New Results On Probabilistic Verification: Automata, Logic and Satisfiability

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

Souymodip Chakraborty

aus

Kolkata, India

Berichter: Prof. Dr. Ir. Dr. h. h. Joost-Pieter Katoen
Prof. Dr. Lijun Zhang

Tag der mündlichen Prüfung: 2019-05-27

Abstract

Probabilistic (or quantitative) verification is a branch of formal methods dealing with stochastic models and logic. Probabilistic models capture the behavior of randomized algorithms and other physical systems with certain uncertainty, whereas probabilistic logic expresses the quantitative measure on the probabilistic space defined by the models.

Most often, the formal techniques used in studying the behavior of these models and logic are not just mundane extension of its non-probabilistic counterparts. The complexity of these mathematical structures is surprisingly different. The thesis is an effort at improving our continued understanding of these models and logic.

We will begin by looking at few interesting formal representations of discrete stochastic models. Namely, we will address the parameter synthesis problem for parametric linear time temporal logic and model checking of convex Markov decision processes with open intervals.

The primary focus of the thesis is however on the satisfiability (or validity) problem of different probabilistic logics. This includes a bounded fragment of probabilistic logic and a simple quantitative (probabilistic) extension of mu-calculus. Decision procedures for the satisfiability problems are developed and a detailed complexity analysis of these problems is provided.

The study of automata has been very effective in understanding logic. We will look at the newly conceived notion of p-automata, which are a probabilistic extension of alternating tree automata. As we will see, probabilistic logic exhibits both non-deterministic and stochastic behavior. The semantics of p-automata are extended to capture non-determinism and hence model Markov decision processes.

Zusammenfassung

Probabilistische (oder quantitative) Verifikation ist ein Teilgebiet der Formalen Methoden, das sich mit stochastischen Modellen und Logiken beschätigt. Probabilistische Modelle beschreiben das Verhalten randomisierter Algorithmen und anderer physikalischer Systeme, die einer gewissen Unsicherheit unterworfen sind, während probabilistische Logik quantitative Maße über dem Wahrscheinlichkeitsraum eines solchen Modells spezifiziert.

In den allermeisten Fällen sind die formalen Techniken, die zur Analyse des Verhaltens probabilistischer Modelle und Logiken eingesetzt werden, mehr als einfache Erweiterungen ihrer nicht-quantitativen Varianten. Vielmehr weisen die resultierenden mathematischen Strukturen eine überraschende Komplexität auf. Das Ziel dieser Arbeit besteht darin, das Verständnis dieser Strukturen grundlegend zu verbessern.

Wir beginnen mit der Vorstellung einiger interessanter formaler Darstellungen diskreter stochastischer Modelle. Insbesondere werden wir uns dem Problem der Parametersynthese für parametrische Linearzeit-Temporallogik sowie des Model Checking für konvexe Markov-Entscheidungsprozesse mit offenen Intervallen widmen.

Der Schwerpunkt der Arbeit liegt auf dem Erfüllbarkeits- und Gültigkeitsproblem für verschiedene probabilistische Logiken. Letztere umfassen ein beschränktes Fragment der probabilistischen Logik sowie eine einfache quantitative (probabilistische) Erweiterung des mu-Kalküls. Wir entwickeln Entscheidungsprozeduren für die zugehörigen Erfüllbarkeitsprobleme und führen eine detaillierte Komplexitätsanalyse durch.

Automaten haben sich als ein äußerst effektives Hilfsmittel für das Verständnis von Logiken erwiesen. Wir führen das neue Konzept der p-Automaten ein, welche eine probabilistische Erweiterung alternierender Baumautomaten darstellen. Wie sich zeigen wird, weist probabilistische Logik sowohl nichtdeterministisches als auch stochastisches Verhalten auf. Daher wird die Semantik von p-Automaten um die Behandlung von Nichtdeterminismus erweitert, so dass auch Markov-Entscheidungsprozesse modelliert werden können.

For Priyanka,
without her I couldn't have finished this.

Acknowledgement

I thank Prof. Dr. Ir. Dr. h. c. Joost-Pieter Katoen for his continued supervision and guidance towards my doctoral studies. He was gracious to call me for research visit during my masters and subsequently offered me a position of research assistant at the chair of informatik 2 in RWTH Aachen university. He provided me with projects and financial support without which my thesis would not be possible.

My four years of stay at Aachen have been a quite memorable. I have grown not only as a researcher but also as person. It will definitely have an ever lasting impression on my life. During by little journey there have been many who accompanied me. I sincerely thank Friedrich Gretz and Viet Yen Nguyen to make me feel at home. I thank Friedrich, Christian, Arpit, Christoph, Harold, Stephen and many other colleagues and friends at the chair for always lending me an ear when I had things to discuss. I specially thank apl. Prof. Dr. Thomas Noll for translating my abstract.

My work was financially supported by the MoVeS (Modeling, Verification and control of complex Systems: From the foundations to power network applications) EU FP7 project SENSATION and the EU FP7 IRSES project MEALS and RWTH Aachen University, Germany.

One of the perks as a PhD student was traveling to various universities and conferences. I also thank, Prof. Dr. Nir Piterman for the interesting discussion we had on my visit to Oxford.

Publications

Following is a complete list of technical papers and journal. We will give a brief introduction to the contribution of these publications.

- **On the Satisfiability of Some Simple Probabilistic Logic. [23]**
  The paper solves the satisfiability problems for two important fragments of probabilistic CTL (PCTL). Namely, a bounded fragment of PCTL(called bounded PCTL) and an extension of the modal μ-calculus with probabilistic with probabilistic quantification over next-modalities (called PμTL). Firstly, the paper shows that bounded PCTL has small model property where the model size is independent from the probability bounds in the formula. An NEXP-TIME algorithm for deciding satisfiability of that the bounded PCTL is developed. The paper also establishes that the satisfiability problem of a simple sub-logic of bounded PCTL is PSPACE-complete.

  Secondly, the paper proves that PμTL too has a small model property and employs a decision procedure using 2 player parity games, showing that the satisfiability problem is decidable. These results have significant ramification as it goes to shows that PμTL and qualitative PCTL are incomparable. The paper also contrasts PCTL by showing that every satisfiable PμTL formula has a rational model, i.e. a model with only rational probabilities.

- **P-Automata for Markov Decision Processes. [22]**
  P-automata provide an automata-theoretic approach to probabilistic verification. Similar to alternating tree automata which accepts labeled transition systems, p-automata accept labeled Markov chains. The paper extends the syntax of p-automata to accept the set of all Markov chains (modulo bisimulation) obtained from a Markov decision process under various schedulers. The aim is to enrich the semantics of the automata to capture various probabilistic tree logic.

- **Model Checking of Open Interval Markov Chains [21]**
  The paper solves the model checking problem for interval Markov

chains with open intervals. Interval Markov chains are generalizations of discrete time Markov chains where the transition probabilities are intervals, instead of constant values. The focus of the paper is the singular case of open intervals (where the boundaries of the interval are not included). Open intervals is semantically challenging, as optimal (min, max) value for reachability does not always exist. The paper solves the model checking (and reachability) problem with minor modification to existing algorithms for model checking Markov chains against PCTL formulas.

- **Parametric LTL on Markov Chains. [20]**
  The paper tackles the verification problem of finite Markov chains against parametrized LTL(pLTL) formulas. In pLTL, the until-modality is equipped with a variable bound. For example, $\diamond_{\leqslant x}\varphi$ asserts that φ holds within x time steps, where x is a variable on natural numbers. Given a Markov chain, a pLTLformula φ and a threshold $\prec p$ (where $\prec$ is a comparison on reals and p is some probability), the problem is to determining the set of parameter valuations, represented by $V_{\prec p}(\varphi)$, for which the probability of satisfying the φ in the Markov chain meets a given threshold $\prec p$. As determining the emptiness of $V_{\geq 0}(\varphi)$ is undecidable, we consider several fragments of the logic with increasing expressiveness. Namely, we consider parametric reachability logic, a sub-logic of pLTL restricted to next and eventually operator, parametric Büchi properties and finally, a maximal subclass of pLTL for which emptiness of $V_{\geq 0}(\varphi)$ is decidable.

- **Modeling and statistical model checking of a micro-grid. [24]**
  The journal provides a use case study of statistical model checking of micro-grids. A micro-grid with wind, micro-turbines, and the main grid as generation resources . The micro-grid is modeled as a parallel composition of various stochastic hybrid automata. Extensive simulation of the behaviour of the individual components give insight into the complex dynamics of the system and provide useful information to determine adequate parameter settings. The study focuses on the application of statistical methods in determining the probability of linear temporal logic properties expressed in the logic LTL. The statistical model checker Uppaal-SMC was used to perform numerical analysis.

Contents

Chapter 1

Introduction

Since the time of antiquity, logic has been an integral part of mathematics. Logic, typically consists of a formal language used for writing the statements of the logic, a deductive systems to deduce new statement from a given set of statements and (or) a model-theoretic semantics to define meaning of each sentence. Loosely speaking, logic is the study of the truthfulness of statements. The language (*syntax*) of the logic is simply a collection of statements (also called formulas) *recursively* defined by a *grammar*. On its own it has no meaning, the meaning is given by the deductive system and (or) by the *semantics*. In a model-theoretical framework, the semantics of the logic defines (inductively) the set of structures for which a given formula is *true*. These structures are called the models of the formula. A formula is said to be *valid* iff every structure is a model of the formula. The two fundamental questions of any logic are:

1. Given a formula f and a model M, whether f is true at M.
2. Given formula f, whether f is *valid*.

We call the first, the *model checking problem* and the second the *validity problem* (or its complement the *satisfiability problem*).

Logic was envisaged as to mitigate the problem of ambiguity in the meaning of sentences of natural languages. Logic allows us to *unambiguously express* properties (formulas of the logic) and formally verify if the property is (true) fulfilled by a given system (model). Directed graphs are the natural mathematical structures for abstracting the behaviour of physical systems. First order logic was found to be unsatisfactory for describing the dynamic behaviour of systems. *Modal logic*, on the other hand, are reasoned over directed graphs, also called Kripke structure, named after Saul Kripke who gave the *possible world semantics* [69]. Thus, modal logic and its many extensions have become the most widely studied logic for verification.

Modal logic paved the way for different kinds of (more expressible) *temporal* logics, which are useful in describing the change in behaviour with time. Temporal logic extends modal logic, specially **K** [83] logic where only local properties of a Kripke structure can be defined, with temporal constructs for expressing global properties. This allows us to express things like reachability and invariants, which are sine qua non for program verification. Temporal logic was introduced around 1960 by Arthur Prior [85] under the name of *Tense Logic* and developed further by many logicians and computer scientists, most prolifically by, Amir Pnueli [82], E. Allen Emerson [36], Moshe Y. Vardi [95] and Dexter Kozen [65], to name a few.

Temporal logic are broadly classified on the basis of how passage of time is perceived. The school of linear temporal logics, as the name suggests, perceives time to flow linearly. Thus the models of these kind of temporal logic formulas are totally ordered sequences of events. Prior's original semantics for Tense logic assumed a linear time flow. One of the most popular and widely used linear temporal logic in computer science is the linear time temporal logic LTL, proposed in the seminal paper by Pnueli (1977) [82] and first explicitly axiomatized and studied by Gabbay et al. (1980) [45]. LTL has $\langle \mathbb{N}, < \rangle$ as models, and is very useful for expressing safety, liveness, fairness, and precedence properties of infinite computations in reactive systems (Manna and Pnueli (1992) [75]). For example, reachability is expressed with the operator F. A linear sequence (also called path) satisfies $\mathsf{F}a$ if it eventually reaches a state satisfying a. Invariant is expressed with the operator G. A path satisfies $\mathsf{G}a$ if at every state of the path a holds. Statement, "*Every time when a message is sent, an acknowledgment of receipt will eventually be returned*", can be easily expressed in LTL as $\mathsf{G}(\mathrm{msg} \rightarrow \mathsf{F}(\mathrm{ack}))$.

The other school of temporal logics considered *branching future*, where time does not follow deterministically as in linear temporal logics, rather non-deterministically with many possible future extensions. It is based on the assumption that while the past cannot be changed, the future can take different possible courses from the present moment. Formally, this means that the natural flows of time for branching time temporal logic semantics are tree-like rather than linear. A models of any branching time temporal logic formula is a tree $\langle T, \prec \rangle$, where $\prec$ is a partial order (contrast to the total order for linear temporal logics) on T, and T is a collection of observable time instances. These structures in computer science are called *computation trees*, and are naturally obtained by unfoldings of Kripke structures. *Computation tree logic* CTL introduced by Clarke and Emerson (1982) [37], CTL^* introduced by Emerson and Halpern (1985) [38] and *Modal μ calculus* ($\mathsf{L}\mu$), introduced by Dexter Kozen [65] are some of the popular branching

time logics used in computer science. In CTL formulas alternate between *path formula* and *state formula*. The path formulas are interpreted over paths of the tree (and hence are formulas of linear temporal logic). State formulas are obtained by universally ($\forall$) or existentially ($\exists$) quantifying path formulas. For example, $\forall \mathsf{F}a$, is a CTL state formula, where $\mathsf{F}a$ is a path formula (also an LTL formula), which is quantified by $\forall$. A state of the tree satisfies $\forall \mathsf{F}a$, if every path from that state satisfies $\mathsf{F}a$. It is important to note that in CTL, there is strict alternation between the linear temporal formulas and the quantifiers. CTL* is CTL without this restriction. The logic Lμ extends standard **K** modal logic with greatest and least fixed point operators. This allows us to define properties recursively, whose semantics is obtained inductively by relying on fixed point theory over complete lattices. Lμ was proved to be the most expressive fragment of second order logic which cannot distinguish between bisimilar models [59], and hence includes CTL and CTL*.

So far we have only talked about logics whose semantics are defined in an absolute sense. That is, given a formula f and a model M, there are only two possibilities, either f is true for M (M satisfies f) or f is false for M (M does not satisfies f). The Branching time semantics opened the door for quantitative satisfaction of formulas. These logic are equipped with constructs which can reason about the number (or the measure) of possible future branches where the formula holds. Hence, they are useful in representing important aspects of computations such as *probabilistic behaviour* [52], *timing behaviour* [64] and other quantitative aspect involving counting [40]. The kind of quantitative models that we will extensively talk in this thesis are probabilistic systems. Broadly classified as *probabilistic transition systems*, these transitions systems are mathematical objects that generalize standard transition systems. Each state of the transition system is equipped with (one or more) probability distribution(s) on the set of states. Labeled Markov chains, concurrent Markov chains, probabilistic automata, among others are some of the well known probabilistic transition systems.

One of the major direction of research in the past few decades has been the development of appropriate structure for expressing qualitative properties, and modeling probabilistic behaviour. Probabilistic aspect are essential for:

1. *Randomized algorithms.* They are modeled by probabilistic programs. Study of sequential programs essentially culminates to the analysis of *Markov chains*, and concurrent programs to *Markov decision processes.*

2. Modeling *unreliable* system behaviour. Phenomena like message loss,

processes failure can be quantified and used to develop a more accurate model of the system.

Once such a model has been chosen, it is necessary to have logics that are able to express the interesting properties of the model catering to its quantitative aspect, since for such systems absolute guarantee of correctness make little sense. For example, in a client-server setting we may want to know whether: "*50% of the time a request sent by a client is serviced by the server*", or in sequential randomized program, "*the program terminates with probability 1*", or in a randomized distributed leader election protocol "*every member has equal probability of being the leader*".

The initial approaches to develop probabilistic logics were to extend existing branching time temporal logics, primarily CTL and CTL^* with *threshold operators* for specifying quantitative properties. For instance, consider the CTL formula $\forall \mathsf{F}a$ which is true for a state s of a transition system if every path from s satisfy $\mathsf{F}a$. In the probabilistic version we have $[\mathsf{F}a]_{\leq \frac{1}{3}}$, which is true for a state of a probabilistic transition system if the probability of the set of paths from s satisfying $\mathsf{F}a$ has a measure $\leq \frac{1}{3}$ (the definition of measure vary from system to system). This method of extending traditional temporal logics led to *probabilistic computation tree logic* (PCTL) and *probabilistic CTL** (PCTL^*) [52]. The semantics of such logics are still boolean, either a state of probabilistic transition system satisfies a formula or it does not satisfies a formula. Only qualitative aspect lies in the measure of the set of paths that satisfies a given linear temporal formula.

Earliest works where the satisfaction of a formula f by a model M was lifted from a boolean valued function to a real valued function were done by M. Huth and M. Kwiatkowska [57], and independently by A. McIver and C. Morgan [78]. Both works extend the interpretation of $\mathsf{L}\mu$ to the probabilistic models. The principal insight is to lift the satisfaction of a formula on a model from a boolean function to a real valued function with range $[0, 1]$. For example, in the case of M. Huth and M. Kwiatkowska [57], the formulas are interpreted over the set of states of a Markov chains. For each formula f and state s, the satisfaction relation $[\![f]\!](s)$ is the value of the a function $[\![f]\!] : S \to [0, 1]$, where S is the set of states of the probabilistic system. Subsequently, many different interpretation of probabilistic μ calculus has been proposed [43, 79, 34]. The central difference lies in the choice of interpretation of different logical operators.

Unlike $\mathsf{L}\mu$ which includes CTL and CTL^*, quantitative interpretation of μ calculus over probabilistic systems, are not comparable to PCTL and PCTL^*. The probabilistic μ calculus ($\mathsf{pL}\mu^{\odot}$ and $\mathsf{pL}\mu^{\odot}_{\oplus}$) proposed by Matteo Mio [80] remedies this shortcoming, by equipping the logic $\mathsf{pL}\mu^{\odot}$ with $\odot$ operator,

called the *independent product* and $\mathsf{pL}\mu^{\odot}_{\oplus}$ with $\oplus$. The logic $\mathsf{pL}\mu^{\odot}$ is expressive enough to capture *qualitative* fragment of PCTL^* and $\mathsf{pL}\mu^{\odot}_{\oplus}$ can capture the full PCTL^*. Fragments of $\mathsf{pL}\mu^{\odot}_{\oplus}$ have also been studied by restricting the syntax of the logic [18].

The Probabilistic μ calculus and its various fragments have been subjected to great scrutiny to illicit correct meaning to its syntax. The most interesting concept that emerged from these investigations is the representation of the semantics of these logics as a two player game. The idea draws from $\mathsf{L}\mu$ where satisfaction of Kripke structure by a formula is seen as two player parity game (player 1 and player 2) [39] (which in turn was inspired by the seminal work of Hentikka[56] on game semantics for logics). A Kripke structure satisfies a formula if there exists a winning strategy for player 1 in the corresponding game. Similarly, the semantics of probabilistic μ calculus is defined by a 2 *player stochastic parity game* [78]. The value of the satisfaction function for a formula f at a state s, i.e., $[\![f]\!](s)$, thus becomes the value of certain configuration in the corresponding 2 player stochastic game.

The main focus of the research in probabilistic logics has been on the development of efficient algorithm for the model checking problem. The model checking problem of finite Markov chain for Probabilistic computation tree logic or PCTL, has been extensively studied [52, 5]. Many variant of PCTL have been proposed for Markov decision processes (MDPs) [54, 88] as well. Subsequently, the model checking problem of PCTL and PCTL^* for MDPs was solved by Bianco and de Alfaro [9]. Furthermore, the complexity of the model checking problem of probabilistic μ calculus was shown to be in $\mathsf{NP}\cap\mathsf{co\text{-}NP}$ [18]. Complexity result for the model checking problem for different fragments of probabilistic μ calculus has also been investigated [18]. A detailed analysis of the algorithmic complexity of calculating the the probability measure of a set of paths that satisfies a LTL property was done by Courcoubetis and Yannakakis (1995) [31]. Researchers have also developed algorithms for parametric model checking. In parametric model checking the models under scrutiny are parametrized, where the transition probability of the models (such as Markov chains or MDPs) are defined using parameters. One of the contribution of this thesis is related to qualitative model checking of parametric linear time properties. Instead of considering parametric models, here we consider *parametrized* version of LTL, where the integer bounds on bounded finally (F) operators are parameters. Thus, given a Markov chain and a probability bound, the problem is to identify the set of parameter valuations for which the required probability bound is achievable. The problem is thus the dual of quantitative model checking on LTL.

We show that the general problem is undecidable, and investigate various restriction on the parametrized logic which makes the problem decidable.

In contrast to model checking of probabilistic logics, very few result exist on the satisfiability problem (or the validity problem) for these logics. Hart and Sharir [55] and Lehmann and Shelah [73], were one of the first to consider the validity problem for the probabilistic branching time logic. These logics were probabilistic extensions of Branching time logics like CTL, and can be categorized as the qualitative fragment of PCTL. Their focus was mainly providing a deductive system in the same spirit as was provided by Ben-Ari and Manna [6] for various branching time temporal logics. More recently, Kucera et al [13], has shown that the satisfiability problem for qualitative PCTL is decidable and is 2-EXPTIME complete.

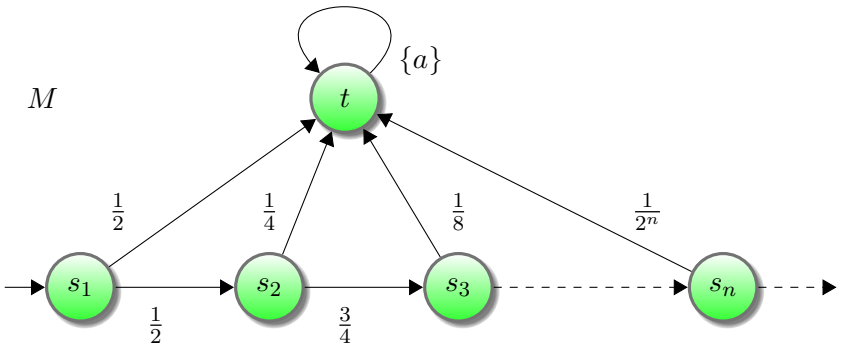


Figure 1.0.1: The Markov chain M satisfies $f := [\mathsf{G}[\sim a \wedge [\mathsf{F}a]_{>0}]]_{>0}$ at state s_1.

One of the main obstacle in obtaining the a decision procedure for probabilistic logics is the fact that there are formulas for which there exists no finite probabilistic transition system that satisfies the formula but are satisfiable by infinite transition systems. The problem already exists for qualitative fragment of PCTL as demonstrated by Hart and Sharir [55] and Kucera et al [13]. For example, consider the PCTL formula $f := [\mathsf{G}[\sim a \wedge [\mathsf{F}a]_{>0}]]_{>0}$. It can be shown that no finite Markov chain satisfy f. But an infinite Markov chain as shown in Figure 1.0.1 is a model of f. The crux of the matter is that at each state s_n (for $n > 0$), there is a non-zero probability of satisfying $\mathsf{F}a$, albeit small, and the path which satisfies $\mathsf{G}(\sim a)$, i.e., $(s_1, s_2, \cdots)$ also has a non-zero probability, since the product of the sequence $\Pi_{n=1}^{\infty}(1 - \frac{1}{2^n}) > 0$. Observe that the exact choice of the sequence $\Pi_{n=1}^{\infty}(1 - \frac{1}{2^n})$ is immaterial. Any sequence with non-zero infinite product would suffice. This is possible since we are dealing with qualitative PCTL, the exact values of the

probability distributions are of little consequence. However, this problem is compounded if we were considering general PCTL. Not only the models could be infinite, but we have to deal with sequences of real number where the sum of the infinite product satisfies a given threshold of the sub-formula.

The contribution of this thesis is to forward our understanding of the complexity of the satisfiability problem for some of the fragments of probabilistic μ calculus. Though the satisfiability problem for PCTL still remains open in general, but as we will see later, we can discern some useful information about logics various fragments of probabilistic μ calculus. We will consider a bounded fragments of PCTL. These logics are endowed with *small model property* (a fact which is reproachfully absent in PCTL as demonstrated by Figure 1.0.1). This means that given any formula f of the logic, f is satisfiable if and only if there exists a (canonical) model of f whose size is exponential in the size of the description of f. In this thesis we establish the exact complexity classes in which the satisfiability problem for these logics belong.

We will also consider a simple probabilistic extension of modal μ calculus (PμTL). PμTL is expressively incomparable to PCTL but is subsumed in probabilistic μ calculus. We show that PμTL has a small model property in the sense that every satisfiable formula f has a model of size exponential in $|f|$ (and has a bounded out-degree at most $|f| + 1$. These results imply (using []) that PμTL and qualitative PCTL are incomparable. The constructive proof uses (ordinary) parity games the satisfiability problem of PμTL. Similar to results for the modal μ-calculus, we obtain that a PμTL-formula f is satisfiable iff player zero has a winning strategy in the game arena that corresponds to f. Using these results we establish that every satisfiable PμTL-formula has a rational model, i.e., a model with rational probabilities only. Our results show that one needs to solve a parity game of exponential size in order to decide PμTL satisfiability. This is the strongest possible bound since PμTL can encode μ-calculus, and the result shows that satisfiability of PμTL lies in the same complexity class as the satisfiability of μ-calculus.

Treating the models of logics as languages of automata is a very effective and well studied mechanism for understanding the properties of the logic. For example, finite regular languages correspond to finite state machines, ω-regular and MSO on infinite words correspond to Büchi automata, modal μ calculus corresponds to alternating tree automata, etc. Similar strides had been made in probabilistic logic in the form of *p-Automata* [] . The language of a p-automaton is a set of Markov chains. This gives us a uniform framework for a language theoretic treatment to the set of Markov chains.

For example, we can consider a formula of any probabilistic logic as the set of Markov chain satisfying the formula. We can even consider any MDP as a set of Markov chains induced by its various schedulers. The final contribution of the thesis is to extend the theory of p-automata so that we can represent the set of Markov chains defined by any MDP as the language of a p-automaton.

1.1 Survey of the chapters

Chapter 2, introduces some preliminary mathematical concepts and definitions that would be used through out the thesis. The definitions are neither elaborate nor exhaustive. Interested reader can follow the provided citations for a more in-depth analysis of the topics.

In chapter 3, we will learn about parametric linear time temporal logic. Parametric linear time temporal can be thought as a description of linear temporal behaviour (the set of infinite words) that is partially define. We get the exact description of the set only when we know the values of the parameters. In this chapter, we will primarily study the synthesis of the values of these parameters for which the probability of the linear time property in a Markov Chain reaches certain threshold.

In chapter 4, we will study interval Markov chains. They differ from Markov chains, as they allow transition probabilities to be intervals. We will see how to model check interval Markov chains against PCTL properties even in the presence of open intervals.

In chapter 5, we explore various fragments of PCTL with finite model property (unlike the general PCTL). We will study the complexity of the decision problems for these logics.

In chapter 6, we will study probabilistic logic with recursion, which extends modal μ-calculus with quantified next operators. These logic are broadly classified under probabilistic μ-calculus. We will see that this logic are orthogonal to PCTL in terms of expressibility and is decidable and possess small model property.

In chapter 7, we will visit the recently developed theory of p-automata. P-automata are reminiscent of tree automata, as they take labeled Markov chains as inputs. We will see how to represent a Markov decision process as p-automata and study the acceptance and language inclusion algorithms.

Chapter 2

Preliminaries

2.1 Important complexity classes

This section will review some of the important concepts and notation for classification of problems according to their complexity. We start with the familiar definition of Turing machine.

Definition 2.1.1. A non-deterministic one-tape Turing machine is a 9-tuple $M = (Q, \Sigma, \Gamma, \triangle, \vdash, \delta, I, F, R)$ where

- Q is a finite set of states.
- Σ is a finite input alphabet.
- Γ is a finite tape alphabet containing Σ as a subset.
- $\triangle \in \Gamma \setminus \Sigma$ is the blank symbol.
- $\vdash$ is the left marker.
- $\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$ is the transition relation.
- $I \subseteq Q$ is the set of start states.
- $F \subseteq Q$ is the set of accepting states.
- $R \subseteq Q$ is the set of rejecting states, $F \cap R = \varnothing$.

Intuitively, $(q, \sigma, q', \sigma', d) \in \delta$ means, "When in state q scanning symbol σ, write σ' on that tape cell, move the head in direction d, and enter state q'. The symbols L and R stand for left and right, respectively.

A configuration is a tuple $(xqay)$, where $x, y \in \Gamma^*$, $a \in \Sigma$ and $q \in Q$. This means that the string xay is on the tape, the head is on the cell containing a and the current state is q. Thus for a configuration $\sigma = xqay \in \Gamma^* \times Q \times \Gamma^+$,

the tape content is $xay = \mathsf{tape}(\sigma) \in \Gamma^+$ with $a \in \Gamma$, the head is at position $|x| + 1$, presently reading input a and the current state is $q = \mathsf{state}(\sigma)$. The successor configuration $c' = (x'q'a'y')$ of $c = (xqay)$ is defined as follows:

- If $(q, a, q', b, L) \in \delta$ then $x'a' = x$ and $y' = by$.
- If $(q, a, q', b, R) \in \delta$ then $x' = xb$ and $y = a'y'$.

A trace ρ is a sequence of successor configurations. A trace is accepting (or rejecting) iff it ends in a configuration $xaqy$ where $q \in F$ (or $q \in R$). A finite word w is accepted (or rejected) iff the there is an accepting trace (or rejecting trace) from the configuration $(q \vdash w)$ for some $q \in I$. The set of word with an accepting run constitute the language of the Turing machine M. Denote it by $L(M)$.

Important variations of non-deterministic Turing machines are:

1. *Deterministic machine*: Each word w has exactly one trace. The transition relation is thus a function $\delta : (Q \times \Sigma) \to (Q \times \Sigma \times \{L, R\})$.
2. *Unambiguous machine*: Each word w has at most one accepting trace.
3. *Alternating Turing machine*: (ATM) [25] is similar to a non-deterministic Turing machine except there is a function in the specification of the machine called type. The function type tells us whether a state is an *and-state* or an *or-state*, where $\mathsf{type} : Q \to \{\wedge, \vee\}$. A *trace* (or a computation) ρ of an ATM A for an input x_{in} is a set of configuration such that, $(q_0 \vdash x_{in}) \in \rho$ and for every $\sigma \in \rho$ with $\mathsf{state}(\sigma) \notin F$, if $\mathsf{type}(\mathsf{state}(\sigma)) = \vee$ then one of the *successor* configurations σ' of σ is in ρ, if $\mathsf{type}(\mathsf{state}(\sigma)) = \wedge$ then every successor configuration of σ is in ρ. Pictorially, ρ is a tree where each node is a configuration and edges are defined by the *successor* relation. A trace ρ is *accepting* for an input x if ρ is finite and only configurations without a successor configuration in ρ are accepting. The language of an ATM A is thus:

$$L(A) = \{x \in \Sigma^* : \text{ there exists an accepting trace } \rho \text{ for } x \}$$

Let $T : \mathbb{N} \to \mathbb{N}$ and $S : \mathbb{N} \to \mathbb{N}$ be numeric functions, which serve as asymptotic time and space bounds for Turing machine computations. Generally, these functions are written as a functions of the length of the input word. For example, $\log n, n, n \log n, n^2, 2^n, n!, 2^{2^n}$, etc.

Definition 2.1.2. A Turing machine TM traces in time $T(n)$ (or is $T(n)$ time-bounded) if for all (*but finitely many*) inputs x, all traces of x are of length $T(|x|)$, where $|x|$ denotes the length of x.

Similarly, a Turing machine TM traces in space $S(n)$ (or is $S(n)$ space-bounded) if for all (*but finitely many*) inputs x, every configuration of every trace of x uses at most $S(|x|)$ work-tape cells.

Now we can define the basic time and space complexity classes:

Definition 2.1.3. Elementary complexity classes:

$$\begin{aligned}
\mathsf{DTIME}(T(n)) &= \{L(M) : M \text{ is a deterministic } T(n) \text{ time bounded TM } \}\\
\mathsf{NTIME}(T(n)) &= \{L(M) : M \text{ is a non-deterministic } T(n) \text{ time bounded TM } \}\\
\mathsf{UTIME}(T(n)) &= \{L(M) : M \text{ is a unambiguous } T(n) \text{ time bounded TM } \}\\
\mathsf{ATIME}(T(n)) &= \{L(M) : M \text{ is an alternating } T(n) \text{ time bounded TM } \}\\
\\
\mathsf{DSPACE}(S(n)) &= \{L(M) : M \text{ is a deterministic } S(n) \text{ space bounded TM } \}\\
\mathsf{NSPACE}(S(n)) &= \{L(M) : M \text{ is a non-deterministic } S(n) \text{ space bounded TM } \}\\
\mathsf{ASPACE}(S(n)) &= \{L(M) : M \text{ is an alternating } S(n) \text{ space bounded TM } \}
\end{aligned}$$

If $\mathfrak{A}$ is a complexity class, the set of complements of the sets in $\mathfrak{A}$ is denoted by $\mathsf{co}\text{-}\mathfrak{A}$. Note that $\mathsf{co}\text{-}\mathfrak{A}$ is not the complement of $\mathfrak{A}$.

