\vartheta) \cap \left\{ m \in \mathbb{N} \;\middle|\; \iota'_m \in g \right\} \right) \right| \\
&\leq \left[B_{N,M,(t_i)_{i\in N}}\right](\vartheta) \cdot \sum_{i\in M} T'_i(\vartheta) \\
&\leq \left[B_{N,M,(t_i)_{i\in N}}\right](\vartheta) \cdot \sum_{i\in \mathbb{N}} T'_i(\vartheta) \qquad (\dagger\dagger)
\end{aligned}$$

The last step holds because $M \subseteq N \subseteq \mathbb{N}$.

As the sets $B_{N,M,(t_i)_{i\in N}}$ are (almost-surely) a countable partition of Conf^ω we have

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{R}(g)) \\
&\leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{i\in\mathbb{N}} T'_i \right) && \text{(by } (\dagger\dagger)\text{)} \\
&= \sum_{i\in\mathbb{N}} \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(T'_i \right) \\
&\leq \sum_{i\in\mathbb{N}} \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(R_{i,0} \right) && \text{(by Cor. B.45.17)} \\
&\leq \sum_{i\in\mathbb{N}} \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(E_i \right) && \text{(by Lemmas A.6.4 and B.45.20)} \\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{i\in\mathbb{N}} E_i \right). && (\ddagger\ddagger)
\end{aligned}$$

With $(\ddagger)$ and $(\ddagger\ddagger)$ we can conclude (B.45.10):

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{R}(g)) \\
&\leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{i\in\mathbb{N}} E_i \right) && \text{(by } (\ddagger\ddagger)\text{)} \\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[\biguplus_{\substack{M\subseteq N\subseteq\mathbb{N},\\ |N|\in\mathbb{N},\\ t_i\in\mathcal{T}^{in}_{\mathcal{GT}_{\mathrm{ni}}} \text{ for all } i\in N}} B_{N,M,(t_i)_{i\in N}} \right] \cdot \sum_{i\in\mathbb{N}} E_i \right)
\end{aligned}$$

$$= \sum_{\substack{M\subseteq N\subseteq\mathbb{N},\\ |N|\in\mathbb{N},\\ t_i\in\mathcal{T}^{in}_{\mathcal{GT}_{\mathrm{ni}}} \text{ for all } i\in N}} \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[B_{N,M,(t_i)_{i\in N}}\right]\cdot\sum_{i\in N}E_i\right)$$

$$= \sum_{\substack{M\subseteq N\subseteq\mathbb{N},\\ 0<|N|\in\mathbb{N},\\ t_i\in\mathcal{T}^{in}_{\mathcal{GT}_{\mathrm{ni}}} \text{ for all } i\in N}} \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[B_{N,M,(t_i)_{i\in N}}\right]\cdot\sum_{i\in N}E_i\right) \qquad (\text{as } \left[B_{\emptyset,\emptyset,()}\right]\cdot\textstyle\sum_{i\in\emptyset}E_i\equiv 0)$$

$$\leq \sum_{\substack{M\subseteq N\subseteq\mathbb{N},\\ 0<|N|\in\mathbb{N},\\ t_i\in\mathcal{T}^{in}_{\mathcal{GT}_{\mathrm{ni}}} \text{ for all } i\in N}} \left(\sum_{\substack{\ell\in\mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}}},\\ h\in\mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}}\left(\sum_{t=(_,_,_,_,\ell)\in h}|s_0|\left(\mathcal{RB}(t)\right)\right)\cdot\right.$$

$$\left.\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[B_{N,M,(t_i)_{i\in N}}\right]\cdot\mathcal{S}(h,\ell,\cdot)\left(\lceil\mathfrak{r}(\ell)\rceil\right)\right)+ \qquad (\text{by } (\ddagger))\right.$$

$$\left.\sum_{\substack{\ell\in\mathcal{EL}_{\mathrm{c},\mathcal{GT}_{\mathrm{ni}}},\\ h\in\mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}}\lceil\mathfrak{r}(\ell)\rceil\cdot\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[B_{N,M,(t_i)_{i\in N}}\right]\cdot\mathcal{R}(h)\right)\right)$$

$$= \sum_{\substack{\ell\in\mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}}},\\ h\in\mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}}\left(\sum_{t=(_,_,_,_,\ell)\in h}|s_0|\left(\mathcal{RB}(t)\right)\right)\cdot\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathcal{S}(h,\ell,\cdot)\left(\lceil\mathfrak{r}(\ell)\rceil\right)\right)$$

$$+\sum_{\substack{\ell\in\mathcal{EL}_{\mathrm{c},\mathcal{GT}_{\mathrm{ni}}},\\ h\in\mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}}\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{R}(h))\cdot\lceil\mathfrak{r}(\ell)\rceil$$

$$\leq \sum_{\substack{\ell\in\mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}}},\\ h\in\mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}}\left(\sum_{t=(_,_,_,_,\ell)\in h}|s_0|\left(\mathcal{RB}(t)\right)\right)\cdot\left(\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathcal{S}(h,\ell,\cdot)\right)\left(\lceil\mathfrak{r}(\ell)\rceil\right)\right)$$

$$+\sum_{\substack{\ell\in\mathcal{EL}_{\mathrm{c},\mathcal{GT}_{\mathrm{ni}}},\\ h\in\mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}}|s_0|\left(\mathcal{RB}_{\mathbb{E}}(h)\right)\cdot\lceil\mathfrak{r}(\ell)\rceil$$

$$\leq \sum_{\substack{\ell \in \mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} \left(\sum_{t=(_,_,_,_,\ell)\in h} |s_0|\left(\mathcal{RB}(t)\right) \right) \cdot \left(|s_0|\left(\mathcal{SB}_{\mathbb{E}}(h,\ell,\cdot)\right)\left(\lceil \mathfrak{r}(\ell) \rceil\right)\right)$$

(since $\lceil \mathfrak{r}(\ell) \rceil$ is linear)

$$+ \sum_{\substack{\ell \in \mathcal{EL}_{\mathrm{c},\mathcal{GT}_{\mathrm{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} |s_0|\left(\mathcal{RB}_{\mathbb{E}}(h)\right) \cdot \lceil \mathfrak{r}(\ell) \rceil$$

$$= |s_0| \Bigg(\sum_{\substack{\ell \in \mathcal{EL}_{\mathrm{nc},\mathcal{GT}_{\mathrm{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} \left(\sum_{t=(_,_,_,_,\ell)\in h} \mathcal{RB}(t) \right) \cdot \left(\mathcal{SB}_{\mathbb{E}}(h,\ell,\cdot)\right)\left(\lceil \mathfrak{r}(\ell) \rceil\right)$$

$$+ \sum_{\substack{\ell \in \mathcal{EL}_{\mathrm{c},\mathcal{GT}_{\mathrm{ni}}}, \\ h \in \mathcal{ET}_{\mathcal{GT}_{\mathrm{ni}}}(\ell)}} \mathcal{RB}_{\mathbb{E}}(h) \cdot \lceil \mathfrak{r}(\ell) \rceil \Bigg),$$

i.e., $\mathcal{RB}'_{\mathbb{E}}$ is indeed a valid expected time bound. □

We illustrate the need for constant ranking functions with an example.

Example B.45.27. Consider the PITS $\mathbf{T}_{ndc}$ in Fig. B.1 from our additional benchmarks. Here, we have $\mathcal{PV} = \{x\}$, i.e., y is a temporary variable. As usual, we omit trivial updates and probabilities that are 1.

We start with finding a PLRF for g_1. Here, we compute the ranking function $\mathfrak{r}$ with $\mathfrak{r}(\ell_1) = 4 \cdot x$ and $\mathfrak{r}(\ell_2) = 4 \cdot x - 4$ for $\mathcal{GT}_{>} = \{g_1\}$ and $\mathcal{GT}_{\mathrm{ni}} = \{g_1, g_2\}$. First of all, g_1 satisfies the condition Boundedness (b) since for any admissible configuration (ℓ, t, s) with $t \in g_1$ we have $s(\mathfrak{r}(\ell)) \geq 0$. Moreover, g_1 satisfies the condition Decrease as for any state s with $s(\varphi_{g_1}) = \text{true}$, i.e., $s(x) > 0$, we have

$$\begin{aligned} &\exp_{\mathfrak{r},g_1,s} \\ &= \tfrac{1}{2} \cdot 4 \cdot s(x) + \tfrac{1}{4} \cdot (4 \cdot (s(x)+1) - 4) + \tfrac{1}{4} \cdot (4 \cdot s(x) - 4) = 4 \cdot s(x) - 1 \\ &\leq s(\mathfrak{r}(\ell_1)) - 1. \end{aligned}$$

The transition g_2 trivially satisfies Boundedness (a) since it only consists of one transition. More precisely, for any admissible path $\cdots\ c'\ c$ there is a unique

$\bowtie_{g,c'} \in \{<, \geq\}$ such that for all $c = (\ell, t, s)$ with $t \in g_2$, i.e., $t = t_4$ and $\ell = \ell_1$, we have $s(\mathfrak{r}(\ell_1)) \bowtie_{g,c'} 0$ since the value for temporary variable y which the scheduler chooses depends only on c'.