Some common complexity classes are as follows:

Example 2.1.1.

$$\begin{aligned}
\mathsf{LOGSPACE} &= \mathsf{DSPACE}(\log n)\\
\mathsf{NLOGSPACE} &= \mathsf{NSPACE}(\log n)\\
\mathsf{P} &= \mathsf{DTIME}(n^{O(1)}) = \bigcup_{i>0} \mathsf{DTIME}(n^i)\\
\mathsf{UP} &= \mathsf{UTIME}(n^{O(1)}) = \bigcup_{i>0} \mathsf{UTIME}(n^i)\\
\mathsf{NP} &= \mathsf{NTIME}(n^{O(1)}) = \bigcup_{i>0} \mathsf{NTIME}(n^i)\\
\mathsf{PSPACE} &= \mathsf{DSPACE}(n^{O(1)}) = \bigcup_{i>0} \mathsf{DSPACE}(n^i)\\
\mathsf{NPSPACE} &= \mathsf{NSPACE}(n^{O(1)}) = \bigcup_{i>0} \mathsf{NSPACE}(n^i)\\
\mathsf{EXPTIME} &= \mathsf{DTIME}(2^{n^{O(1)}}) = \bigcup_{i>0} \mathsf{DTIME}(2^{n^i})\\
\mathsf{NEXPTIME} &= \mathsf{NTIME}(2^{n^{O(1)}}) = \bigcup_{i>0} \mathsf{NTIME}(2^{n^i})\\
\mathsf{2EXPTIME} &= \mathsf{DTIME}(2^{2^{n^{O(1)}}}) = \bigcup_{i>0} \mathsf{DTIME}(2^{2^{n^i}})\\
\mathsf{ATIME}(T(n)^{O(1)}) &= \mathsf{DSPACE}(T(n)^{O(1)})\\
\mathsf{ASPACE}(S(n)) &= \mathsf{DTIME}(2^{O(S(n))}).
\end{aligned}$$

2.2 Probability and stochastic processes

Let X and Y be two non-empty sets. The set of functions from X to Y is denoted by Y^X. For $\varphi \in Y^X$, let $\mathsf{img}(\varphi) \subseteq X$ be the image and $\mathsf{dom}(\varphi) = Y$

be the domain of the function φ. The set of all subsets (called the power set) of X is defined as 2^X. Any subset $\mathcal{Z} \subseteq 2^X$ is called a *collection* of X. A *topology* on a set X is a collection $\mathcal{T} \subseteq 2^X$ such that:

1. $\varnothing \in \mathcal{T}$ and $X \in \mathcal{T}$.
2. For every $U, V \in \mathcal{T}$, $U \cap V \in \mathcal{T}$.
3. For any collection $\mathcal{U} \subseteq \mathcal{T}$, $\bigcup_{U \in \mathcal{U}} U \in \mathcal{T}$.

The elements of $\mathcal{T}$ are called *open set*, and any set whose complement is in $\mathcal{T}$ is called *closed*. In that respect, the sets $X, \varnothing$ are closed and open at the same time. The pair $(X, \mathcal{T})$ is called a topological space. A collection $\mathcal{C} \subseteq \mathcal{T}$ is a *base* of $\mathcal{T}$, iff for every $V \in \mathcal{T}$, $V = \bigcup\{U \in \mathcal{C} : U \subseteq V\}$.

Definition 2.2.1 ((Weighted) cover)**.** Let H be a set of objects. A *cover* c is a set of sets of objects of H, such that $\bigcup_{e \in c} e = H$. The cardinality of c is called the *width* of the cover c. A *weighted cover* of H is a cover c with a mapping $w : c \to (0, 1]$, such that $\sum_{e \in c} w(c) = 1$.

For weighted cover (c, w) of $H = \{o_1, \cdots, o_n\}$, let $H(o_i) = \{e \in c : o_i \in e\}$ and with little abuse of notation, let $w(o_i) = \sum_{e \in H(o_i)} w(e)$.

Definition 2.2.2. A σ-algebra on a set X is a collection $\mathcal{F}$ of X which obeys the following properties:

1. (Empty set) $\varnothing \in \mathcal{F}$.
2. (Complement) If $E \in \mathcal{F}$, then the complement $E^c := X \setminus E$ also is in $\mathcal{F}$.
3. (Countable unions) If $E_1, E_2, \cdots \in \mathcal{F}$ then $\bigcup_{n=1}^{\infty} E_n \in \mathcal{F}$.

We refer to the pair $(X, \mathcal{F})$ of a set X together with the σ-algebra on that set as a *measurable space*. Elements of $\mathcal{F}$ are called events.

A σ-algebra can be constructed from any collection $\mathcal{F}$ with the notion of generation.

Definition 2.2.3. Let $\mathcal{F}$ be any family of sets in X. We define $\langle \mathcal{F} \rangle$ to be the intersection of all σ-algebra that contains $\mathcal{F}$.[1] Equivalently, $\langle \mathcal{F} \rangle$ is the coarsest σ-algebra that contains $\mathcal{F}$.

We now turn to an important example of a σ-algebra .

[1] Fact that the intersection of a collection of σ-algebra is σ-algebra can be proved, easily.

Definition 2.2.4. Let $(X, \mathcal{T})$ be a topological space. The *Borel* σ-algebra $\mathcal{B}[X]$ of X is defined to be the σ-algebra generated by the open sets of X, i.e., $\mathcal{B}[X] = \langle \mathcal{T} \rangle$. Elements of $\mathcal{B}[X]$ are called *Borel measurable*.

We define a the concept of measure on the σ-algebra . We endow these structures with *countably additive measure* μ.

Definition 2.2.5. Let $(X, \mathcal{B})$ be a measurable space. A countable additive measure μ on $\mathcal{B}$, or measure for short, is a map $\mu \in [0, +\infty]^{\mathcal{B}}$ that obey the following axioms:

1. (Empty set) $\mu(\varnothing) = 0$.
2. (Countable additivity) Whenever $E_1, E_2, \cdots \in \mathcal{B}$ are countable sequence of disjoint measurable set, then $\mu(\bigcup_{n=1}^{\infty} E_n) = \sum_{n=1}^{\infty}(E_n)$.

A triplet $(X, \mathcal{B}, \mu)$ where $(X, \mathcal{B})$ is a measurable space and $\mu \in [0, +\infty]^{\mathcal{B}}$ is a countable additive measure, is known as a *measure space*.

We are interested in a particular class of measures:

Definition 2.2.6. Let $(X, \mathcal{B})$ be a measurable space. A *probability* measure (also called distribution) is a countable additive finite measure $\mu \in [0, 1]^{\mathcal{B}}$, with $\mu(X) = 1$. If $(X, \mathcal{B})$ is a discrete space (i.e. X is countable and $\mathcal{B}$ is a discrete topology) then the set of probability measure over the set X is denoted by $\mathcal{D}_X$ where $\vec{d} \in \mathcal{D}_X$ iff $\vec{d} \in [0, 1]^X$ and $\vec{d}^T \cdot \vec{1} = 1$. For $\mu \in \mathcal{D}_X$, let $\mathsf{supp}(\mu) = \{x \in X \ : \ \mu(x) > 0\}$ be the support of the distribution μ. If $|\mathsf{supp}(\mu)| = 1$ then μ is called a Dirac distribution.

We can construct a new space from existing spaces by product operations.

Definition 2.2.7. For each $i \in \mathbb{N}$, let $(X_i, \mathcal{F}_i)$ be measurable spaces, and let $X = \prod_{i=1}^{\infty} X_i$ be the set of tuples (of infinite length), such that $(x_1, x_2, \cdots) \in X$ whenever $x_i \in X_i$. A *cylinder set* for a measurable set $A \in \prod_{i=1}^{n} \mathcal{F}_i$ is defined as $\mathsf{Cyl}(A) = \{(x_1, \cdots, x_n, \cdots) \in X : (x_1, \cdots, x_n) \in A\}$.

Consider the topology $\mathcal{X}$ of $X = \prod_{i=1}^{\infty} X_i$ formed by all infinite union of all cylinder sets. Observe that every cylinder set is both open and close. As we will see subsequently, the *Borel* σ-algebra of X, $\mathcal{B}[X]$ is of particular interest to us.

Next we use the above definitions to define the probability measure space generated by stochastic processes.

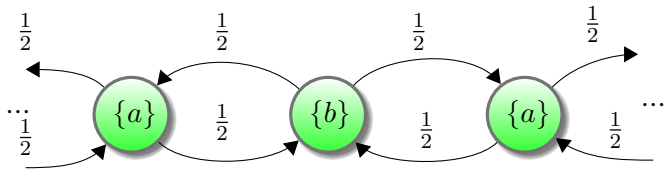


Figure 2.2.1: The Markov chain M defines a random walk. The set of atomic propositions are $\{a, b\}$.

Definition 2.2.8. A labeled Markov chain (MC) M is a quintuple $(S, \mathsf{P}, \mathsf{AP}, L, s_{in})$ where S is a (countable) set of states, $\mathsf{P}(s) \in \mathcal{D}_S$ for all $s \in S$, AP is a set of atomic propositions, $L : (2^{\mathsf{AP}})^S$ is a labeling function, and $s_{in} \in S$ is the initial state.

An example of a Markov chain is given in Figure 2.2.1. An *infinite path* σ of MC M is a sequence of states $\sigma = \{\sigma_i\}_{i \geq 0}$, where for all $i \geq 0$, $\mathsf{P}(\sigma_i, \sigma_{i+1}) > 0$. Let $\mathsf{path}(s)$ denote the set of (finite or infinite) paths starting from state s. For a path σ, let $\sigma\downarrow$ denote the last state of σ if this exists (i.e., if σ is finite) and $|\sigma|$ denote the length of σ. The *trace* of a σ is defined as the sequence of sets of atomic propositions $\mathsf{trace}(\sigma) = (L(\sigma_0), L(\sigma_1), \cdots)$. Let $\mathsf{succ}(s) = \{t : \mathsf{P}(s, t) > 0\}$ be the direct successors of state s.

Each state s defines a probability measure space on the set of states, $X_s = (S, 2^S, \mu_s)$, where $\mu_s(A) = \sum_{s' \in A} \mathsf{P}(s, s')$. A probability measure on sets of infinite paths is obtained in the following way. Let Ω_s be an infinite cross product of spaces $X_s \times \prod_{i=1}^{\infty} \bigcup_{s' \in S} X_{s'}$. Any finite path $\sigma = (\sigma_0, \cdots, \sigma_n)$ is measurable in the finite product space $X_s \times \prod_{i=1}^{n} \bigcup_{s' \in S} X_{s'}$. The collection $\mathcal{C} = \{\mathsf{Cyl}(\sigma) : \sigma \text{ is a finite path}\}$, the set of cylinder sets, is the base of the topology generated by the countable union and finite intersections of elements in $\mathcal{C}$. The measurable space $(\Omega_s, \mathcal{F}, \Pr)$, where $\mathcal{F} = \mathcal{B}[\Omega_s]$ is the Borel σ-algebra generated from topology with $\mathcal{C}$ as base. Pr is the probability measure on $\mathcal{F}$ is deduced from the measure of cylinder sets. A finite path $\sigma = (\sigma_0, \cdots, \sigma_n)$, has a measure $\Pr(\mathsf{Cyl}(\sigma)) = \prod_{0 < i \leq |\sigma|} \mathsf{P}(\sigma_{i-1}, \sigma_i)$.

We will often define a property as a set of words in Σ^*, where $\Sigma = 2^{\mathsf{AP}}$. The set of paths corresponding to a property f is the set $\mathsf{trace}^{-1}(f) = \{\sigma : \mathsf{trace}(\sigma) \in f\}$. When $\mathsf{trace}^{-1}(f)$ is measurable in $(\Omega_s, \mathcal{F}, \Pr)$, then $\Pr(s \models f) = \Pr(\mathsf{trace}^{-1}(f))$.

Definition 2.2.9. A *Markov decision process* D is a quintuple $(S, \Delta, \mathsf{AP}, L, s_{in})$ where S, AP, L, and s_{in} are as before, and $\Delta : (2^{\mathcal{D}_S})^S$ such that $\Delta(s)$ is a finite set of distributions. We assume S and $\Delta(s)$ for each $s \in S$ to be finite (unless the contrary is explicitly specified).

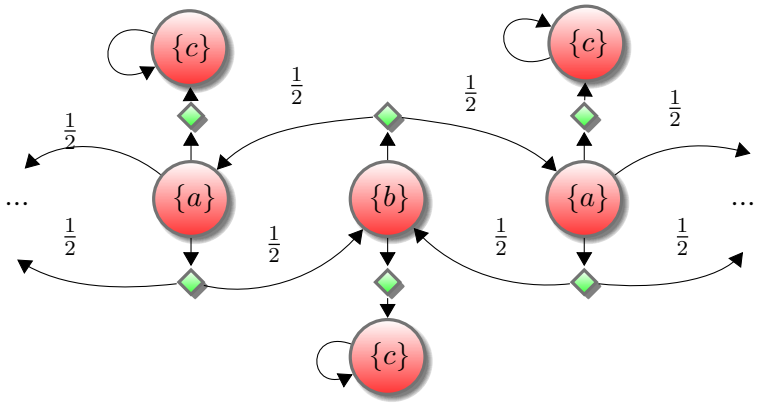


Figure 2.2.2: A sample MDP D with countable number of states. At each state has two distributions.

A finite path of an MDP is a sequence of states $\sigma = \sigma_0 \dots \sigma_n$ such for each $0 < i \leq n$ $\sigma_i \in \mathsf{supp}(\mu)$ for some $\mu \in \Delta(\sigma_{i-1})$. Let $\mathsf{path}(s)$ be the set of (finite and infinite) paths from the state s. Let $\mathsf{succ}(s) = \{t \ : \ t \in \bigcup_{\mu \in \Delta(s)} \mathsf{supp}(\mu)\}$ be the set of successors of s. As usual, we use *schedulers* to resolve the possible non-determinism in a state.

Definition 2.2.10. A *scheduler* of MDP $D = (S, \Delta, \mathsf{AP}, L, s_{in})$ is a function $\eta : (\mathcal{D}_{\mathcal{D}_S})^{S^+}$ with $\eta(\sigma) \in \mathcal{D}_{\Delta(\sigma\downarrow)}$. The scheduler η induces the MC $D_\eta = (S^+, \mathsf{P}, \mathsf{AP}, L', s_{in})$ with $L'(\sigma) = L(\sigma\downarrow)$, and $\mathsf{P}(\sigma, \sigma{\cdot}t) = \sum_{\mu \in \Delta(\sigma\downarrow)} \eta(\sigma)(\mu){\cdot}\mu(t)$.

These schedulers are history-dependent and randomized. Let $\mathsf{HR}(D)$ denote the set of history-dependent randomized schedulers of MDP D. If for all σ,$\eta(\sigma)$ is a Dirac distribution on $\Delta(\sigma\downarrow)$, then η is a *history dependent deterministic* scheduler. Let $\mathsf{HD}(D)$ be the set of all such schedulers of D.

Definition 2.2.11. A stochastic game G is a tuple $(V, E, V_0, V_1, V_p, \mathsf{P}, \Omega)$, where (V, E) is a directed graph and (V_0, V_1, V_p) is a partition of V. V_0 is the set of Player 0 configurations, V_1 is the set of Player 1 configurations and V_p is the set of stochastic (or probabilistic) configurations. P is a probability transition function $P \in (\mathcal{D}_V)^{V_p}$ and Ω is an acceptance property that determines whether a path of the directed graph (V, E) is accepting. A path (also called a play) in the graph (V, E) is winning for Player 0 if it is finite and ends in Player 1 configuration, or it is infinite and satisfies the acceptance property Ω.

The 2-player game proceeds from a configuration u by the following rule. If u is a Player 0 (or Player 1) configuration, then Player 0 (or Player 1, resp.) chooses a configuration v such that $(u, v) \in E$. If no configurations can

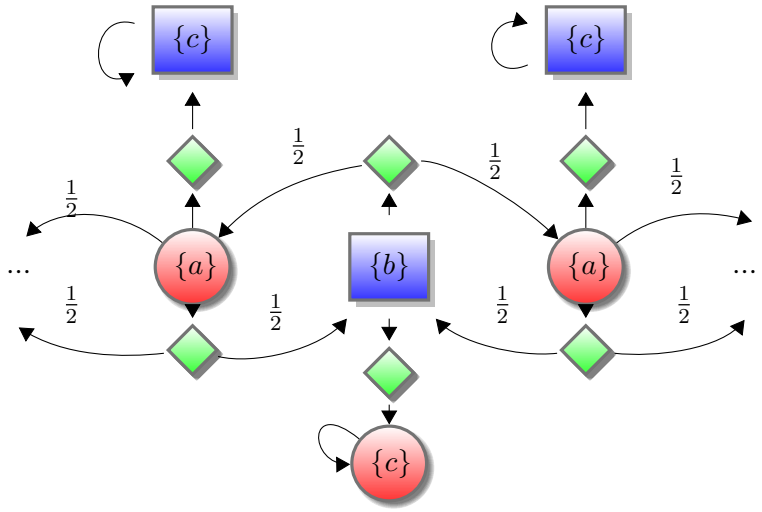


Figure 2.2.3: A two player stochastic game G. The Player 0 configurations are represented by circular nodes, Player 1 configurations are represented by rectangular nodes and probabilistic nodes are denoted by rhombus nodes. A co-Büchi acceptance condition is $\Omega_0 = \{v : L(v) = \{c\}\}$.

be reached from u then u is losing for Player 0 (or Player 1, resp.). On the other hand, if u is a probabilistic configuration then the next configuration v is chosen with probability $\mathsf{P}(u, v)$. If this process continues ad infinitum, we get a sequence of configuration which forms an infinite path (also called a play) of the game. The path is winning for Player 0 iff it satisfies the property Ω. Different acceptance properties define various 2-player games. The acceptance property that will be of particular interest to us (in Chapter 7) is called *co-Büchi* property. It is represented by a set of configurations $\Omega_0 \subseteq V$ of G. An infinite path is accepting iff there is a suffix (tail) of the path which lies in Ω_0, i.e., a path $\sigma = (v_0, v_1, \cdots)$ is winning for Player 0, iff there exists $n \in \mathbb{N}$, such that for all $m \geq n$, $v_m \in \Omega_0$.

A subset C of V is called strongly connected component if for any two configurations $u, v \in S$, v is reachable from u. A strongly connected complement C is maximal (MSCC, for short) iff there is no superset of C that is strongly connected. A stochastic game is called a *weak stochastic game* iff it has a co-Büchi acceptance condition and for all maximal connected components (MSCC) C, either $C \subseteq \Omega_0$ or $C \cap \Omega_0 = \varnothing$. If $V_p = \varnothing$ then it is a *weak game*. A *strategy* of a Player 0 is a function $\rho : (\mathcal{D}_V)^{(V^* \times V_0)}$, with $\rho(w \cdot u)(v) > 0$ implies $(u, v) \in E$. A play $w = v_0 v_1 \cdots$ is consistent with strategy ρ if for every $i \geq 0$, $v_i \in V_0$ implies $\rho(v_0 \cdots v_i)(v_{i+1}) > 0$. Strategies of Player 1 are similarly defined. Let Υ and Π be the set of all strategies

for Player 0 and Player 1, respectively. A player 0 strategy ρ is memoryless iff $\rho(w{\cdot}v) = \rho(w'{\cdot}v)$, for any $w, w' \in V^*$, and it pure iff ρ is Dirac (similarly definitions apply to strategies of player 1).

A pair of strategies $(\rho, \pi) \in \Upsilon \times \Pi$ of a game G determines a MC $M^{\rho,\pi}$ (configurations without an out-going transition are made absorbing) whose paths are plays of G according to ρ, π. The measure of the set of winning plays of Player 0 starting from a configuration c in $M^{\rho,\pi}$, is denoted by $\mathsf{val}_0^{\rho,\pi}(c)$. We have $\mathsf{val}_1^{\rho,\pi}(c) = 1{-}\mathsf{val}_0^{\rho,\pi}(c)$. Let $\mathsf{val}_0(c) = \sup_{\rho\in\Upsilon} \inf_{\pi\in\Pi} \mathsf{val}_0^{\rho,\pi}(c)$ and $\mathsf{val}_1(c) = \sup_{\pi\in\Pi} \inf_{\rho\in\Upsilon} \mathsf{val}_1^{\rho,\pi}(c)$. If a strategy achieves these values then it is called *optimal*.

Theorem 2.2.1. *[77, 30, 27] Let G be a stochastic game and c be one of its configurations. Then G is determined, that is $\mathsf{val}_0(c){+}\mathsf{val}_1(c) = 1$. If G is finite and weak, then optimal strategies for both players exist and they are memoryless and pure. If G is a stochastic weak game, then the problem whether $\mathsf{val}_0(c)$ exceeds than a given quantity $v \in \mathbb{Q}$ can be decided in $\mathsf{NP}\cap\mathsf{co}$-$\mathsf{NP}$, and if G is weak game then $\mathsf{val}_0(c) = 1$ can be decided in linear time.*

Stochastic games (the structure rather than acceptance properties) generalizes both Markov chains and Markov decision processes. A stochastic game without Player 1 states (i.e., $V_1 = \varnothing$) is called $1\frac{1}{2}$-player game and its stochastic behaviour is identical a to MDP. Similarly, a stochastic game with only probabilistic states (i.e., $V_0 = V_1 = \varnothing$) behaves like a Markov chain.

Definition 2.2.12 (Probabilistic bisimulation [72])**.** Let MC $M = (S, P, \mathsf{AP}, L, s_{in})$ and $H \subseteq \mathsf{AP}$. The equivalence relation $\mathcal{R}_H \subseteq S \times S$ is a *probabilistic bisimulation* iff for every $(s, s') \in \mathcal{R}_H$ it holds:

1. $L(s) \cap H = L(s') \cap H$, and
2. for every $C \in S/\mathcal{R}_H$, we have $\sum_{t\in C} P(s,t) = \sum_{t'\in C} P(s',t')$.

Let $\approx_H$ denote the largest probabilistic bisimulation on S. The MCs M_1 and M_2 are probabilistically bisimilar, denoted $M_1 \approx_H M_2$, if $s_{in}^1 \approx_H s_{in}^2$ in the disjoint union of M_1 and M_2.

2.3 Temporal logic

Temporal logic broadly covers reasoning about temporal events within a logical framework. Temporal logic is an important class of modal logic. Temporal logic can have both dense and discrete models, in this thesis we are interested in discrete models. Next we introduce two different notion of perceiving the flow of time.

2.3.1 Linear temporal logic

Linear temporal logic, as the name suggests, considers a linear flow of time. The (forward) models of this logic are totally ordered structures (w, i), where $w \in N^{2^{\mathsf{AP}}}$ (called a word), AP is the set of atomic propositions, 2^{AP} is sometimes referred as the alphabet (Σ) and $i \in N$ (called a time instance). If N is finite (i.e., $N = \{0, 1, \cdots, n\}$) then w is a finite word, and if $N = \mathbb{N}$ then w is an infinite word. Next, we will describe few important linear time logics.

ω-regular languages ω-regular languages generalize the definition of *regular languages* to infinite words. Similar to regular expression, the formulas of ω-regular languages are defined by ω-regular expressions. Consider the following operations on sets of words:

- *Concatenation:* Let L_1 be a set of finite words and L_2 be a set of finite or infinite words. Then $L_1 \cdot L_2 = \{w \cdot w' : w \in L_1, w' \in L_2\}$.
- **-iteration:* Let L be a finite set of finite words. $L^* = \bigcup_{n \in \mathbb{N}} L^n$, where $L^n = \{w_1 \cdot w_2 \cdots w_n : w_i \in L, 1 \leq i \leq n\}$, and $L^0 = \{\epsilon\}$.
- *ω-iteration:* Let L be a finite set of finite words. $L^\omega = \{w_1 \cdot w_2 \cdot w_3 \cdots : w_i \in L \setminus \{\epsilon\}\}$. Observe that $\{\epsilon\}^\omega = \varnothing$, where as $\{\epsilon\}^* = \{\epsilon\}$.

The syntax of ω-regular expressions is defined as follows:

Definition 2.3.1.

$$\begin{aligned} s &::= \varnothing \mid r^\omega \mid r \cdot s \mid s + s \\ r &::= \epsilon \mid r \cdot r \mid r^* \mid r + r \end{aligned}$$

where r defines regular expression and s defines ω-regular expression. The language $L(s)$ of a ω-regular expression is defined inductively as follows:

- $L(\varnothing) = \varnothing$.
- $L(r^\omega) = (L(r))^\omega$.
- $L(r \cdot s) = L(r) \cdot L(s)$.
- $L(t_1 + t_2) = L(t_1) \cup L(t_2)$, where t_1, t_2 are either regular expressions or ω-regular expressions.
- $L(r^*) = (L(r))^*$.

A language L is ω-regular iff there is an ω-regular expression s, such that $L = L(s)$.

ω-automata are generalizations of finite automata, which are acceptors infinite words. A non-deterministic Büchi automaton, introduced by J.R. Büchi [15], is an ω-automaton defined as follows:

Definition 2.3.2. Non-deterministic Büchi automaton $A = (Q, \Sigma, \delta, I, F)$.

- Q is a finite set of states.
- Σ is the finite alphabet set.
- $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation.
- $I \subseteq Q$ is the set of initial states.
- $F \subseteq Q$ is the set of accepting states.

A *run* is a sequence of states of the automaton, $\rho = q_0 q_1 q_2 \cdots$, such that $q_0 \in I$ and for all $i \geq 0$, $(q_i, \sigma, q_{i+1}) \in \delta$ for some $\sigma \in \Sigma$. $\mathsf{inf}(\rho)$ is the set of states in ρ that occur infinitely often. A run ρ is *accepting* iff $\mathsf{inf}(\rho) \cap F \neq \varnothing$. An infinite word $w = \sigma_0 \sigma_1 \cdots$ has a run $\rho = q_0 q_1 q_2 \cdots$ if for all $i \geq 0$, $(q_i, \sigma_i, q_{i+1}) \in \delta$. The language of a Büchi automaton is defined as follows:

$$L(A) = \{w \in \Sigma^\omega : w \text{ has an accepting run}\}$$

The equivalence of expressive power of ω-regular languages and Büchi automata was established by the following theorem.

Theorem 2.3.1. *([15]) An ω-language is recognized by a Büchi automaton iff it is an ω-regular language.*

There are many other ω-regular automata with equivalent expressive power. These include, Muller, Street, Rabin, parity, etc. They all differ in their accepting condition. Deterministic parity automata is of particular interest to us.

Definition 2.3.3. A deterministic parity automaton $A = (Q, \Sigma, \delta, I, \Omega)$.

- Q is a set of states.
- Σ is the alphabet.
- $\delta : Q \times \Sigma \to Q$ is the transition function.
- $I \subseteq Q$ is the set of initial states.
- $\Omega : Q \to C$, where $C = \{0, 1, \cdots, n\}$ for some $n \in \mathbb{N}$.

An infinite run ρ is accepting iff $\max\limits_{q \in \mathsf{inf}(\rho)} \{\Omega(q)\}$ is even.

Theorem 2.3.2. *([81])From a nondeterministic Büchi automaton A with n states a deterministic parity automaton B with n^{2n+2} states and an index of size $2n$ can be constructed such that $L(A) = L(B)$.*

Linear-time temporal logic (LTL)

Definition 2.3.4. Linear-time temporal logic [75] is defined by the following grammar:

$$f \;:=\; a \mid \sim f \mid f \wedge f \mid \mathsf{X} f \mid f \,\mathsf{U}\, f,$$

where $a \in \mathsf{AP}$. The semantics is inductively defined as follows:

$$\begin{array}{lll}
(w,i) \vDash a & \text{iff} & a \in w(i), \\
(w,i) \vDash \sim f & \text{iff} & (w,i) \nvDash f, \\
(w,i) \vDash f \wedge g & \text{iff} & (w,i) \vDash f \text{ and } (w,i) \vDash g, \\
(w,i) \vDash \mathsf{X} f & \text{iff} & (w,i+1) \vDash f, \\
(w,i) \vDash f \,\mathsf{U}\, g & \text{iff} & \exists k : \forall j < k : (w,k) \vDash g \text{ and } (w,j) \vDash f.
\end{array}$$

As a convention, the time instance i is not specified when $i = 0$, i.e., we write $w \vDash f$ instead of $(w,0) \vDash f$. The language of a LTL formula f is $L(f) = \{w : w \vDash f\}$. When there is no ambiguity, a formula also denotes a set of models where it is true.

Theorem 2.3.3. *([92]) ω-regular languages are strictly more expressive than LTL.*

We can define sets of infinite paths of a labeled Markov chain as linear temporal properties, using LTL formulas, ω-regular expressions or ω-automata. Let M be a Markov chain, s be a particular state and f be a temporal property. We are generally interested in the measure of the following set:

$$\{\pi \in \mathsf{path}(s) : \mathsf{trace}(\pi) \in L(f)\}$$

It was shown in [5] that when f is an ω-regular property then the set $\{\pi \in \mathsf{path}(s) : \mathsf{trace}(\pi) \vDash f\}$ is measurable. Hence we can write:

$$\Pr(s \vDash f) \;=\; \Pr\{\pi \in \mathsf{path}(s) : \mathsf{trace}(\pi) \in L(f)\}$$

2.3.2 Branching-timetime temporal logic

Branching-time logic revel in the assumption that, "the future can take different possible courses from the present moment". Thus the models of branching-time temporal logic are not linear as in linear temporal logics, but partially ordered tree like structure.

Kripke structures forms the natural models for branching-time logic (though they were conceived to give possible world semantics to Modal logic [69]). A Kripke structure K is a tuple (S, P, L), where S is a (countable) set of states, $P \subseteq S \times S$ is the transition relation and $L \in \Sigma^S$ is a

labeling function. A path of a Kripke structure $\pi \in S^{\mathbb{N}}$ is a sequence of states $(s_1, s_2, \cdots)$ such that for all $i > 1$, $(s_i, s_{i+1}) \in P$. Let $\mathsf{path}(s)$ denote the set of paths starting from the state s.

Computation tree logic (CTL)

We will now define an important branching-time logic called *Computation Tree Logic (CTL)*.

Definition 2.3.5. The syntax of computation tree logic is as follows. The state formula is defined by the following grammar:

$$f \coloneqq a \mid \sim f \mid f \wedge f \mid \exists g,$$

where g is a path formula and path formulas are defined by the following grammar:

$$g \coloneqq \mathsf{X} f \mid f \,\mathsf{U}\, f.$$

where f is a state formula. The pointed satisfaction relation is define on a Kripke structure K and one of its state s.

$$\begin{array}{lll} (K,s) \vDash a & \text{iff} & a \in L(s), \\ (K,s) \vDash \sim f & \text{iff} & (K,s) \nvDash f, \\ (K,s) \vDash f_1 \wedge f_2 & \text{iff} & (K,s) \vDash f_1 \text{ and } (K,s) \vDash f_2 \\ (K,s) \vDash \exists g & \text{iff} & \exists \pi \in \mathsf{path}(s) : (\pi, 0) \vDash g. \end{array}$$

A path π satisfies a path formula iff:

$$\begin{array}{lll} (\pi,i) \vDash \mathsf{X} f & \text{iff} & \pi(i+1) \vDash f, \\ (\pi,i) \vDash f_1 \,\mathsf{U}\, f_2 & \text{iff} & \exists k : \forall j < k : \pi(k) \vDash f_2 \text{ and } \pi(j) \vDash f. \end{array}$$

For a given Kripke structure K and formula f, let $[\![f]\!]_K = \{s : K, s \vDash f\}$. $[\![f]\!]_K$ can also be viewed as a function in $\{0,1\}^S$, where $[\![f]\!]_K(s) = 1$ if and only if $K, s \vDash f$. We will call $[\![f]\!]_K$ the satisfaction function.

Modal μ-calculus

Modal μ-calculus (L_μ), introduced by Dexter Kozen [65], is a more expressive logic that encompasses CTL. In simplest terms, modal μ-calculus is modal logic augmented with greatest and least fixed point operators (here we only consider unimodal operators). The models of L_μ are Kripke structures. Let $V = \{x, y, \cdots\}$ be an enumerable set of variables. These variables are interpreted over functions, i.e., a function from the set of states (say S of a Kripke structure) to a real set (in this case it is $\{0,1\}$). The syntax of the logic is as follows:

$$f \coloneqq x \mid \sim x \mid f \wedge f \mid f \vee f \mid \Box f \mid \Diamond f \mid \mu x.f \mid \nu y.f$$

where the variable x in $\mu x.f$ and $\nu x.f$ occur only positively in f. The fixed point operators are viewed as quantifiers, and the standard terminology and notation used with quantifiers are used. Variables occurring freely in a formula f is denoted by $\mathsf{free}(f)$. Normal scope-rule for quantifiers is adopted. A *fixed point formula* is a L_μ formula where a fixed point operator is the outermost connective.

It is instructive to view a L_μ formula f as a function which assigns values to each state of a Kripke structure for a given interpretation of the free variables in f. For defining the semantics of L_μ, we will need the following mathematical constructs. Let $f, f_1, \cdots, f_n$ be L_μ formula with $\{x_1, \cdots, x_n\} \subseteq \mathsf{free}(f)$, where each x_i occurs positively in f. Then

$$f[f_1/x_1, \cdots, f_n/x_n]$$

denotes the formula that is obtained from f by simultaneously substituting f_i for x_i in f. We will sometimes use the *lambda notation* for a function description, i.e, a function $\varphi(x)$ with parameter x can be written as $\lambda x.\varphi$. Let φ be a function in X^Y, where X and Y are partial orders. φ is monotonic if for all $a, a' \in Y$, $a \prec a'$ implies $\varphi(a) \prec \varphi(a')$. Let φ be a monotone function over a complete lattice L, i.e., $\varphi \in L^L$. By Knaster-Tarski fixed point theorem we have:

Theorem 2.3.4. *If φ is monotonic then:*

1. *$\bigsqcup\{a : a \prec \varphi(a)\}$ is the greatest fixed point of φ.*
2. *$\bigsqcap\{a : a \succ \varphi(a)\}$ is the least fixed point of φ.*

Observe that the set of functions from the set of states S to the real set $\{0, 1\}$ forms a complete lattice. That is, for $\varphi, \varphi' \in \{0,1\}^S$, $\varphi \sqcap \varphi' = \lambda s. \min\{\varphi(s), \varphi'(s)\}$ and $\varphi \sqcup \varphi' = \lambda s. \max\{\varphi(s), \varphi'(s)\}$. We are now ready to define the semantics of a L_μ formula. Let I be an interpretation of variables and $K = (S, P, L)$:

$$\begin{array}{lcllcl}
[\![x]\!]^I_K & = & I(x) & [\![\sim x]\!]^I_K & = & \mathbf{1} - I(x) \\
[\![f_1 \wedge f_2]\!]^I_K & = & [\![f_1]\!]^I_K \sqcap [\![f_2]\!]^I_K & [\![f_1 \vee f_2]\!]^I_K & = & [\![f_1]\!]^I_K \sqcup [\![f_2]\!]^I_K \\
[\![\Diamond f]\!]^I_K & = & \lambda s. \max_{(s,s') \in P} [\![f]\!]^I_K(s') & [\![\Box f]\!]^I_K & = & \lambda s. \min_{(s,s') \in P} [\![f]\!]^I_K(s') \\
[\![\mu x.f]\!]^I_K & = & \bigsqcap\{\varphi : [\![f]\!]^{I[\varphi/x]}_K \leq \varphi\} & [\![\nu x.f]\!]^I_K & = & \bigsqcup\{\varphi : \varphi \leq [\![f]\!]^{I[\varphi/x]}_K\}
\end{array}$$

We state without proof that each of the logical operators which can be viewed as functions on the set $\{0,1\}^S$ are monotonic (it is thus necessary for fixed point variables to occur positively in its scope). Hence from Theorem 2.3.4, we know that greatest and the least fixed points exist and the semantics is well defined.