The transition g_2 also satisfies Non-Increase since for any state s such that $s(\varphi_{g_2}) = \text{true}$, i.e., $s(y) > 0$, we have

$$\begin{aligned} & \exp_{\mathfrak{r},g_2,s} \\ = \; & 4 \cdot (s(x) - s(y)) \\ \leq \; & 4 \cdot s(x) - 4 \\ = \; & s(\mathfrak{r}(\ell_2)) \,. \end{aligned}$$

Here, $\mathcal{EL}_{\mathcal{GT}_{\text{ni}}} = \{\ell_1\}$ and $\mathcal{ET}_{\mathcal{GT}_{\text{ni}}}(\ell_1) = \{g_0\}$. So let us assume that we have already computed $\mathcal{RB}(t_0) = 1$ and $\mathcal{SB}_{\mathbb{E}}(g_0, \ell_1, x) = x$. By using Thm. 17.3.8, we infer

$$\mathcal{RB}_{\mathbb{E}}(g_1) = \mathcal{RB}(t_0) \cdot \mathcal{SB}_{\mathbb{E}}(g_0, \ell_1, \cdot)(\lceil \mathfrak{r}(\ell_1) \rceil) = 4 \cdot x.$$

However, there is no PLRF $\mathfrak{r}'$ for $\mathcal{GT}'_{>} = \{g_2\}$ and $\mathcal{GT}'_{\text{ni}} = \{g_1, g_2\}$ since t_3 increases the value of x, and t_1 and t_2 leave the value of x unchanged, while t_4 decreases the value of x. Thus, we consider $\mathcal{GT}'_{>} = \{g_2\} = \mathcal{GT}'_{\text{ni}}$. In this case, $\mathcal{EL}_{\mathcal{GT}_{\text{ni}}'} = \{\ell_2\}$ and $\mathcal{ET}_{\mathcal{GT}_{\text{ni}}}(\ell_2) = \{g_1\}$. Unfortunately, we cannot compute finite *non-probabilistic* runtime bounds for t_2 and t_3 since they might occur arbitrarily often in an admissible run, and we cannot compute a finite expected size bound $\mathcal{SB}_{\mathbb{E}}(g_1, \ell_2, x)$ without a finite expected runtime bound for g_2 as (g_1, ℓ_2, x) and (g_2, ℓ_1, x) are members of the same non-trivial SCC of the general result variable graph (see Thm. 17.4.10). Thus, by using Thm. 17.3.8, we would always obtain

$$\mathcal{RB}_{\mathbb{E}}(g_2) = \left(\sum_{t \in g_1} \mathcal{RB}(t) \right) \cdot \mathcal{SB}_{\mathbb{E}}(g_1, \ell_2, \cdot)\left(\left\lceil \mathfrak{r}'(\ell_2) \right\rceil\right) = \infty$$

since $\mathfrak{r}'(\ell_2)$ is non-zero. However, we can infer the *constant* ranking function $\mathfrak{r}'$ with $\mathfrak{r}'(\ell_2) = 1$ and $\mathfrak{r}'(\ell_1) = 0$ for $\mathcal{GT}'_{>} = \{g_2\} = \mathcal{GT}'_{\text{ni}}$. Using Thm. B.45.26, we obtain

$$\mathcal{RB}_{\mathbb{E}}(g_2) = \mathcal{RB}_{\mathbb{E}}(g_1) \cdot \mathfrak{r}(\ell_2) = 4 \cdot x.$$

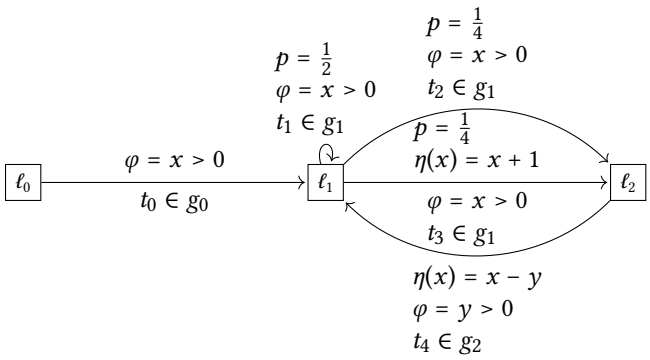


Figure B.1: PITS $\mathbf{T}_{ndc}$

B.46 Proof of Lemma 17.3.12

Proof. Every linear polynomial is clearly concave. For the other direction, by Def. 17.1.1, if $b \in \mathcal{B}$ does not contain ∞, then b is built from variables in $\mathcal{PV}$, non-negative constants in $\mathbb{R}_{\geq 0}$, and the operators $+$, $\cdot$, and $v^{\bullet}$ for $v \geq 1$. Thus, b defines a function which is infinitely often continuously differentiable.

It is a standard result from calculus (see, e.g., [BV04]), that such a b is concave as a function $(\mathcal{PV} \to \mathbb{R}_{\geq 0}) \to \mathbb{R}$ iff its Hessian matrix H, i.e., the matrix of its second derivatives, is negative semi-definite on $(\mathcal{PV} \to \mathbb{R}_{>0})$. A quadratic real matrix $A \in \mathbb{R}^{n \times n}$ is *negative semi-definite* if for all $\vec{v} \in \mathbb{R}^n$ we have $\vec{v}^T \cdot A \cdot \vec{v} \leq 0$.

So, let b be a bound which is not a linear polynomial. Then its Hessian matrix H contains only terms built from variables in $\mathcal{PV}$, constants in $\mathbb{R}_{\geq 0}$, and the operators $+$, $\cdot$, and $v^{\bullet}$ for $v \geq 1$. Moreover, H is not the zero matrix. Now let A be H evaluated at the assignment from $(\mathcal{PV} \to \mathbb{R}_{>0})$ which assigns every program variable the value 1. Then each entry of A is non-negative and at least one entry is positive (since only non-negative numbers, addition, multiplication and exponentiation with strictly positive basis are involved, no expressions can cancel each other out). Now let $\vec{v}$ be the vector where each entry is 1. Then $A \cdot \vec{v}$ is a vector with only non-negative entries and at least one positive entry. Thus,

$$\vec{v}^T \cdot A \cdot \vec{v} > 0,$$

since this value is just the sum of the entries of $A \cdot \vec{v}$.

Hence, A is *not* negative semi-definite, i.e., H is *not* negative semi-definite on $(\mathcal{PV} \to \mathbb{R}_{>0})$. Thus, b is not concave. □

B.47 Proof of Thm. 17.4.3

Proof. Let $\mathfrak{s}(c') = (g_{\mathfrak{s}}, s'_{\mathfrak{s}})$, i.e., $s'(x) = s'_{\mathfrak{s}}(x)$ for all $x \in \mathcal{PV}$. If $g_{\mathfrak{s}} \neq g$, the right hand side of the inequation is zero and thus, there is nothing to show. So let $g_{\mathfrak{s}} = g$. By definition of the underlying probability mass function $\mu^{\mathrm{pt}}_{\bullet,\bullet}$ (see Sect. 17.1), we have:

$$
\begin{aligned}
&\sum_{\substack{c=(\ell,t,s)\in \mathsf{Conf},\\ t\in g}} \mu^{\mathrm{pt}}_{c',\mathfrak{s}}(c)\cdot \left|s(x)-s'(x)\right| \\
&= \sum_{\substack{c=(\ell,t,s)\in \mathsf{Conf},\\ t=(_,p,_,_,\ell)\in g}} \left|s(x)-s'(x)\right| \cdot p \cdot \prod_{y\in\mathcal{PV}} \mathrm{q}_{s'_{\mathfrak{s}},t,s(y)}(y) \cdot \prod_{u\in\mathcal{TV}} \left[s'_{\mathfrak{s}}(u)=s(u)\right] \\
&= \sum_{\substack{v\in\mathbb{Z},\\ t=(_,p,_,\eta,\ell)\in g}} |v|\cdot p \cdot \begin{cases} \left[s'_{\mathfrak{s}}(\eta(x))-s'(x)=v\right], & \text{if } \eta(x)\in\mathbb{Z}[\mathcal{V}] \\ D(s'_{\mathfrak{s}})(v), & \text{if } \eta(x)=D\in\mathcal{D} \end{cases} \\
&= \sum_{\substack{v\in\mathbb{N},\\ t=(_,p,_,\eta,\ell)\in g}} v\cdot p \cdot \begin{cases} \left[\left|s'_{\mathfrak{s}}(\eta(x))-s'(x)\right|=v\right], & \text{if } \eta(x)\in\mathbb{Z}[\mathcal{V}] \\ D(s'_{\mathfrak{s}})(v)+D(s'_{\mathfrak{s}})(-v), & \text{if } \eta(x)=D\in\mathcal{D} \end{cases} \\
&= \sum_{t=(_,p,_,\eta,\ell)\in g} p\cdot\left(\sum_{v\in\mathbb{N}} v\cdot \begin{cases} \left[\left|s'_{\mathfrak{s}}(\eta(x))-s'(x)\right|=v\right], & \text{if } \eta(x)\in\mathbb{Z}[\mathcal{V}] \\ D(s'_{\mathfrak{s}})(v)+D(s'_{\mathfrak{s}})(-v), & \text{if } \eta(x)=D\in\mathcal{D} \end{cases}\right) \\
&\leq \sum_{t=(_,p,_,\eta,\ell)\in g} p\cdot \begin{cases} \left|s'_{\mathfrak{s}}\right|(\lceil\eta(x)-x\rceil), & \text{if } \eta(x)\in\mathbb{Z}[\mathcal{V}] \\ \left|s'_{\mathfrak{s}}\right|(\mathfrak{E}(D)), & \text{if } \eta(x)=D\in\mathcal{D} \end{cases} \\
&\qquad \text{(since } s'_{\mathfrak{s}}(x)=s'(x) \text{ for all } x\in\mathcal{PV} \text{ and } \left|s'_{\mathfrak{s}}\right|(\lceil\eta(x)-x\rceil)\geq\left|s'_{\mathfrak{s}}\right|(|\eta(x)-x|)) \\
&\leq \left|s'_{\mathfrak{s}}\right|(\mathcal{CB}_{\mathbb{E}}(g,\ell,x)) \\
&= \left|s'\right|(\mathcal{CB}_{\mathbb{E}}(g,\ell,x)) \qquad \text{(since } s'_{\mathfrak{s}}(x)=s'(x) \text{ for all } x\in\mathcal{PV})
\end{aligned}
$$

□

B.48 Proof of Thm. 17.4.7

Proof. First of all, for a general result variable $\beta \neq \alpha$ there is nothing to prove, since $\mathcal{SB}_{\mathbb{E}}$ is already a valid expected size bound. Let $s_0 \in \Sigma$ and $\mathfrak{s} \in \mathrm{Scheds}$.

Initial Transition: We now consider the case $g \in \mathcal{GT}_0$. It suffices to consider admissible runs $\vartheta = (\ell_0, t_{\text{in}}, s_0)(\ell_1, t_1, s_1)\cdots$ with $t_1 \in g$ and $\ell_1 = \ell$ since there is no transition leading back to the initial location. Therefore, the transition g is always the general transition that is executed first (if it is executed at all). Hence, for all considered runs ϑ we have:

$$\mathcal{S}(\alpha)(\vartheta) = \sup\{|s_1(x)|\} = |s_1(x)|$$

Thus, we use Thm. 17.4.3 with the (trivially) admissible configuration $c_0 = (\ell_0, t_{\text{in}}, s_0)$. Note that c_0 also forms an admissible finite path. Moreover, any admissible finite path starts with c_0. So we have

$$\begin{aligned}
&\mathbb{E}^{\text{pt}}_{s_0,\mathfrak{s}}(\mathcal{S}(\alpha)) \\
&= p^{\text{pt}}_{s_0,\mathfrak{s}}(c_0) \cdot \sum_{\substack{c=(\ell,t,s')\in\mathsf{Conf},\\ t\in g}} p^{\text{pt}}_{c_0,\mathfrak{s}}(c) \cdot |s'(x)| \\
&= \sum_{\substack{c=(\ell,t,s')\in\mathsf{Conf},\\ t\in g}} p^{\text{pt}}_{c_0,\mathfrak{s}}(c) \cdot |s'(x) - s_0(x) + s_0(x)| && (\text{since } p^{\text{pt}}_{s_0,\mathfrak{s}}(c_0) = 1) \\
&\leq \sum_{\substack{c=(\ell,t,s')\in\mathsf{Conf},\\ t\in g}} p^{\text{pt}}_{c_0,\mathfrak{s}}(c) \cdot (|s_0(x)| + |s'(x) - s_0(x)|) \\
&\leq |s_0(x)| + \sum_{\substack{c=(\ell,t,s')\in\mathsf{Conf},\\ t\in g}} p^{\text{pt}}_{c_0,\mathfrak{s}}(c) \cdot |s'(x) - s_0(x)| \\
&\leq |s_0(x)| + |s_0|\,(\mathcal{CB}_{\mathbb{E}}(\alpha)) && (\text{by Thm. 17.4.3}) \\
&= |s_0|\,(x + \mathcal{CB}_{\mathbb{E}}(\alpha)) \\
&= |s_0|\,\left(\mathcal{SB}'_{\mathbb{E}}(\alpha)\right)
\end{aligned}$$

Non-Initial Transition: Now let $g \in \mathcal{GT} \setminus \mathcal{GT}_0$. This case is similar to the previous case except that we look at all finite paths f leading to an execution of g. This means we have to look for all f's ending in a configuration $c = (\ell_g, t_n, s_n)$ with $t_n \in h$ and $h \in \text{pre}(g)$.

Recall that the active variables consist of all variables that influence $x + \mathcal{CB}_{\mathbb{E}}(\alpha)$. So the set of active variables of $\alpha = (g, \ell, x)$ is never empty since we always have $x \in \text{actVar}(\alpha)$. Hence, as g is not an initial general transition, it has a predecessor in the general result variable graph. Since α forms a trivial SCC of the system, this means that g cannot be applied more than once in any admissible run. In other words, we can restrict ourselves to finite paths f where g has not been applied before the

last evaluation step. Therefore, for all considered runs ϑ that start with the prefix fc for some configuration $c = (\ell', t, s)$ with $t \in g$, we get

$$\mathcal{S}(\alpha)(\vartheta) = \sup\{|s(x)|\} = |s(x)|.$$

Moreover, we define $\mathcal{S}^{\alpha}_{in}(\cdot) = \sum_{h \in \mathrm{pre}(g)} \mathcal{S}(h, \ell_g, \cdot)$, i.e., for every variable $y \in \mathcal{PV}$ we have

$$\mathcal{S}^{\alpha}_{in}(y) = \sum_{h \in \mathrm{pre}(g)} \mathcal{S}(h, \ell_g, y).$$

To ease notation, we allow to apply $\mathcal{S}^{\alpha}_{in}(y)$ also to finite prefixes instead of just runs. Again, note that $h \in \mathrm{pre}(g)$ can never be executed in an admissible run *after* g was already executed since (g, ℓ, x) is a trivial SCC of the result variable graph. So we get

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{S}(\alpha)) \\
&= \sum_{\substack{f=(\ell_0,t_0,s_0)\cdots(\ell_n,t_n,s_n),\\ t_n\in h,\\ h\in\mathrm{pre}(g)}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) \cdot \sum_{\substack{c=(\ell',t,s)\in\mathsf{Conf},\\ t\in g}} p^{\mathrm{pt}}_{(\ell_n,t_n,s_n),\mathfrak{s}}(c) \cdot |s(x)| \\
&= \sum_{\substack{f=(\ell_0,t_0,s_0)\cdots(\ell_n,t_n,s_n),\\ t_n\in h,\\ h\in\mathrm{pre}(g)}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) \cdot \sum_{\substack{c=(\ell',t,s)\in\mathsf{Conf},\\ t\in g}} p^{\mathrm{pt}}_{(\ell_n,t_n,s_n),\mathfrak{s}}(c) \cdot |s(x) - s_n(x) + s_n(x)| \\
&\leq \sum_{\substack{f=(\ell_0,t_0,s_0)\cdots(\ell_n,t_n,s_n)\\ t_n\in h,\\ h\in\mathrm{pre}(g)}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) \cdot \sum_{\substack{c=(\ell',t,s)\in\mathsf{Conf},\\ t\in g}} p^{\mathrm{pt}}_{(\ell_n,t_n,s_n),\mathfrak{s}}(c) \cdot (|s_n(x)| + |s(x) - s_n(x)|) \\
&\leq \sum_{\substack{f=(\ell_0,t_0,s_0)\cdots(\ell_n,t_n,s_n),\\ t_n\in h,\\ h\in\mathrm{pre}(g)}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f) \cdot (|s_n(x)| + |s_n|\,(\mathcal{CB}_{\mathbb{E}}(\alpha))) \qquad \text{(by Thm. 17.4.3)}
\end{aligned}$$

$$\leq \sum_{\substack{f=(\ell_0,t_0,s_0)\cdots(\ell_n,t_n,s_n),\\ t_n\in h,\\ h\in\mathrm{pre}(g)}} p^{\mathrm{pt}}_{s_0,\mathfrak{s}}(f)\cdot\left(\left|\mathcal{S}^{\alpha}_{in}(x)(f)\right| + \left|\mathcal{S}^{\alpha}_{in}(\cdot)(f)\right|(\mathcal{CB}_{\mathbb{E}}(\alpha))\right)$$
$$= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathcal{S}^{\alpha}_{in}(x) + \left|\mathcal{S}^{\alpha}_{in}(\cdot)\right|(\mathcal{CB}_{\mathbb{E}}(\alpha))\right)$$

Now, if $\mathcal{CB}_{\mathbb{E}}(\alpha)$ is *linear*:

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathcal{S}^{\alpha}_{in}(x) + \left|\mathcal{S}^{\alpha}_{in}(\cdot)\right|(\mathcal{CB}_{\mathbb{E}}(\alpha))\right)\\
\leq\ &\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathcal{S}^{\alpha}_{in}(x)\right) + \left|\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathcal{S}^{\alpha}_{in}(\cdot)\right)\right|(\mathcal{CB}_{\mathbb{E}}(\alpha))\\
\leq\ &|s_0|\left(\mathrm{size}^{\alpha}_{\mathbb{E}}(x + \mathcal{CB}_{\mathbb{E}}(\alpha))\right)\\
=\ &|s_0|\left(\mathcal{SB}'_{\mathbb{E}}(\alpha)\right)
\end{aligned}$$

This proves the second case of Thm. 17.4.7, i.e., $g \notin \mathcal{GT}_0$, $\beta = \alpha$, and $\mathcal{CB}_{\mathbb{E}}(\alpha)$ is linear.

If $\mathcal{CB}_{\mathbb{E}}(\alpha)$ *is not linear*, then we get:

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathcal{S}^{\alpha}_{in}(x) + \left|\mathcal{S}^{\alpha}_{in}(\cdot)\right|(\mathcal{CB}_{\mathbb{E}}(\alpha))\right)\\
\leq\ &\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathcal{S}^{\alpha}_{in}(x)\right) + \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left|\mathcal{S}^{\alpha}_{in}(\cdot)\right|(\mathcal{CB}_{\mathbb{E}}(\alpha))\right)\\
\leq\ &|s_0|\left(\mathrm{size}^{\alpha}_{\mathbb{E}}(x) + \mathrm{size}^{\alpha}(\mathcal{CB}_{\mathbb{E}}(\alpha))\right)\\
=\ &|s_0|\left(\mathcal{SB}'_{\mathbb{E}}(\alpha)\right)
\end{aligned}$$

□

B.49 Proof of Thm. 17.4.10

Fix a non-trivial SCC $C \subseteq \mathcal{GRV}$ of the general result variable graph. Furthermore, we fix an initial state s_0. We define $\mathsf{Conf}_C = \{(\ell, t, _) \mid t \in g, (g, \ell, _) \in C\}$ and $\mathsf{Conf}_{\neg C} = \mathsf{Conf} \setminus \mathsf{Conf}_C$ to ease readability.

Moreover, for any $x \in \mathcal{PV}$ we define the random variable

$$\mathcal{S}_C(x) = \sup_{\alpha'=(_,_,x)\in C} \mathcal{S}(\alpha').$$

So for any $\vartheta \in \mathsf{Conf}^{\omega}$, $\mathcal{S}_C(x)(\vartheta)$ is the maximal absolute value that x takes in a location ℓ in ϑ that was entered with g for general result variables (g, ℓ, x) from C. Then we immediately get $\mathcal{S}(\alpha') \leq \mathcal{S}_C(x)$ for each $\alpha' = (_, _, x) \in C$, so in particular

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathcal{S}(\alpha')\right) \leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathcal{S}_C(x)\right) \quad \text{for all } \alpha' = (_, _, x) \in C, \tag{B.49.1}$$

by monotonicity of the expected value operator.

We fix $x \in \mathcal{PV}$ from now on. To over-approximate the expected value of $S_C(x)$, in analogy to the size random variable $\mathcal{S}(\alpha)$ for $\alpha \in \mathcal{GRV}$, we define a random variable $S(x)_n$ corresponding to the maximal size of the variable x after evaluating general transitions $g \in \mathcal{GT}_C$ at most n times. By definition of Conf_C, we also take the locations into account, i.e., we only consider evaluations of $g \in \mathcal{GT}_C$ ending in a location ℓ such that $(g, \ell, _) \in C$. The special case $S(x)_0$ defaults to the value before the first such general transition is executed. The underlying idea is similar to the stochastic process $R_{i,j}$ which we defined in Def. B.45.3 to prove the correctness of our technique for inferring expected time bounds.

By $I(\vartheta) \subseteq \mathbb{N}$ we denote the set containing all indices corresponding to a configuration from Conf_C in a run $\vartheta \in \mathsf{Conf}^\omega$ which is formally defined as follows.

Definition B.49.1 (Indices for Conf_C)**.** Let $\vartheta = c_0\, c_1 \cdots \in \mathsf{Conf}^\omega$. Then we define

$$I(\vartheta) = \{i \in \mathbb{N} \mid c_i \in \mathsf{Conf}_C\}.$$

The k-th index (for $k > 0$) is then given by:

$$I_k(\vartheta) = \begin{cases} \inf(\mathrm{cuts}^{k-1}(I(\vartheta))), & \text{if } \mathrm{cuts}^{k-1}(I(\vartheta)) \neq \emptyset \\ 0, & \text{otherwise,} \end{cases}$$

i.e., $I_k(\vartheta)$ is the k-th-smallest element of $I(\vartheta)$ (see Def. B.45.2).