Note that, if f has no free variable then the choice of the interpretation is immaterial. The pointed satisfaction of a L_μ formula f at a state s of a Kripke structure K with interpretation I, is defined as

$$(K, I, s) \vDash f \text{ iff } [\![f]\!]^I_K(s) = 1$$

If K is understood from the context and f has no free variable then we simply write $s \vDash f$.

Example 2.3.1. Consider the following formula $f = \nu x.(\Box x \wedge \Diamond \mathsf{true})$. A state $s \vDash f$ if and only if s cannot reach any deadlock state. Whereas, the formula $\nu x.(\Box x)$ is satisfied by every state s.

Next we define the *alternation depth* of a formula. The alternation depth is an index of complexity of a L_μ formula. Let $f = \delta x.f'$, where $\delta \in \{\mu, \nu\}$. A sub-formula g of f is a *active* sub-formula, iff $g \neq f$ and the variable x appears in g.

- For a least fixed point formula f, the alternation depth of f is $\mathsf{alt}(f) = 0$ if it does not have any greatest fixed point sub-formula. Otherwise, $\mathsf{alt}(f) = 1 + \max\{\mathsf{alt}(g) : g \text{ is an active greatest fixed point formula of } f\}$.
- For a greatest fixed point formula f, the alternation depth of f is $\mathsf{alt}(f) = 0$ if it does not have any least fixed point sub-formula. Otherwise, $\mathsf{alt}(f) = 1 + \max\{\mathsf{alt}(g) : g \text{ is a active least fixed point formula of } f\}$.
- For any formula f, $\mathsf{alt}(f) = \max\{\mathsf{alt}(g) : g \text{ is a fixed point sub-formula of } f\}$.

Example 2.3.2. For $f = \nu x.(\mu y.(p \vee \Diamond y) \wedge \Box x)$, $\mathsf{alt}(f) = 0$, whereas for $f' = \nu x.(\mu y.(\Box x \vee \Diamond y))$, $\mathsf{alt}(f') = 1$.

2.3.3 Probability and logic

The logic that we have studied so far are qualitative in the sense, that the satisfaction function returned value either 0 or 1. Even when we were measuring the set of paths that satisfies a given linear temporal property, we where only considering paths for which the satisfaction function returned 1. In this section we consider logic where the satisfaction function can have real values in $[0, 1]$. To prepare for this paradigm shift, we first consider Probabilistic CTL (or PCTL) where the quantitative aspects are hidden and the satisfaction function is still boolean.

Probabilistic Computation Tree Logic Probabilistic CTL (PCTL) [53] is a probabilistic extension of the well-known branching-time logic for specifying properties of stochastic systems. In PCTL, the existential and universal path quantifiers of CTL are replaced with the probabilistic operator which allows to quantify the probability of all runs that satisfy a given path formula. The syntax of PCTL is built upon atomic propositions, using Boolean connectives and operators *next* and *until* of the form $[\mathsf{X}f]_{\bowtie p}$ and $[f\,\mathsf{U}\,g]_{\bowtie p}$, respectively, where $\bowtie \in \{\leq, <, \geq, >\}$, and $p \in [0,1]$ is a rational constant. Other operators such as F, G and W can be derived from U. It has the following syntax:

$$\begin{array}{lcl} f & ::= & a \mid {\sim}f \mid f \wedge f \mid [g]_{\bowtie p} \\ g & ::= & \mathsf{X}f \mid f\,\mathsf{U}\,f \end{array}$$

where $a \in AP$, f is called a state formula, g is called a path formula, $\bowtie \in \{<, \leq, >, \geq\}$ and p is a rational number in $[0,1]$. The *qualitative* fragment of PCTL allows sub-formulas with only two kinds of probability operators, $[g]_{>0}$ and $[g]_{=1}$. The PCTL semantics is define on Markov chains. An MC $M = (S, \mathsf{P}, \mathsf{AP}, L)$ satisfies a state formula f at a state s if:

$$\begin{array}{lll} M, s \vDash a & \text{iff} & a \in L(s) \\ M, s \vDash {\sim}f & \text{iff} & M, s \nvDash f \\ M, s \vDash f_1 \wedge f_2 & \text{iff} & M, s \vDash f_1 \text{ and } M, s \vDash f_2 \\ M, s \vDash [g]_{\bowtie p} & \text{iff} & Pr\{s \vDash g\} \bowtie p, \end{array}$$

where $\{s \vDash g\} = \{w \;:\; w_0 = s \text{ and } M, w \vDash g\}$. A path formula g is true for a path w of M if:

$$\begin{array}{lll} M, w \vDash \mathsf{X}f & \text{iff} & M, w_1 \vDash f \\ M, w \vDash f_1\,\mathsf{U}\,f_2 & \text{iff} & \exists i : M, w_i \vDash f_2 \text{ and } \forall j < i : M, w_j \vDash f_1 \end{array}$$

We will denote the satisfaction relation by $s \vDash f$ (and $w \vDash g$) when M is fixed. Observe that satisfaction function is still boolean, i.e., for any state s and formula f, $[\![f]\!]_M(s) \in \{0,1\}$. Let us consider path sub-formulas in little more detail. Path formulas are linear temporal formulas and hence have linear sequence (traces of a paths) as models. But observe that, path formulas are always quantified by a probabilistic operator $[g]_{\bowtie p}$, and the satisfaction function is defined on $[g]_{\bowtie p}$. We can enrich the satisfaction function by defining $[\![g]\!]_M(s)$ as the probability of set of paths from s that satisfies g. That is:

$$[\![g]\!]_M(s) \;=\; Pr\{s \vDash g\}$$

The satisfaction function can now have value in $[0,1]$. We will extend this idea to give a quantitative semantics to probabilistic μ-calculus.

Probabilistic μ-calculus

The probabilistic μ-calculus presented in the thesis is a generalization of modal μ-calculus by allowing quantification over path formulas. Though this a subset of the probabilistic μ-calculus $p\mathsf{L}\mu^{\odot}_{\oplus}$ proposed by Matteo Mio [80], the semantics is almost identical to L_μ and is easier to understand. It also gives a clean separation of formulas with boolean valued satisfaction function from formulas with real valued satisfaction function.

Let $M = (S, \mathsf{P}, \mathsf{AP}, L)$ be a Markov chain. As before, let $V = \{x, y, z, \cdots\}$ be an enumerable set of variables, where the variables are interpreted over set of function $[0,1]^S$. The syntax of the logic is given by the following grammar, where f is a state formula and g is a path formula.

$$\begin{array}{rcl} f & ::= & a \mid \sim a \mid f \wedge f \mid f \vee f \mid [\mathsf{X} f]_{\bowtie p} \mid \nu x.f \mid \mu x.f \\ g & ::= & f \mid x \mid f \wedge f \mid f \vee f \mid \Diamond f \mid \Box f \mid \nu x.f \mid \mu x.f \end{array}$$

The semantics of the logic is inductively defined as follows:

$$\begin{array}{lcllcl} [\![a]\!]^I_M & = & \mathbf{1}_{L^{-1}(a)} & [\![\sim a]\!]^I_M & = & \mathbf{1} - [\![a]\!]^I_M \\ [\![f_1 \wedge f_2]\!]^I_M & = & \min\{[\![f_1]\!]^I_M, [\![f_2]\!]^I_M\} & [\![f_1 \vee f_2]\!]^I_M & = & \max\{[\![f_1]\!]^I_M, [\![f_2]\!]^I_M\} \\ [\![\Diamond f]\!]^I_M & = & \lambda s. \max\limits_{(s,s')\in P} [\![f]\!]^I_M(s') & [\![\Box f]\!]^I_M & = & \lambda s. \min\limits_{(s,s')\in P} [\![f]\!]^I_M(s') \\ [\![x]\!]^I_M & = & I(x) & [\![\mathsf{X} f]\!]^I_M & = & \lambda s. \sum\limits_{s'\in S} \mathsf{P}(s,s') \cdot [\![f]\!]^I_M(s) \\ [\![\mu x.f]\!]^I_M & = & \bigsqcap\{\varphi : [\![f]\!]^{I[\varphi/x]}_M \leq \varphi\} & [\![\nu x.f]\!]^I_M & = & \bigsqcup\{\varphi : \varphi \leq [\![f]\!]^{I[\varphi/x]}_M\} \end{array}$$

It is not difficult to see that the satisfaction function for a state formula will have value in $\{0,1\}$, whereas for path formula it can have any real value in the range $[0,1]$. PCTL can be easily encoded in this logic. For example, $a \,\mathsf{U}\, b$ can be defined as $\mu y.b \vee (a \wedge \mathsf{X} y)$. Similar to L_μ, for a state formula f, we say $M, I, s \vDash f$ if and only if $[\![f]\!]^I_M(s) = 1$. If the formula has no free variable and the Markov chain is understood from the context, then we simply write $s \vDash f$. We conclude this section with the following important theorem.

Theorem 2.3.5. *[18] The problem whether $s \vDash f$ for a state s of a finite Markov chain M and a state formula f can be decided in NP∩co-NP.*

Chapter 3

Parametric Linear Time Temporal Logic

In this chapter we will consider the parameterized version of linear time temporal logic (pLTL). We determine the set of parameter valuations $V_{\prec p}(\varphi)$ for which the probability of the set of paths of a Markov chain that satisfies the pLTL-formula φ is above (or below) some threshold p, where is some rational in $[0, 1]$. Since determining the emptiness of $V_{>0}(\varphi)$ for any arbitrary pLTL formula φ is undecidable, we look at several fragments of the logic. We consider parametric reachability properties, then a sub-logic of pLTL restricted to next and $\mathsf{F}_{\leqslant x}$, parametric Büchi properties and finally, a maximal subclass of pLTL for which emptiness of $V_{>0}(\varphi)$ is decidable.

3.1 Introduction

Parameterization of the quantitative specifications is a type of abstraction which allows the system designer to express quantitative information about the system under study. Parametrized specifications may be necessary when the quantitative aspects of the specifications change over different environmental conditions, (thus instilling robustness to the system design), or when the precise information is absent in the initial phases of the system development. For example, consider the request-response property of a server-client model. We want to verify that: "*For every request sent by the client it is ultimately received, the server eventually responses to every received request*". This can be easily represented in LTL as:

$$\mathsf{G}(\text{req} \to \mathsf{F}(\text{rec})) \to \mathsf{G}(\text{rec} \to \mathsf{F}(\text{res}))$$

But realistically, we expect the response to happen within a certain time period of the received request, where the precise length of the time period

may vary. This type of specifications can be expressed in *parametric linear time temporal logic* (pLTL, for short).

$$\mathsf{G}(\text{req} \to \mathsf{F}(\text{rec})) \to \mathsf{G}(\text{rec} \to \mathsf{F}_{\leq x}(\text{res}))$$

This says that whenever a request (req) is sent, it is received (rec) and the response (res) happens within x-steps of the received request.

In pLTL [2], temporal operators can be subscripted by variables ranging over the natural numbers. The formula '$\mathsf{F}_{\leqslant x}\, a$' means that in at most x steps 'a' occurs, and '$\mathsf{GF}_{\leqslant y}\, a$' means that at every index 'a' occurs within y steps. Note that x and y are variables whose value is not fixed in advance. The central problem of this chapter is to determine the values of x and y such that the probability of the set of paths of a given Markov chain satisfying a given pLTL-formula φ meets a certain threshold p. This is referred to as the valuation set $V_{\sim p}(\varphi)$ for comparison operator $\sim$. This problem has both a qualitative (threshold > 0 and $= 1$) and a quantitative variant ($0 < p < 1$). Since determining the emptiness of $V_{>0}(\varphi)$ for any arbitrary pLTL formula φ is undecidable, we look at several fragments of the logic. We consider parametric reachability properties, then a sub-logic of pLTL restricted to next and $\mathsf{F}_{\leqslant x}$, parametric Büchi properties and finally, a maximal subclass of pLTL for which emptiness of $V_{>0}(\varphi)$ is decidable.

3.2 Parameter synthesis Problem for pLTL

In this section we define the syntax and semantics of pLTL. We describe the parameter synthesis problem for pLTL on labeled Markov chains.

Parametric LTL.

Parametric LTL extends propositional LTL with bounded temporal modalities, for which the bound is either a constant or a variable. Let *Var* be a finite set of variables ranged over by x, y, and AP be a finite set of propositions ranged over by a and b. Let $c \in \mathbb{N}$. Parametric LTL formulas adhere to the following syntax:

$$\varphi \ ::= \ a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi\, \mathsf{U}\, \varphi \mid \mathsf{F}_{\sim x}\, \varphi \mid \mathsf{F}_{\sim c}\, \varphi$$

where $\sim \in \{=, \leqslant, <, >, \geqslant\}$. A pLTL structure is a triple $(w, i, \boldsymbol{v})$ where $w \in \Sigma^\omega$ with $\Sigma = 2^{\mathsf{AP}}$ is an infinite word over sets of propositions, $i \in \mathbb{N}$ is an index, and $\boldsymbol{v} : \mathit{Var} \to \mathbb{N}$ is a variable valuation. Analogously, we consider a valuation $\boldsymbol{v}$ as a vector in $\mathbb{N}^d$, where d for a pLTL formula φ is the number of variables occurring in φ. We compare valuations $\boldsymbol{v}$ and $\boldsymbol{v}'$ as $\boldsymbol{v} \leqslant \boldsymbol{v}'$ iff $\boldsymbol{v}(x) \leqslant \boldsymbol{v}'(x)$

for all x. Let $w[i]$ denote the i-th element of w. The satisfaction relation $\vDash$ is defined by structural induction over φ as follows:

$$\begin{array}{lll}
(w,i,\boldsymbol{v}) \vDash a & \text{iff} & a \in w[i] \\
(w,i,\boldsymbol{v}) \vDash \sim\varphi & \text{iff} & (w,i,\boldsymbol{v}) \nvDash \varphi \\
(w,i,\boldsymbol{v}) \vDash \varphi_1 \wedge \varphi_2 & \text{iff} & (w,i,\boldsymbol{v}) \vDash \varphi_1 \text{ and } (w,i,\boldsymbol{v}) \vDash \varphi_2 \\
(w,i,\boldsymbol{v}) \vDash \mathsf{F}_{\prec x}\,\varphi & \text{iff} & (w,j,\boldsymbol{v}) \vDash \varphi \text{ for some } j \prec \boldsymbol{v}(x)+i.
\end{array}$$

For the sake of brevity, we have omitted the semantics of the standard LTL modalities, which can be found in the preliminaries. As usual, $\varphi_1 \mathsf{R} \varphi_2 \equiv \sim(\sim\varphi_1 \,\mathsf{U}\, \sim\varphi_2)$, $\mathsf{F}\varphi \equiv \mathsf{true}\,\mathsf{U}\,\varphi$ and $\mathsf{G}\varphi \equiv \sim\mathsf{F}\sim\varphi$. The language of φ is defined by $\mathcal{L}(\varphi) = \{(w,\boldsymbol{v}) \,:\, (w,0,\boldsymbol{v}) \vDash \varphi\}$. Alur et al. [2] have shown that other modalities such as $\mathsf{U}_{\leqslant x}$, $\mathsf{F}_{>x}$, $\mathsf{G}_{>x}$, $\mathsf{U}_{>x}$, $\mathsf{R}_{\leqslant x}$ and $\mathsf{R}_{>x}$, can all be encoded in our syntax. For instance, the following equivalences hold:

$$\begin{array}{rclcrcl}
\mathsf{F}_{>x}\varphi & \equiv & \mathsf{G}_{\leqslant x}\,\mathsf{F}\mathsf{X}\varphi, & & \mathsf{G}_{>x}\varphi & \equiv & \mathsf{F}_{\leqslant x}\,\mathsf{G}\mathsf{X}\varphi, \\
\varphi\,\mathsf{U}_{\leqslant x}\,\psi & \equiv & (\varphi\,\mathsf{U}\,\psi) \wedge \mathsf{F}_{\leqslant x}\,\psi, & \quad \varphi\,\mathsf{U}_{>x}\,\psi & \equiv & \mathsf{G}_{\leqslant x}\,(\varphi \wedge \mathsf{X}(\varphi\,\mathsf{U}\,\psi))
\end{array} \tag{3.1}$$

For valuation $\boldsymbol{v}$ and pLTL-formula φ, let $\boldsymbol{v}(\varphi)$ denote the LTL formula obtained from φ by replacing variable x by its valuation $\boldsymbol{v}(x)$; e.g., $\boldsymbol{v}(\mathsf{F}_x\,\varphi)$ equals $\mathsf{F}_{\boldsymbol{v}(x)}\,\boldsymbol{v}(\varphi)$.

Valuation set.

The central problem addressed in this chapter is to determine the valuation set of a pLTL formula φ. Let $M = (S, \mathsf{P}, s_0, L)$ be an MC, $p \in [0,1]$ a probability bound, and $\prec \in \{=, \leqslant, <, >, \geqslant\}$. Then we are interested in determining:

$$V_{\prec p}(\varphi) \;=\; \{\boldsymbol{v} \;:\; \Pr(M, s_0 \vDash \boldsymbol{v}(\varphi)) \prec p\},$$

where $\Pr(M, s_0 \vDash \boldsymbol{v}(\varphi)) = \Pr\{\pi \in \mathsf{Path}(s_0) : (\mathsf{trace}(\pi), \boldsymbol{v}) \vDash \varphi\}$. i.e., the set of valuations under which the probability of satisfying φ meets the bound $\prec p$. In particular, we will focus on the decidability and complexity of the emptiness problem for $V_{\prec p}(\varphi)$, i.e., the decision problem whether $V_{\prec p}(\varphi) = \varnothing$ or not, on algorithms (if any) determining the set $V_{\prec p}(\varphi)$, and on the size of the minimal representation of $V_{\prec p}(\varphi)$. In the qualitative setting, the bound $\prec p$ is either > 0, or $= 1$.

Proposition 3.2.1. *For $\varphi \in$ pLTL and MC M, the problem if $V_{>0}(\varphi) = \varnothing$ is undecidable.*

Proof. The proof is based on [2, Th. 4.1], where the problem of deciding the existence of a halting computation of a two-counter machine[1] is reduced

[1]Formal description of two counter machine can be found in the appendix A.

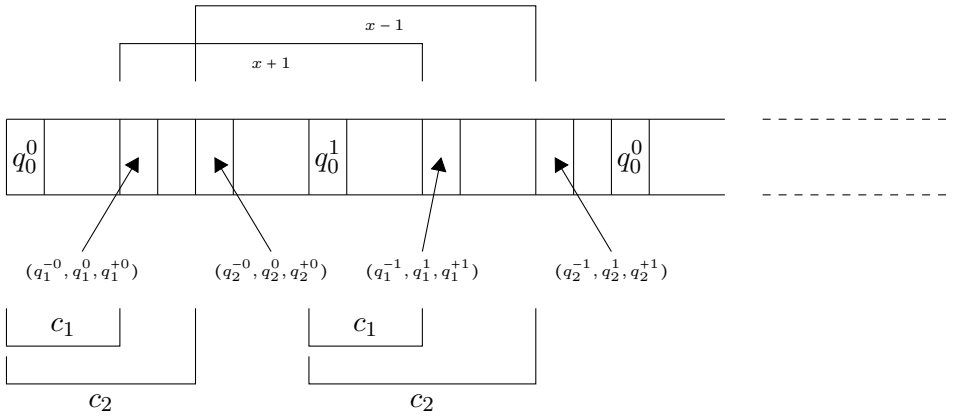


Figure 3.2.1: Shows a word satisfying the formula φ_T.

to the emptiness problem of $\mathcal{L}(\varphi)$ (i.e. whether there exists a path and a valuation pair that satisfies the formula).

Let T be a counter machine with two counters $\{c_1, c_2\}$ and $k+1$ states $\{s_0, s_1, \cdots, s_k\}$, s_0 being the initial state and s_k the halting state. We construct a pLTL formula φ_T with a single parameter x such that any satisfiable structure $(w, \boldsymbol{v})$, represents a sequence of configurations of T that constitute a halting computation. In other words, the sequence of letters in the word w, will encode a halting computation of T for the valuation $\boldsymbol{v}$. Crux of the reduction is that, the parameter x is used to guess the maximum value of each counter in any halting computation of T. Thus, each configuration of a halting computation can be stored in length x.
We have propositions p_j for each state s_j and propositions

$$\{q_i^{-b}, q_i^b, q_i^{+b}, q_i^{-\bar{b}}, q_i^{\bar{b}}, q_i^{+\bar{b}}\}$$

to keep track of the value of counter c_i, for $i \in \{1, 2\}$, and $\{q_0^b, q_0^{\bar{b}}\}$ to keep track of the start and end of the encodings of the configurations, where if b is 0 (or 1) and $\bar{b}$ is 1 (or 0, respectively). A configuration of the counter machine, is stored in a substring of a word w, and $\{q_0^0, q_0^1\}$ denoting the start and end of a configuration. To be precise, if the encoding of a configuration starts with q_0^0 at position i and ends with q_0^1 at position j, then the encoding of the next configuration starts with q_0^1 at position j and ends with q_0^0 in some position k (so on and so forth) (see Figure 3.2.1). The distance between q_0^0, q_0^1 is exactly x. This is imposed by the formula:

$$\varphi_1 := \bigwedge_b \left(q_0^b \rightarrow \mathsf{X}(\sim q_0^{\bar{b}} \,\mathsf{U}_{=x}\, q_0^{\bar{b}}) \wedge \mathsf{X}(q_0^b \,\mathsf{U}\, q_0^{\bar{b}}) \right).$$

The propositions $\{q_1^{-0}, q_1^0, q_1^{+0}\}$ (or $\{q_1^{-1}, q_1^1, q_1^{+1}\}$) will be used to keep track of counter c_1 in the configuration starting with q_0^0 (or q_0^1, respectively). Similarly, $\{q_2^{-0}, q_2^0, q_2^{+0}\}$, $\{q_2^{-1}, q_2^1, q_2^{+1}\}$ do the same for counter c_2. We impose the condition that all these propositions occur exactly once between q_0^0 and q_0^1, and $\{q_i^{-b}, q_i^b, q_i^{+b}\}$ $(i = 1, 2)$ always occur consecutively.

$$\varphi^{bi} \coloneqq \Big((q_0^b \to {\sim}q_i^{\bar{b}} \,\mathsf{U}\, q_i^b) \wedge (q_i^b \to \mathsf{X}({\sim}q_i^b \,\mathsf{U}\, q_0^b)) \wedge (q_i^{+b} \to \mathsf{X}({\sim}q_i^{+b} \,\mathsf{U}\, q_0^b)) \\ \wedge (q_i^{-b} \to \mathsf{X}({\sim}q_i^{-b} \,\mathsf{U}\, q_0^b)) \wedge (q_i^b \to \mathsf{X}q_i^{+b}) \wedge (q_i^{-b} \to \mathsf{X}q_i^b) \Big).$$

Let $\varphi_2 \coloneqq \bigwedge_{b,i=1,2} \varphi^{bi}$. Consider configuration (s_i, c_1, c_2) of T. If this configuration occurs in a halting computation, then it is encoded in w as a sub-sequence of propositions (of length x) between q_0^0 and q_0^1. Exactly one of the state propositions, p_i in this case is true at the start of the configuration q_0^b. This is imposed by:

$$\varphi_3 \coloneqq \bigwedge_b \Big(q_0^b \to P \wedge \mathsf{X}(P' \,\mathsf{U}\, q_0^b) \Big)$$

where $P \coloneqq (p_1 \wedge {\sim}p_2 \wedge \cdots \wedge {\sim}p_k) \vee \cdots \vee ({\sim}p_1 \wedge \cdots \wedge {\sim}p_{k-1} \wedge p_k)$ and $P' \coloneqq ({\sim}p_1 \wedge \cdots \wedge {\sim}p_k)$. The distance of q_1^b from q_0^b will be used to keep track of the value of c_1. To be precise, at a distance c_1 from q_0^b the sequence $q_1^{-b}, q_1^b, q_1^{+b}$ occurs (Figure 3.2.1). Similarly, for the second counter.

Consider a transition $e = (s_i \xrightarrow{c_1 \coloneqq c_1 + 1} s_j)$. So if we are in a configuration where the distance of q_1^{-b} from q_0^b is c_1 then in the next configuration, the distance of $q_1^{-\bar{b}}$ from $q_0^{\bar{b}}$ is $c_1 + 1$ or the distance of q_1^b to $q_1^{-\bar{b}}$ is x. This can be encoded as:

$$\varphi_e \coloneqq \bigwedge_b \Big((q_0^b \wedge p_i) \to ({\sim}q_0^{\bar{b}} \,\mathsf{U}\, p_j \wedge {\sim}q_0^{\bar{b}} \,\mathsf{U}\, (q_1^b \to \mathsf{X}({\sim}q_1^b \,\mathsf{U}_{=x}\, q_1^{-\bar{b}}))) \Big)$$

A similar formula can be defined for transitions where the counter is decremented. For a transition where a counter value is compared to 0, the the proposition denoting the start of a configuration, say q_0^b is followed by q_1^{-b}. That is for, $e = (s_i \xrightarrow{c_1 = 0} s_j)$ is encoded as:

$$\varphi_e \coloneqq \bigwedge_b \Big((q_0^b \wedge p_i) \to ({\sim}q_0^{\bar{b}} \,\mathsf{U}\, p_j \wedge \mathsf{X}q_1^{-b}) \Big).$$

Thus, the entire transition relation of T can be encoded as $\varphi_4 \coloneqq \bigvee_e \varphi_e$.

$$\varphi_T \coloneqq q_0^0 \wedge (\bigwedge_{i=1}^{4} \varphi_i) \,\mathsf{U}\, p_k.$$

As a satisfiable structure of φ_T encodes a halting computation of T (vice-versa), satisfiability of φ_T becomes undecidable. Furthermore, if $(w, \boldsymbol{v})$ satisfies φ_T then p_k is true at some *finite* length of w. We can easily construct a Markov chain M such that the set of finite traces of M is Σ^* (Σ^* is the set of sets of propositions used). We know that the probability measure of any finite trace of M is greater than 0. Thus, we can decide whether φ_T is satisfiable iff we can decide $\Pr(M \vDash \boldsymbol{v}(\varphi_T)) > 0$ for some valuation $\boldsymbol{v}$. Hence, we conclude that the emptiness problem of $V_{>0}(\varphi)$ is undecidable. □

Since $V_{>0}(\varphi) = \varnothing$ iff $V_{=1}(\sim\varphi) \neq \varnothing$, it follows that deciding whether $V_{=1}(\varphi) = \varnothing$ is undecidable. As a combination of $\mathsf{F}_{\leqslant x}$ and $\mathsf{G}_{\leqslant x}$ modalities can encode $\mathsf{U}_{=x}$, e.g.,

$$\sim a \wedge \mathsf{X}(\sim a \,\mathsf{U}_{=x}\, a) \;\equiv\; \mathsf{X}(\sim a \,\mathsf{U}_{\leqslant x}\, a) \wedge (\sim a \,\mathsf{U}_{>x}\, a),$$

where $\mathsf{U}_{>x}$ and $\mathsf{U}_{\leqslant x}$ can be expressed by $\mathsf{F}_{\leqslant x}$ and $\mathsf{G}_{\leqslant x}$ (Equations 3.1), we will restrict ourselves to fragments of pLTL where each formula is in negative normal form and the only parametrized operator is $\mathsf{F}_{\leqslant x}\,\varphi$. We refer to this fragment as $\mathrm{pLTL}_{\mathsf{F}}$:

$$\varphi \;::=\; a \mid \sim a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \varphi \mathsf{R} \varphi \mid \mathsf{G}\varphi \mid \mathsf{F}_{\leqslant x}\,\varphi \mid \mathsf{F}_{\leqslant c}\,\varphi \mid \mathsf{G}_{\leqslant c}\,\varphi. \quad (3.2)$$

We show it is a sub-logic of pLTL for which the emptiness problem for $V_{>0}(\varphi)$ is decidable. The logic has a favourable *monotonicity* property, i.e.,

Remark. For every $\mathrm{pLTL}_{\mathsf{F}}$-formula φ, infinite word w and valuations $\boldsymbol{v}, \boldsymbol{v}'$, $\boldsymbol{v} \leqslant \boldsymbol{v}'$ implies $(w, \boldsymbol{v}) \vDash \varphi \implies (w, \boldsymbol{v}') \vDash \varphi$.

Here $(w, \boldsymbol{v}) \vDash \varphi$ is a shorthand for $(w, 0, \boldsymbol{v}) \vDash \varphi$.

3.3 Parameter synthesis for Markov chains

We start off briefly with (only) parametric reachability and then consider the sub-logic $\mathrm{pLTL}(\mathsf{F}, \mathsf{X})$ restricted to X and F_x. Later on, we also consider parametric Büchi formulas, and finally, $\mathrm{pLTL}_{\mathsf{F}}$.

3.3.1 Parametric reachability

In this section, we consider pLTL-formulas of the form $\mathsf{F}_x\, a$, where $a \in \mathsf{AP}$, and MC $M = (S, \mathsf{P}, s_0, L)$. Define the set of target states $T = \{s \in S \;:\; a \in L(s)\}$. We consider bounds of the form $\geqslant p$ with $0 < p \leqslant 1$, and we are interested in $V_{\geqslant p}(\mathsf{F}_x\, a)$. Let μ_i be the probability of reaching T within i steps; the sequence $\{\mu_i\}_{i \in \omega}$ is ascending. There can be two cases: (a) the

sequence reaches a constant value in m steps ($m = |S|$) or (b) the sequence monotonically increases and converges to μ_∞. This gives us an algorithm for checking emptiness of $V_{\geqslant p}(\mathsf{F}_x\, a)$. In the first case, we check $\mu_m \geqslant p$, and in the second case, emptiness can be checked in time polynomial in the size of the MC, by determining $\mu_\infty = \Pr(s_0 \vDash \mathsf{F}a)$ which can be done by solving a system of linear equations with at most m variables. Then, $V_{\geqslant p}(\mathsf{F}_x\, a) \neq \varnothing$ iff $p < \mu_\infty$.

Assume in the sequel that T is non-empty. Let $\min V_{\geqslant p}(\mathsf{F}_x\, a) = n_0$, then the valuation set can be represented by n_0 (this gives a minimal representation of the set). A membership query, i.e., does $n \in V_{\geqslant p}(\mathsf{F}_x\, a)$, then simply boils down to checking whether $n_0 \leqslant n$, which can be done in constant time (modulo the size of n_0). The only catch is that n_0 can be very large if p is close to μ_∞. A simple example elucidates this fact.

Example 3.3.1. Consider the MC M with $S = \{s_0, t\}$, $L(t) = \{a\}$, $L(s_0) = \varnothing$, $\mathsf{P}(s_0, s_0) = \frac{1}{2} = \mathsf{P}(s_0, t)$ and $\mathsf{P}(t, t) = 1$. Then $\Pr(M \vDash \mathsf{F}_n\, a) = 1 - \left(\frac{1}{2}\right)^n$. It follows that $\min V_{\geqslant p}(\mathsf{F}_x\, a)$ goes to infinity when p approaches one.

An upper bound for n_0 can nonetheless be provided. This bound allows for obtaining the minimum value n_0 by a binary search.

Proposition 3.3.1. *For MC M, $\min V_{\geqslant p}(\mathsf{F}_x a) \leqslant \log_\gamma(1 - (1-\gamma)\frac{p}{b})$, where $0 < \gamma < 1$ and $b > 0$, when $p < \Pr(s_0 \vDash \mathsf{F}_x a)$.*

Proof. Collapse all a-states into a single state t and make it absorbing (i.e., replace all outgoing transitions by a self-loop with probability one). Let t be the only *bottom strongly connected component* (BSCC) of M (other BSCCs can be safely ignored). Let $\{1, \cdots, m\}$ be the states of the modified MC M, with the initial state s_0 and the target state t represented by 1 and m, respectively. Let $\mathbf{Q}$ be the $(m-1) \times (m-1)$ transition matrix of the modified MC without the state t. That is, $\mathbf{Q}(i, j) = \mathbf{P}(i, j)$ iff $j \neq m$ where $\mathbf{P}$ is the transition probability matrix of M. We have the following observation:

1. Let the coefficient of ergodicity $\tau(\mathbf{Q})$ of $\mathbf{Q}$ defined as

$$\tau(\mathbf{Q}) \;=\; 1 - \min_{i,j}\left(\sum_k \min\{\mathbf{Q}(i,k), \mathbf{Q}(j,k)\}\right).$$

   As $\mathbf{Q}$ is sub-stochastic and no row of $\mathbf{Q}$ is zero, it follows $0 < \tau(\mathbf{Q}) < 1$.

2. Let vector $\mathbf{r}^T = (r_1, \cdots, r_{m-1})$ with $r_i = \mathbf{P}(i, m)$, $r_{\max}$ be the maximum element in $\mathbf{r}$ and $\mathbf{i}^T$ be $(1, 0, \cdots, 0)$. The probability of reaching the state m from the state 1 in at most $n+1$ steps is the probability of being in some state $i < m$ within n steps and taking the next transition to m:

$$\mu_{n+1} = \sum_{j=0}^{n+1} \mathbf{i}^T \mathbf{Q}^j \mathbf{r} \;\leqslant\; \sum_{j=0}^{n+1} \tau(\mathbf{Q})^j r_{\max}.$$

Let $\tau(\mathbf{Q}) = \gamma$ and $r_{\max} = b$. The integer n_0 is the smallest integer such that $\mu_{n_0} \geqslant p$. Using the above results this is equivalent to $b \cdot \frac{1-\gamma^{n_0}}{1-\gamma} \geqslant p$. This yields $n_0 \leqslant \log_\gamma(1 - (1-\gamma)\frac{p}{b})$. □

As in the non-parametric setting, it follows that (for finite MCs) the valuation sets $V_{>0}(\mathsf{F}_x\, a)$ and $V_{=1}(\mathsf{F}_x\, a)$ can be determined by a graph analysis, i.e. no inspection of the transition probabilities is necessary for qualitative parametric reachability properties.

Proposition 3.3.2. *The problem $V_{>0}(\mathsf{F}_x\, a) = \varnothing$ is* NL*-complete.*

Proof. The problem is the same as reachability in directed graphs. □

Proposition 3.3.3. *The sets $V_{>0}(\mathsf{F}_x\, a)$ and $V_{=1}(\mathsf{F}_x\, a)$ can be determined in polynomial time by a graph analysis of MC M.*

Proof. Collapse all the a-states into a target state t and make t absorbing. If $V_{>0}(\mathsf{F}_x\, a)$ is non-empty, it suffices to determine $\min V_{>0}(\mathsf{F}_x\, a)$ which equals the length of a shortest path from s_0 to t. To determine whether $V_{=1}(\mathsf{F}_x\, a)$ is empty or not, we proceed as follows. If a cycle without t is reachable from s_0, then no finite n exists for which the probability of reaching t within n steps equals one. Thus, $V_{=1}(\mathsf{F}_x\, a) = \varnothing$. If this is not the case, then the graph of M is a DAG (apart from the self-loop at t), and $\min V_{=1}(\mathsf{F}_x\, a)$ equals the length of a longest path from s_0 to t. □

3.3.2 The fragment pLTL(F, X)

This section considers the fragment pLTL(F, X) which is defined by:

$$\varphi \;::=\; a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \mathsf{F}\varphi \mid \mathsf{F}_{\leqslant x}\, \varphi \mid \mathsf{F}_{\leqslant c}\, \varphi$$

Our first result is a necessary and sufficient condition for the emptiness of $V_{>0}(\varphi)$.

Theorem 3.3.4. *For $\varphi \in$ pLTL(F, X) and MC M with m states, $V_{>0}(\varphi) \neq \varnothing$ iff $\bar{\boldsymbol{v}} \in V_{>0}(\varphi)$ with $\bar{\boldsymbol{v}}(x) = m \cdot |\varphi|$.*

Proof. The direction from right to left is trivial. Consider the other direction. Let φ be a pLTL(F, X)-formula and assume $V_{>0}(\varphi) \neq \varnothing$. By monotonicity, it suffices to prove that $\boldsymbol{v} \in V_{>0}(\varphi)$ with $\boldsymbol{v} \not\leqslant \bar{\boldsymbol{v}}$ implies $\bar{\boldsymbol{v}} \in V_{>0}(\varphi)$. The proof proceeds in a number of steps. (1) We show that it suffices to

consider formulas without disjunction. (2) We show that if path fragment $\pi[0..l] \models \bar{\varphi}$, (where LTL($\mathsf{F},\mathsf{X}$)-formula $\bar{\varphi}$ is obtained from φ by omitting all parameters from φ) then $\pi[0..l] \models \boldsymbol{v}_l(\varphi)$ with $\boldsymbol{v}_l(x) = l$ for every x. (3) We construct a deterministic Büchi automaton (DBA) $A_{\bar{\varphi}}$ for $\bar{\varphi}$ such that its initial and final state are at most $|\bar{\varphi}|$ transitions apart. (4) We show that reachability of a final state in the product of MC M and DBA $A_{\bar{\varphi}}$ implies the existence of a finite path in M of length at most $m{\cdot}|\varphi|$ satisfying $\bar{\varphi}$.

1. As disjunction distributes over $\wedge, \mathsf{X}, \mathsf{F}$, and F_x, each formula can be written in disjunctive normal form. Let $\varphi \equiv \varphi_1 \vee \ldots \vee \varphi_k$, where each φ_i is disjunction-free. Evidently, $|\varphi_i| \leqslant |\varphi|$. Assume $\boldsymbol{v} \in V_{>0}(\varphi)$. Then, $\boldsymbol{v} \in V_{>0}(\varphi_i)$ for some $0 < i \leqslant k$. Assuming the theorem holds for φ_i (this will be proven below), $\bar{\boldsymbol{v}}_i \in V_{>0}(\varphi_i)$ with $\bar{\boldsymbol{v}}_i(x) = |\varphi_i|{\cdot}m$. Since $\bar{\boldsymbol{v}} \geqslant \bar{\boldsymbol{v}}_i$, it follows by monotonicity that $\bar{\boldsymbol{v}} \in V_{>0}(\varphi_i)$, and hence, $\bar{\boldsymbol{v}} \in V_{>0}(\varphi)$. It thus suffices in the remainder of the proof to consider disjunction-free formulas.

2. For pLTL(F,X)-formula φ, let $\bar{\varphi}$ be the LTL(F,X)-formula obtained from φ by replacing all occurrences of F_x by F, e.g., for $\varphi = \mathsf{F}_x(a \wedge \mathsf{F}_y b)$, $\bar{\varphi} = \mathsf{F}(a \wedge \mathsf{F} b)$. We claim that $\pi[0...l] \models \bar{\varphi}$ implies $\pi[0...l] \models \boldsymbol{v}_l(\varphi)$ with $\boldsymbol{v}_l(x) = l$ for all x. This is proven by induction on the structure of φ. The base cases a and $\neg a$ are obvious. For the induction step, conjunctions, $\mathsf{X}\varphi$ and $\mathsf{F}\varphi$ are straightforward. It remains to consider $\mathsf{F}_x\,\varphi$. Assume $\pi[0...l] \models \mathsf{F}\,\bar{\varphi}$. Thus, for some $i \leqslant l$, $\pi[i...l] \models \bar{\varphi}$. By induction hypothesis, $\pi[i...] \models \boldsymbol{v}_{il}(\varphi)$ with $\boldsymbol{v}_{il}(y) = l{-}i$ for each variable y in φ. Thus, $\pi[0..l] \models \boldsymbol{v}_l(\mathsf{F}_x\,\varphi)$ with $\boldsymbol{v}_l(x) = l$ and for all y in φ, $\boldsymbol{v}_l(y) = l$.

3. We provide a DBA $A_{\bar{\varphi}} = \langle Q, \Sigma, \delta, q_0, F\rangle$ with $\Sigma = 2^{\mathsf{AP}}$ for each LTL(F,X)-formula $\bar{\varphi}$ using the construction from [4]. We first treat $\bar{\varphi} = a$ and $\bar{\varphi} = \mathsf{F}a$. As every LTL($\mathsf{F},\mathsf{X}$)-formula can be obtained from $\mathsf{F}(a \wedge \varphi)$, $\varphi_1 \wedge \varphi_2$ and $\mathsf{X}\varphi$, we then treat these inductive cases. (Negations are treated similarly.) For $\bar{\varphi} = a$, $A_a = \langle\{q_0, q_1\}, \Sigma, \delta, q_0, \{q_1\}\rangle$ with $\delta(q_0, a) = q_1$ and $\delta(q_1, \mathsf{true}) = q_1$, cf. Figure 3.3.2. For $\bar{\varphi} = \mathsf{F}a$, the DBA $A_{\mathsf{F}a} = \langle\{q_0, q_1\}, \Sigma, \delta, q_0, \{q_1\}\rangle$, where $\delta(q_0, a) = q_1$, $\delta(q_0, \neg a) = q_0$ and $\delta(q_1, \mathsf{true}) = q_1$. cf. Fig. 3.3.3. This completes the base cases. For the three inductive cases, the DBA is constructed as follows.

   (a) Let $A_{\bar{\varphi}} = \langle Q, \Sigma, \delta, q_0, F\rangle$. $A_{\mathsf{F}(a\wedge\bar{\varphi})} = \langle Q \cup \{q_0'\}, \Sigma, \delta', q_0', F\rangle$ where q_0' is fresh, $\delta'(q, \cdot) = \delta(q, \cdot)$ if $q \in Q$, $\delta'(q_0', a) = \delta(q_0, a)$, and $\delta'(q_0', \neg a) = q_0'$. (Figure 3.3.4).

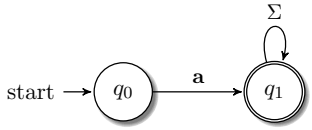


Figure 3.3.2: DBA for $\bar{\varphi} = \mathbf{a}$

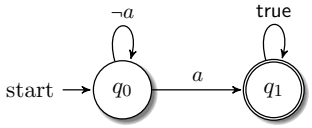


Figure 3.3.3: DBA for $\bar{\varphi} = \mathsf{F}\mathbf{a}$

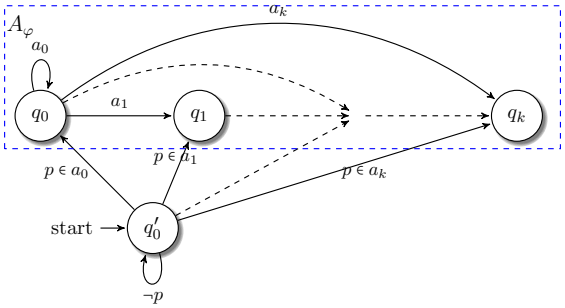


Figure 3.3.4: The DBA $A_{\mathsf{F}(\mathbf{a} \wedge \varphi)}$

(b) For $\bar{\varphi}_1 \wedge \bar{\varphi}_2$, the DBA is a standard synchronous product of the DBA for $\bar{\varphi}_1$ and $\bar{\varphi}_2$.

(c) Let $A_{\bar{\varphi}} = \langle Q, \Sigma, \delta, q_0, F \rangle$. $A_{\mathsf{X}\bar{\varphi}} = \langle Q \cup \{q_0'\}, \Sigma, \delta', q_0', F \rangle$ where q_0' is fresh, $\delta'(q_0', a) = q_0$ for all $a \in \Sigma$ and $\delta'(q, a) = \delta(q, a)$ for every $q \in Q$.

A few remarks are in order. The resulting DBA have a single final state. In addition, the DBA enjoy the property that the reflexive and transitive closure of the transition relation is a partial order [4]. Formally, $q \preceq q'$ iff $q' \in \delta^*(q, w)$ for some $w \in \Sigma^\omega$. The diameter of $A_{\bar{\varphi}}$ is the length of a longest simple path from the initial to the final state. This implies that the diameter of $A_{\mathsf{F}(a \wedge \bar{\varphi})}$ and $A_{\mathsf{X}\bar{\varphi}}$ is $n{+}1$ where n is this diameter of $A_{\bar{\varphi}}$, and the diameter of $A_{\bar{\varphi}_1 \wedge \bar{\varphi}_2}$ is $n_1 + n_2$ where n_i is the diameter of $A_{\bar{\varphi}_i}$, $i \in \{1, 2\}$.

4. Let $\varphi \equiv \varphi_1 \vee \ldots \vee \varphi_k$, where each φ_i is disjunction-free, with DBA $A_{\bar{\varphi}_i}$. Evidently, $V_{>0}(\varphi) \neq \varnothing$ iff $V_{>0}(\varphi_i) \neq \varnothing$ for some disjunct φ_i. Consider the product of MC M and DBA $A_{\bar{\varphi}_i}$, denoted $M \otimes A_{\bar{\varphi}_i}$; see, e.g., [5, Def. 10.50]. By construction, $M \otimes A_{\bar{\varphi}_i}$ is partially ordered and has diameter at most $m{\cdot}|\varphi_i|$. We have that $\Pr(M \vDash \bar{\varphi}_i) > 0$ iff an accepting state in $M \otimes A_{\bar{\varphi}_i}$ is reachable. Thus, there exists a finite path $\pi[0..m{\cdot}|\varphi_i|]$ in M with $\pi[0..m{\cdot}|\varphi_i] \vDash \bar{\varphi}$, or, $\pi[0..m{\cdot}|\varphi|] \vDash \bar{\boldsymbol{v}}(\varphi)$. This concludes the proof.

$M \otimes A_{\bar{\varphi}_i}$ can also be used to show that, if we have a valuation $\boldsymbol{v}$ such that

$\boldsymbol{v}(x) > m{\cdot}|\varphi|$ and for all other variables $y \neq x$, $\boldsymbol{v}(x) \leqslant m{\cdot}|\varphi|$ and $\boldsymbol{v} \in V_{>0}(\varphi)$ then $\boldsymbol{v}' \in V_{>0}(\varphi)$, where $\boldsymbol{v}'(x) = m{\cdot}|\varphi|$ and for $y \neq x$, $\boldsymbol{v}'(y) = \boldsymbol{v}(y)$. The argument proceed by induction on $\bar{\varphi}_i$. □

The above Theorem 3.3.4 leads to the following proposition.

Proposition 3.3.5. *For $\varphi \in$ pLTL(F,X), deciding if $V_{>0}(\varphi) = \varnothing$ is NP-complete.*

Proof. Similar to the NP-hardness proof of satisfiability of LTL(F,X) formulas [90, Th. 3.7], we give a polynomial reduction from the 3-SAT problem. For a 3-CNF formula ϕ with boolean variables $\{x_1, \cdots, x_n\}$, we define MC M and pLTL(F,X) formula φ such that ϕ is satisfiable iff $V_{>0}(\varphi)$ is not empty. Let 3-CNF formula $\phi = C_1 \wedge \cdots \wedge C_k$ with $C_i = d_{i1} \vee d_{i2} \vee d_{i3}$, where literal d_{il} is either x_j or $\sim x_j$ (for $1 \leq j \leq n$). Let MC $M = (S, \mathsf{P}, s_0, L)$ and

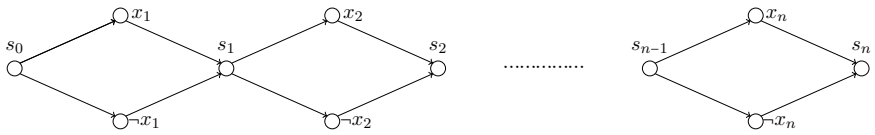


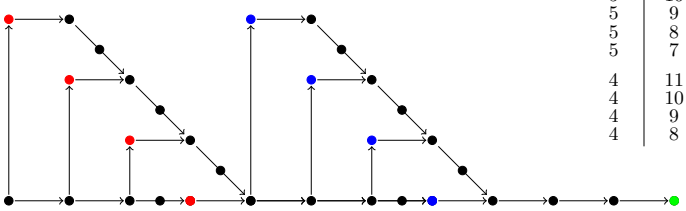
Figure 3.3.5: The Markov chain M used for reducing 3-CNF satisfiability problem to a pLTL(F,X) model checking problem. Observe that there are exponentially many orders in which C_is can be visited by a path in M satisfying, $\varphi = \mathsf{F}_{y_1} C_1 \wedge \ldots \wedge \mathsf{F}_{y_k} C_k$.

$\mathsf{AP} = \{C_i \ : \ 0 < i \leqslant k\}$ (Figure 3.3.5):

- $S = \{s_i \ : \ 0 \leqslant i \leqslant n\} \cup \{x_i \ : \ 0 < i \leqslant n\} \cup \{\sim x_i \ : \ 0 < i \leqslant n\}$. Thus for each variable x we have two states x and $\sim x$.
- The non-zero probabilities are given by: $\mathsf{P}(s_i, x_{i+1}) > 0$, $\mathsf{P}(s_i, \sim x_{i+1}) > 0$ for $0 \leqslant i < n$, $\mathsf{P}(x_i, s_i) > 0$ and $\mathsf{P}(\sim x_i, s_i) > 0$ for $0 < i \leqslant n$, and $\mathsf{P}(s_n, s_n) = 1$ (the actual probabilities are not relevant),
- $C_i \in L(x_j)$ iff $d_{il} = x_j$ for some $0 < l \leqslant 3$, and $C_i \in L(\sim x_j)$ iff $d_{il} = \sim x_j$ for some $0 < l \leqslant 3$, and $L(s_j) = \varnothing$ for all $0 \leqslant j \leqslant n$.

Let pLTL(F,X)-formula $\varphi = \mathsf{F}_{y_1} C_1 \wedge \ldots \wedge \mathsf{F}_{y_k} C_k$. Then ϕ is satisfiable iff $V_{>0}(\varphi)$ is not empty. Evidently, M and φ are obtained in polynomial time.

It remains to show membership in NP. By the proof of Theorem 3.3.4, $V_{>0}(\varphi) \neq \varnothing$ iff there is a finite path of M of length $m{\cdot}|\varphi|$ satisfying $\bar{\varphi}$. Thus, we non-deterministically select a path of M of length $m{\cdot}|\varphi|$ and check (using standard algorithms) in polynomial time whether it satisfies $\bar{\varphi}$. □



x_1	x_2	x_3
5	10	14
5	9	15
5	8	16
5	7	17
4	11	15
4	10	16
4	9	17
4	8	18

x_1	x_2	x_3
3	10	16
3	11	17
3	10	18
3	9	19
2	13	17
2	12	18
2	11	19
2	10	20

Figure 3.3.6: MC and $\min V_{>0}(\varphi)$ for $pLTL(\mathsf{F},\mathsf{X})$-formula $\varphi = \mathsf{F}_{x_1}\, r \wedge \mathsf{F}_{x_2}\, b \wedge \mathsf{F}_{x_3}\, g$

For almost sure properties, a similar approach as for $V_{>0}(\varphi)$ suffices.

Theorem 3.3.6. *For $\varphi \in pLTL(\mathsf{F},\mathsf{X})$ and MC M with m states, $V_{=1}(\varphi) \neq \varnothing$ iff $\bar{\boldsymbol{v}} \in V_{=1}(\bar{\varphi})$ with $\bar{\boldsymbol{v}}(x) = m{\cdot}|\varphi|$.*

Proof. The proof goes along similar lines as the proof of Theorem 3.3.4. □

Theorem 3.3.4 suggests that $\min V_{>0}(\varphi)$ lies in the hyper-cube $H = \{0,\cdots,N\}^d$, where $N = m{\cdot}|\varphi|$. A possible way to find $\min V_{>0}(\varphi)$ is to apply the bisection method in d-dimensions. We recursively choose a middle point of the cube, say $\boldsymbol{v} \in H$ —in the first iteration $\boldsymbol{v}(x) = N/2$— and divide H in 2^d equally sized hypercubes. If $\boldsymbol{v} \in V_{>0}(\varphi)$, then the hypercube whose points exceed $\boldsymbol{v}$ is discarded, else the cube whose points are below $\boldsymbol{v}$ is discarded. The asymptotic time-complexity of this procedure is given by the recurrence relation:

$$T(k) = (2^d - 1) \cdot T(k{\cdot}2^{-d}) + F \tag{3.3}$$

where k is the number of points in the hypercube and F is the complexity of checking $\boldsymbol{v} \in V_{>0}(\varphi)$ where $|\boldsymbol{v}| \leqslant N$. Section 3.3.4 presents an algorithm working in $\mathcal{O}(m{\cdot}N^d{\cdot}2^{|\varphi|})$ for a somewhat more expressive logic. From (3.3), this yields a complexity of $\mathcal{O}(m{\cdot}N^d{\cdot}2^{|\varphi|}{\cdot}\log N)$. The size of a set of minimal points can be exponential in the number of variables, as shown below.

Proposition 3.3.7. $|\min V_{>0}(\varphi)| \leqslant (N{\cdot}d)^{d-1}$.

Proof. Let $H = \{0,\ldots,N\}^d$. $(H,\leqslant)$ is a partially ordered set where $\leqslant$ is element-wise comparison. A subset $S^{(k)}$ of H has rank k if the summation of the coordinates of every element of S is k. By [61], the largest set of incomparable elements (anti-chain) is given by $Z^{(k)}$ where k is $N{\cdot}d/2$ if even, else k is $(N{\cdot}d{-}1)/2$. Then $|Z| = \binom{\lfloor N{\cdot}d/2\rfloor + d - 1}{d-1}$. □

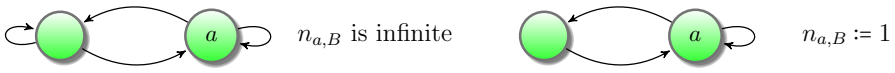


Figure 3.3.7: n_{aB} in the left BSCC B is infinity, while n_{aB} for the right one is 1

Example 3.3.2. There exist MCs for which $|\min V_{>0}(\varphi)|$ grows exponentially in d, the number of parameters in φ, whereas the number m of states in the MC grows linearly in d. For instance, consider the MC M in Fig. 3.3.6 and $\varphi = \mathsf{F}_{x_1}\, r \wedge \mathsf{F}_{x_2}\, b \wedge \mathsf{F}_{x_3}\, g$, i.e., d=3. We have $|\min V_{>0}(\varphi)| = 4^2$ as indicated in the table.

We conclude this section by briefly considering the membership query: does $\boldsymbol{v} \in V_{>0}(\varphi)$ for pLTL(F,X)-formula φ with d parameters? Checking membership of a valuation $\boldsymbol{v} \in V_{>0}(\varphi)$ boils down to deciding whether there exists a $\boldsymbol{v}' \in \min V_{>0}(\varphi)$ such that $\boldsymbol{v} \geqslant \boldsymbol{v}'$. A representation of $\min V_{>0}(\varphi)$ facilitating an efficient membership test can be obtained by putting all elements in this set in lexicographical order. This involves sorting over all d coordinates. A membership query then amounts to a recursive binary search over d dimensions. This yields:

Proposition 3.3.8. *For pLTL(F,X)-formula φ with d parameters, $\boldsymbol{v} \in V_{>0}(\varphi)$? takes $\mathcal{O}(d{\cdot}\log N{\cdot}d)$ time, provided a representation of $\min V_{>0}(\varphi)$ is given.*

3.3.3 Qualitative parametric Büchi

In this section, we consider pLTL-formulas of the form $\varphi = \mathsf{GF}_x\, a$, for proposition a. We are interested in $V_{>0}(\varphi)$, i.e., does the set of infinite paths visiting a-states that are maximally x apart infinitely often, have a positive measure? Let MC $M = (S, \mathsf{P}, s_0, L)$. A *bottom strongly-connected component* (BSCC) $B \subseteq S$ of M is a set of mutually reachable states with no edge leaving B. For BSCC B, let $n_{a,B} = \max\{|\pi| \,:\, \forall i \leqslant |\pi|, \pi[i] \in B \wedge a \notin L(\pi[i])\}$.

Proposition 3.3.9. *Let B be a BSCC and $s \in B$. Then, $\forall n \in \mathbb{N}, n > n_{a,B} \Leftrightarrow \Pr(s \vDash \mathsf{GF}_n\ a) = 1$ and $n \leqslant n_{a,B} \Leftrightarrow \Pr(s \vDash \mathsf{GF}_n\ a) = 0$.*

Proof. If $n > n_{a,B}$, then each path π from any state $s \in B$ will have at least one a-state in finite path fragment $\pi[i, \cdots, i{+}n]$ for all i. Hence, $\Pr(s \vDash \mathsf{GF}_n\ a) = 1$. If $n \leqslant n_{a,B}$, then there exists a finite path fragment ρ of B, such that, for all $i \leqslant n$, $a \notin L(\rho[i])$. Consider an infinite path π starting from any arbitrary $s \in B$. As $s \in B$, π will almost surely infinitely often visit the initial state of ρ. Therefore, by [5, Th.10.25], π will almost surely visit

every finite path fragment starting in that state, in particular ρ. Path π thus almost surely refutes $\mathsf{GF}_n\ a$, i.e. $\Pr(s \models \mathsf{GF}_n\ a) = 0$.(Figure 3.3.7) $\square$

For any BSCC B and $\mathsf{GF}_x\ a$, $n_{a,B} < \infty$ iff every cycle in B has at least one a-state. Hence, $n_{a,B}$ can be obtained by analysing the digraph of B (in $\mathcal{O}(m^2)$, the number of edges). BSCC B is called *accepting* for $\mathsf{GF}_x\, a$ if $n_{a,B} < \infty$ and B is reachable from the initial state s_0. Note that this may differ from being an accepting BSCC for $\mathsf{GF}a$. Evidently, $V_{>0}(\mathsf{GF}_x\, a) \neq \varnothing$ iff $n_{a,B} < \infty$. This result can be extended to *generalized Büchi* formula $\varphi = \mathsf{GF}_{x_1}\, a_1 \wedge \cdots \wedge \mathsf{GF}_{x_d}\, a_d$, by checking $n_{a_i,B} < \infty$ for each a_i.

As a next problem, we determine $\min V_{>0}(\mathsf{GF}_x\, a)$. For the sake of simplicity, let MC M have a single accepting BSCC B. For states s and t in MC M, let $d(s,t)$ be the distance from s to t in the graph of M. (Recall, the distance between state s and t is the length of the shortest path from s to t.) For BSCC B, let $d_{a,B}(s) = \min_{t \in B, a \in L(t)} d(s,t)$, i.e., the minimal distance from s to an a-state in B. Let the proposition a_B hold in state s iff $s \in B$ and $a \in L(s)$. Let $G_a = (V, E)$ be the digraph defined as follows: V contains all a-states of M and the initial state s_0 and $(s, s') \in E$ iff there is path from s to s' in M. Let c be a cost function defined on a finite path $s_0 \ldots s_n$ in graph G_a as: $c(s_0 \ldots s_n) = \max_i d(s_i, s_{i+1})$, ($d$ is defined on the graph of M). Using these auxiliary notions we obtain the following characterization for $\min V_{>0}(\mathsf{GF}_x\, a)$:

Theorem 3.3.10. $\min V_{>0}(\mathsf{GF}_x\, a) = n_0$ *where* $n_0 = \max\left(n_{a,B}, \min_{\pi = s_0 \ldots s_n, s_n \models a_B} c(\pi)\right)$ *if* $n_{a,B} < d_{a,B}(s_0)$ *and* $n_0 = n_{a,B}$ *otherwise.*

Proof. We show for $n \geqslant n_0$, $\Pr(\mathsf{GF}_n\, a) > 0$, and for $n < n_0$, $\Pr(\mathsf{GF}_n\, a) = 0$. Distinguish:

1. $n_{a,B} \geqslant d_{a,B}(s_0)$. Then, from s_0 an a-state in B can be reached within $n_{a,B}$ steps, i.e., $\Pr(s_0 \models \mathsf{F}_{n_{a,B}}\, a_B) > 0$. For this a_B-state, s, say, by Proposition 3.3.9 it follows $\Pr(s \models \mathsf{GF}_{n_{a,B}}\, a) = 1$. Together this yields $\Pr(s_0 \models \mathsf{GF}_n\, a) > 0$ for each $n \geqslant n_{a,B} = n_0$. For $n < n_0 = n_{a,B}$, it follows by Proposition 3.3.9 that $\Pr(s \models \mathsf{GF}_n\, a) = 0$ for every a_B-state s. Thus, $\Pr(s_0 \models \mathsf{GF}_n\, a) = 0$.

2. $n_{a,B} < d_{a,B}(s_0)$. As B is accepting, $d_{a,B}(s_0) \neq \infty$. Consider a simple path π from s_0 to an a-state in B. Let $c(\pi)$ be the maximal distance between two consecutive a-states along this path. Then it follows $\Pr(s_0 \models \mathsf{GF}_k\, a) > 0$ where $k = \max(c(\pi), n_{a,B})$. By taking the minimum c_{min} over all simple paths between s_0 and B, it follows $\Pr(s_0 \models \mathsf{GF}_n\, a) > 0$ for each $n \geqslant n_0 = \max(n_{a,B}, c_{min})$ with $c_{min} = \min_{\pi \in \mathsf{Paths}(s_0 \models \mathsf{F} a_B)} c(\pi)$,

where $\mathsf{Paths}(s \models \varphi) = \{w \ : \ w \in \mathsf{Paths}(s), w \models \varphi\}$. For $n < n_0$, distinguish between $n_0 = n_{a,B}$ and $n_0 = c_{min}$. In the former case, it follows (as in the first case) by Proposition 3.3.9 that $\Pr(s_0 \models \mathsf{GF}_n\, a) = 0$ for all $n \geqslant n_0$. Consider now $n_0 = c_{min} \geqslant n_{a,B}$. Let $n < n_0$. By contraposition. Assume $\Pr(s_0 \models \mathsf{GF}_n\, a) > 0$. Let $\pi = s_0 \dots s_{1,a} \dots s_{2,a} \dots \dots s_{k,a}$ be a finite path fragment in M where $s_{i,a} \models a$ and $s_{k,a}$ is the first a-state along π which belongs to B. Then, by definition of the digraph G_a, the sequence $\pi = s_0 s_{1,a} s_{2,a} \dots s_{k,a}$ is a path in G_a satisfying $c(s_{i,a}, s_{i+1,a}) \leqslant n$ for all $0 \leqslant k < n$. But then $c_{min} \leqslant n$. Contradiction.

This concludes the proof. □

If MC M has more than one accepting BSCC, say $\{B_1, \dots, B_k\}$ with $k > 1$, then $n_0 = \min_i n_{0,B_i}$, where n_{0,B_i} for $0 < i \leqslant k$ is obtained as in Theorem 3.3.10.

Proposition 3.3.11. *The sets $V_{>0}(\mathsf{GF}_x\, a)$ and $V_{=1}(\mathsf{GF}_x\, a)$ can be determined in polynomial time by a graph analysis of MC M.*

Proof. We argue that $\min V_{>0}(\mathsf{GF}_x\, a)$ can be determined in polynomial time. The proof for $V_{=1}(\mathsf{GF}_x\, a)$ goes along similar lines and is omitted here. We can determine both $n_{a,B}$ and $d_{a,B}(s_0)$ in linear time. It remains to obtain $c_{min} = \min_{\pi = s_0 \dots s_n, s_n \models a_B} c(\pi)$ in case $n_{a,B} < d_{a,B}(s_0)$. This can be done as follows. The distances $d(s, s')$, required for the function c in the digraph $G_a = (V, E)$, can be obtained by applying Floyd-Warshall's all-pairs shortest path algorithm on the graph of M. This takes $\mathcal{O}(m^3)$. To obtain c_{min}, we use a cost function $F : V \to \mathbb{N}$ which is initially set to 0 for initial state s_0 and ∞ otherwise. Let pQ be a min priority queue, initially containing all vertices of G_a, prioritized by the value of F. Algorithm 1 finds c_{min} in $\mathcal{O}(m^2 \cdot \log m)$. Its correctness follows from the invariant $F(v) \leqslant \max(F(u), c(u, v))$. Using

Algorithm 1 Input: MC M Output: c_{min}

```
1: Initialize F, found := false and pQ.
2: while (¬found and pQ ≠ ∅) do
3:     u := pop(pQ); found := (a_B ∈ L(u));
4:     for v ∈ pQ do F(v) := min(F(v), max(F(u), c(u, v)))
5:     end for
6: end while
```

this we can find the minimum n for which we can reach an accepting BSCC via a finite path satisfying $\mathsf{GF}_n\, a$. □

Determining $\min V_{\geqslant p}(\mathsf{GF}_x\, a)$ for arbitrary p reduces to reachability of *accepting* BSCCs. In a similar way as for parametric reachability (cf. Sec-

tion 3.3.1), this can be done by searching. For generalized Büchi formula $\varphi = \mathsf{GF}_{x_i} a_i \wedge \cdots \wedge \mathsf{GF}_{x_d} a_d$ and BSCC B, $n_{a_i B}$ is at most m. Thus, $\min V_{>0}(\varphi) \in \{0, \cdots, m{\cdot}d\}^d$ and can be found by the bisection method, similar to the procedure described in Section 3.3.2.

Example 3.3.3. Rusty decides to represents the frequency of repeated reachability of guards as parametric Büchi properties. Danny then uses the algorithm described above to elicit the parameter values for which there is non-zero probability of failure. He then identifies possible targets based on this information.

3.3.4 The fragment pLTL$_{\mathsf{F}}$

This section is concerned with the logical fragment pLTL$_{\mathsf{F}}$, as defined in (Eq. 3.2 pg. 44):

$$\varphi \ ::= \ a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \varphi \mathsf{R} \varphi \mid \mathsf{G}\varphi \mid \mathsf{F}_{\leqslant x}\, \varphi.^2$$

We will focus on the emptiness problem: is $V_{>0}(\varphi) = \varnothing$. The decision problem whether $V_{=1}(\varphi)$ is very similar. Similar as for $pLTL(\mathsf{F}, \mathsf{X})$, we obtain necessary and sufficient criteria for both cases. The proofs for these criteria depend on an algorithm that checks whether $\boldsymbol{v} \in V_{>0}(\varphi)$. This algorithm is presented first.