As usual, for some finite path f we define $I_k(f) = i_k \in \mathbb{N}$ iff for all runs $\vartheta = f \cdots$ we have $I_k(\vartheta) = i_k$ (and $\bot$ otherwise).

It is worth noting that in any admissible run ϑ we have $t_0 = t_{\text{in}}$, i.e., if $I(\vartheta)$ is non-empty, its minimal element is *positive*.

Moreover, we can now define the function $S(x)_n$ mapping a run ϑ to the maximal absolute value that x takes in the first n configurations from Conf_C and the function $S(x)_0$ returning the size immediately before entering the SCC C.

Definition B.49.2 ($S(x)_0, S(x)_n$ for Runs)**.** For a run $\vartheta = (\ell_0, t_0, s_0) \cdots$ and $n \in \mathbb{N}_{\geq 1}$ we define:

$$S(x)_0(\vartheta) = \begin{cases} \left| s_{I_1(\vartheta)-1}(x) \right|, & \text{if } I_1(\vartheta) > 0 \\ 0, & \text{otherwise} \end{cases}$$

$$S(x)_n(\vartheta) = \begin{cases} \max\left\{ S_{n-1}(x)(\vartheta), \left| s_{I_n(\vartheta)}(x) \right| \right\}, & \text{if } I_n(\vartheta) > 0 \\ S_{n-1}(x)(\vartheta), & \text{otherwise} \end{cases}$$

By construction, for every run ϑ we have

$$S(x)_n(\vartheta) \leq S_{n+1}(x)(\vartheta) \tag{B.49.2}$$

and

$$\mathcal{S}_C(x)(\vartheta) \leq \lim_{n \to \omega} S(x)_n(\vartheta). \tag{B.49.3}$$

Note that in general we do not have equality in (B.49.3) since $\mathcal{S}_C(x)$ does *not* take the value *before* entering Conf_C into account.

As usual, we generalize the just defined random variables to finite paths.

Definition B.49.3 ($S(x)_n$ for Finite Paths)**.** The functions $S(x)_n$ for $n \in \mathbb{N}$ are generalized from runs to finite paths $f \in \mathsf{Conf}^+$ as follows:

$$S(x)_n(f) = \begin{cases} z, & \text{if } S(x)_n(\vartheta) = z \neq 0 \text{ for all runs } \vartheta = f \cdots \\ \bot, & \text{otherwise} \end{cases}$$

for all $n \in \mathbb{N}$.

We now define a filtration $(\mathcal{F}_n)_{n \in \mathbb{N}}$ such that the stochastic process $(S(x)_n)_{n \in \mathbb{N}}$ becomes adapted to this filtration. However, there is one thing which we have to keep in mind when doing so.

By Def. 17.4.5, the set Conf_C can only be entered once. So if in an *admissible* run ϑ a configuration $c' \in \mathsf{Conf}_C$ is followed by some configuration $c \in \mathsf{Conf}_{\neg C}$, we know that in the remainder of ϑ there cannot occur another configuration from Conf_C afterwards. Hence, in an admissible run ϑ, the part related to executions in Conf_C is an infix of ϑ. So, we now distinguish the runs by the "shape" of the infix belonging to the first execution in Conf_C (remember that admissibility is related to a probability measure which we do not know yet). Therefore, for some $n > 0$ and $f = c_1 \cdots c_n \in \mathsf{Conf}^n$ consider the set Inf_f defined by

$$\mathrm{Inf}_f = \left\{ \vartheta \in \mathsf{Conf}^\omega \;\middle|\; \vartheta = \cdots c_1 \cdots c_n \cdots \right\}$$

i.e., the set Inf_f consists exactly of the runs in which f occurs as an infix.

However, taking $f \neq f' \in \mathsf{Conf}_C^n$ still does *not* yield $\mathrm{Inf}_f \cap \mathrm{Inf}_{f'} = \emptyset$. The problem that arises here is that whereas in admissible runs Conf_C is only entered once, there are still non-admissible runs in which Conf_C can be entered multiple times which then are contained in the intersection above. Furthermore, this intersection contains (for example) $\mathrm{Inf}_{ff'}$, which can also contain admissible runs. Nevertheless, we are only interested in the *first* occurrence of Conf_C, i.e., for $f = c_1 \cdots c_n \in \mathsf{Conf}_C^n$ we are interested in

$$\mathrm{Inf}_f^C = \left\{ \vartheta \in \mathrm{Inf}_f \mid c_i = \vartheta[I_1(\vartheta) + i - 1] \text{ for } 1 \leq i \leq n \right\} \tag{B.49.4}$$

Then, for $f \neq f' \in \mathsf{Conf}_C^n$ we have indeed $\mathrm{Inf}_f^C \cap \mathrm{Inf}_{f'}^C = \emptyset$. Thus, it seems natural to define $\mathcal{F}_n$ as the σ-field generated by all Inf_f^C for $f \in \mathsf{Conf}_C^n$ for some fixed $n > 0$.

However, in this case we ignore that the complement of $\biguplus_{f \in \mathsf{Conf}_C^n} \mathrm{Inf}_f^C$ is more fine-grained since it does not only contain those runs where Conf_C is not visited at all but it also contains the runs in which the infix belonging to the first occurrence of Conf_C has a length $0 < k < n$. Hence, we also include these runs by adapting our definition of Inf_f^C. Moreover, the definition of Inf_f^C in (B.49.4) also has the drawback that we do *not* know the configuration before entering Conf_C, i.e., we cannot determine the value of $S(x)_0$ and thus, we cannot determine the value of any of the $S(x)_n$. To abbreviate notation, we first define

$$A_{\neg C} = \mathsf{Conf}_{\neg C}^{\omega},$$

i.e., $A_{\neg C}$ is the set of runs that never visit Conf_C.

Definition B.49.4 (Prefix Sets for Conf_C)**.** Let $c_0 \in \mathsf{Conf}_{\neg C}$ and let $f = c_1 \cdots c_n \in \mathsf{Conf}_C^k \, \mathsf{Conf}_{\neg C}^{n-k}$ for some $0 < k \leq n$. We then define

$$\mathrm{Inf}_{c_0 f}^C = \left\{ \vartheta \in \mathrm{Inf}_{c_0 f} \mid c_i = \vartheta[I_1(\vartheta) + i - 1] \text{ for } 0 \leq i \leq n \right\}$$

Intuitively, $\mathrm{Inf}_{c_0 f}^C$ is the set of runs where the first visit of Conf_C is entered from c_0 and right after c_0 the infix part f occurs. Now we have the right concepts at hand to define the filtration.

Definition B.49.5 (Filtration for Size Process)**.** For $n > 0$ we define

$$\mathcal{F}_n = \left\langle \left\{ \mathrm{Inf}_{c_0 f}^C \mid f \in \mathsf{Conf}_C^k \, \mathsf{Conf}_{\neg C}^{n-k}, 0 < k \leq n \wedge c_0 \in \mathsf{Conf}_{\neg C} \right\} \cup \{A_{\neg C}\} \right\rangle_{\sigma}$$

and

$$\mathcal{F}_0 = \left\langle \left\{ \biguplus_{c_1 \in \mathsf{Conf}_C} \mathrm{Inf}^C_{c_0 c_1} \;\middle|\; c_0 \in \mathsf{Conf}_{\neg C} \right\} \cup \{A_{\neg C}\} \right\rangle_\sigma .$$

In this way, whenever $f, f' \in \mathsf{Conf}_C^k \, \mathsf{Conf}_{\neg C}^{n-k}$ and $c_0, c_0' \in \mathsf{Conf}$ with $(f, c_0) \neq (f', c_0')$ we have $\mathrm{Inf}^C_{c_0 f} \cap \mathrm{Inf}^C_{c_0' f'} = \emptyset$. Furthermore, we have for any $n > 0$

$$\biguplus_{k=1}^{n} \; \biguplus_{f \in \mathsf{Conf}_C^k \, \mathsf{Conf}_{\neg C}^{n-k}} \mathrm{Inf}^C_f = \mathsf{Conf}^\omega \setminus A_{\neg C}.$$

For $n \in \mathbb{N}$, the σ-field $\mathcal{F}_n$ captures exactly the information that we need to evaluate $S_n(x)$ on a given ϑ. Intuitively, two runs $\vartheta_1, \vartheta_2 \in \mathsf{Conf}^\omega$ are distinguishable in $\mathcal{F}_n$ for $n > 0$, if they do not coincide on the infix of length $n+1$ starting from the point directly before entering Conf_C. Here, $\mathcal{F}_0$ is a special case: it can only distinguish between two runs whenever their first visits of Conf_C are entered from different configurations.

Lemma B.49.6 ($(S(x)_n)_{n\in\mathbb{N}}$ is Adapted to the Filtration $(\mathcal{F}_n)_{n\in\mathbb{N}}$)**.** *Consider $(\mathcal{F}_n)_{n\in\mathbb{N}}$ as defined in Def. B.49.5.*

(i) The sequence of σ-fields $(\mathcal{F}_n)_{n\in\mathbb{N}}$ forms a filtration of $\mathcal{F}$.

(ii) The random variable $S(x)_n$ is $\mathcal{F}_n$-measurable.

Proof. We prove the two statements separately.

(i) Obviously, it is enough to prove $\mathcal{F}_n \subseteq \mathcal{F}_{n+1}$ for all $n > 0$ since the case $n = 0$ is trivial by construction. It is enough to prove that each of the generators given in Def. B.49.5 for $\mathcal{F}_n$ is contained in $\mathcal{F}_{n+1}$. By construction, $A_{\neg C} \in \mathcal{F}_{n+1}$. So, let $c_1 \cdots c_n \in \mathsf{Conf}_C^k \, \mathsf{Conf}_{\neg C}^{n-k}$ for some $0 < k \leq n$ and $c_0 \in \mathsf{Conf}_{\neg C}$. Then we have

$$\mathrm{Inf}^C_{c_0 f} = \biguplus_{c \in \mathsf{Conf}} \mathrm{Inf}^C_{c_0 f c} \in \mathcal{F}_{n+1},$$

so we indeed have $\mathcal{F}_n \subseteq \mathcal{F}_{n+1}$.

(ii) To prove that $S(x)_n$ is $\mathcal{F}_n$-measurable we proceed as follows. First of all, $S(x)_n$ is a discrete random variable, i.e., its image is countable.