Automata constructions. Let φ be a pLTL$_{\mathsf{F}}$-formula, and $\boldsymbol{v}$ a variable valuation. W.l.o.g. we assume that each variable occurs once in φ. We will extend the classical automaton-based approach for LTL by constructing a nondeterministic Büchi automaton for φ that is amenable to treat the variables occurring in φ. To that end, inspired by [99], we proceed in a number of steps:

1. Construct an automaton G_φ for φ, independent from the valuation $\boldsymbol{v}$, with two types of acceptance sets, one for treating until and release-modalities (as standard for LTL [94]), and one for treating the parameter constraints.

2. Establish how for a given valuation $\boldsymbol{v}$, a Büchi automaton $B_\varphi(\boldsymbol{v})$ can be obtained from G_φ such that for infinite word w, $(w, \boldsymbol{v}) \in \mathcal{L}(\varphi)$ iff w is an accepting run of $B_\varphi(\boldsymbol{v})$.

3. Exploit the technique advocated by Couvreur *et al.* [32] to verify MC M versus $B_\varphi(\boldsymbol{v})$.

[2] The modalities $\mathsf{F}_{\leqslant c}$ and $\mathsf{G}_{\leqslant c}$ can be removed with only quadratic blow up.

We start with constructing G_φ. Like for the LTL-approach, the first step is to consider consistent sets of sub-formulas of φ. Let $\mathsf{cl}(\varphi)$ be the set of all sub-formulas of φ. Set $H \subseteq \mathsf{cl}(\varphi)$ is *consistent*, when:

- $a \in H$ iff $\neg a \notin H$,
- $\varphi_1 \wedge \varphi_2 \in H$ iff $\varphi_1 \in H$ and $\varphi_2 \in H$,
- $\varphi_1 \vee \varphi_2 \in H$ iff $\varphi_1 \in H$ or $\varphi_2 \in H$,
- $\varphi_2 \in H$ implies $\varphi_1 \,\mathsf{U}\, \varphi_2 \in H$,
- $\varphi_1, \varphi_2 \in H$ implies $\varphi_1 \mathsf{R} \varphi_2 \in H$,
- $\varphi_1 \in H$ implies $\mathsf{F}_x\, \varphi_1 \in H$.

We are now in a position to define G_φ, an automaton with two acceptance sets. For $\varphi \in \mathrm{pLTL}_\mathsf{F}$, let $G_\varphi = (Q, 2^{\mathsf{AP}}, Q_0, \delta, \mathsf{Acc}_B, \mathsf{Acc}_P)$ where

- Q is the set of all consistent sub-sets of $\mathsf{cl}(\varphi)$ and $Q_0 = \{H \in Q \,:\, \varphi \in H\}$.
- $(H, a, H') \in \delta$, where $a \in 2^{\mathsf{AP}}$ whenever:
  - $H \cap \mathsf{AP} = \{a\}$,
  - $\mathsf{X}\varphi_1 \in H \iff \varphi_1 \in H'$,
  - $\varphi_1 \,\mathsf{U}\, \varphi_2 \in H \iff \varphi_2 \in H$ or $(\varphi_1 \in H$ and $\varphi_1 \,\mathsf{U}\, \varphi_2 \in H')$,
  - $\varphi_1 \mathsf{R} \varphi_2 \in H \iff \varphi_2 \in H$ and $(\varphi_1 \in H$ or $\varphi_1 \mathsf{R} \varphi_2 \in H')$,
  - $\mathsf{F}_x\, \varphi_1 \in H \iff \varphi_1 \in H$ or $\mathsf{F}_x\, \varphi_1 \in H'$,
- (generalized) Büchi acceptance Acc_B and parametric acceptance Acc_P:
  - $\mathsf{Acc}_B = \{F_{\varphi'} \,:\, \varphi' \in \mathsf{cl}(\varphi) \wedge (\varphi' = \varphi_1 \,\mathsf{U}\, \varphi_2 \vee \varphi' = \varphi_1 \mathsf{R} \varphi_2)\}$ where
    - $F_{\varphi'} = \{H \,:\, \varphi' \in H \Rightarrow \varphi_2 \in H\}$ if $\varphi' = \varphi_1 \,\mathsf{U}\, \varphi_2$, and
    - $F_{\varphi'} = \{H \,:\, \varphi_2 \in H \Rightarrow \varphi' \in H\}$ if $\varphi' = \varphi_1 \mathsf{R} \varphi_2$,
  - $\mathsf{Acc}_P = \{F_{x_i} \,:\, \mathsf{F}_{x_i}\, \varphi_i \in \mathsf{cl}(\varphi)\}$ with $F_{x_i} = \{H \,|\, \mathsf{F}_{x_i}\, \varphi_i \in H \Rightarrow \varphi_i \in H\}$.

A run $\rho \in Q^\omega$ of G_φ is *accepting* under valuation $\boldsymbol{v}$ if it visits each set in Acc_B infinitely often and each $F_{x_i} \in \mathsf{Acc}_P$ in every infix of length $\boldsymbol{v}(x_i)$. $\mathcal{L}(G_\varphi)$ contains all pairs $(w, \boldsymbol{v})$ such that there is an accepting run of w under the valuation $\boldsymbol{v}$. G_φ is *unambiguous* if $q \xrightarrow{a} q'$ and $q \xrightarrow{a} q''$ implies $\mathcal{L}(q') \cap \mathcal{L}(q'') = \varnothing$, where $\mathcal{L}(q)$ is the language starting from the state q.

Proposition 3.3.12 ([99]). *For $\varphi \in pLTL_\mathsf{F}$, the automaton G_φ is unambiguous and $\mathcal{L}(G_\varphi) = \mathcal{L}(\varphi)$.*

The automaton G_φ can be constructed in $\mathcal{O}(2^{|\varphi|})$. Apart from the parametric acceptance condition, G_φ behaves as a generalized Büchi automaton (GNBA) if we consider only the accepting set Acc_B. Let Acc_B be

$\{F_1, \ldots, F_k\}$. In order to obtain a non-deterministic automaton, we first apply a similar transformation as for GNBA to NBA [5]. We convert G_φ to $U_\varphi = (Q', 2^{\mathsf{AP}}, Q'_0, \delta', \mathsf{Acc}'_B, \mathsf{Acc}'_P)$ where $Q' = Q \times \{1, \ldots, k\}$, $Q'_0 = Q_0 \times \{1\}$. If $(q, a, q') \in \delta$, then $((q, i), a, (q', i')) \in \delta'$ with $i{=}i'$ if $q \notin F_i$ else $i' = (i \bmod k){+}1$. $\mathsf{Acc}_B = F_1 \times \{1\}$ and $\mathsf{Acc}'_P = \{F'_{x_i} : F_{x_i} \in \mathsf{Acc}_P\}$, where $F'_{x_i} = F_{x_i} \times \{1, \ldots, k\}$. Note that the construction preserves unambiguity and the size of U_φ is in $\mathcal{O}(|\varphi| \cdot 2^{|\varphi|})$.

For a given valuation $\boldsymbol{v}$, U_φ can be converted into an NBA $B_\varphi(\boldsymbol{v})$. This is done as follows. Let $U_\varphi = (Q', 2^{AP}, Q'_0, \delta', \mathsf{Acc}'_B, \mathsf{Acc}'_P)$ and $\boldsymbol{v}$ a valuation of φ with d parameters. Then $B_\varphi(\boldsymbol{v}) = (Q'', 2^{AP}, Q''_0, \delta'', \mathsf{Acc})$ with:

- $Q'' \subseteq Q' \times \{0, \cdots, \boldsymbol{v}(x_1)\} \times \cdots \times \{0, \cdots, \boldsymbol{v}(x_d)\}$,
- $((q, \mathbf{n}), a, (q', \mathbf{n}')) \in \delta''$ if $(q, a, q') \in \delta'$ and for all x_i:
  - if $q' \in F'_{x_i}$ and $\mathbf{n}(x_i) < \boldsymbol{v}(x_i)$ then $\mathbf{n}'(x_i) = 0$,
  - if $q' \notin F'_{x_i}$ and $\mathbf{n}(x_i) < \boldsymbol{v}(x_i)$ then $\mathbf{n}'(x_i) = \mathbf{n}(x_i) + 1$.
- $Q''_0 = Q'_0 \times 0^d$ and $\mathsf{Acc} = \mathsf{Acc}'_B \times \{0, \cdots, \boldsymbol{v}(x_1)\} \times \cdots \times \{0, \cdots, \boldsymbol{v}(x_d)\}$.

It follows that $B_\varphi(\boldsymbol{v})$ is unambiguous for any valuation $\boldsymbol{v}$. Furthermore, every run of $B_\varphi(\boldsymbol{v})$ is either finite or satisfies the parametric acceptance condition for valuation $\boldsymbol{v}$. Thus we have:

Proposition 3.3.13. *An infinite word $w \in \mathcal{L}(B_\varphi(\boldsymbol{v}))$ if and only if $(w, \boldsymbol{v}) \in \mathcal{L}(\varphi)$.*

The size of $B_\varphi(\boldsymbol{v})$ is in $\mathcal{O}(c_{\boldsymbol{v}} \cdot |\varphi| \cdot 2^{|\varphi|})$ where $c_{\boldsymbol{v}} = \prod_{x_i} (\boldsymbol{v}(x_i){+}1)$. As a next step, we exploit the fact that $B_\varphi(\boldsymbol{v})$ is unambiguous, and apply the technique by Couvreur *et al.* [32] for verifying MC M against $B_\varphi(\boldsymbol{v})$. Let $M \otimes B_\varphi(\boldsymbol{v})$ be the synchronous product of M and $B_\varphi(\boldsymbol{v})$ [5], Π_1 the projection to M and Π_2 the projection to $B_\varphi(\boldsymbol{v})$. Let $\mathcal{L}(s, q) = \{\pi \in \mathsf{Paths}(s) : \mathsf{trace}(\pi) \in \mathcal{L}(q)\}$ and $\Pr(s, q) = \Pr(\mathcal{L}(s, q))$. Let $\Pr(M \otimes B_\varphi(\boldsymbol{v})) = \sum_{q_0 \in Q_0} \Pr(s_0, q_0)$. As $B_\varphi(\boldsymbol{v})$ is unambiguous, we have for any (s, q):

$$\Pr(s, q) \;=\; \sum_{(t, q') \in \delta(s, q)} \mathsf{P}(s, t) \cdot \Pr(t, q'),$$

where δ is the transition relation of $M \otimes B_\varphi(\boldsymbol{v})$ and $\mathsf{P}(s, t)$ is the one-step transition probability from s to t in MC M. A (maximal) strongly connected component (SCC, for short) $C \subseteq S$ is *complete* if for any $s \in \Pi_1(C)$:

$$\mathsf{Paths}(s) = \bigcup_{(s, q) \in C} \mathcal{L}_C(s, q)$$

where $\mathcal{L}_C(s,q)$ *restricts* runs to C (runs only visits states from C). The SCC C is *accepting* if $\mathsf{Acc} \cap \Pi_2(C) \neq \varnothing$ (where Acc is the set of accepting states in $B_\varphi(\boldsymbol{v})$).

Proposition 3.3.14 ([32])**.** *Let C be a complete and accepting SCC in $M \otimes B_\varphi(\boldsymbol{v})$. Then for all $s \in \Pi_1(C)$:*

$$\Pr\left(\bigcup_{(s,q)\in C} \mathcal{L}_C(s,q)\right) = 1.$$

Moreover, since $B_\varphi(\boldsymbol{v})$ is unambiguous, $\Pr(M \otimes B_\varphi(\boldsymbol{v})) > 0$ implies there exists a reachable, complete and accepting SCC.

Finding complete and accepting SCC in $M \otimes B_\varphi(\boldsymbol{v})$ is done by standard graph analysis. Altogether, $\boldsymbol{v} \in V_{>0}(\varphi)$ is decided in $\mathcal{O}(m{\cdot}c_{\boldsymbol{v}}{\cdot}|\varphi|{\cdot}2^{|\varphi|})$. The space complexity is polynomial in the size of the input (including the valuation), as $M \otimes B_\varphi(\boldsymbol{v})$ can be stored in $\mathcal{O}(\log m + |\varphi| + \log c_{\boldsymbol{v}})$ bits. In the sequel, we exploit these results to obtain a necessary and sufficient criterion for the emptiness of $V_{>0}(\varphi)$ for φ in $\mathrm{pLTL}_{\mathsf{F}}$.

Theorem 3.3.15. *For $\varphi \in pLTL_{\mathsf{F}}$, $V_{>0}(\varphi) \neq \varnothing$ iff $\bar{\boldsymbol{v}} \in V_{>0}(\varphi)$ s.t. $\bar{\boldsymbol{v}}(x) = m{\cdot}|\varphi|{\cdot}2^{|\varphi|}$.*

Proof. Consider the direction from left to right. The only non-trivial case is when there exists a valuation $\boldsymbol{v} \not\leqslant \bar{\boldsymbol{v}}$ such that $\boldsymbol{v} \in V_{>0}(\varphi)$ implies $\bar{\boldsymbol{v}} \in V_{>0}(\varphi)$. In the model checking algorithm described above, we first construct G_φ, and then U_φ with a single Büchi accepting set Acc'_B and d parametric accepting sets F'_{x_i}, one for each variable x_i in φ. For the sake of clarity, assume $d = 1$, i.e., we consider valuation v. The explanation extends to the general case in a straightforward manner. For valuation v, consider $M \otimes B_\varphi(v)$. We show that, for $r < v$, $\Pr(M \otimes B_\varphi(v)) > 0$ implies $\Pr(M \otimes B_\varphi(r)) > 0$, where $r = m{\cdot}|U_\varphi|$, which is in $\mathcal{O}(m{\cdot}|\varphi|{\cdot}2^{|\varphi|})$.

Note that every cycle in $M \otimes B_\varphi(r)$ contains a state (s,q,i) with $i = 0$. Moreover, the graph of $M \otimes B_\varphi(r)$ is a sub-graph of $M \otimes B_\varphi(v)$. We now prove that, if a (maximal) SCC C of $M \otimes B_\varphi(r)$ is not complete (or accepting) then any SCC C' of $M \otimes B_\varphi(v)$ containing C is also not complete (or accepting, respectively).

(a) Suppose C is not complete. Then there exists a finite path $\sigma = s\,s_1 \ldots s_k$ of M, such that from any q, with $(s,q,0) \in C$, the run $\rho = (s,q,0)(s_1,q_1,1)\ldots(s_j,\, q_j, j)$ leads to a deadlock state. This can have two causes: either (s_j,q_j,j) has no successor for any j. Then, C' is not complete. Or, the path ρ terminates in (s_j,q_j,j) where $j = r$. This means, for all $(s',q',j{+}1) \in \delta(s_j,q_j,j)$ in C', $q' \notin F_x$. As the length of ρ exceeds

r, there are states in the run whose first and second component appear multiple times. Thus, we can find another path σ' (possibly longer than σ) for C' which goes through states where the first and the second component of some of its states are repeated sufficiently many times to have a run $(s,q,0)(s_1,q_1,1)\ldots(s_j,q_j,v)$ which is a deadlock state. Thus, C' is not complete.

(b) Suppose C' is accepting. Then there exists (s',q',i') with $q' \in \mathsf{Acc}$. Since C' is an SCC and $C \subseteq C'$, there is a path from $(s,q,0) \in C$ to (s',q',i'). If the length of the path is less than r, then we are done. If $i' > r$, then some (s'',q'') pair in the path must be repeated. Thus, we can find another path of length less than r to a state (s',q',i), where $i \leqslant r$. Therefore, C is accepting. The rest of the proof follows from Proposition 3.3.14. □

For almost sure properties, a similar approach as for $V_{>0}(\varphi)$ suffices.

Theorem 3.3.16. *For $\varphi \in pLTL_{\mathsf{F}}$, $V_{=1}(\varphi) \neq \varnothing$ iff $\bar{\boldsymbol{v}} \in V_{=1}(\bar{\varphi})$ with $\bar{\boldsymbol{v}}(x) = m{\cdot}|\varphi|{\cdot}2^{|\varphi|}$.*

Let $N_{\varphi M} = m{\cdot}|\varphi|{\cdot}2^{|\varphi|}$. Note that $c_{\bar{\boldsymbol{v}}}$ equals $(N_{\varphi M})^d$. Thus, we have:

Proposition 3.3.17. *For $\varphi \in pLTL_{\mathsf{F}}$, deciding if $V_{>0}(\varphi) = \varnothing$ is PSPACE-complete.*

Proof. Theorem 3.3.15 gives an algorithm in PSPACE, as $M \otimes B_{\varphi}(\bar{\boldsymbol{v}})$ can be stored in $O(\log m + |\varphi| + d \log N_{\varphi M})$ bits. PSPACE hardness follows trivially, as for LTL formula φ and MC M, deciding $\Pr(M \vDash \varphi) > 0$ (which is known to be a PSPACE complete problem) is the same as checking the emptiness of $V_{>0}(\varphi)$. □

Just as for pLTL(F,X), we can use the bisection method to find $\min V_{>0}(\varphi)$. The search procedure invokes the model checking algorithm multiple times. We can reuse the space each time we check $\Pr(M \vDash \boldsymbol{v}(\varphi)) > 0$. Hence, $\min V_{>0}(\varphi)$ can be found in polynomial space. The time complexity of finding $\min V_{>0}(\varphi)$ is $\mathcal{O}(m{\cdot}(N_{\varphi M})^d{\cdot}2^{|\varphi|}{\cdot}\log N_{\varphi M})$. Membership can also be similarly solved.

Proposition 3.3.18. *For $pLTL_{\mathsf{F}}$-formula φ, $\boldsymbol{v} \in V_{>0}(\varphi)$? takes $\mathcal{O}(d{\cdot}\log \frac{N_{\varphi M}}{d})$ time, provided a representation of $V_{>0}(\varphi)$ is given.*

3.3.5 Parametric (0/1)-counter automata

We have seen that $\mathrm{pLTL}_{\mathsf{F}}$ formula can be converted to an equivalent Büchi automata with parametrized acceptance condition. We can implement the parametrized repeated reachability by parametric 0/1-counter automata with

Büchi acceptance condition. 0/1-counters automata are counter automata were the counter can either increase or reset to zero (similar to clocks of timed automata). A counter is said to be *parametric* if it is compared with a parameter. Parametric 0/1-counter automata has one or more parametric counters. The principal problem for parametric 0/1-counter is the language emptiness. They were first studied by Alur *et al.* [3], where the class of 0/1 counter automata with one parametric counter was shown to be decidable. It was also shown that large classes of parametric automata have undecidable emptiness problem. Recently, in [14] the class of 0/1-counter automata with two parametric counter was shown to be decidable as well.

The language emptiness of parametric (0/1) counter automata is analogous to the language emptiness of parametric LTL. We can improve the undecidability result of [3], where the reachability problem for automata with three (0/1) parametric counters was shown to be undecidable, using the proof technique used in showing undecidability of language emptiness of pLTL. Closer scrutiny of the demonstration given by [3] reveals that the number of parameters (six in total) plays a crucial part in the encoding to the halting problem of a two counter machine. We show that the reachability problem is still undecidable with three (0/1) counters and *one* parameter. We also show undecidability for two (0/1) parametric counters which can only be compared with a non-parametric (0/1) counter using a single parameter. The details can be found in the appendix A.

3.4 Related work and conclusion

The verification of parametric probabilistic models in which certain transition probabilities are given as parameters (or functions thereof) has recently received considerable attention. Most of these works are focused on parameter synthesis: for which parameter instances does a given (LTL or PCTL) formula hold? To mention a few, Han *et al.* [49] considered this problem for timed reachability in continuous-time MCs, Hahn *et al.* [47] and Pugelli *et al.* [86] for Markov decision processes (MDPs), and Benedikt *et al.* [7] for ω-regular properties of interval MCs. Hahn *et al.* [48] and Dehnert *et al.* [35] provide an algorithm for computing the rational function expressing the probability of reaching a given set of states in a parametric (reward) MDPs and Markov chains based on exploiting regular expressions as initially proposed by Daws [33]. Other related work includes the synthesis of loop invariants for parametric probabilistic programs [62]. To the best of our knowledge, verifying parametric properties on MCs has not been considered so far. The closest related works are on parametrized computation tree

logic by Emerson *et al.* [40] and the computation of quantiles by Ummels and Baier [93].

3.4.1 Summary

In this chapter we considered the verification of finite MCs against parametric LTL. The main problem of this chapter was, given a pLTL formula f and a finite Markov chain M, how to synthesize parameter values $V_{>0}(f)$ for which the resultant LTL formula has non-zero probabilities. We saw that deciding the emptiness of the set of parameter values $V_{>0}(f)$ for any arbitrary pLTL formula is undecidable. Thus, we looked at various fragments of pLTL for which the emptiness problem is decidable. We obtained several results on the necessary and sufficient conditions for emptiness of $V_{>0}(f)$, as well as studied the complexity of the emptiness problem for various fragments. The necessary and sufficient conditions are in the form of specific parameter values that $V_{>0}(f)$ contains if and only if $V_{>0}(f)$ is non-empty. We used these results to define algorithms for synthesizing the set $V_{>0}(f)$, which basically searches the maximal set of *non-dominating* parameter values. Future work consists of devising more efficient algorithms for the quantitative verification problems, and lifting the results to extended temporal logics [96] and stochastic games, possibly exploiting the results of optimal bounds on parametric LTL games [99].

Chapter 4

PCTL and Interval Markov Chains

In this chapter we recall model checking of PCTL formulas on uncertain stochastic models whose probability distributions are not exactly know. These uncertain models are commonly referred as *convex Markov chains*, where the transition probabilities are known to lie within a convex set. Most often these sets are represented by intervals. We will study the technical problem that arises with open intervals. The main focus of this chapter is on model checking of uncertain models, some of whose transition probability intervals are not closed intervals, against PCTL formulas.

4.1 Introduction

Formal verification is the exhaustive validation of the functionalities of a system. Once a model (a representation of the system) has been fixed, the next step is to check whether a property is satisfied by every possible behaviour of the model. Hence, *model-checking* is the heart of formal verification. The behaviour of many (physical) systems are not completely deterministic rather they exhibit uncertainties, i.e. they have stochastic characteristics. There are many well studied mathematical models that can be used to model such systems. For example, discrete time Markov chain (MCs), they has emerged as an useful stochastic model for analyzing the reliability and performance for system with stochastic behaviour.

But the next immediate question is, what are the values of the probability distributions? Most often than not, the precise definition of these values may not be always available [60, 89, 97]. This is precisely the case when transition probabilities are obtained by statistical methods. It is ironic since we not only know that the system behaves uncertainly, but also that we are

uncertain even about the degree of uncertainty in its behaviour.

This leads us to *Interval Markov chains* (IMCs) [60, 100] where the value of the transition probabilities are not stated precisely. IMCs generalize discrete time Markov chains by allowing intervals of possible probabilities on the state transitions in order to capture the system uncertainty more faithfully. For example, instead of specifying that the probability of moving from state s to t is 0.5, one can specify an interval $[0.3, 0.7]$ which captures the uncertainty in the probability of moving from state s to t. There are many cases where it is beneficial to be able to specify an interval [97]. In some cases, the transition probabilities may depend on an unknown environment, and are approximately known, in other cases the interval may be introduced to make the model more robust.

There are two prevalent semantics of interval Markov chains. *Uncertain Markov Chains* (UMC) [60, 89] semantics interprets an interval Markov chain as a set of (possibly uncountably many) discrete time Markov chains where each element of the set is a DTMC whose transition probabilities lie within the interval range defined by the IMC. In *Interval Markov Decision Processes* semantics (IMDP) [89], the uncertainty of the transition probabilities are resolved non-deterministically. It requires the notion of *scheduler*, which chooses a distribution from a (possibly uncountable) set of distributions defined by the intervals on the transitions, each time a state is visited in an execution.

The problem of model-checking PCTL properties for IMCs was studied in [89]; it provides PSPACE algorithms for both UMC and IMDP semantics for interval Markov chains. Furthermore, NP and co-NP hardness was shown for model-checking in UMC semantics and PTIME hardness for IMDP semantics which follows from PTIME hardness of model-checking PCTL formulas on DTMCs. [26] improved the upper bound and showed that model-checking problem for IMDP semantics is in co-NP. This result is shown for a richer class of logic, called ω-*PCTL*, which allow Büchi and co-Büchi properties in the formula. These results rely on the construction of an MDP which encodes all the behaviours of the IMDP under analysis. For each state in the new MDP, the set of possible distribution is mapped to the Basic Feasible Solutions (BFS) of the set of inequalities specifying the transition probabilities of the IMDP. Since, in the worst case, the number of BFS is exponential in the number of states in the IMDP, the equivalent MDP can have size exponential in the size of the IMDP. Recently, in [86] a polynomial time algorithm for PCTL verification was presented. The central idea was to represent the reachability problem in IMDP as a linear programming problem.

In the literature, the intervals of IMCs are always assumed to be closed. This assumption is sensible from the model-checking perspective, since a model with open interval may not have an optimal value of satisfying a temporal property (in IMDP semantics). The focus of this chapter is to study IMDP semantics of IMCs with open intervals. We will later contrast the result with the UMC semantics, and see that the outcome may differ from IMDP semantics. The main intuition is that the (optimal) value of the reachability probability in a IMC with open intervals can be made arbitrarily close to the (optimal) value of the property obtained by *closing* the intervals.

4.2 Interval Markov chains

Definition 4.2.1. Let $\mathcal{I}$ be the set of intervals (open or closed) in the range $[0,1]$. The subsets $\mathcal{I}_0 \triangleq \{(a,b] \mid 0 \leq a < b \leq 1\}$, $\mathcal{I}_1 \triangleq \{(a,b) \mid 0 \leq a < b \leq 1\}$, $\mathcal{I}_2 \triangleq \{[a,b) \mid 0 \leq a < b \leq 1\}$ and $\mathcal{I}_3 \triangleq \{[a,b] \mid 0 \leq a \leq b \leq 1\}$. $\mathcal{I} = \bigcup_{i \in \{0,1,2,3\}} \mathcal{I}_i$.

Let $I \triangleq \langle a,b \rangle$ be an interval in $\mathcal{I}$, where $\langle \in \{(,[\}$ and $\rangle \in \{),]\}$. The lower bound $I{\downarrow} = a$ [1] and upper bound is $I{\uparrow} = b$. Point intervals ($[a,a]$) are closed intervals where the upper and lower bounds are equal. The closure of an interval I, denoted by $\bar{I}$, is the smallest closed interval that includes I.

Definition 4.2.2. An Interval Markov chain (IMC) is a tuple $\mathcal{M} \triangleq (S, L, \delta)$, where S is a (finite) set of states and L is a labeling function $L : S \to 2^{AP}$, where AP is the set of atomic propositions. δ is a function $\delta : S \to \mathcal{D}$, where $\mathcal{D}$ is the set of functions from the set of states to the set of intervals $\mathcal{I}$, i.e., $\mathcal{D} = S \to \mathcal{I}$.

We will use the un-Curry notation $\delta(s,t)$ for $\delta(s)(t)$. For a state s, the probability of a single step from s to t lies in the interval $\delta(s,t)$. Thus an IMC defines a collection of Markov chains, where the single step transition probability of moving from state s to t lies in the interval $\delta(s,t)$. Not every IMC defines a collection of Markov chains. Thus, we have the notion of realizability.

Definition 4.2.3. Let $\mathcal{M} = (S, L, \delta)$ be an IMC with states $S = \{s_1, \cdots, s_m\}$. Let $D^{\mathcal{M}}$ be the set of $m \times 1$ vectors $\vec{d}$, such that $\vec{d}^T \cdot \vec{1} = 1$, which represents the set of distributions on states of $\mathcal{M}$. Where $\mathcal{M}$ is fixed we denote the set as D.

$\mathcal{M}$ is said to be *realizable* if for each set of intervals defined by $\delta(s)$, there exists a distribution $\vec{d}$ such that for all $i \in [1,m]$ $\vec{d}_i$ (the i^{th} component

[1] Recall, for a sequence σ, $\sigma{\downarrow}$ denotes the last element of the sequence if σ is finite. The meaning of $\downarrow$ should be clear from the contex.

of $\vec{d}$) is in $\delta(s, s_i)$. The distribution $\vec{d}$ is said to be a solution of $\delta(s)$. Let $sol(s)$ be the set of solutions of $\delta(s)$.

Next we give two semantics of IMCs: 1) *Uncertain Markov Chains* (UMC), 2) *Interval Markov Decision Process* (IMDP).

Definition 4.2.4. (**Uncertain Markov chain semantics**) An IMC $\mathcal{M} = (S, L, \delta)$ represents a set of DTMCs, denoted by $[\mathcal{M}]_u$, such that for each DTMC $M = (S, L, \delta_M)$ in $[\mathcal{M}]_u$, $\delta_M(s)$ is a solution of $\delta(s)$ for every state $s \in S$.

In UMC semantics, we assume that nature non-deterministically picks a solution of $\delta(s)$ for each state $s \in S$, and then all transitions behave according to the chosen transition probability matrix.

To define interval Markov decision process semantics, we need the notion of *schedulers*. The schedulers resolve the non-determinism at each state s by choosing a particular distribution from $sol(s)$.

Definition 4.2.5. A *scheduler* of an IMC $\mathcal{M} = (S, L, \delta)$ is a function $\eta : S^+ \to D^{\mathcal{M}}$, such that for every finite sequence of states $\pi{\cdot}s$ of $\mathcal{M}$, $\eta(\pi{\cdot}s)$ is a solution of $\delta(s)$.

A path $w = s_0 s_1 s_2 \cdots$ of an IMC $\mathcal{M}$ is an infinite sequence of states. A path w starting from a state s (i.e., $w_0 = s$) is said to be *according* to the scheduler η if for all $i \geq 0$, $\eta(w_0, \cdots, w_i)(w_{i+1}) > 0$. A scheduler is *memoryless* if the choice of the distribution depends solely on the current state, that is, $\eta : S \to D^{\mathcal{M}}$.

Definition 4.2.6. (**Interval Markov decision process semantics**) In IMDP semantics, before every transition from a state s of a IMC $\mathcal{M} = (S, L, \delta)$, nature chooses a solution of $\delta(s)$ and then takes a one-step probabilistic transition according to the chosen distribution. In other words, nature chooses a scheduler η which then defines a DTMC M. The set of all DTMC in this semantics is denoted by $[\mathcal{M}]_d$.

Obviously, for any IMC $\mathcal{M}$ we have:

$$[\mathcal{M}]_u \subseteq [\mathcal{M}]_d.$$

Given an IMC $\mathcal{M}$ and a state s, let σ-algebra $(\Omega_s, \mathcal{F})$ be the smallest σ-algebra on the cylinder sets of Ω_s, where Ω_s is the set of infinite paths starting from s. For each scheduler η we have a probability measure Pr^{η} (also denoted by $\mu^{\eta}_{\mathcal{M}}$) on the events in $\mathcal{F}$.

Next, we define the satisfaction relation of a PCTL formula f for an IMC $\mathcal{M}$ for the two semantics. In UMC semantics,

$$\mathcal{M}, s \vDash_u f \text{ iff for every DTMC} M \in [\mathcal{M}]_u, M, s \vDash f$$

Note that for a PCTL formula f, $\mathcal{M}, s \vDash_u f$ does not imply $\mathcal{M}, s \nvDash_u \sim f$. In IMDP semantics, the satisfaction of a PCTL formula f by a state s of $\mathcal{M}$ ($\mathcal{M}, s \vDash_d f$) is as follows:

$$\begin{array}{lll} M, s \vDash a & \text{iff} & a \in L(s) \\ M, s \vDash \sim f & \text{iff} & M, s \nvDash f \\ M, s \vDash f_1 \wedge f_2 & \text{iff} & M, s \vDash f_1 \text{ and } M, s \vDash f_2 \\ \mathcal{M}, s \vDash [g]_{\bowtie p} & \text{iff} & \forall \eta : Pr^{\eta}_{\mathcal{M}}\{s \vDash g\} \bowtie p \end{array} \tag{4.1}$$

Particularly,

$$\begin{array}{lll} M, s \vDash [g]_{\leq c} & \text{iff} & \sup_{\eta} Pr^{\eta}\{s \vDash g\} \leq c \\ M, s \vDash [g]_{< c} & \text{iff} & \sup_{\eta} Pr^{\eta}\{s \vDash g\} < c \\ M, s \vDash [g]_{\geq c} & \text{iff} & \inf_{\eta} Pr^{\eta}\{s \vDash g\} \geq c \\ M, s \vDash [g]_{> c} & \text{iff} & \inf_{\eta} Pr^{\eta}\{s \vDash g\} > c \end{array}$$

where, $(s \vDash g) = \{w \mid w_0 = s \text{ and } M, w \vDash g\}$. Thus for an event $E \in \mathcal{F}$, defining a set of paths, we are interested in the values:

$$\inf_{\eta} Pr^{\eta}_{\mathcal{M}}(E) \quad \text{and} \quad \sup_{\eta} Pr^{\eta}_{\mathcal{M}}(E)$$

Open intervals present a problem for model checking in IMDP semantics. There might not exist a scheduler that gives the optimal values. Consider the reachability problem for IMCs in the following example:

Example 4.2.1. It is possible that an optimal scheduler may not exist for IMCs with open intervals. Consider the following example Figure 4.2.1, E is the set of paths that eventually reach the state s_1 from s_0. $\inf_{\eta} Pr^{\eta}(E) = 0.6$, but no scheduler gives the probability of reaching s_1 from s_0 as 0.6. The reason for this is the open lower bound of $(0.3, 1]$.

4.2.1 ϵ-Approximate Scheduler for Reachability

In this section we consider the reachability problem in IMDP semantics for IMCs with open intervals. As observed in the previous example, an optimal scheduler may not exists, thus we will construct ϵ-approximate schedulers.