Secondly, if $n = 0$, for any $z \in \mathbb{N}$ we have

$$S(x)_0^{-1}(\{z\}) = \biguplus_{\substack{(_,_,s_0)=c_0 \in \mathsf{Conf}_{\neg C}, \\ |s_0(x)|=z}} \left(\biguplus_{c_1 \in \mathsf{Conf}_C} \mathrm{Inf}^C_{c_0 c_1} \right) \in \mathcal{F}_0.$$

Thirdly, to evaluate $S(x)_n$ for $n > 0$ on some $\vartheta \in \mathsf{Conf}^\omega$ we need to know the states in the run which occur at the positions $I_1(\vartheta) - 1, I_1(\vartheta), \ldots, I_1(\vartheta) + n - 1$. But this is exactly the information provided by $\mathcal{F}_n$. So, we have for $z \in \mathbb{N}_{\geq 1}$

$$\begin{aligned} &(S(x)_n)^{-1}(\{z\}) \\ &= \biguplus_{c_0=(_,_,s_0)\in \mathsf{Conf}_{\neg C}} \biguplus_{k=1}^{n} \biguplus_{\substack{f=(_,_,s_1)\cdots(_,_,s_n)\in \mathsf{Conf}_C^k \mathsf{Conf}_{\neg C}^{n-k}, \\ \max\{|s_0(x)|,\ldots,|s_k(x)|\}=z}} \mathrm{Inf}^C_{c_0 f} \in \mathcal{F}_n. \end{aligned}$$

□

Still, in this section we want to over-approximate the expected value of $\mathcal{S}_C(x)$. We do this by using the stochastic process $(S(x)_n)_{n\in\mathbb{N}}$ and its adaptedness to the filtration $(\mathcal{F}_n)_{n\in\mathbb{N}}$.

In (B.49.2) we have seen that the $S(x)_n$ form a monotonic sequence whose limit over-approximates $\mathcal{S}_C(x)$ (see (B.49.3)). So, to achieve our goal, we use the Monotone Convergence Theorem Cor. A.5.1. Using this result, from (B.49.3) we directly obtain the following corollary for our process $(S(x)_n)_{n\in\mathbb{N}}$.

Corollary B.49.7 (Over-Approximating the Expected Value of $\mathcal{S}_C(x)$)**.** *Let $s_0 \in \Sigma$ and let $\mathfrak{s} \in \mathsf{Scheds}$ be a scheduler. Then we have*

$$\mathbb{E}^{pt}_{s_0,\mathfrak{s}}(\mathcal{S}_C(x)) \leq \lim_{n\longrightarrow\omega} \mathbb{E}^{pt}_{s_0,\mathfrak{s}}(S(x)_n).$$

From now on, we fix $s_0 \in \Sigma$ and $\mathfrak{s} \in \mathsf{Scheds}$. We now over-approximate the expected value $\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S(x)_n)$ in such a way that we can over-approximate the limit $\lim_{n\to\omega} \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S(x)_n)$. Since $S(x)_n$ is $\mathcal{F}_n$-measurable, we get the following relation

between $S(x)_n$ and $S(x)_{n+1}$ for all $n \geq 0$:

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S(x)_{n+1}\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S(x)_{n+1} \mid \mathcal{F}_n\right)\right) && \text{(by Lemma A.6.4)}\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S(x)_{n+1} + S(x)_n - S(x)_n \mid \mathcal{F}_n\right)\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S(x)_n \mid \mathcal{F}_n\right) + \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S(x)_{n+1} - S(x)_n \mid \mathcal{F}_n\right)\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S(x)_n + \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S(x)_{n+1} - S(x)_n \mid \mathcal{F}_n\right)\right). && \text{(B.49.5)}
\end{aligned}$$

The last step is due to Thm. A.6.5 (ii) as $S(x)_n$ is $\mathcal{F}_n$-measurable by Lemma B.49.6 (ii).

So, to begin with, we over-approximate $\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S(x)_{n+1} - S(x)_n \mid \mathcal{F}_n\right)$. Note that this conditional expected value exists since $S(x)_{n+1} \geq S(x)_n$, i.e., $S(x)_{n+1} - S(x)_n$ only takes non-negative values (see Lemma A.6.2).

To do so, we introduce a sequence of random variables. It approximates the expected change of x that can happen when entering the $(n+1)$-th configuration of C during the run $\vartheta \in \mathsf{Conf}^\omega$ for $n \in \mathbb{N}$. We define

$$\begin{aligned}
&\left(\sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0|\left(\mathrm{size}^\alpha\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\right)(\vartheta)\\
&= \begin{cases}
\sum_{\alpha=(g,_,x)\in C} |s_0|\left(\mathrm{size}^\alpha\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right), & \text{if } \vartheta = (_,_,s_0)\cdots, I_{n+1}(\vartheta) > 0,\\
 & \vartheta[I_{n+1}(\vartheta)] = (_,t,_) \text{ with } t \in g\\
0, & \text{otherwise}
\end{cases}
\end{aligned}$$

Using this definition we can state the following lemma.

Lemma B.49.8. *For any $n \in \mathbb{N}$ we have*

$$\mathbb{E}^{pt}_{s_0,\mathfrak{s}}\left(S(x)_{n+1} - S(x)_n \mid \mathcal{F}_n\right) \leq \mathbb{E}^{pt}_{s_0,\mathfrak{s}}\left(\sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0|\left(\mathrm{size}^\alpha\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right) \,\middle|\, \mathcal{F}_n\right)$$

Proof. First note that the conditional expectation

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0|\left(\mathrm{size}^\alpha\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right) \,\middle|\, \mathcal{F}_n\right)$$

is $\mathcal{F}_n$-measurable by definition. We prove the result by analyzing the cases $n = 0$ and $n > 0$ separately.

Case $n = 0$: By Cor. A.6.3, we only need to consider the generators of $\mathcal{F}_0$. Here, it suffices to consider only the sets

$$A_{c_0} = \biguplus_{c_1 \in \mathsf{Conf}_C} \mathrm{Inf}^C_{c_0 c_1}$$

for some fixed $(_, _, s) = c_0 \in \mathsf{Conf}_{\neg C}$ as both sides are zero on $A_{\neg C}$. Moreover, w.l.o.g. we can assume that A_{c_0} is not a null set w.r.t. the chosen initial state s_0 and scheduler $\mathfrak{s}$, i.e., $\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(A_{c_0}\right) > 0$. We have to prove

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left(S(x)_1 - S(x)_0\right) \cdot \left[A_{c_0}\right]\right) \leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\alpha=(g_{I_1},_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}\left(\alpha\right)\right)\right) \,\middle|\, \mathcal{F}_0\right) \cdot \left[A_{c_0}\right]\right).$$

By the definition of conditional expected values (Def. 14.4.24), $A_{c_0} \in \mathcal{F}_0$ implies

$$\begin{aligned}&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\alpha=(g_{I_1},_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}\left(\alpha\right)\right)\right) \,\middle|\, \mathcal{F}_0\right) \cdot \left[A_{c_0}\right]\right)\\ &= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left(\sum_{\alpha=(g_{I_1},_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}\left(\alpha\right)\right)\right)\right) \cdot \left[A_{c_0}\right]\right).\end{aligned}$$

Thus, we now prove that

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left(S(x)_1 - S(x)_0\right) \cdot \left[A_{c_0}\right]\right) \leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left(\sum_{\alpha=(g_{I_1},_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}\left(\alpha\right)\right)\right)\right) \cdot \left[A_{c_0}\right]\right).$$

We have

$$\begin{aligned}&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left(S(x)_1 - S(x)_0\right) \cdot \left[A_{c_0}\right]\right)\\ &= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left(S(x)_1 - S(x)_0\right) \cdot \left[\biguplus_{c_1 \in \mathsf{Conf}_C} \mathrm{Inf}^C_{c_0 c_1}\right]\right)\\ &= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{c_1 \in \mathsf{Conf}_C} \left(S(x)_1 - S(x)_0\right) \cdot \left[\mathrm{Inf}^C_{c_0 c_1}\right]\right)\end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{c_1=(_,_,s_1)\in \mathsf{Conf}_C} (S(x)_1 - S(x)_0)\cdot\left[\mathrm{Inf}^C_{c_0c_1}\right]\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\substack{c_1=(_,_,s_1)\in \mathsf{Conf}_C,\\ |s_1(x)|\geq|s(x)|}} (S(x)_1 - S(x)_0)\cdot\left[\mathrm{Inf}^C_{c_0c_1}\right]\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\substack{c_1=(_,_,s_1)\in \mathsf{Conf}_C,\\ |s_1(x)|\geq|s(x)|}} (|s_1(x)| - |s(x)|)\cdot\left[\mathrm{Inf}^C_{c_0c_1}\right]\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\substack{c_1=(_,_,s_1)\in \mathsf{Conf}_C,\\ |s_1(x)|\geq|s(x)|}} (||s_1(x)| - |s(x)||)\cdot\left[\mathrm{Inf}^C_{c_0c_1}\right]\right)\\
&\leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\substack{c_1=(_,_,s_1)\in \mathsf{Conf}_C,\\ |s_1(x)|\geq|s(x)|}} (|s_1(x) - s(x)|)\cdot\left[\mathrm{Inf}^C_{c_0c_1}\right]\right)\\
&= \sum_{\substack{c_1=(_,_,s_1)\in \mathsf{Conf}_C,\\ |s_1(x)|\geq|s(x)|}} (|s_1(x) - s(x)|)\cdot\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[\mathrm{Inf}^C_{c_0c_1}\right]\right)\\
&= \sum_{\substack{c_1=(_,_,s_1)\in \mathsf{Conf}_C,\\ |s_1(x)|\geq|s(x)|}} (|s_1(x) - s(x)|)\cdot\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Inf}^C_{c_0c_1}\right)
\end{aligned}
$$

Now we would like to "split" the set $\mathrm{Inf}^C_{c_0c_1}$. We are only interested in runs where c_0 is the configuration directly before entering Conf_C. However, if we know that a run ϑ satisfies $\vartheta[I_1(\vartheta)-1] = c_0$ it could still be the case that c_0 occurs multiple times in ϑ and, e.g., the third or fourth occurrence leads into Conf_C. Thus, we now explain why this cannot happen in any admissible run before we continue with the proof.