An IMC is called a *closed* IMC if the probability interval of every transition is closed. We can obtain a closed IMC from an arbitrary IMC by taking the closure of the probability intervals.

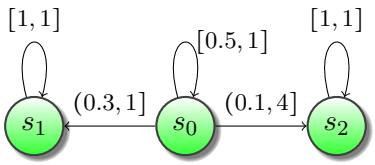


Figure 4.2.1: A interval Markov chain

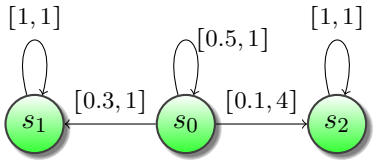


Figure 4.2.2: A closed interval Markov chain

Definition 4.2.7. Given an IMDP $\mathcal{M} \triangleq (S, L, \delta)$, a *closed* IMDP $\bar{\mathcal{M}}$ is defined as (S, L, δ'), where for every s, t, $\delta'(s,t) = \bar{\delta}(s,t)$.

Example 4.2.2. The closed IMC $\bar{\mathcal{M}}$ for $\mathcal{M}$ in the example 4.2.1 is shown below:
Evidently, if an IMC $\mathcal{M}$ is realizable then $\bar{\mathcal{M}}$ is also realizable.

Definition 4.2.8. Basic feasible solution (BFS). Given a set of closed intervals $R \triangleq \{I_1, \cdots, I_m\}$ a basic feasible solution $\vec{d}$ is an $m \times 1$ vector, such that there exists a set $H \subseteq R$ with $|H| \geq |R| - 1$ and for all $I_i \in H$, $\vec{d}_i = I_i\downarrow$ or $\vec{d}_i = I_i\uparrow$, and $\vec{d}^T \cdot \vec{1} = 1$.

BFSs of a set of intervals $\mathcal{J}$ that contains open intervals are the BFSs of the set of closed intervals $\bar{\mathcal{J}}$. We have the following observation.

Proposition 4.2.1. *Every solution of a set of (open or closed) intervals, can be represented as the convex combination of the BFSs.*

Proposition 4.2.2 ([26])**.** *Let $\mathcal{M}$ be a closed IMC, and E be an event defining the reachability of some set of states $T \subseteq S$. There exists a memoryless scheduler η such that the probability of the event E is optimal.*

The proposition says that, if $\mathcal{M}$ is closed then we have a scheduler $\eta : S \to D^{\mathcal{M}}$ such that $Pr^{\eta}(E) = \inf_{\eta'} Pr^{\eta'}(E)$ (or $\sup_{\eta'} Pr^{\eta'}(E)$), and η chooses at each state s one of the BFSs of $\delta(s)$ (pure scheduler). The proposition follows directly from the existence of an optimal scheduler for reachability in Markov Decision Processes [9].
The main theorem of this chapter is as follows:

Theorem 4.2.3. *Let E be the event describing the set of paths of an IMC $\mathcal{M}$ starting from a state s and eventually reaching some goal states T. Then:*

$$\forall \varepsilon > 0 \quad \exists \hat{\eta} : |\min_{\eta} Pr^{\eta}_{\bar{\mathcal{M}}}(E) - Pr^{\hat{\eta}}_{\mathcal{M}}(E)| \leq \varepsilon$$

and

$$\forall \varepsilon > 0 \quad \exists \hat{\eta} : |\max_{\eta} Pr^{\eta}_{\bar{\mathcal{M}}}(E) - Pr^{\hat{\eta}}_{\mathcal{M}}(E)| \leq \varepsilon$$

Proof. Let $\mathcal{M} \triangleq (S, L, \delta)$ and $\bar{\mathcal{M}} \triangleq (S, L, \delta')$. $\bar{\mathcal{M}}$ is closed, thus by Prop. 4.2.2 an optimal scheduler exists. Let $\overset{*}{\eta}$ be an optimal scheduler that minimizes $\Pr^{\eta}_{\bar{\mathcal{M}}}(E)$. Furthermore, $\overset{*}{\eta}$ is memoryless, deterministic and chooses one of the BFS of $\delta'(s)$ at each state s. Hence, $\overset{*}{\eta}$ induces a DTMC on $\bar{\mathcal{M}}$, and $\overset{*}{\eta}(s,t)$ defines the single step transition probability from a state s to a state t.

Let the stochastic matrix $\overset{*}{P}$ be such that each row is identified with a state of $\bar{\mathcal{M}}$. We have :

$$\overset{*}{P}(s,t) = \overset{*}{\eta}(s,t) \quad \text{if } s \notin T \quad \text{and} \quad \overset{*}{P}(s,s) = 1 \quad \text{if } s \in T \tag{4.2}$$

Let $A = (1 + \overset{*}{P} + (\overset{*}{P})^2 + (\overset{*}{P})^3 \cdots)$, A is well-defined stochastic matrix as the series converges. Let $\gamma = \|A\|_\infty$.

Now we are in a position to define a scheduler $\hat{\eta}$ for the IMC $\mathcal{M}$. The scheduler $\hat{\eta}$ is a function, $\hat{\eta} : S \times \mathbb{N} \to D^{\mathcal{M}}$. We assume that there are no positive point intervals. (We can set the value of $\hat{\eta}$ if point intervals are present.) Define the following:

$$\begin{aligned}
Q_s &= \{t \mid \overset{*}{\eta}(s,t) > 0,\ \overset{*}{\eta}(s,t) \notin \delta(s)\} \\
L_s &= \{t \mid \overset{*}{\eta}(s,t) \in \delta(s,t),\ \overset{*}{\eta}(s,t) = \delta(s,t)\downarrow\} \\
R_s &= \{t \mid \overset{*}{\eta}(s,t) \in \delta(s,t),\ \overset{*}{\eta}(s,t) = \delta(s,t)\uparrow\} \\
I_s &= \{t \mid \overset{*}{\eta}(s,t) \in \delta(s,t),\ \overset{*}{\eta}(s,t) \neq \delta(s,t)\uparrow,\ \overset{*}{\eta}(s,t) \neq \delta(s,t)\downarrow\} \\
\rho &= \min\{\{x \mid \exists s, \exists t \in L_s \cup I_s : x = \overset{*}{\eta}(s,t) - \delta(s,t)\downarrow\}, \\
&\quad \{x \mid \exists s, \exists t \in R_s \cup I_s : x = \delta(s,t)\uparrow - \overset{*}{\eta}(s,t)\}, \\
&\quad \{x \mid \exists s, \exists t \in Q_s : x = \delta(s,t)\uparrow = \delta(s,t)\downarrow\}\}
\end{aligned}$$

Observe that ρ is a constant of the model $\mathcal{M}$. Let $\hat{\eta}$ be defined as follows:

- Let $t \in Q_s$. This implies $\overset{*}{\eta}(s,t) = \delta(s,t)\uparrow$ or $\overset{*}{\eta}(s,t) = \delta(s,t)\downarrow$. If $\overset{*}{\eta}(s,t) = \delta(s,t)\uparrow$ then $\delta(s,t)$ is open from above and $\hat{\eta}(s,n,t) = \overset{*}{\eta}(s,t) - 2^{-n}\frac{\kappa\rho}{|Q_s|}$, where $\kappa = \frac{\varepsilon}{1+\gamma}$. Similarly, if $\overset{*}{\eta}(s,t) = \delta(s,t)\downarrow$ then $\delta(s,t)$ is open from below and $\hat{\eta}(s,n,t) = \overset{*}{\eta}(s,t) + 2^{-n}\frac{\kappa\rho}{|Q_s|}$.

- Let $t \in R_s$ and $\alpha \triangleq \sum_{t \in Q_s} \hat{\eta}(s,n,t) - \overset{*}{\eta}(s,t)$. If $\alpha < 0$ then for all $t \in R_s \cup I_s$, $\hat{\eta}(s,n,t) = \overset{*}{\eta}(s,t) + \frac{\alpha}{|R_s \cup I_s|}$ and for $t \in L_s$, $\hat{\eta}(s,n,t) = \overset{*}{\eta}(s,t)$. If $\alpha > 0$ then for all $t \in L_s \cup I_s$, $\hat{\eta}(s,n,t) = \overset{*}{\eta}(s,t) + \frac{\alpha}{|L_s \cup I_s|}$ and for $t \in R_s$, $\hat{\eta}(s,n,t) = \overset{*}{\eta}(s,t)$. If $\alpha = 0$ then for all $t \in L_s \cup I_s \cup R_s$, $\hat{\eta}(s,n,t) = \overset{*}{\eta}(s,t)$.

It remains to prove that $\vec{d} = \overset{*}{\eta}(s,n)$, defined above, is a solution to $\delta(s)$. From the construction it follows that $\sum_{t \in S} \vec{d}_t = 1$ and hence it is a valid

distribution on the states of the IMC $\mathcal{M}$. Consider the following cases: $t \in Q_s$ and $\overset{*}{\eta}(s,t) = \delta(s,t)\uparrow$, the upper bound of $\delta(s,t)$ is open. The lower bound of $\delta(s,t)$ is strictly smaller than $2^{-n}\kappa\rho$ for any $n \in \mathbb{N}$ i.e., $\delta(s,t)\downarrow < \kappa\rho$ since ρ is at the most as large as the smallest interval in $\mathcal{M}$. Thus $\vec{d}_t \in \delta(s,t)$. Similarly, for every $t \in Q_s$, $\vec{d}_t \in \delta(s,t)$. Suppose $\alpha < 0$, then $R_s \cup I_s$ is not empty, else $\delta(s)$ will not be realizable. The changes to the probability for a transition s to t, where $t \in R_s \cup I_s$ is small enough so that $\vec{d}_t \in \delta(s,t)$. Thus, for every t, $\vec{d}_t \in \delta(s,t)$, or equivalently $\vec{d}$ is a solution to $\delta(s,t)$. Identical argument holds when $\alpha > 0$.

Let $\hat{P}_n$ be a sub-stochastic matrix defined as follows: $\hat{P}_n(s,t) = \hat{\eta}(s,t)$ if $\overset{*}{P}(s,t) > 0$ else $\hat{P}_n(s,t) = 0$. In other words, $\hat{P}_n(s,t) > 0$ if the state t is in $support(\overset{*}{\eta}(s))$.

$$\hat{P}_n = \overset{*}{P} + P_n \tag{4.3}$$

where $|P_n(s,t)| \leq 2^{-n}\kappa\rho$ for every (s,t).

Let $\overset{*}{\eta}$ and $\hat{\eta}$ induce DTMCs M' and M on the IMCs $\bar{\mathcal{M}}$ and $\mathcal{M}$, respectively. Let the corresponding σ-algebra be $\mathcal{S} \triangleq (\Omega_s, \mathcal{F}, \overset{*}{\mu})$ and $\mathcal{S}' \triangleq (\Omega_s, \mathcal{F}, \hat{\mu})$, where s is some state of $\mathcal{M}$ and Ω_s is the set of paths starting from state s. Define $\overset{*}{R} \triangleq \{w \in \Omega_s \mid w \text{ is according to } \overset{*}{\eta}\}$ and $\hat{R} \triangleq \{w \in \Omega_s \mid w \text{ is according to } \hat{\eta}\}$, i.e., $\overset{*}{R}$ and R are set of paths in M' and M, respectively. Let $B \in \mathcal{F}$ be the event of reaching the goal states T, and $E = \overset{*}{R} \cap B$ and $E' = \hat{R} \cap B$. It follows from the construction that $E \subseteq E'$. Define $A_i \triangleq \{w \mid \exists u \in E : w_0 \cdots w_i = u_0 \cdots u_i \text{ and } \overset{*}{\eta}(w_i, w_{i+1}) = 0, \hat{\eta}(w_i, i, w_{i+1}) > 0\}$. Let $A = \bigcup_i A_i$. It is easy to see that, $E' \cap \bar{A} = E$. We will first show that the event A has a very small probability measure in $\mathcal{S}'$:

$$\hat{\mu}(A) = Pr_M^{\hat{\eta}}(A) = \sum_{i=0} Pr_M^{\hat{\eta}}(A_i)$$

If $w \in A_i$ then $\delta(w_i, w_{i+1})\uparrow > 0$ and $\overset{*}{\eta}(s,t) = 0$. Thus,

$$Pr_M^{\hat{\eta}}(A_i) \leq 2^{-i}\kappa\rho \quad \text{or} \quad Pr_M^{\hat{\eta}}(A) \leq \kappa\rho$$

Thus,

$$\hat{\mu}(A) \leq \kappa\rho \tag{4.4}$$

We will now show that the probability of E' can be made infinitesimally close to the probability of E. Formally, we will show, $|\hat{\mu}(E') - \bar{\mu}(E)| \leq \varepsilon$. The left hand side can be written as:

$$\begin{aligned} |\hat{\mu}(E') - \bar{\mu}(E)| &= |\hat{\mu}(E' \cap A) + \hat{\mu}(E' \cup \bar{A}) - \bar{\mu}(E)| \\ &\leq |\hat{\mu}(E) - \bar{\mu}(E)| + \kappa\rho \end{aligned} \tag{4.5}$$

That is, we restrict to the paths that belong to E. Let x_s^n denote the probability of reaching the goal states T at the n^{th} step in M' from the state s. Let E_n be the event of reaching the goal states T at the n^{th} step in the Markov chain M such that $E_n \subseteq E$ and thus $\bigcup_n E_n = E$. Let $y_s^n = \hat{\mu}(E_n)$. Thus, we can write the following:

$$\begin{aligned} x_s^{n+1} &= \sum_{t \in support(\overset{*}{\eta}(s))} \overset{*}{P}(s,t) x_t^n, \\ y_s^{n+1} &= \sum_{t \in support(\overset{*}{\eta}(s))} \hat{P}_n(s,t) y_t^n. \end{aligned}$$

Or, using vector notation, $\vec{x}_{n+1} = \overset{*}{P}\vec{x}_n$ and $\vec{y}_{n+1} = \hat{P}_n \vec{y}_n$. Therefore:

$$\begin{aligned} \vec{y}_{n+1} - \vec{x}_{n+1} &= \overset{*}{P}(\vec{y}_n - \vec{x}_n) + P_n \vec{y}_n && \text{from equation (4.3)} \\ &\leq \overset{*}{P}(\vec{y}_n - \vec{x}_n) + 2^{-n}\kappa\rho\vec{1} \\ &\leq 2^{-n}\kappa\rho(1 + \overset{*}{P} + \overset{*}{P}^2 + \cdots)\vec{1} \\ \text{Thus,} \quad \|\vec{y}_{n+1} - \vec{x}_{n+1}\|_\infty &\leq 2^{-n}\kappa\rho\gamma. \end{aligned}$$

We have,

$$|\hat{\mu}(E) - \bar{\mu}(E)| \leq |\sum_n (y_s^n - x_s^n)| \leq \sum_n 2^{-n}\kappa\rho\gamma \leq \kappa\rho\gamma$$

Combining this with equation (4.5) we can conclude:

$$|\hat{\mu}(E') - \bar{\mu}(E)| \leq (1+\gamma)\kappa\rho \leq \varepsilon$$

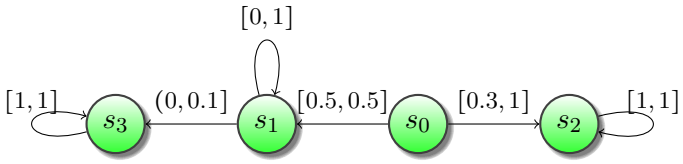
By similar argument we conclude $\forall \varepsilon > 0 \ \exists \hat{\eta} : |\max_\eta Pr^\eta_{M'}(E) - Pr^{\hat{\eta}}_M(E)| \leq \kappa$. □

Corollary. *Let E be the set of paths that reach some goal states T of IMC $\mathcal{M}$. Then:*

$$\min_\eta Pr^\eta_{\bar{\mathcal{M}}}(E) = \inf_\eta Pr^\eta_{\mathcal{M}}(E) \ \textit{and} \ \max_\eta Pr^\eta_{\bar{\mathcal{M}}}(E) = \sup_\eta Pr^\eta_{\mathcal{M}}(E).$$

Proof. We need to show $\forall \kappa > 0 \ \exists \hat{\eta} : |\min_\eta Pr^\eta_{M'}(E) - Pr^{\hat{\eta}}_M(E)| \leq \kappa$. Observe that, $\hat{\eta}$ is also a scheduler of M', thus, $Pr^{\hat{\eta}}_M(E) - \min_\eta Pr^\eta_{M'}(E) \leq \kappa$. Similarly, for all $\kappa > 0$ there exists a scheduler $\hat{\eta}$ of M such that $\max_\eta Pr^\eta_{M'}(E) - Pr^{\hat{\eta}}_M(E) \leq \kappa$. □

Example 4.2.3. In UMC semantics, the nature picks the probability transition matrix and the model behaves according to it. The infimum (or supremum) probability of reaching some state is different than the infimum probability in IMDP semantics. This becomes apparent in the following IMC with an open interval:



The minimum and maximum probability of reaching state s_3 from s_0 in the UMC semantics is 0.5. But for any $\epsilon > 0$ there exists a scheduler for which the probability of reaching s_3 is smaller than ϵ. That is, the infimum of the probability of reaching state s_3 is 0.

4.2.2 Model checking interval Markov chains with open intervals

In this section we briefly recall PCTL model checking on DTMC and IMCs with closed intervals (for the two semantics), and then show how to use the result of previous section to do model checking for IMCs with open intervals.

Model checking of PCTL [31, 5] formula f on DTMC M proceeds much like the CTL model checking on Kripke structures [29]. The satisfiability of a (state) sub-formula f' of f for a state s of M is iteratively calculated and the labeling functions are updated accordingly. For example, for the until formula $f = \mathsf{P}_{\bowtie p}(f_1 \mathsf{U} f_2)$ and a state s, the formula f is added to the label of s iff the probability of reaching states with label f_2, via states with label f_1 satisfies $\bowtie p$. This can be done in polynomial time by solving linear constraints. Finally, a state $s \models f$ if $f \in L(s)$ and the model checking problem can be solved in polynomial time.

Model checking in UMC semantics uses the *existential* theory of the reals [87]. An IMC $\mathcal{M}, s \models_u f$ in UMC semantics iff for all DTMC $M \in [\mathcal{M}]_u$, $M, s \models f$, or equivalently, $\mathcal{M}, s \not\models_u f$ iff there exists a $M \in [\mathcal{M}]_u$ such that $M, s \models \sim f$. Basically, we use parameters to encode the transition probabilities which are constrained by the intervals and construct a formula Γ in existential theory of reals such that Γ is satisfiable iff there exists a $M \in [\mathcal{M}]_u$ such that $M, s \models \sim f$ [26]. Observe, that the presence (or absence) of open intervals does not affect the algorithm and the algorithm operates in PSPACE.

Model checking in IMDP semantics is done by first transforming the IMC into an MDP and then doing model checking on the MDP [9]. Let $\mathcal{M} = (S, L, \delta)$ be a *closed* IMC and for each state $s \in S$, let B_s be the set of basic feasible solution of $\delta(s)$. Let $D_{\mathcal{M}} = (S, L, \mu)$ be the MDP with $\mu : S \to S \to [0, 1]$, where $\mu(s) = B_s$. From Proposition 4.2.1, we can deduce that, a DTMC $M \in [\mathcal{M}]_d$ iff M is induced by some scheduler η of

$D_{\mathcal{M}}$. Model checking of MDP proceeds the same way as model checking of DTMC. We iteratively update the labels of the state with (state) subformulas. Conjunctions and disjunctions are handled as in the DTMC model checking. Interesting cases are formulas with probabilistic operator and negations. Let g be a path formula and $\mathsf{P}_{\succ p}g$ (or $\mathsf{P}_{\prec p}g$) is added to the label of a state $s \in S$, iff

$$\min_{\eta} Pr^{\eta}_{D_{\mathcal{M}}}(s \vDash g) \succ p \text{ (or } \max_{\eta} Pr^{\eta}_{D_{\mathcal{M}}}(s \vDash g) \prec p)$$

where $\succ \in \{\geq, >\}$ ($\prec \in \{\leq, <\}$). This is done by solving a linear optimization problem. We use the following proposition to handle formulas with negations.

Proposition 4.2.4. *For any $E \in \mathcal{F}$ of $(\Omega_s, \mathcal{F})$ on MDP M,*

$$\inf_{\eta} Pr^{\eta}(E) \;=\; 1 - \sup_{\eta} Pr^{\eta}(\bar{E})$$

Thus, model checking MDPs boils down to solving successive reachability optimization problems. Note that direct application of this method to IMCs with open interval is not possible since no scheduler exists which may yields the value $\inf_{\eta} Pr^{\eta}_{D_{\mathcal{M}}}(s \vDash g)$.

In the rest of the section we use the above mentioned model checking mechanism to show that model checking IMCs with open interval in IMDP semantics, reduces to model checking its closure.

Theorem 4.2.5. *Given a PCTL formula f and an IMC $\mathcal{M}$,*

$$\mathcal{M}, s \vDash f \quad \textit{iff} \quad \bar{\mathcal{M}}, s \vDash f$$

Proof. We assume that $\mathcal{M}$ has open intervals. We proceed by induction on the structure of the formula f. We have the following cases:

1. Let $f := a$. The labeling function of s in $\mathcal{M}$ and $\bar{\mathcal{M}}$ are identical. Thus, $\mathcal{M}, s \vDash f$ iff $\bar{\mathcal{M}}, s \vDash f$.

2. Let $f := {\sim}f'$. From the induction hypothesis, $\mathcal{M}, s \nvDash f'$ iff $\bar{\mathcal{M}}, s \nvDash f'$. Thus, $\mathcal{M}, s \vDash f$ iff $\bar{\mathcal{M}}, s \vDash f$.

3. Let $f := f_1 \wedge f_2$. From the induction hypothesis, $\mathcal{M}, s \vDash f_1$ iff $\bar{\mathcal{M}}, s \vDash f_1$ and $\mathcal{M}, s \vDash f_2$ iff $\bar{\mathcal{M}}, s \vDash f_2$. Thus, $\mathcal{M}, s \vDash f$ iff $\bar{\mathcal{M}}, s \vDash f$.

4. Let $f := [\mathsf{X}f']_{\bowtie c}$. Consider the case $\bowtie \in \{\geq, >\}$. Suppose $\overset{*}{\eta}$ be the optimal scheduler of $\bar{\mathcal{M}}$ such that $\Pr^{\overset{*}{\eta}}_{\bar{\mathcal{M}}}(\mathsf{X}f') = \min_{\eta} \Pr^{\eta}_{\bar{\mathcal{M}}}(\mathsf{X}f')$.

We show that for every ε we can construct a scheduler $\hat{\eta}$ of $\mathcal{M}$ such that

$$Pr^{\hat{\eta}}_{\mathcal{M}}(\mathsf{X}f') - Pr^{\overset{*}{\eta}}_{\bar{\mathcal{M}}}(\mathsf{X}f') \leq \varepsilon.$$

Observe that, any scheduler of $\mathcal{M}$ is also a scheduler of $\bar{\mathcal{M}}$, since for any states $s, t \in S$ $\delta(s,t) \subseteq \bar{\delta}(s,t)$. Thus, Corollary 4.2.1. is applicable. Let $Q_s \triangleq \{t \mid \overset{*}{\eta}(s,t) > 0, \overset{*}{\eta}(s,t) \notin \delta(s)\}$ and $R_s \triangleq \{t \mid \overset{*}{\eta}(s,t) > 0, \overset{*}{\eta}(s,t) \in \delta(s,t)\}$. We assume that Q_s, R_s are not empty and there are no point intervals. Let $\hat{\eta}(s) = \vec{d}$, where $\vec{d}$ is defined as follows:

- Let $t \in Q_s$. This implies $\overset{*}{\eta}(s,t) = \delta(s,t)\uparrow$ or $\overset{*}{\eta}(s,t) = \delta(s,t)\downarrow$. If $\overset{*}{\eta}(s,t) = \delta(s,t)\uparrow$ then $\delta(s,t)$ is open from above and $\vec{d}_t = \overset{*}{\eta}(s,t) - \frac{\varepsilon\rho}{|S|}$, where ρ is the minimum of the length of the non-zero interval in $\mathcal{M}$ and the $\overset{*}{\eta}(s,t)$ for $t \in R_s$. Similarly, if $\overset{*}{\eta}(s,t) = \delta(s,t)\downarrow$ then $\delta(s,t)$ is open from below and $\vec{d}_t = \overset{*}{\eta}(s,t) + \frac{\varepsilon\rho}{|S|}$.
- Let $t \in R_s$ and $\alpha \triangleq 1 - \sum_{t \in Q_s} \vec{d}_t - \sum_{t \in R_s} \overset{*}{\eta}(s,t)$. We have $\vec{d}_t = \overset{*}{\eta}(s,t) + \frac{\alpha}{|R_s|}$.

It follows that $\vec{d}$ is a distribution on the states of $\mathcal{M}$ and is a solution to $\delta(s)$. Let $E \triangleq \{w \mid \overset{*}{\eta}(w_0, w_1) > 0 \text{ and } \bar{\mathcal{M}}, w_1 \vDash f'\}$ and $E' \triangleq \{w \mid \hat{\eta}(w_0, w_1) > 0 \text{ and } \mathcal{M}, w_1 \vDash f'\}$.

$$|\Pr^{\overset{*}{\eta}}_{\bar{\mathcal{M}}}(E) - \Pr^{\hat{\eta}}_{\mathcal{M}}(E')| \leq \sum_{t \in support(\hat{\eta}(s))} \frac{\varepsilon\rho}{|S|} \quad \leq \quad \varepsilon$$

Thus we can conclude that $\inf_\eta \Pr^\eta_{\mathcal{M}}(\mathsf{X}f') = \min_\eta \Pr^\eta_{\bar{\mathcal{M}}}(\mathsf{X}f')$. By similar argument:

$$\sup_\eta Pr^\eta_{\mathcal{M}}(\mathsf{X}f') = \max_\eta Pr^\eta_{\bar{\mathcal{M}}}(\mathsf{X}f').$$

$\mathcal{M}, s \vDash [\mathsf{X}f']_{\bowtie c}$ iff $\bar{\mathcal{M}}, s \vDash [\mathsf{X}f']_{\bowtie c}$, where $\bowtie \in \{\leq, <\}$.

5. Let $f := [f_1 \mathsf{U} f_2]_{\bowtie c}$. Suppose $\bowtie \in \{\geq, >\}$. By induction hypothesis, for every s, $\mathcal{M}, s \vDash f_1$ iff $\bar{\mathcal{M}}, s \vDash f_1$ and $\mathcal{M}, s \vDash f_2$ iff $\bar{\mathcal{M}}, s \vDash f_2$. Let $S_1 \triangleq \{s \mid s, \mathcal{M} \vDash f_1\}$ and $T \triangleq \{s \mid s, \mathcal{M} \vDash f_2\}$. The IMC $\mathcal{M}'$ is obtained from $\mathcal{M}$ by omitting states not present in the set $S_1 \cup T$. It is easy to see that, if E is the event of reaching T in $\mathcal{M}'$, then $\inf_\eta \Pr^\eta_{\mathcal{M}'}(E) = \inf_\eta \Pr^\eta_{\mathcal{M}}(f)$.

   From Corollary 4.2.1 it follows that for any $0 < \varepsilon \leq 1$ we can find $\hat{\eta}$ such that $\Pr^{\hat{\eta}}_{\mathcal{M}'}(E) - \min_\eta \Pr^\eta_{\bar{\mathcal{M}}'}(E) \leq \varepsilon$, where E is the event of reaching T in $\mathcal{M}'$. Thus $\inf_\eta \Pr^\eta_{\mathcal{M}}(f) = \min_\eta \Pr^\eta_{\bar{\mathcal{M}}}(f)$. Similar argument holds for $\bowtie \in \{<, \leq\}$.

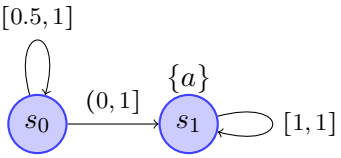


Figure 4.2.3: An interval Markov chain, where the result of PCTL model checking result for UMC semantics differ from the IMDP semantics.

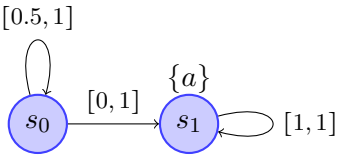


Figure 4.2.4: The corresponding closed interval Markov chain where model checking results are identical for the two semantics.

This concludes the proof. □

Example 4.2.4. Consider PCTL model checking of IMCs in UMC semantics. This involves existentially quantifying the transition probabilities and creating a formula in closed real field [26]. This captures a strict set of DTMC as compared to IMDP semantics, i.e, $[\mathcal{M}]_u \subsetneq [\mathcal{M}]_d$. For example, DTMC where the transition probability between two states s, t change over time cannot be represented in UMC semantics. This is exemplified by the IMC $\mathcal{M}$ in Figure 4.2.3.

The probability of satisfying the path formula $g = \mathsf{G}(\sim a \wedge [\mathsf{X}a]_{>0})$ in the UMC semantics is 0. But we can find schedulers which can make the probability of going to a-state arbitrarily close to 1. The scheduler has the freedom to defines an infinite Markov chain by assigning monotonically increasing probabilities for the transition $s_0 \to s_0$).

The model checking of the open IMC $\mathcal{M}$ is done by first closing it (Figure 4.2.4). This gives us the closed IMC $\bar{\mathcal{M}}$:

The maximum probability of satisfying g in $\bar{\mathcal{M}}$ is 1. Which implies, for every $0 < \varepsilon \leq 1$, there exists a scheduler $\hat{\eta}$, for which the probability of staying in a state that satisfies $\sim a \wedge [\mathsf{X}a]_{>0}$ (s_0) is greater than $1 - \varepsilon$, by Theorem 4.2.5.

4.3 Strategy synthesis for MDPs for PCTL objectives

We have seen that an IMC is useful for model with uncertainties. Topologically, an IMC is a set of Markov chains and the actual system behaviour can be any one of the many Markov chains in the set. We have also seen that we can model check an IMC (even in the presence of not closed intervals) against a PCTL formula. The natural question to ask is: whether every

Markov chain defined by an IMC is satisfies a given PCTL formula? Or the complement problem of *scheduler synthesis*, is to find a scheduler such that the induced Markov chain satisfies a given PCTL formula.

Let us first consider the UMC semantics. By definition, the satisfaction relation in UMC semantics decides whether all Markov chains defined by the UMC semantics satisfies a PCTL formula. Thus, strategy synthesis is just model checking the negation of the given formula in UMC semantics. Form [89] we know that the problem is solvable in PSPACE. But strategy synthesis in IMDP semantics is not so trivial. Let $\mathcal{M} = (S, L, \delta)$ be an IMC and $D = (S, L, \Delta)$ is a Markov decision process where, for each $s \in S$, $\Delta(s)$ if the set of basic feasible solutions of $\delta(s)$. We have seen that, $\mathcal{M}$ is behaviorally identical to D, that is, for any MC M, $M \in [\mathcal{M}]_d$ if and only if there exists a scheduler η, such that $D_\eta = M$. To be precise, $[\mathcal{M}]_u$ corresponds to all memoryless randomized (MR) schedulers and $[\mathcal{M}]_d$ corresponds to all history dependent randomized schedulers (HR) of D. Thus, scheduler synthesis in IMC naturally leads to the scheduler synthesis in MDPs.

4.3.1 Scheduler Synthesis problem for MDPs.

The scheduler synthesis problem is defined as follows: Given a MDP D and PCTL formula f, does there exist a scheduler η such that D_η satisfies f? The first guess would be: Doesn't the model checking algorithm of a PCTL formula on MDPs (as defined in § 4.2) gives us a method to solve the scheduler synthesis problem? After all, in the semantics (equation 4.1) we explicitly define that the probability measure of the set of paths satisfying a path (sub)formula is the optimal (supremum or infimum) over *all* schedulers. Unfortunately, this is not always the case, and the following example elucidates this fact. Consider the MDP D (as shown in Figure 4.3.5), and

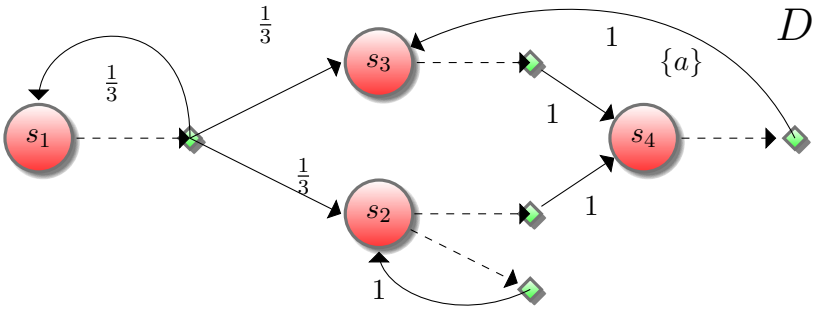


Figure 4.3.5: The state s_1 of the MDP D satisfies $f = [\mathsf{F}\ [\mathsf{X}a]_{\geq 1}]_{\leq \frac{1}{2}}$.

a PCTL formula $f = [\mathsf{F}\ [\mathsf{X}a]_{\geq 1}]_{\leq \frac{1}{2}}$. The s_2 satisfies $[\mathsf{X}a]_{\geq 1}$, and s_3 does not satisfies $[\mathsf{X}a]_{\geq 1}$, since there are schedulers for which the probability measure of $\mathsf{X}a$ is zero. The state s_1 the maximum probability of reaching a state satisfying $[\mathsf{X}a]_{\geq 1}$ (state s_2 in this case) is $\geq \frac{1}{2}$. But observe that there exists a scheduler η, which induces the MC D_η (shown in Figure 4.3.6) where the state s_1 does not satisfy f. So there exists MDPs D and PCTL formulas f

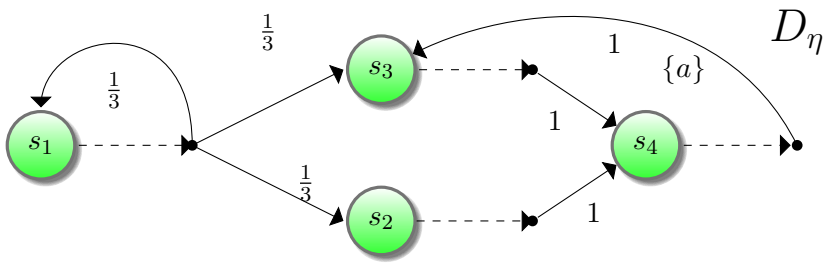


Figure 4.3.6: The MC D_η induced by η on MDP D which does not satisfy $f = [\mathsf{F}\ [\mathsf{X}a]_{\geq 1}]_{\leq \frac{1}{2}}$.

where, $D, s \vDash f$ (according to Equations 4.1), yet there exists scheduler η such that $D_\eta, s \nvDash f$ (according to Equation 4.1). The problem is obviously the presence of non-determinism of the probability distribution at each state. In the above example only state with non-deterministic choice over the distribution is s_2. It is interesting to note that neither the formulas $[\mathsf{X}a]_{\geq 1}$ nor its negation ($[\mathsf{X}\sim a]_{>0}$) are satisfied at s_2. This causes the problem since, we are assuming if s_2 does not satisfies $[\mathsf{X}a]_{\geq 1}$, then s_2 satisfies ($[\mathsf{X}\sim a]_{>0}$), which is not the case here.