So let ϑ be an admissible run which enters C and let $\vartheta[I_1(\vartheta)-1] = c_0 = (\ell, t, s)$ such that ϑ contains multiple occurrences of c_0. Moreover, let $\vartheta[I_1(\vartheta)] = c_1 = (\ell', t_1, _)$. By definition of the scheduler, we have $\mathfrak{s}(c_0) = (g_1, _)$, where $t_1 \in g_1$. Since from c_0 we can visit C but do not need to, there are locations $\ell' \neq \ell''$ such that $(g_1, \ell', x) \in C$ for some variable x but $(g_1, \ell'', y) \notin C$ for every $y \in \mathcal{PV}$. Since $(g_1, \ell', x) \in C$ and C is an

SCC, there is some general transition g_2 reachable from g_1 (in possibly several steps) such that after g_2 the general transition g_1 can be executed again. Thus, ℓ is a target location of g_2. Hence, there is an edge from (g_2, ℓ, x) to (g_1, ℓ', x) in the general result variable graph. Moreover, there is also an edge from (g_2, ℓ, x) to (g_1, ℓ'', x) since g_2 can be executed before g_1. So there is a path from (g_1, ℓ', x) via (g_2, ℓ, x) to (g_1, ℓ'', x) in the graph.

Since c_0 occurs multiple times in the run before C is entered, there is also a general transition g_3 with target location ℓ which is reachable from g_1 when ending with g_1 in ℓ''. Thus, there is some path from (g_1, ℓ'', x) via (g_3, ℓ, x) to (g_1, ℓ', x). Combining both parts, there is a path from (g_1, ℓ', x) to (g_1, ℓ'', x) in the general result variable graph and vice versa, i.e., $(g_1, \ell'', x) \in C$, which is a contradiction. Therefore, c_0 can occur at most once in any admissible run.

So, we can now continue our reasoning from above. Again, g_1 is the unique general transition which the scheduler chooses in c_0, i.e., $\mathfrak{s}(c_0) = (g_1, _)$.

$$
\begin{aligned}
&\sum_{\substack{c_1=(_,_,s_1)\in \mathsf{Conf}_C,\\ |s_1(x)|\geq|s(x)|}} (|s_1(x) - s(x)|) \cdot \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Inf}^{C}_{c_0 c_1}\right) \\
= &\sum_{\substack{c_1=(_,_,s_1)\in \mathsf{Conf}_C,\\ |s_1(x)|\geq|s(x)|}} (|s_1(x) - s(x)|) \cdot p^{\mathrm{pt}}_{c_0,\mathfrak{s}}(c_1) \cdot \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(A_{c_0}) \\
= &\left(\sum_{\substack{c_1=(_,_,s_1)\in \mathsf{Conf}_C,\\ |s_1(x)|\geq|s(x)|}} (|s_1(x) - s(x)|) \cdot p^{\mathrm{pt}}_{c_0,\mathfrak{s}}(c_1)\right) \cdot \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(A_{c_0}) \\
= &\left(\sum_{\substack{\ell\in\mathcal{L},\\ c_1=(\ell,_,s_1)\in \mathsf{Conf}_C,\\ |s_1(x)|\geq|s(x)|}} (|s_1(x) - s(x)|) \cdot p^{\mathrm{pt}}_{c_0,\mathfrak{s}}(c_1)\right) \cdot \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(A_{c_0}) \\
\leq &\sum_{\ell\in\mathcal{L}} \left(\sum_{\substack{c_1=(\ell,t,s_1)\in \mathsf{Conf}_C,\\ t\in g_1}} p^{\mathrm{pt}}_{c_0,\mathfrak{s}}(c_1) \cdot (|s_1(x) - s(x)|)\right) \cdot \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(A_{c_0}) \qquad (\dagger) \\
\leq &\left(\sum_{\alpha=(g_1,_,x)\in C} |s_0| \left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\right) \cdot \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(A_{c_0}) \qquad \text{(by Thm. 17.4.3)}
\end{aligned}
$$

$$= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left(\sum_{\alpha=(g_{I_1},_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\right)\cdot\left[A_{c_0}\right]\right) \qquad \text{(by definition)}$$

Note that in (†) we have used that the general transition g_1 which is used to continue c_0 is unique. This proves the case $n = 0$.

Case $n > 0$: Since we have constructed $\mathcal{F}_n$ using a generating system of countably many atoms which cover the sample space Conf^{ω}, by Cor. A.6.3 it is sufficient to only consider the generators of $\mathcal{F}_n$, similar to the case $n = 0$. Thus, let $0 < k \leq n$, $c_0 \in \mathsf{Conf}_{\neg C}$, and $c_1 \cdots c_n = f \in \mathsf{Conf}_C^k\,\mathsf{Conf}_{\neg C}^{n-k}$. We have to prove that

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left((S(x)_{n+1} - S(x)_n)\cdot\left[\mathrm{Inf}^C_{c_0 f}\right]\right) \leq$$
$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\,\middle|\,\mathcal{F}_n\right)\cdot\left[\mathrm{Inf}^C_{c_0 f}\right]\right).$$

By the definition of conditional expected values (Def. 14.4.24), $\mathrm{Inf}^C_{c_0 f} \in \mathcal{F}_n$ implies

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\,\middle|\,\mathcal{F}_n\right)\cdot\left[\mathrm{Inf}^C_{c_0 f}\right]\right)$$
$$= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left(\sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\right)\cdot\left[\mathrm{Inf}^C_{c_0 f}\right]\right).$$

Hence, it suffices to prove that

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left((S(x)_{n+1} - S(x)_n)\cdot\left[\mathrm{Inf}^C_{c_0 f}\right]\right)$$
$$\leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left(\sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\right)\cdot\left[\mathrm{Inf}^C_{c_0 f}\right]\right).$$

First of all, we left out the generator $A_{\neg C}$ since here the claim holds trivially, as both sides are zero. Secondly, whenever $k < n$, then $S(x)_{n+1} - S(x)_n$ is zero on all admissible runs in $\mathrm{Inf}^C_{c_0 f}$, i.e., the left-hand side of the inequation is zero and the claim holds trivially. So, w.l.o.g., we can assume that $n = k$. Let $c_i = (_, _, s_i')$ for $0 \leq i \leq n$. Due to the chosen scheduler $\mathfrak{s}$, the transition that is executed to continue in an admissible

run from c_n is an element of a *unique* general transition g_{n+1}. Furthermore, if a run ϑ leaves Conf_C after c_n, then $S(x)_{n+1}(\vartheta) - S(x)_n(\vartheta)$ is zero as well.

$$
\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left((S(x)_{n+1} - S(x)_n)\cdot\left[\mathrm{Inf}^C_{c_0 f}\right]\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left((S(x)_{n+1} - S(x)_n)\cdot\left[\biguplus_{c_{n+1}\in\mathsf{Conf}} \mathrm{Inf}^C_{c_0 f c_{n+1}}\right]\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{c_{n+1}\in\mathsf{Conf}} (S(x)_{n+1} - S(x)_n)\cdot\left[\mathrm{Inf}^C_{c_0 f c_{n+1}}\right]\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{c_{n+1}=(_,_,s'_{n+1})\in\mathsf{Conf}_C} (S(x)_{n+1} - S(x)_n)\cdot\left[\mathrm{Inf}^C_{c_0 f c_{n+1}}\right]\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\substack{c_{n+1}=(_,_,s'_{n+1})\in\mathsf{Conf}_C,\\ |s'_{n+1}(x)|\geq\max_{0\leq i\leq n}|s'_i(x)|}} (S(x)_{n+1} - S(x)_n)\cdot\left[\mathrm{Inf}^C_{c_0 f c_{n+1}}\right]\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\substack{c_{n+1}=(_,_,s'_{n+1})\in\mathsf{Conf}_C,\\ |s'_{n+1}(x)|\geq\max_{0\leq i\leq n}|s'_i(x)|}} \left(|s'_{n+1}(x)| - \max_{0\leq i\leq n}|s'_i(x)|\right)\cdot\left[\mathrm{Inf}^C_{c_0 f c_{n+1}}\right]\right)\\
&\leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\substack{c_{n+1}=(_,_,s'_{n+1})\in\mathsf{Conf}_C,\\ |s'_{n+1}(x)|\geq\max_{0\leq i\leq n}|s'_i(x)|}} \left(|s'_{n+1}(x)| - |s'_n(x)|\right)\cdot\left[\mathrm{Inf}^C_{c_0 f c_{n+1}}\right]\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\substack{c_{n+1}=(_,_,s'_{n+1})\in\mathsf{Conf}_C,\\ |s'_{n+1}(x)|\geq\max_{0\leq i\leq n}|s'_i(x)|}} \left(\big||s'_{n+1}(x)| - |s'_n(x)|\big|\right)\cdot\left[\mathrm{Inf}^C_{c_0 f c_{n+1}}\right]\right)\\
&\leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\substack{c_{n+1}=(_,_,s'_{n+1})\in\mathsf{Conf}_C,\\ |s'_{n+1}(x)|\geq\max_{0\leq i\leq n}|s'_i(x)|}} \left(|s'_{n+1}(x) - s'_n(x)|\right)\cdot\left[\mathrm{Inf}^C_{c_0 f c_{n+1}}\right]\right)
\end{aligned}
$$

$$
\begin{aligned}
&= \sum_{\substack{c_{n+1}=(_,_,s'_{n+1})\in \mathsf{Conf}_C,\\ |s'_{n+1}(x)|\geq \max_{0\leq i\leq n}|s'_i(x)|}} \left(\left|s'_{n+1}(x)-s'_n(x)\right|\right)\cdot \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[\mathrm{Inf}^C_{c_0 f c_{n+1}}\right]\right)\\
&= \sum_{\substack{c_{n+1}=(_,_,s'_{n+1})\in \mathsf{Conf}_C,\\ |s'_{n+1}(x)|\geq \max_{0\leq i\leq n}|s'_i(x)|}} \left(\left|s'_{n+1}(x)-s'_n(x)\right|\right)\cdot \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Inf}^C_{c_0 f c_{n+1}}\right)\\
&= \left(\sum_{\substack{c_{n+1}=(_,_,s'_{n+1})\in \mathsf{Conf}_C,\\ |s'_{n+1}(x)|\geq \max_{0\leq i\leq n}|s'_i(x)|}} \left(\left|s'_{n+1}(x)-s'_n(x)\right|\right)\cdot p^{\mathrm{pt}}_{c_n,\mathfrak{s}}(c_{n+1})\right)\cdot \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Inf}^C_{c_0 f}\right)\\
&\leq \left(\sum_{\alpha=(g_{n+1},_,x)\in C}\left|s'_n\right|\left(C\mathcal{B}_{\mathbb{E}}(\alpha)\right)\right)\cdot \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Inf}^C_{c_0 f}\right) \qquad \text{(by Thm. 17.4.3)}\\
&\leq \left(\sum_{\alpha=(g_{n+1},_,x)\in C}|s_0|\left(\mathrm{size}^{\alpha}\left(C\mathcal{B}_{\mathbb{E}}(\alpha)\right)\right)\right)\cdot \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Inf}^C_{c_0 f}\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left(\sum_{\alpha=(g_{n+1},_,x)\in C}|s_0|\left(\mathrm{size}^{\alpha}\left(C\mathcal{B}_{\mathbb{E}}(\alpha)\right)\right)\right)\cdot\left[\mathrm{Inf}^C_{c_0 f}\right]\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left(\sum_{\alpha=(g_{I_{n+1}},_,x)\in C}|s_0|\left(\mathrm{size}^{\alpha}\left(C\mathcal{B}_{\mathbb{E}}(\alpha)\right)\right)\right)\cdot\left[\mathrm{Inf}^C_{c_0 f}\right]\right),
\end{aligned}
$$

which proves the case $n > 0$. □

Moreover, we can over-approximate the expected value

$$
\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\alpha=(g_{I_{n+1}},_,x)\in C}|s_0|\left(\mathrm{size}^{\alpha}\left(C\mathcal{B}_{\mathbb{E}}(\alpha)\right)\right)\right).
$$