The example shows that scheduler synthesis problem in MDP for a given PCTL formula may not be as simple as model checking. It turns out that, this problem is indeed very hard, Σ_1^1-hard in the analytic hierarchy.

Theorem 4.3.1 ([12])**.** *The scheduler synthesis problem in MDPs for PCTL formula is Σ_1^1-hard.*

4.4 Conclusion

We studied the problem of model checking Interval Markov chains with open intervals, and seen how to model check them against PCTL formulas. Interval Markov chains are but special cases of more complex Markovian models, called *constraint Markov chains* (CMC) [16]. Transition probabilities in these models are defined as a solution to linear equations. A constraint Markov chain is a tuple $\mathcal{M} = (S, L, \delta)$, where the transition function

$\delta : (2^{F_V})^S$, maps each state to a set of linear in-equations, where F_V is the set of linear in-equations on variables V. Thus IMCs are a strict sub-class of convex Markov decision process. The behaviour of a CMC can again be defined in the UMC and IMDP semantics. Model checking CMCs with non-strict inequalities in the system on linear equations, suffers from the same kinds of problems as described for open IMCs. Theorem 4.2.5 can be easily extended to CMCs as well. We can define *basic feasible solutions* for a system of linear in-equations as follows. For a state s of a CMC $\mathcal{M}$, the transition $\delta(s)$ forms a convex hull and its BFSs are the vertices of the convex hull. The same argument as in the proof of Theorem 4.2.5 shows that, model checking of PCTL formulas on CMCs can be done by first closing the system of in-equations, this is done by replacing the strict inequalities $(<, >)$ with non-strict inequalities $(\leq, \geq)$, and then model checking the closed model.

Our study of interval Markov chain with open interval was motivated by the investigation into the Hintikka game semantics for PCTL. Hentikka game semantics for PCTL on labeled Markov chain [42] "gives an operational account of the denotational semantics of PCTL". Given a denumerable Markov chain M, a state s and a PCTL formula f, a 2-player game $\mathsf{G}_{s,f}$ is obtained with Büchi acceptance property, such that $M, s \vDash f$ iff Player 0 has a winning strategy. This perspective of defining the meaning of a logical formula in terms of the existence of a winning strategy of a 2-player game can be found as early as 1959 in the works of Henkin (where the game semantics was implicit) and formally presented in 1982 in the seminal work of Jaakko Hintikka [56] for first order logic.

We can define the semantics of PCTL for MDPs (or an IMC) as follows. An MDP D satisfies a PCTL formula f iff there exists a scheduler η such that the induced MC D_η satisfies f. We can easily envisage a similar 2-player game with Büchi acceptance property for defining the acceptance of PCTL formula for an MDP. We can immediately show that D satisfies f if and only if there exists a winning strategy for Player 0, by relying on the Martin's determinacy theorem [77]. It is important to note that this point of view, though intuitive, cannot be used to fashion an decision algorithm, as the game thus formed may have uncountably many configurations.

Chapter 5

Decidable Fragments of PCTL

In this chapter, we give a decision procedure for the satisfiability problems for a bounded fragment of probabilistic CTL (called bounded PCTL). We provide an NEXPTIME-algorithm for the satisfiability problem and show that the logic has *small model property* where the size of a canonical model is independent from the probability bounds in the formula. We show that the satisfiability problem of a simple sub-logic of bounded PCTL is PSPACE-complete. In this chapter we will also discuss some of the reason what makes the satisfiability problem for PCTL difficult.

5.1 Introduction

The synthesis problem for a specification language (or a logic) is to create a model from a given sentence. Synthesis of models of a logic can be viewed as the functional equivalent of the satisfiability decision problem. From a practical point of view, the model one wants to synthesize from a specification need to be finite (or at least finitely representable). As we have discussed in the introduction of this thesis, this is generally not the case for PCTL. Even qualitative PCTL fail to possess finite model property. In our attempts to tackle the satisfaction problem, we have found that the difficulty of providing a decision procedure for probabilistic logics primarily lies in the presence of recursively defined path formulas with probabilities. This includes formulas of the form $[a \mathsf{U} b]_{\geqslant \frac{1}{2}}$ expressing that a b-state is to be reached via a-states with probability at least $\frac{1}{2}$. This chapter therefore considers probabilistic logics in which syntactic restrictions are imposed on recursively defined path formulas.

Bounded PCTL is a PCTLfragment in which until-modalities are bounded by the number of steps that can be taken; e.g., $[a \,\mathsf{U}^n b]_{\geqslant \frac{1}{2}}$ expresses that a b-state is to be reached within n steps. Bounded PCTL thus abandons the unbounded until-modality. We will see that the logic has a finite (tree) model property where the size of the model is independent from the probability bounds (like $\frac{1}{2}$) in the formula. To study the computational complexity of bounded PCTL satisfiability, we will first show that the satisfiability problem of a simpler sub-logic of bounded PCTL that (besides propositional logic) only contains nested quantified next-modalities is PSPACE-complete. The main result is an NEXPTIME-algorithm for the entire bounded PCTL satisfiability. This is based on a novel variable elimination method for solving the satisfiability problem for specific class of formulas in the theory of the reals. Finally, we show that the satisfiability of bounded PCTL-formula f is EXPTIME-hard in the encoding of f.

5.2 Bounded PCTL

We consider the sub-logic bounded PCTL which contains the next operator X and bounded until U^n. The syntax of bounded PCTL is as follows:

Definition 5.2.1 (Bounded PCTL)**.**

$$\begin{array}{lcl} f & ::= & a \mid \sim f \mid f \wedge f \mid [g]_{\succ p} \\ g & ::= & \mathsf{X}f \mid f \,\mathsf{U}^n f. \end{array}$$

where $a \in \mathsf{AP}$, $\succ \in \{>, \geq\}$ is the comparison operator, $p \in [0,1] \cap \mathbb{Q}$ is a rational number and $n \in \mathbb{N}$ is an integer.

For an MC M, and state s, the pointed satisfaction is defined as:

$$\begin{array}{lll} M, s \vDash a & \text{iff} & a \in L(s) \\ M, s \vDash \sim f & \text{iff} & M, s \nvDash f \\ M, s \vDash f_1 \wedge f_2 & \text{iff} & M, s \vDash f_1 \text{ and } M, s \vDash f_2 \\ M, s \vDash [g]_{\succ p} & \text{iff} & \Pr\{w \in \Omega_s \mid M, w \vDash g\} \succ p. \end{array}$$

For an infinite path w, the satisfaction relation for the path formulas is defined as:

$$\begin{array}{lll} M, w \vDash \mathsf{X}f & \text{iff} & M, w_1 \vDash f \\ M, w \vDash f \,\mathsf{U}^0 g & \text{iff} & M, w_0 \vDash g \\ M, w \vDash f \,\mathsf{U}^n g & \text{iff} & M, w_0 \vDash g \text{ or } (M, w_0 \vDash f \text{ and } M, w_1 \vDash f \,\mathsf{U}^{n-1} g) \quad \text{if } n > 0 \end{array}$$

Thus, an infinite path w satisfies $f \,\mathsf{U}^n g$ iff $M, w_i \vDash g$ for some $i \leq n$ and for every $j < i$, $M, w_j \vDash f$. Note that, $[f \,\mathsf{U}^n g]_{\succ p}$ cannot be expressed in PCTL.[1] Let $\mathsf{F}^n g$ denote $\mathsf{true}\,\mathsf{U}^n g$ and $\mathsf{G}^n g \equiv \sim \mathsf{F}^n \sim g$.

[1] $[f \,\mathsf{U}^n g]_{\succ p}$ can be easily represented in PCTL*.

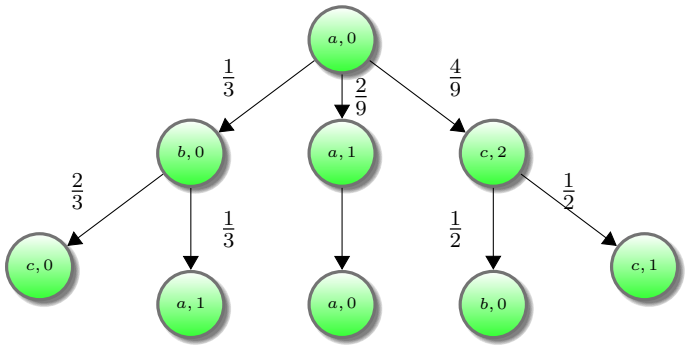


Figure 5.2.1: A finite-depth probabilistic computation tree.

Example 5.2.1. The sentence $[\mathsf{F}^3[\mathsf{G}^{10}\, b]_{=1}]_{>\frac{1}{3}}$ in bounded PCTL expresses that the probability to reach a state within three steps from which almost surely b holds for at least the next ten steps exceeds $\frac{1}{3}$.

Next we define the structure of canonical models of bounded PCTL. These notions are inspired by [76].

Definition 5.2.2. A probabilistic computation tree (PT) is a tuple $T = (W, P, L)$ where $W \neq \varnothing$, and:

- $W \subseteq \mathbb{N}^+$ is an unlabeled tree, i.e., prefix-closed,
- $P : W \to \mathcal{D}_W$, which is a partial function satisfying $P(\pi)(\pi') > 0$ iff $\pi' = \pi{\cdot}n \in W$ for some $n \in \mathbb{N}$. Furthermore, $\sum_n P(\pi, \pi{\cdot}n) = 1$.
- $L : W \to 2^{\mathsf{AP}}$ is a node labeling function.

The node $\pi = 0$ is called the *root*, while all nodes π such that $P(\pi)$ is undefined are referred to as the leaves. A PT T has a finite depth if there exists a $n \in \mathbb{N}$ such that for all $\pi \in W$, $|\pi| \leq n$, and T is total if for every $\pi \in W$ there exists a $\pi' \neq \pi \in W$, such that $\pi \prec \pi'$ (π is a proper prefix of π'). Let $\mathbb{T}^*$ and $\mathbb{T}^\omega$ denote the sets of all finite depth and total trees, respectively. $\mathbb{T}^\infty = \mathbb{T}^\omega \cup \mathbb{T}^*$.

Example 5.2.2. Consider a PT $T = (W, P, L)$, shown in Figure 5.2.1. The set of nodes of the tree $W = \{0, 00, 01, 02, 000, 001, 010, 020, 021\}$, the labeling function L and probability distribution is defined as shown in the figure.

Topologically a property $P \subseteq T^\omega$ is a set of total PTs.

Observation 1. A labeled Markov chain M satisfies a property P means that the PT created from the unrolling of M belongs to P.

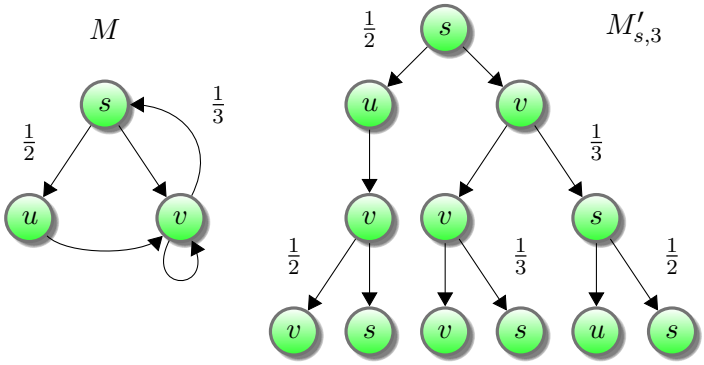


Figure 5.2.2: Markov chain M can be unfolded to probabilistic tree $M'_{s,3}$.

For a MC M and a state s, $M_{s,n}$ is a finite probabilistic tree that is obtained from M by unfolding M starting from s, such that the length of any path from s to a leaf is of length less than n (Figure 5.2.2). The satisfiability of bounded PCTL over finite trees obey the monotonicity property, i.e., $M_{s,n} \models f$ implies $M_{s,m} \models f$ for every $m \geqslant n$ (can be easily proved by structural induction). For bounded PCTL-formula f, let $\mathsf{ord}(f)$ be recursively defined as follows:

$$
\begin{array}{lcl}
\mathsf{ord}(a) & = & 1 \quad \text{for } a \in \mathsf{AP} \\
\mathsf{ord}(f_1 \wedge f_2) & = & \max\{\mathsf{ord}(f_1), \mathsf{ord}(f_2)\} \\
\mathsf{ord}(\sim f) & = & \mathsf{ord}(f) \\
\mathsf{ord}([f_1 \,\mathsf{U}^{\,n}\, f_2]_{\succ p}) & = & n + \max\{\mathsf{ord}(f_1), \mathsf{ord}(f_2)\} \\
\mathsf{ord}([\mathsf{X} f]_{\succ p}) & = & 1 + \mathsf{ord}(f)
\end{array}
$$

The function $\mathsf{ord}(f)$ give us a handle on the depth of the tree models of bounded PCTL formula f. This is proved by the following proposition.

Proposition 5.2.1. *For every bounded* PCTL *formula f and MC M: $M, s \models f$ implies $M_{s,n}, s \models f$ with $n = \mathsf{ord}(f)$.*

Proof. If $M_{s,n}, s \models f$ with $n = \mathsf{ord}(f)$ then obviously $M, s \models f$. For the other direction, we proceed by induction on the structure of the sentence f. Assume $M, s \models f$, we will see that $M_{s,n}, s \models f$.

1. $f := a$ Then, $n = \mathsf{ord}(f) = 1$. By definition, $M_{s,n}$ consists of a single node s equipped with a self-loop. If $M, s \models f$ then $a \in L(s)$. Hence, $M_{s,n}, s \models f$.

2. $f := f_1 \wedge f_2$. Let $n = \max\{\mathsf{ord}(f_1), \mathsf{ord}(f_2)\}$. By induction hypothesis and monotonicity, it follows $M_{s,n}, s \models f_1$ and $M_{s,n}, s \models f_2$. Thus, $M_{s,n}, s \models f$.

3. $f := {\sim}g$. For $f = {\sim}a$ the argument is similar to case 1. If $M, s \vDash {\sim}g$ then $M, s \nvDash g$, which implies $M_{s,n}, s \nvDash g$.

4. $f := [\mathsf{X}g]_{\succ p}$. Let $M, s \vDash f$, $S' = \{t \mid M, t \vDash g \text{ and } P(s,t) > 0\}$ and $m = \mathsf{ord}(g)$. By induction hypothesis, $M_{t,m}, t \vDash g$ for every $t \in S'$. By construction, $M_{t,m}$ is a subtree of $M_{s,m+1}$ for every $t \in S'$ and $\sum_{t \in S'} P(s,t) \succ p$. Thus, $M_{s,m+1}, s \vDash f$.

5. $f := [g \,\mathsf{U}^n h]_{\succ p}$. Suppose $M, s \vDash f$, $n_1 = \mathsf{ord}(g)$ and $n_2 = \mathsf{ord}(h)$. If $M, s \vDash h$ then $1 \succ p$ and the statement follows from the induction hypothesis. Assume $M, s \nvDash h$. Consider a path w starting in s with $w \vDash g \,\mathsf{U}^n h$. Thus, there exists a $0 < i \leq n$ such that $M, w_i \vDash h$ and for every $j < i$, $M, w_j \vDash g$. By induction hypothesis, $M_{w_i,n_2}, w_i \vDash h$ and for any predecessor w_j $M_{w_j,n_1}, w_j \vDash g$. Or, $M_{w_{i-1},m'}, w_{i-1} \vDash g$ and $M_{w_{i-1},m'}, w_i \vDash h$, where $m' = \max\{\mathsf{ord}(g)+1, \mathsf{ord}(h)\}$. For $m = m' + n$, $M_{w_{i-1},m'}$ is a sub-tree of $M_{s,m}$, therefore $M_{s,m}, w_{i-1} \vDash g$ and $M_{s,m}, w_i \vDash h$. This is true for any path w from s, that satisfies $g \,\mathsf{U}^n h$. Thus, $M_{s,m}, s \vDash f$.

This concludes the proof. □

The set of *sub-formulas* of bounded PCTL-formula f is denoted by $\mathsf{sub}(f)$.

$$
\begin{array}{lcl}
\mathsf{sub}(a) & = & 1 \text{ for } a \in \mathsf{AP} \\
\mathsf{sub}(f_1 \wedge f_2) & = & \{f_1 \wedge f_2\} \cup \mathsf{sub}(f_1) \cup \mathsf{sub}(f_2) \\
\mathsf{sub}({\sim}f) & = & \{{\sim}f\} \cup \mathsf{sub}f \\
\mathsf{sub}([f_1 \,\mathsf{U}^n f_2]_{\succ p}) & = & \{[f_1 \,\mathsf{U}^n f_2]_{\succ p}\} \cup \mathsf{sub}(f_1) \cup \mathsf{sub}(f_2) \\
\mathsf{sub}([\mathsf{X}f]_{\succ p}) & = & \{[\mathsf{X}f]_{\succ p}\} \cup \mathsf{sub}(f)
\end{array}
$$

Let $\mathsf{sub}_{\mathsf{path}}(f) = \{g \,\mathsf{U}^k h, \mathsf{X}g \mid [g \,\mathsf{U}^n h]_{\succ p}, [\mathsf{X}g]_{\succ p} \in \mathsf{sub}(f), 0 \leq k \leq n\}$. These definitions are lifted to sets of formulas in the usual way, i.e., $\mathsf{sub}(H) = \bigcup_{f \in H} \mathsf{sub}(f)$ and $\mathsf{sub}_{\mathsf{path}}(H) = \bigcup_{f \in H} \mathsf{sub}_{\mathsf{path}}(f)$. We will now prove that bounded PCTL-formulas can be satisfied by MCs of bounded width. A similar result has been obtained in [13], though the argument here is simpler on the account that we are dealing with bounded until. First we appeal to an elementary result from linear algebra.

Proposition 5.2.2 (Dual of Helly's theorem)**.** *Let T be a countable set of vectors in an n-dimensional space ($\mathbb{R}^n$). If a vector $\vec{v}$ is a convex combination of vectors from T, then there exists a set $T' \subseteq T$ such that $\vec{v}$ is a convex combination of vectors from T' and $|T'| \leq n{+}1$.*

Proof. The vector $\vec{v}$ is inside the convex polytope defined by T. A *triangulation* of a polytope is a partitioning of the space inside the convex polytope

using $(n{+}1)$-simplexes (tetrahedrons) in n-dimensions. Such a triangulation always exists even if the convex polytope is generated by a countable set of points. Thus, $\vec{v}$ is inside (or on) some $n+1$-simplex whose vertices are in $T' \subseteq T$. Thus, $\vec{v}$ can also be defined as a convex combination of vectors in T'. □

Proposition 5.2.3. *Every satisfiable bounded PCTL-sentence f has a tree model with bounded out-degree at most $|\mathsf{sub}(f)|{+}1$.*

Proof. Let M be a model of f. As the statement trivially holds for propositional formulas, we focus on path sentences. Consider state s in M and let $H = \{g \in \mathsf{sub}(f) : s \vDash g\}$. Assume w.l.o.g. that no two sub-sentences are syntactically identical. Let $\{1, \cdots, n\}$ be an enumeration of H, i.e., each formula in $\mathsf{sub}(f)$ is assigned a unique index. Assume s has more than $n{+}1$ descendants, i.e., $\mathsf{succ}(s) = \{t_1, \cdots, t_k\}$ for $k > n{+}1$. Let for path sentence g, $\Pr(s \vDash g)$ abbreviate $\Pr\{w \in \Omega_s : w \vDash g\}$. Define the vectors $\{\vec{s}, \vec{t}_1, \cdots, \vec{t}_k\}$ in the Euclidean space $[0,1]^n$ as follows:

1. for $[\mathsf{X}\, g]_{\succ p}$ with index i, $\vec{s}(i) = p$ where $\Pr(s \vDash \mathsf{X}\, g) = p$, and $t(i) = 1$ if $t \vDash g$ else $t(i) = 0$, for each $t \in \mathsf{succ}(s)$.

2. for $[f_1\, \mathsf{U}^k f_2]_{\succ p}$ with index i and $s \nvDash f_2$, $\vec{s}(i) = p$ where $\Pr(s \vDash f_1\, \mathsf{U}^k f_2) = p$, and $t(i) = q$ with $q = \Pr(t \vDash f_1\, \mathsf{U}^{k-1} f_2)$, for each $t \in \mathsf{succ}(s)$.

3. for any other index i, $\vec{s}(i) = \vec{t}(i) = 0$.

For the semantics of a path sentence of the form $[g]_{\succ p}$, we obtain the following relation:

$$\vec{s} \;=\; \sum_{t \in \mathsf{succ}(s)} P(s,t) \cdot \vec{t}.$$

That is, $\vec{s}$ is a linear combination of the vectors $\{\vec{t}_1, \cdots, \vec{t}_k\}$. By Proposition 5.2.2 (see page 81), there exists a set $G \subseteq \mathsf{succ}(s)$ with $|G| \leqslant n{+}1$ and a distribution $P'(s)$ such that:

$$\vec{s} \;=\; \sum_{t \in G} P'(s,t) \cdot \vec{t}.$$

It is easy to see that using G as set of direct successors (rather than $\mathsf{succ}(s)$) s still satisfies H. Applying this procedure to every state of M yields a model with out-degree at most $n{+}1$. □

Form Propositions 5.2.1 and 5.2.3, we obtain the small model theorem of bounded PCTL.

Theorem 5.2.4. *If a bounded* PCTL *formula f is satisfiable then it is satisfiable by a finite probabilistic tree of depth* $\mathsf{ord}(f)$ *and degree* $|\mathsf{sub}(f)|+1$.

The *size* of a bounded PCTL formula f is defined as $\mathsf{size}(f) = |\mathsf{ord}(f)| + |\mathsf{sub}(f)|$. Note that the small model theorem states that every satisfiable sentence f, there exists a probabilistic tree model of f whose number of nodes is exponential in $\mathsf{size}(f)$ (but not the space needed to encode f).

5.3 Complexity of satisfiability problem for bounded PCTL

In this section we give a hierarchical complexity analysis for various fragments of bounded PCTL.

5.3.1 Complexity of Px_ω satisfiability

We will now show that the satisfiability problem for bounded PCTL without the bounded *until* is PSPACE-complete. We distinguish the following sublogics. Let Px_0 be the set of formula defined by the syntax:

$$f ::= \ a \mid f \wedge f \mid {\sim}f$$

where $a \in \mathsf{AP}$. Px_0 is identical to Propositional logic. The logic Px_i is defined inductively as follows:

$$f ::= \ a \mid f \wedge f \mid {\sim}f \mid g \mid [\mathsf{X}g]_{\succ p}$$

where $g \in \mathsf{Px}_{i-1}$, $\succ \in \{>, \geq\}$ and $p \in [0,1]$. With little abuse of notation, we will denote the set of all formulas of the logic Px_i, by the set Px_i. Thus, Px_ω is the set of formulas with an unbounded number of nested *next* operators. Px_ω coincides with bounded PCTL without bounded until.

We will show PSPACE-hard of the satisfiability problem of Px_ω. The reduction follows the exact line of reasoning for proving PSPACE-hardness for K-systems in modal logics. The hardness proof uses only the operator $[\mathsf{X}g]_{=1}$. The semantics of $[\mathsf{X}g]_{=1}$ is then similar to the $\Box$ operator of modal logic K [83][2]. Henceforth, we will use $\Box g$ to denote $[\mathsf{X}g]_{=1}$ and $\Diamond g$ to denote ${\sim}[\mathsf{X}{\sim}g]_{=1}$ (which is equivalent to $[\mathsf{X}g]_{>0}$). We can now directly use the results of K-logic. In the sequel we present the construction.

Proposition 5.3.1. *The satisfiability problem for Px_ω is PSPACE-hard.*

[2]The more appropriate modal logic system would be with K and *serial* axioms.

Proof. The main idea behind the reduction ([70]) is to give a logspace transducer to convert every instance of a QBF to a formula in Px_ω. Let f be a QBF $Q_1x_1 \cdots Q_mx_m\varphi(x_1, \cdots, x_m)$, where $Q_i \in \{\exists, \forall\}$, x_i is a boolean variable $(1 \leq i \leq m)$ and $\varphi(x_1, \cdots, x_m)$ is a quantifier free boolean formula with variables $x_1, \cdots, x_m$.

We will use new propositions $y_0, \cdots, y_m$ to uniquely encode the index $0 \leq i \leq m$. For that purpose, let $z_1, \cdots, z_n$, where $n = \lceil \log m \rceil$ be new propositions such that $y_i \equiv \beta_{i,1}z_1 \wedge \cdots \wedge \beta_{i,n}z_n$ for $0 \leq i \leq m$, where $\beta_{i,j} = \sim$ if the j^{th} bit of (binary) i is zero else $\beta_{i,j}$ is a empty string $(1 \leq j \leq n)$. Let g_1 represent the conjunction of all such equivalences. Next we define the Px_ω formula g which uses propositions $x_1, \cdots, x_m, y_0, \cdots, y_m, z_1, \cdots, z_n$. The formula g is a conjunction of the following formulas:

$$\Box^m g_1 \quad (F1)$$

$$y_0 \quad (F2)$$

$$\Box^m(y_i \rightarrow \Diamond y_{i+1}) \text{ for each } 0 \leq i < m \quad (F3)$$

$$\Box^m(y_i \rightarrow ((x_i \rightarrow \Box^{m-i}x_{i+1}) \wedge (\sim x_i \rightarrow \Box^{m-i}\sim x_{i+1}))) \text{ for each } 0 < i < m \quad (F4)$$

$$\Box^m(y_i \rightarrow (\Diamond(y_{i+1} \wedge x_{i+1}) \wedge (\Diamond(y_{i+1} \wedge \sim x_{i+1})))) \text{ if } Q_i = \forall,\ 0 \leq i < m \quad (F5)$$

$$\Box^m(y_m \rightarrow \varphi) \quad (F6)$$

where $\Box^m h = h \wedge \Box(\Box^{m-1}h)$ and $\Box^0 h = h$. Intuitively, $\Box^m h$ is true at s if h is true at every state reachable from s within m steps. The idea behind the reduction is that any model of g *simulates* the formula f. Suppose s satisfies g, the variable y_i marks the states of the tree (rooted at s) at depth i, (implemented by (F1), (F2) and (F3)). If the i^{th} quantifier is universal, then (F5) guarantees that there are two descendants, one of which makes x_i true and the other makes $\sim x_i$ true. Once, x_i (or $\sim x_i$) is chosen at a branch, it remains unaltered for every descendant, this is guaranteed by (F4). Finally, we want to evaluate the quantifier free boolean formula φ. This is implemented by (F6).

To see that only logspace is sufficient to produce the output g, observe that at each step we need to be able to count the index i $(0 \leq i \leq m)$, which can be stored in logspace of the working tape, and write the corresponding string (the formula as defined by (F1), (F2), (F3), (F4), (F5) and (F6)) in the output tape. □

Next we show that the satisfiability problem can be decided in PSPACE.

Proposition 5.3.2. *The satisfiability problem for Px_ω is in PSPACE.*

Proof. We show that the satisfiability problem for sentences in Px_n is in Σ_n^P of the polynomial-time hierarchy. Let T_n be a non-deterministic Turing

machine (NTM) with an oracle Ω_{n-1}. Oracle Ω_n can foretell whether a set of sentences in Px_n is satisfiable[3]. W.l.o.g. we assume that Px_n sentences are in negated normal form. Let H be the set of Px_n sentences that the input to NTM T_n. T_n proceeds as follows:

1. If $f = f_1 \wedge f_2 \in H$, then remove f from H and add f_1 and f_2 to H.

2. If $f = \sim(f_1 \wedge f_2) \in H$, then remove f from H and non-deterministically choose $i \in \{1, 2\}$ and add $\sim f_i$ to H.

3. If $f = \sim[\mathsf{X}g]_{\succ p}$ and $f \in H$, then remove f from H and add $[\mathsf{X}g]_{\succcurlyeq p}$ to H.

This takes linear time in the size of the input set H. In the end, H only contains atomic propositions a, negated atomic propositions $\sim a$ or formulas with *next* operator, $[\mathsf{X}\psi]_{\succ p} \in \mathsf{Px}_i$. The machine T_i executes the following steps:

1. $H \cap \mathsf{Px}_0$ is unsatisfiable then T_i *rejects*.

2. Otherwise, T_n selects a weighted cover (see Def. 2.2.1) (c, w) of $\{g : [\mathsf{X}\, g]_{\succ p} \in H\}$ with

   (a) $w(g) \succ p$ for each $[\mathsf{X}\, g]_{\succ p} \in H$, and

   (b) $\bigwedge_{g \in G} g \not\equiv \text{false}$ for each $G \in c$.

By Proposition 5.2.3, we restrict to covers whose widths are at most $|H|+1$. Checking (1) can be done by solving linear equations, and (2) satisfiability of formulas for each $G \in c$ is delegated to the oracle Ω_{n-1}. This is possible since the set G only contains sentences in Px_{n-1}. The NTM T_n accepts if such a weighted cover exists, else it rejects.

The correctness of the above algorithm is straightforward. The algorithm generates a model (and accepts H) iff H is satisfiable. We omit the details. The satisfiability of a set of Px_n sentences can thus be solved by an NTM with an oracle Ω_{n-1} in polynomial time. Hence, the satisfiability problem for Px_ω is in $\mathsf{NP}^{\mathsf{NP}^{\mathsf{NP}^{\cdots}}}$, and hence in PSPACE. □

From propositions 5.3.1 and 5.3.2, we get the following theorem.

Theorem 5.3.3. *The satisfiability for Px_ω is PSPACE-complete.*

Algorithm 2 closure(H)

1: **for each** $f \in H$ **do**
2: **if** $f = a \in AP$ or $f = \sim a$ **then** skip
3: **if** $f = f_1 \wedge f_2$ **then** $H := (H \setminus \{f\}) \cup \{f_1, f_2\}$
4: **if** $f = f_1 \vee f_2$ **then**
5: $H := (H \setminus \{f\}) \cup \{f_i\}$, where $i = 1$ or 2
6: **if** $f = [\mathsf{X} g]_{\sim p}$ **then** skip
7: **if** $f = [f_1 \,\mathsf{U}^0 f_2]_{\sim p}$ **then** $H := (H \cup \{f_2\}) \setminus \{f\}$
8: **if** $f = [f_1 \,\mathsf{U}^n f_2]_{\sim p}$ and $n > 0$ **then**
9: either $H := (H \cup \{f_2\}) \setminus \{f\}$ or $H := H \cup \{f_1\}$
10: **end for**
11: **return** H

5.3.2 Complexity of bounded PCTL satisfiability

Now we consider the full bounded PCTL logic (with bounded until). We will need the following machinery to solve the satisfiability problem.

Proposition 5.3.4. *Given a finite tree T and a bounded* PCTL *formula f, we can decide in NP-time whether there exists a probabilistic tree M satisfying f, with T as the underlying graph.*

Proof. Let tree $T = (V, E, s_0)$, where V is a set of vertices, $E \subseteq V \times V$ a set of directed edges, and s_0 is the root. Every edge $e \in E$ is assigned a variable x_e denoting the weight of e. Let $\mathcal{P} = \{x_e : e \in E\}$ be the set of weights in T. To construct an MC M with T as underlying graph, we non-deterministically select a labelling function L using Alg. 3. Function L labels every vertex in T with a set of bounded PCTL-formulas. This goes as follows. We initialize $L(s_{in})$ to $\{f\}$, and invoke *label*(s_{in}) (see Alg. 3.). Line 2 covers the case when s is only labeled with propositional formulas. If s is labeled with a non-propositional formula, its labelling is adapted to the *Hintikka set* of $L(s)$ (line 3). The computation of the Hintikka set is done using Alg. 2. This procedure is non-deterministic (see lines 5 and 9). After labelling s, a non-deterministic selection of its direct successors is labeled in the for-loop (line 5–22). During this loop, a set H (initially empty) of multi-variate polynomial inequations is computed over the variables x_e and newly introduced variables p_t for vertex t (line 16–17). Each vertex of T is visited twice: once to calculate the polynomial inequations and once in the recursive call. Thus, the labelling algorithm is in NP.

[3] See [67] for background information on oracle Turing machines and the polynomial time-hierarchy.