For this expected value, we can ignore the set of non-admissible runs w.r.t. the chosen scheduler $\mathfrak{s}$ and the chosen initial state s_0 since this set has probability zero. Furthermore, we can ignore those runs, in which the random variable takes the value 0, i.e., especially those ones, which visit Conf_C less then $n+1$ times. Thus, to compute the expected value we only need to consider the set $\biguplus_{c_0\in \mathsf{Conf}_{\neg C},\, f\in \mathsf{Conf}_C^{n+1}} \mathrm{Inf}^C_{c_0 f}$ since

$\sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0| \left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)$ takes the value zero on any admissible run in the complement. Then we have

$$
\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0| \left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[\biguplus_{\substack{c_0\in \mathsf{Conf}_{\neg C},\\ f=c_1\cdots c_{n+1}\in \mathsf{Conf}_C^{n+1}}} \mathrm{Inf}^{C}_{c_0 f}\right]\cdot \sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0| \left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\right)\\
&= \sum_{\substack{c_0\in \mathsf{Conf}_{\neg C},\\ f=c_1\cdots c_{n+1}\in \mathsf{Conf}_C^{n+1}}} \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[\mathrm{Inf}^{C}_{c_0 f}\right]\cdot \sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0| \left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\right)\\
&= \sum_{\substack{c_0\in \mathsf{Conf}_{\neg C},\\ f=c_1\cdots c_{n+1}\in \mathsf{Conf}_C^{n+1}}} \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[\mathrm{Inf}^{C}_{c_0 f}\right]\cdot \sum_{\alpha=(g_{n+1},_,x)\in C} |s_0| \left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\right)\\
&= \sum_{\substack{c_0\in \mathsf{Conf}_{\neg C},\\ f=c_1\cdots c_{n+1}\in \mathsf{Conf}_C^{n+1}}} \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\left[\mathrm{Inf}^{C}_{c_0 f}\right]\right)\cdot \sum_{\alpha=(g_{n+1},_,x)\in C} |s_0| \left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\\
&= \sum_{\substack{c_0\in \mathsf{Conf}_{\neg C},\\ f=c_1\cdots c_{n+1}\in \mathsf{Conf}_C^{n+1}}} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Inf}^{C}_{c_0 f}\right)\cdot \sum_{\alpha=(g_{n+1},_,x)\in C} |s_0| \left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right) \qquad \text{(B.49.6)}
\end{aligned}
$$

Finally, we have all information at hand to prove Thm. 17.4.10.

Proof of Thm. 17.4.10. Let $n \geq 0$. Then using our results obtained so far in this subsection we have

$$
\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S(x)_{n+1}\right)\\
&= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S(x)_n + \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S(x)_{n+1} - S(x)_n \mid \mathcal{F}_n\right)\right) && \text{(by (B.49.5))}\\
&\leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S(x)_n + \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0| \left(\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right) \,\middle|\, \mathcal{F}_n\right)\right)\\
& && \text{(by Lemma B.49.8)}
\end{aligned}
$$

$$= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S(x)_n) + \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\sum_{\alpha=(g_{I_{n+1}},_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}(\mathcal{CB}_{\mathbb{E}}(\alpha))\right)\right) \quad \text{(by Lemma A.6.4)}$$

$$= \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S(x)_n) + \sum_{\substack{c_0\in \mathsf{Conf}_{\neg C},\\ f=c_1\cdots c_{n+1}\in \mathsf{Conf}^{n+1}_C}} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Inf}^C_{c_0 f}\right)\cdot \sum_{\alpha=(g_{n+1},_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}(\mathcal{CB}_{\mathbb{E}}(\alpha))\right)$$

(by (B.49.6))

Iterating this argument $(n+1)$-times then yields:

$$\begin{aligned}&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S(x)_{n+1})\\ \leq\ &\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S(x)_0)\\ &+\sum_{i=1}^{n+1}\sum_{\substack{c_0\in \mathsf{Conf}_{\neg C},\\ f=c_1\cdots c_i\in \mathsf{Conf}^{i}_C}} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Inf}^C_{c_0 f}\right)\cdot \sum_{\alpha=(g_i,_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}(\mathcal{CB}_{\mathbb{E}}(\alpha))\right). \qquad \text{(B.49.7)}\end{aligned}$$

Remember that with g_i we denote the general transition which leads to configuration c_i, i.e., $c_i = (t_i, _, _)$ and $t_i \in g_i$. This notation is used again in the next calculations.

Let us elaborate on

$$\sum_{i=1}^{n+1}\sum_{\substack{c_0\in \mathsf{Conf}_{\neg C},\\ f=c_1\cdots c_i\in \mathsf{Conf}^{i}_C}} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Inf}^C_{c_0 f}\right)\cdot \sum_{\alpha=(g_i,_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}(\mathcal{CB}_{\mathbb{E}}(\alpha))\right).$$

To increase readability of the following reasoning, we introduce a shorthand notation.

$$ch_{\Sigma}{}^{C}(g) = \begin{cases}\sum_{\alpha=(g,_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}(\mathcal{CB}_{\mathbb{E}}(\alpha))\right), & \text{if } g\in \mathcal{GT}_C\\ 0, & \text{otherwise}\end{cases}.$$

Then we have

$$\sum_{i=1}^{n+1}\sum_{\substack{c_0\in \mathsf{Conf}_{\neg C},\\ f=c_1\cdots c_i\in \mathsf{Conf}^{i}_C}} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Inf}^C_{c_0 f}\right)\cdot \sum_{\alpha=(g_i,_,x)\in C} |s_0|\left(\mathrm{size}^{\alpha}(\mathcal{CB}_{\mathbb{E}}(\alpha))\right)$$

$$= \sum_{i=1}^{n+1}\sum_{\substack{c_0\in \mathsf{Conf}_{\neg C},\\ f=c_1\cdots c_i\in \mathsf{Conf}^{i}_C}} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathrm{Inf}^C_{c_0 f}\right)\cdot ch_{\Sigma}{}^{C}(g_i)$$

$$= \sum_{i=1}^{n+1} \sum_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ f = c_1 \cdots c_i \in \mathsf{Conf}_C^i}} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \left(\biguplus_{c_{i+1}\cdots c_{n+1} \in \mathsf{Conf}^{n+1-i}} \mathrm{Inf}^C_{c_0 f c_{i+1}\cdots c_{n+1}} \right) \cdot {ch_\Sigma}^C(g_i)$$

$$= \sum_{i=1}^{n+1} \sum_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ f = c_1 \cdots c_i \in \mathsf{Conf}_C^i}} \sum_{c_{i+1}\cdots c_{n+1} \in \mathsf{Conf}^{n+1-i}} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \left(\mathrm{Inf}^C_{c_0 f c_{i+1}\cdots c_{n+1}} \right) \cdot {ch_\Sigma}^C(g_i)$$

$$= \sum_{i=1}^{n+1} \sum_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ f = c_1 \cdots c_{i-1} \in \mathsf{Conf}_C^{i-1}}} \sum_{c_i \cdots c_{n+1} \in \mathsf{Conf}^{n+1-(i-1)}} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \left(\mathrm{Inf}^C_{c_0 f c_i \cdots c_{n+1}} \right) \cdot {ch_\Sigma}^C(g_i)$$

$$({ch_\Sigma}^C(g_i) = 0 \text{ if } g_i \notin \mathcal{GT}_C)$$

$$= \sum_{i=1}^{n+1} \sum_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ c_1 \in \mathsf{Conf}_C}} \sum_{c_2 \cdots c_{n+1} \in \mathsf{Conf}^n} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \left(\mathrm{Inf}^C_{c_0 c_1 c_2 \cdots c_{n+1}} \right) \cdot {ch_\Sigma}^C(g_i)$$

(since Conf_C can only be entered once in an admissible run)

$$= \sum_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ c_1 \in \mathsf{Conf}_C}} \sum_{c_2 \cdots c_{n+1} \in \mathsf{Conf}^n} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \left(\mathrm{Inf}^C_{c_0 c_1 c_2 \cdots c_{n+1}} \right) \cdot \sum_{i=1}^{n+1} {ch_\Sigma}^C(g_i)$$

Since we fixed $c_1 \in \mathsf{Conf}_C$, for $f \neq f' \in \mathsf{Conf}^n$ the intersection $\mathrm{Inf}^C_{c_0 c_1 f} \cap \mathrm{Inf}^C_{c_0 c_1 f'}$ only contains non-admissible runs. This is the reason for the slight abuse of notation of Inf^C in the reasoning above which increases readability. Here, if for no $k \geq 0$ we have $f \in \mathsf{Conf}_C^k \mathsf{Conf}_{\neg C}^{n-k}$, the set Inf^C_f just contains non-admissible runs, i.e., it is a null set. Again, we assume that any configuration c_i has the form $(_, t_i, _)$. Hence, we conclude

$$\sum_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ c_1 \in \mathsf{Conf}_C}} \sum_{c_2 \cdots c_{n+1} \in \mathsf{Conf}^n} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \left(\mathrm{Inf}^C_{c_0 c_1 c_2 \cdots c_{n+1}} \right) \cdot \sum_{i=1}^{n+1} {ch_\Sigma}^C(g_i)$$

$$= \sum_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ c_1 \in \mathsf{Conf}_C,\\ c_2 \cdots c_{n+1} \in \mathsf{Conf}^n}} \left(\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathrm{Inf}^C_{c_0 \cdots c_{n+1}}) \cdot \sum_{g \in \mathcal{GT}_C} \sum_{\substack{1 \leq i \leq n+1,\\ t_i \in g}} {ch_\Sigma}^C(g) \right)$$

$$\leq \sum_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ c_1 \in \mathsf{Conf}_C,\\ c_2 \cdots c_{n+1} \in \mathsf{Conf}^n}} \left(\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathrm{Inf}^C_{c_0 \cdots c_{n+1}}) \cdot \sum_{g \in \mathcal{GT}_C} \inf \left\{ \mathcal{R}(g)(\vartheta) \;\middle|\; \vartheta \in \mathrm{Inf}^C_{c_0 \cdots c_{n+1}} \right\} \cdot {ch_\Sigma}^C(g) \right)$$

$$
\begin{aligned}
&= \sum_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ c_1 \in \mathsf{Conf}_C,\\ c_2 \cdots c_{n+1} \in \mathsf{Conf}^n}} \left(\sum_{g \in \mathcal{GT}_C} \left(\mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathrm{Inf}^C_{c_0 \cdots c_{n+1}}) \cdot \inf \left\{ \mathcal{R}(g)(\vartheta) \;\middle|\; \vartheta \in \mathrm{Inf}^C_{c_0 \cdots c_{n+1}} \right\} \right) \cdot {ch_\Sigma}^C(g) \right) \\
&\leq \sum_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ c_1 \in \mathsf{Conf}_C,\\ c_2 \cdots c_{n+1} \in \mathsf{Conf}^n}} \left(\sum_{g \in \mathcal{GT}_C} \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \left(\mathcal{R}(g) \cdot \left[\mathrm{Inf}^C_{c_0 \cdots c_{n+1}} \right] \right) \cdot {ch_\Sigma}^C(g) \right) \\
&= \sum_{g \in \mathcal{GT}_C} \sum_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ c_1 \in \mathsf{Conf}_C,\\ c_2 \cdots c_{n+1} \in \mathsf{Conf}^n}} \left(\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \left(\mathcal{R}(g) \cdot \left[\mathrm{Inf}^C_{c_0 \cdots c_{n+1}} \right] \right) \cdot {ch_\Sigma}^C(g) \right) \\
&= \sum_{g \in \mathcal{GT}_C} \left(\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \left(\mathcal{R}(g) \cdot \left[\biguplus_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ c_1 \in \mathsf{Conf}_C,\\ c_2 \cdots c_{n+1} \in \mathsf{Conf}^n}} \mathrm{Inf}^C_{c_0 \cdots c_{n+1}} \right] \right) \cdot {ch_\Sigma}^C(g) \right) \\
&\leq \sum_{g \in \mathcal{GT}_C} \left({ch_\Sigma}^C(g) \cdot \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{R}(g)) \right) \\
&= \sum_{g \in \mathcal{GT}_C} \left(\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{R}(g)) \cdot \sum_{\alpha=(g,_,x) \in C} |s_0| \left(\mathrm{size}^\alpha \left(\mathcal{CB}_{\mathbb{E}}(\alpha) \right) \right) \right)
\end{aligned}
\tag{B.49.8}
$$

Let us abbreviate $\sum_{(\beta,\alpha) \in \mathcal{GRVE},\, \beta \notin C,\, \alpha \in C,\, \beta = (_,_,x)} \mathcal{SB}_{\mathbb{E}}(\beta)$ by $\mathrm{size}^C_{\mathbb{E}}(x)$. The value $\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S(x)_0)$ is bounded by $|s_0| \left(\mathrm{size}^C_{\mathbb{E}}(x) \right)$ since $\mathcal{SB}_{\mathbb{E}}$ is a valid expected size bound. By combining our previous observations we get:

$$
\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S(x)_{n+1}) \\
&\leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S(x)_0) + \sum_{i=1}^{n+1} \sum_{\substack{c_0 \in \mathsf{Conf}_{\neg C},\\ f = c_1 \cdots c_i \in \mathsf{Conf}^i_C}} \mathbb{P}^{\mathrm{pt}}_{s_0,\mathfrak{s}} \left(\mathrm{Inf}^C_{c_0 f} \right) \cdot \sum_{\alpha=(g_i,_,x) \in C} |s_0| \left(\mathrm{size}^\alpha \left(\mathcal{CB}_{\mathbb{E}}(\alpha) \right) \right) \\
&\qquad\qquad \text{(by (B.49.7))} \\
&\leq \mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S(x)_0) + \sum_{g \in \mathcal{GT}_C} \left(\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(\mathcal{R}(g)) \cdot \sum_{\alpha=(g,_,x) \in C} |s_0| \left(\mathrm{size}^\alpha \left(\mathcal{CB}_{\mathbb{E}}(\alpha) \right) \right) \right) \\
&\qquad\qquad \text{(by (B.49.8))}
\end{aligned}
$$

$$\leq |s_0|\left(\mathrm{size}^C_{\mathbb{E}}(x)\right) + |s_0|\left(\sum_{g\in\mathcal{GT}_C}\left(\mathcal{RB}_{\mathbb{E}}(g)\cdot\sum_{\alpha=(g,_,x)\in C}\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\right)$$

(as $(\mathcal{RB}_{\mathbb{E}}, \mathcal{SB}_{\mathbb{E}})$ is an expected bound pair)

$$= |s_0|\left(\mathrm{size}^C_{\mathbb{E}}(x) + \sum_{g\in\mathcal{GT}_C}\left(\mathcal{RB}_{\mathbb{E}}(g)\cdot\sum_{\alpha=(g,_,x)\in C}\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\right)$$

So we have

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S(x)_{n+1}) \leq |s_0|\left(\mathrm{size}^C_{\mathbb{E}}(x) + \sum_{g\in\mathcal{GT}_C}\mathcal{RB}_{\mathbb{E}}(g)\cdot\sum_{\alpha=(g,_,x)\in C}\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right).$$

Hence, we also have

$$\lim_{n\longrightarrow\omega}\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}(S(x)_{n+1}) \leq |s_0|\left(\mathrm{size}^C_{\mathbb{E}}(x) + \sum_{g\in\mathcal{GT}_C}\mathcal{RB}_{\mathbb{E}}(g)\cdot\sum_{\alpha=(g,_,x)\in C}\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right).$$

By Cor. B.49.7 this means that

$$\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(S_C(x)\right) \leq |s_0|\left(\mathrm{size}^C_{\mathbb{E}}(x) + \sum_{g\in\mathcal{GT}_C}\mathcal{RB}_{\mathbb{E}}(g)\cdot\sum_{\alpha=(g,_,x)\in C}\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right).$$

So for any $\alpha' = (_, _, x) \in C$, (B.49.1) implies

$$\begin{aligned}
&\mathbb{E}^{\mathrm{pt}}_{s_0,\mathfrak{s}}\left(\mathcal{S}(\alpha')\right)\\
&\leq |s_0|\left(\mathrm{size}^C_{\mathbb{E}}(x) + \sum_{g\in\mathcal{GT}_C}\mathcal{RB}_{\mathbb{E}}(g)\cdot\sum_{\alpha=(g,_,x)\in C}\mathrm{size}^{\alpha}\left(\mathcal{CB}_{\mathbb{E}}(\alpha)\right)\right)\\
&= |s_0|\left(\mathcal{SB}'_{\mathbb{E}}(\alpha')\right)
\end{aligned}$$

As s_0 and $\mathfrak{s}$ have been chosen arbitrarily, $\mathcal{SB}'_{\mathbb{E}}$ is indeed an expected size bound. □

Eidesstattliche Erklärung

Ich, Marcel Hark, erkläre hiermit, dass diese Dissertation und die darin dargelegten Inhalte die eigenen sind und selbstständig, als Ergebnis der eigenen originären Forschung, generiert wurden.

Hiermit erkläre ich an Eides statt

(i) Diese Arbeit wurde vollständig oder größtenteils in der Phase als Doktorand dieser Fakultät und Universität angefertigt;

(ii) Sofern irgendein Bestandteil dieser Dissertation zuvor für einen akademischen Abschluss oder eine andere Qualifikation an dieser oder einer anderen Institution verwendet wurde, wurde dies klar angezeigt;

(iii) Wenn immer andere eigene- oder Veröffentlichungen Dritter herangezogen wurden, wurden diese klar benannt;

(iv) Wenn aus anderen eigenen- oder Veröffentlichungen Dritter zitiert wurde, wurde stets die Quelle hierfür angegeben. Diese Dissertation ist vollständig meine eigene Arbeit, mit der Ausnahme solcher Zitate;

(v) Alle wesentlichen Quellen von Unterstützung wurden benannt;

(vi) Wenn immer ein Teil dieser Dissertation auf der Zusammenarbeit mit anderen basiert, wurde von mir klar gekennzeichnet, was von anderen und was von mir selbst erarbeitet wurde;

(vii) Teile dieser Arbeit wurden zuvor veröffentlicht und zwar in:

[FHG20] F. Frohn, M. Hark und J. Giesl. “Termination of Polynomial Loops”. In: *Proc. SAS*. Lecture Notes in Computer Science Vol. 12389. 2020, S. 89–112.

[GGH19a] J. Giesl, P. Giesl und M. Hark. “Computing Expected Runtimes for Constant Probability Programs”. In: *Proc. CADE*. Lecture Notes in Computer Science Vol. 11716. 2019, S. 269–286.

[Har+20] M. Hark, B. L. Kaminski, J. Giesl und J.-P. Katoen. “Aiming Low is Harder: Induction for Lower Bounds in Probabilistic Program Verification”. In: *Proc. ACM Program. Lang.* 4.POPL (2020), 37:1–37:28.

[HFG20] M. Hark, F. Frohn und J. Giesl. “Polynomial Loops: Beyond Termination”. In: *Proc. LPAR*. EPiC Series in Computing Vol. 73. 2020, S. 279–297.

[MHG21b] F. Meyer, M. Hark und J. Giesl. “Inferring Expected Runtimes of Probabilistic Integer Programs Using Expected Sizes”. In: *Proc. TACAS (1)*. Lecture Notes in Computer Science Vol. 12651. 2021, S. 250–269.

Ein detaillierter Überblick über die Zusammenhänge dieser Arbeit mit den oben genannten Veröffentlichungen sowie eine Diskussion meiner Beiträge zu diesen Veröffentlichungen findet sich in Kapitel 2.

Aachen, den . Juni 2021 _____