Algorithm 3 label(s)

1: **if** $L(s) \subseteq \mathsf{Px}_0$ **then**
2: **return** *true* iff $L(s)$ is satisfiable
3: **else** $L(s) = \text{closure}(L(s))$
4: **end if**;
5: **for each** $f \in L(s)$ **do**
6: **if** $f = [\mathsf{X}\, g]_{\succ p}$ **then**
7: choose non-deterministically $S' \subseteq \mathsf{succ}(s)$
8: **for each** $t \in S'$ **do**
9: $L(t) := L(t) \cup \{g\}$
10: $H := H \cup \left(\sum_{t \in S'} x_{(s,t)} \succ p\right)$
11: *extend H with constraint for s and S'*
12: **end for**
13: **elseif** $f = [f_1\, \mathsf{U}^n f_2]_{\succ p}$
14: choose non-deterministically $S' \subseteq \mathsf{succ}(s)$;
15: **for each** $t \in S'$ **do**
16: $(L(t) := L(t) \cup \{[f_1\, \mathsf{U}^{n-1} f_2]_{=p_t}\})$
17: or $(L(t) := L(t) \cup \{f_2\}$ and $p_t = 1)$
18: *where p_t is a new variable*
19: $H := H \cup \left(\sum_{t \in S'} x_{(s,t)} \cdot p_t \succ p\right)$
20: **end for**
21: **end if**
22: **end for**
23: **for each** $t \in \mathsf{succ}(s)$ **do** label(t) **od**

Formula f holds in s_{in} iff the set of (real non-linear) inequations H, with variables in $\mathcal{P}$ is satisfiable. The number of inequations is in $O(|V| \cdot |\mathsf{sub}(f)|)$ and the number of variables is $|E|$, i.e., polynomial in the size of the input. Using the *existential theory of the reals* [17], we can determine the feasibility of the inequations in PSPACE. This complexity can be improved by exploiting the special structure of the inequations. Observe that after some simplification (and removal of new variables introduced in lines 16–17 of Alg. 3 [4]) every equation has the following form: $a_0 \cdot \sigma_0 + a_1 \cdot \sigma_1 + \cdots + a_k \cdot \sigma_k \succ b$, where $a_0, \cdots, a_k, b \in \mathbb{Q}$ and σ_i $(0 \le i \le k)$ is a term of a polynomial of the type $x_{e_{1,i}} x_{e_{2,i}} \cdots x_{e_{n,i}}$ where $e_{1,i} e_{2,i} \cdots e_{n,i}$ is a path in the tree T. Furthermore, the edges $e_{1,i}$ for every $0 \le i \le k$ (Figure 5.3.3) have the same source vertex. In Appendix C we show how to solve the satisfiability problem of such a system

[4] Note that the variable p_t is the lvalue of a single equation of the form $p_t = \cdots$. Thus p_t can be easily substituted.

of (in)equations in NP. □

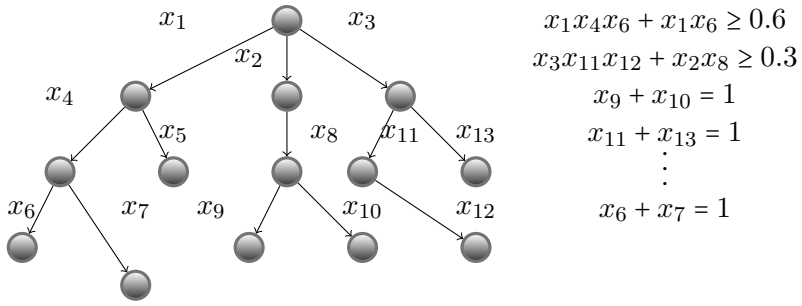


Figure 5.3.3: A typical example of set of equations generated by algorithm 3.

The complexity for the satisfiability problem for bounded PCTL is now straightforward.

Proposition 5.3.5. *The satisfiability problem for bounded* PCTL *is in NEXPTIME in the* size *of the formula.*

Proof. Let f be a bounded PCTL-formula. Theorem 5.2.4 and Proposition 5.3.4 suggest the following algorithm to solve the satisfiability problem. We non-deterministically guess a tree T of size $2^{O(\mathsf{size}(f))}$. Then we check whether there is an MC with the underlying graph T that satisfies f. The algorithm works in $\mathsf{NTIME}(2^{O(\mathsf{size}(f)^2)}) \subseteq \mathsf{NEXPTIME}$ in the size of the formula. □

Proposition 5.3.6. *The satisfiability of bounded* PCTL *formula is EXPTIME-hard in the encoding of the formula.*

Proof. The details of the reduction from the acceptance problem for an alternating polynomial space Turing machine to the satisfiability of bounded PCTL formula can be found in the appendix. □

5.4 Related discussion

We have seen that satisfiability problem for bounded PCTL decidable. Two important observations that make bounded PCTL satisfiable are, first the models are bounded and second the bound on the size of the model is not (intricately) related to the probabilities present in the sentence. Can we find other fragments of PCTL that share this property? In this section we will discuss other fragments of PCTL that are studied in literature. From here on we will consider PTs as models of a logic.

5.4.1 Safety and co-Safety

Alpern and Schneider [1], defined safety and liveness for infinite words. The idea can be carried to define safety for PTs [63]. First, we need to extend the definition of prefix to probabilistic trees.

Definition 5.4.1. Let $T_i = (W_i, L_i, P_i)$ for $i = 1, 2$, and $T_1 \in \mathbb{T}^*$ and $T_2 \in \mathbb{T}^\infty$. T_1 is a prefix of T_2, denoted as $T_1 \preceq T_2$, iff:

$W_1 \subseteq W_2$ and $\forall \pi \in W_1, L_1(\pi) = L_2(\pi)$, and $\forall \pi, \pi' \in W_1, P_1(\pi, \pi') = P_2(\pi, \pi')$.

Let $Pre_{fin}(T) = \{T' : T' \preceq T\}$ denote the set of prefixes of T.

Definition 5.4.2. Let $T_i = (W_i, L_i, P_i)$ with $T_i \in \mathbb{T}^\infty$, $i = 1, 2$ is a suffix of T_1 iff there exists $\pi_1 \in W_1$ such that

- $\{\pi_1 \cdot \pi_2 : \pi_2 \in W_2\} \subseteq W_1$,
- $L_2(\pi_2) = L_1(\pi_1 \cdot \pi_2)$ for each $\pi_2 \in W_2$,
- $P_2(\pi_2, \pi_2') = P_1(\pi_1 \cdot \pi_2, \pi_1 \cdot \pi_2')$ for any $\pi_2, \pi_2' \in W_2$.

Definition 5.4.3. $P \subseteq \mathbb{T}^\omega$ is a safety property iff for all $T \in P$ every $T_1 \in Pre_{fin}(T)$ there exists a $T_2 \in P$ such that $T_1 \preceq T_2$.

Thus a safety property P only consists of PTs for which any finite depth prefix can be extended to a PT in P. Contrapositively, if $T \notin P$ then there is a finite depth prefix of T that cannot be extended to PTs in P. This finite prefix is colloquially known as the "bad prefix". Dual of safety is co-safety.

Definition 5.4.4. P is a co-safety property iff $\mathbb{T}^\omega \setminus P$ is a safety property.

The co-safety properties are of special interest to us since they have finite model property by construction.

Observation 2. If P is a co-safety property then for any $T \in P$ there exists a finite prefix $T_1 \preceq T$, such that for all $T_2 \in \mathbb{T}^\omega$ $T_1 \cdot T_2 \in P$.

Another important class of properties are called liveness.

Definition 5.4.5. $P \subseteq \mathbb{T}^\omega$ is a liveness property iff $\forall T \in \mathbb{T}^*$, there exists $T_2 \in P : T_1 \preceq T_2$.

Example 5.4.1. Consider the following PCTL formulas.

1. $f := [a \,\mathsf{U}\, b]_{\leq \frac{1}{3}}$ is a safety property. Thus, $f' := [a \,\mathsf{U}\, b]_{> \frac{1}{3}}$ is a co-safety property. Since, for any PT that satisfies f', the probabilistic measure of the set of paths satisfying $a \,\mathsf{U}\, b$ must cross $\frac{1}{3}$ after certain depth.

2. $f := [a \mathsf{U} b]_{\geq \frac{1}{3}}$ is neither safe nor co-safe. It is also not live. Consider a finite-depth PT whose every path satisfies $a \mathsf{U} b$ or $\mathsf{F} \sim a$, and the probability measure of paths satisfying $a \mathsf{U} b$ is strictly less than $\frac{1}{3}$. Then no infinite extension of such a tree can satisfy $[a \mathsf{U} b]_{\geq \frac{1}{3}}$.

3. Most importantly (and vacuously) false is a safety as well as co-safety property. Else satisfiability problem for these fragments would be redundant.

The semantics of PCTL(and its fragments) are defined on Markov chains, that is total PTs. We can extend the definition of satisfiability to finite PTs as follows:

Definition 5.4.6. A formula f is satisfiable by a finite depth tree T iff for any $T' \in \mathbb{T}^\omega$ with $T \preceq T'$, T' satisfies f.

5.4.2 Safety and co-Safety PCTL

Now we will define the safety fragment of PCTL.

Definition 5.4.7. Let $\mathsf{PCTL}_{\text{safe}}$ denote the safe fragment of PCTL, defined as the smallest set satisfying:

1. a or $\sim a$ $\mathsf{PCTL}_{\text{safe}}$.
2. If f in $\mathsf{PCTL}_{\text{safe}}$, then $[\mathsf{X} f]_{\geq p}$ in $\mathsf{PCTL}_{\text{safe}}$.
3. If f, g in $\mathsf{PCTL}_{\text{safe}}$, then $f \wedge g$, $f \vee g$, $[f \mathsf{W} g]_{\geq p}$ in $\mathsf{PCTL}_{\text{safe}}$.
4. If $\sim f, \sim g \in \mathsf{PCTL}_{\text{safe}}$, then $[f \mathsf{U} g]_{\leq p}$ in $\mathsf{PCTL}_{\text{safe}}$.

We will also consider the co-safety fragment.

Definition 5.4.8. Let $\mathsf{PCTL}_{\text{co-safe}}$denote the co-safe fragment of PCTL, defined as the smallest set satisfying:

1. a or $\sim a$ in $\mathsf{PCTL}_{\text{co-safe}}$.
2. If f $\mathsf{PCTL}_{\text{co-safe}}$, then $[\mathsf{X} f]_{\geq p}$ in $\mathsf{PCTL}_{\text{co-safe}}$.
3. If f, g in $\mathsf{PCTL}_{\text{co-safe}}$, then $f \wedge g$, $f \vee g$, $[f \mathsf{U} g]_{> p}$ in $\mathsf{PCTL}_{\text{co-safe}}$.
4. If $\sim f, \sim g$ in $\mathsf{PCTL}_{\text{co-safe}}$, then $[f \mathsf{W} g]_{< p}$ in $\mathsf{PCTL}_{\text{co-safe}}$.

If a Markov chain M satisfies a $\mathsf{PCTL}_{\text{co-safe}}$formula f, then there exists a finite prefix T of the unrolling of M such that for all $T' \in \mathbb{T}^\omega$ with $T \preceq T'$, T' also satisfies f. We can now relate Px_ω and bounded PCTLwith safety and co-safety properties. Recall:

Observation 3. (Finite tree property) Let f be a Px_ω formula and g be a bounded PCTLformula.

- A Markov chain M satisfies f if and only if there is a finite depth prefix T of the unrolling of M, which satisfies f. The depth of T is at the most $|f|$.
- A Markov chain M satisfies g if and only if there is a finite depth prefix T of the unrolling of M, which satisfies g. The depth of T is at the most $\mathsf{size}(g)$.

It has the following consequences.

1. Px_ω and bounded PCTL are both safety and co-safety property.
2. To check a formula is satisfiable or *NOT*, it suffices only to look at a finite depth PT.
3. And most importantly, the depth is linear in the size of the input formula. That is, it does not depend on the exact values of the probability bounds.

With these properties in mind we could consider co-safe PCTL restricted to F and G operators (reciprocally, safe-PCTL restricted to F and G).

Definition 5.4.9. Let $\mathsf{PCTL}_{\text{co-safe}}(\mathsf{F},\mathsf{G})$ denote the co-safe fragment of PCTL, defined as the smallest set satisfying:

1. a or $\sim a$ in $\mathsf{PCTL}_{\text{co-safe}}(\mathsf{F},\mathsf{G})$.
2. If f $\mathsf{PCTL}_{\text{co-safe}}(\mathsf{F},\mathsf{G})$, then $[\mathsf{X}f]_{\geq p}$ in $\mathsf{PCTL}_{\text{co-safe}}(\mathsf{F},\mathsf{G})$.
3. If f, g in $\mathsf{PCTL}_{\text{co-safe}}(\mathsf{F},\mathsf{G})$, then $f \wedge g$, $f \vee g$, $[\mathsf{F}f]_{>p}$ in $\mathsf{PCTL}_{\text{co-safe}}(\mathsf{F},\mathsf{G})$
4. If $\sim f$ in $\mathsf{PCTL}_{\text{co-safe}}(\mathsf{F},\mathsf{G})$, then $[\mathsf{G}f]_{<p}$ in $\mathsf{PCTL}_{\text{co-safe}}(\mathsf{F},\mathsf{G})$.

Proposition 5.4.1. *$f \in PCTL_{co\text{-}safe}(\mathsf{F},\mathsf{G})$ is satisfiable iff and only there exists a finite-depth PT of depth $|f|$ that satisfies f.*

Proof. First note that it only suffices to consider formulas $a, \sim a, [\mathsf{X}g]_{\geq p}$ and $[\mathsf{F}g]_{>p}$. Next observe that following: Consider two formulas $[\mathsf{F}g_1]_{>p_1}$ and $[\mathsf{F}g_2]_{>p_2}$ such that g_1 and g_2 are satisfiable by a finite depth probabilistic trees T_1 and T_2, respectively. Now we construct a finite-depth probabilistic tree T in the following way. Select leaves of T_1 such that the probability of reaching these leaves is $> p_2$ and make as many copies of T_2 as the number of leaves selected. From each of these selected leaves of T_1, add an edge to the

roots of a copy of T_2. It is easy to conclude that, if g_1 and g_2 are satisfiable then $[\mathsf{F}g_1]_{>p_1} \wedge [\mathsf{F}g_2]_{>p_2}$ is also satisfiable. In the sequel we generalize this observation.

We proceed in an inductive manner. But we must observe caution, it may be the case that for a formula of the type $f_1 \wedge f_2$, f_1 and f_2 are individually satisfiable, but their conjunction is not. For example consider $[\mathsf{X}a]_{\geq 0.6}$ and $[\mathsf{X}\neg a]_{\geq 0.6}$. Both of these formulas are satisfiable but their conjunction is unsatisfiable. Thus we will follow induction on the nesting depth of the formula. For example, $[\mathsf{X}[\mathsf{F}b]_{\geq q}]_{\geq p} \wedge [\mathsf{F}a]_{\geq r}$ has a nesting depth 2, whereas $[\mathsf{X}a]_{\geq p} \wedge [\mathsf{F}a]_{\geq r} \wedge [\mathsf{X}c]_{\geq r}$ has a nesting depth 1.

Suppose all formulas of nesting depth less than n satisfy the proposition. We consider $f = g_1 \wedge \cdots \wedge g_n \wedge h_1 \wedge \cdots \wedge h_m$ where $g_i = [\mathsf{X}\varphi_i]_{\geq p_i}$, $h_j = [\mathsf{F}\psi_j]_{>q_j}$ and formulas φ_i, ψ_j have nesting depth at most $n-1$. If $H = \{g_1, \cdots, g_n\}$ is satisfiable, then a weighted cover $(C, w) \in 2^{2^H} \times \mathcal{D}_{2^H}$ must exist, such that for all $g_i \in H$, $\sum_{s \in C : g_i \in s} w(s) \geq p_i$, and for all $s \in C$, s is satisfiable. Formulas present in every $s \in C$ has formula depth strictly less than n, thus the induction hypothesis is applicable. Let T_s be the finite-depth probabilistic tree that satisfies s. Construct a finite PT T_H, whose root has an probabilistic transition to the root of each T_s of weight $w(s)$. This takes care of formulas in H.

Each ψ_j has formula depth less than n. By induction hypothesis ψ_j is satisfiable by a probabilistic tree T_j of depth $|\psi_j|$. Now we generalize the observation. We start with $T_f = T_H$ and add T_1 to the leaves of T_f such that set of path satisfying $\mathsf{F}\psi_1$ is $> q_1$. This will yield a new tree and we set T_f to it. Continuing this procedure of adding T_j to for each ψ_j to the current T_f, would yield the necessary PT for f.

The other cases of the induction are trivial. If for instance, we have disjunct $f_1 \vee f_2$, we can select either f_1 or f_2, and apply our induction. If $f = [\mathsf{X}g]_{\geq p}$ then T_g is the probabilistic tree satisfying g, then the required probabilistic tree of f has a root with a transition to the root of T_g with probability mass at least p. For $f = [\mathsf{F}g]_{\geq p}$, the probabilistic tree of f is same as that of g.

For the base case when $n = 0$, we have conjunction only atomic proposition or their negation. Thus, if a conjunction of atomic proposition (or their negation) is satisfiable then the depth of the satisfying PT is 1. □

Note that the above demonstration didn't use the fact that the comparison of the probability bounds are strict, thus the proof is also applicable for formulas with non-strict comparisons. We see that the finite tree property does not depend on it on the strictness of the comparison operator, even

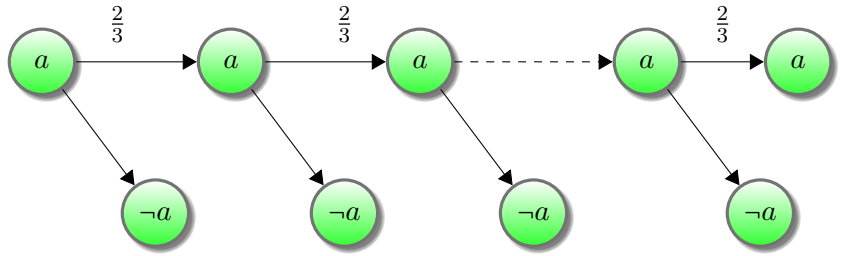


Figure 5.4.4: The model for the formula $f = [[\mathsf{X}a]_{\geq 2/3}\,\mathsf{U}\,\sim a]_{>p} \wedge a$. In this case the depth of the PT can be calculated a priori, which is $\frac{\log(1-p)}{\log(2/3)}$. But a mechanism for finding such formula for any co-safe formula is not known.

Logic	Finite tree Property	finite-depth	SAT
Px_ω	yes	linear	PSPACE
bounded PCTL	yes	linear	EXPTIME
$\mathsf{PCTL}_{\text{co-safe}}(\mathsf{F},\mathsf{G})$	yes	linear	PSPACE
$\mathsf{PCTL}_{\text{co-safe}}$	yes	?	r.e.

Table 5.1: Comparison of various fragments of PCTL with finite PT property.

though a formula is safe or co-safe depends on the strictness of the comparison (see the example in pg 89). The unifying feature of Px_ω, bounded PCTL, and $\mathsf{PCTL}_{\text{co-safe}}(\mathsf{F},\mathsf{G})$ is as follows:

Observation 4. The depth of the PT does not depend on the exact values of the probability bounds.

This is not the case for the general co-safety fragment of PCTL. There are formulas which have finite tree property but the depth of the finite-depth PT depends on the probability bounds.

Example 5.4.2. Let $f = [[\mathsf{X}a]_{\geq 2/3}\,\mathsf{U}\,\sim a]_{>p} \wedge a$. First observe that f is a co-safety property. That is for any value of $p \in [0,1)$ f is satisfiable by a finite depth PT. But the depth of the PT becomes larger as p tends towards 1 (Figure 5.4.4).

Thus, co-safe formulas even though have finite tree property, gives us only a semi-decidable fragment of PCTL. Table 5.1 summarizes various fragments.

5.5 Conclusion

We have presented the sub-logic bounded PCTL which possesses the small model property. This implies that the satisfiability problem for bounded PCTL is decidable and we have given an NEXPTIME algorithm in the *size* of the formula. We have also considered fragments $(\mathsf{Px}_0, \cdots, \mathsf{Px}_\omega)$ of bounded PCTL and shown the hierarchical complexity of their satisfiability problem.

We observe that if a bounded PCTL formula is satisfiable then it is satisfiable in an MC with *rational* transition probabilities (the variable elimination procedure works on rationals). Surprisingly, this statement does not hold for Bertrand *et al.* [8], bounded satisfiability (a priori fixing the number of states of a model). We have mentioned that the EXPTIME hardness in the *encoding* (space) of the formula, but our algorithm runs in NEXPTIME in the *size* of the formula. Reducing this gap is an open problem. The major bottle neck of the algorithm is the variable elimination method, which we *believe* is NP-hard.

We have also discussed on some of the reasons *why* these fragments have small model property and why the argument fails for other fragments of PCTL, namely co-safe fragment of PCTL. Interesting direction of research would be exploring other decidable fragments of PCTL and the inter-relation between them.

Chapter 6

Simple Probabilistic Extensions of μ-calculus

This chapter considers a modal μ-calculus extended with a probabilistic next-modality. After introducing the logic, we define the notions of rank and signature. We then show that satisfiable $\mathsf{P}\mu\mathsf{TL}$-sentences have a model of bounded out-degree. Finally, we provide a decision procedure for $\mathsf{P}\mu\mathsf{TL}$ satisfiability using parity games—in the same vein as for the modal μ-calculus—and yield a small model as well as a rational model property.

6.1 Introduction

In the previous chapter we have studied the satisfiability problem for the bounded fragments of PCTL, shown that it is decidable and discussed the algorithmic complexity of the problem. In this chapter we will follow suit for yet another probabilistic logic called $\mathsf{P}\mu\mathsf{TL}$. $\mathsf{P}\mu\mathsf{TL}$ is an extension of μ-calculus with probabilistic quantification over *next* formulas. The logic, called $\mathsf{P}\mu\mathsf{TL}$ was first studied by [74], were the model checking and satisfiability problem were discussed.

Like PCTL, the models of $\mathsf{P}\mu\mathsf{TL}$ are labeled Markov chains. Given a finite labeled Markov chain, one can easily reduce the model checking problem for a $\mathsf{P}\mu\mathsf{TL}$ formula to a 2-player parity game, such that the game is winning for Player 0 if the Markov chain satisfies a given $\mathsf{P}\mu\mathsf{TL}$ formula. The size of the resulting game is polynomial in the size of the formula and Markov chain. The existence of a winning strategy can be decided in PTIME. For more details, please refer to [18]. The topic of discussion of this chapter is however the satisfiability problem of $\mathsf{P}\mu\mathsf{TL}$. Even though the decidability issue of this problem was settled in [74], we will see some original contribution that will further our understanding on the subject.

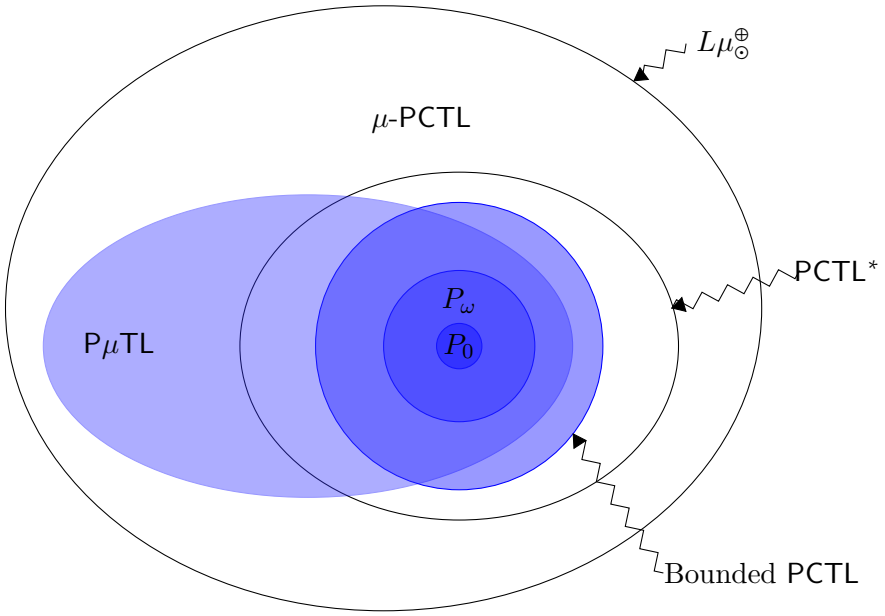


Figure 6.1.1: Expressiveness of different probabilistic logic. Logics with shades have finite model property.

In terms of expressive power, PμTL is orthogonal to PCTL. That is there are PCTL formulas which can not be represented as PμTL formula, and vice-versa. Naturally, PμTL is included in other probabilistic logic with recursion like, μ-PCTL [18], μ^p-calculus [18], and $L\mu^{\oplus}_{\odot}$ [80]. Figure 6.1.1 gives a pictorial view of relative expressiveness of various probabilistic logics.

We believe that PμTL is very closely related to modal μ-calculus, thus any decidability proof for PμTL, can be made to work for modal μ-calculus. In this chapter we will see that the reasoning paradigm [91] used for deciding modal μ-calculus is applicable to PμTL. In this chapter, we address the following important properties of PμTL were.

1. PμTL has small model property (Exponential Size). That is, a satisfiable sentence always has a model of exponential size.

2. PμTL has bounded branching (linear). That is, a satisfiable sentence always has a model with linear branching degree.

3. A satisfiable sentence always has a model where weight of the edges are rational. It is interesting note that in PμTL we can (asymptotically) bound the size of the model a priori, since we have small model property, and still we can find a model where transition probabilities are rational. This is contrary to PCTL [8], where it is shown that there

are PCTL formulas which are only satisfiable by models with irrational weight when we bound the size of the model (a priori). We don't know that such a property holds for PCTL in general.

4. Satisfiability is obtained by playing 2 player parity game, whose size is exponential in the *size* of the input formula.

5. Finally, satisfiability is EXPTIME-hard. This follows directly from EXPTIME-hardness of the satisfiability problem of modal μ-calculus, which in turn follows from PDL [44].

All the above listed properties are shared by modal μ-calculus (except 3. which is not applicable). We will conclude that PμTL cannot capture qualitative PCTL, and the probability bounds play no major role on the complexity of the decision procedure for PμTL. That is, the asymptotic size of the model is not affected by the exact values of the probability bounds. Recall, we have witnessed similar properties for Px_ω, bounded PCTL and $\mathsf{PCTL}_{\text{co-safe}}(\mathsf{F},\mathsf{G})$.

6.2 Preliminaries

Definition 6.2.1. Let H be a set of objects. A *cover* c is a set of sets of objects of H, such that $\bigcup_{e\in c} e = H$. The cardinality of c is the width of the cover c. A *weighted cover* of H is a cover c with a mapping $w : c \to (0,1]$, such that $\sum_{e\in c} w(c) = 1$.

Proposition 6.2.1. *A set of cardinality n has at most $\frac{2^{n\cdot(k+1)}-2^n}{2^n-1}$ different covers of width at most k.*

Proof. Let H be a set with $|H| = n$, and c a cover of H with width $i \le k$. An object of H can be placed in every set of c. Given that c covers H, there are 2^i-1 possibilities. This holds for every object of H. The number of different covers of width i thus is $(2^i-1)^n$ (or, $\leqslant 2^{n\cdot i}$). (Figure 6.2.2, page 98). Summing over all $1 \le i \le k$ gives the desired bound. □

Given a weighted cover c of $H = \{o_1,\cdots,o_n\}$, $H(o_i) = \{e \in c : o_i \in e\}$ and with little abuse of notation $w(o_i) = \sum_{e\in H(o_i)} w(e)$.

Definition 6.2.2. The syntax of PμTL is given by the grammar:

$$f ::= a \mid \sim a \mid Z \mid f \wedge f \mid f \vee f \mid [\mathsf{X} f]_{\succ p} \mid \mu Z.f(Z) \mid \nu Z.f(Z)$$

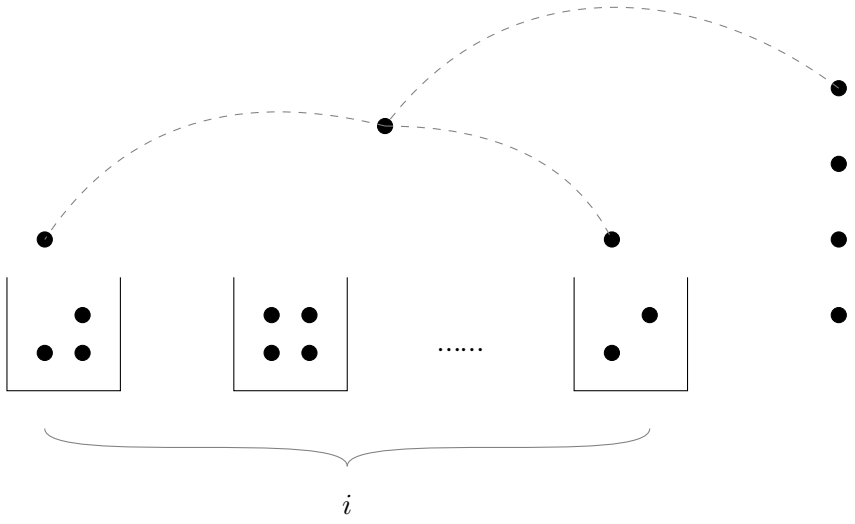


Figure 6.2.2: We have n distinguishable balls and i distinguishable urns. A ball is thrown, and in mid air it duplicates itself and each urn receives at the most one copy. But the order in which balls are received is immaterial.

We have represented the logic in negative normal form, where negation are applied only to the atomic propositions. We also restrict $\succ$ to $\{>, \geq\}$ and p to rational numbers in $[0, 1]$. An occurrence of a variable Z is *bounded* in a sentence (or formula) f, if that occurrence of Z stands within a subformula of f having one of the two forms: $\mu Z.f$ or $\nu Z.f$, else Z occurs *freely* in f. A sentence f is *closed* if all variables occurring in f are bounded. In the sequel, all sentences are assumed to be closed unless stated explicitly. A μ-sentence has the form $\mu Z.f(Z)$, similarly a ν-sentence has the form $\nu Z.f(Z)$.

The semantics of the logic is defined on labeled Markov chains. The pointed satisfaction of a PμTL sentence for a labeled Markov chain M at a state s is defined by the following rules:

Definition 6.2.3.

$s \vDash a$	iff	$a \in L(s)$	$s \vDash \sim a$	iff	$a \notin L(s)$
$s \vDash g \wedge h$	iff	$s \vDash g$ and $s \vDash h$	$s \vDash g \vee h$	iff	$s \vDash g$ or $s \vDash h$
$s \vDash \mu Z.g(Z)$	iff	$s \in \bigcap\{S : g(S) \subseteq S\}$	$s \vDash \nu Z.g(Z)$	iff	$s \in \bigcup\{S : S \subseteq g(S)\}$
$s \vDash [\mathsf{X}g]_{\succ p}$	iff	$\sum_{s' : s' \vDash g} P(s, s') \succ p$			

With little abuse of notation, a sentence f is also a set of states which satisfy f. It should be clear from the context, when f is considered as a sentence and when a set of states.

6.2.1 Motivation and examples

Though the absence of quantified recursively defined path sentences is a severe restriction, none-the-less we can capture many interesting properties in simple probabilistic logics.

- The logic $\mathsf{P}\mu\mathsf{TL}$ can be used to model probabilistic programs where probability distributions are fixed and variables have a finite domain. Consider the probabilistic program

```
while(c == H)
      Toss(c);
```

  where c is a fair coin that initially equals heads or tails. The following $\mathsf{P}\mu\mathsf{TL}$ sentence where proposition t stands for tails and h abbreviates heads:

$$(\mathsf{t} \to [\mathsf{X}\,\mathsf{t}]_{=1}) \wedge \Big(\mathsf{h} \to \nu Z.([\mathsf{X}\,\mathsf{t}]_{\geq \frac{1}{2}} \wedge [\mathsf{X}\,\mathsf{h} \wedge Z]_{\geq \frac{1}{2}})\Big).$$

  expresses that c being initially t implies c stays t almost surely, and that c being initially h implies that it turns into t with at least probability $\frac{1}{2}$ or stays h and continues with the same threshold.

- Application in motion planing [74] and validation of specification for faulty systems, using model checking algorithms.

6.3 Ordinal, Ranks and Signature

Let f be a function on subsets of a universe U, i.e., $f : 2^U \to 2^U$. If f is monotonic, then by the Knaster-Tarski theorem, least and greatest fixed points of f exist. For ordinal α, the least fixed point $\mu(f) = \bigcup_\alpha \mu_\alpha(f)$, and the greatest fixed point $\nu(f) = \bigcap_\alpha \nu_\alpha(f)$, where:

$$\begin{array}{l} \mu_0(f) = \varnothing \;\text{ and }\; \mu_{\alpha+1}(f) = f(\mu_\alpha(f)) \text{ and} \\ \nu_0(f) = U \;\text{ and }\; \nu_{\alpha+1}(f) = f(\nu_\alpha(f)). \end{array}$$

We can view a $\mathsf{P}\mu\mathsf{TL}$ sentence f as characterising a set of states satisfying f. Hence, we denote $\eta_{\alpha+1}(f) = \{s : s \vDash f(\eta_\alpha(f))\}$ for $\eta \in \{\mu, \nu\}$ where $\mu_0(f) = \bot$ and $\nu_0(f) = \top$. With little abuse of notation, we denote $s \vDash \eta_\alpha(f)$ iff $s \in \eta_\alpha(f)$. The satisfaction relation of $\mathsf{P}\mu\mathsf{TL}$ (see Def. 6.2.3) can now be rephrased as follows:

$$\begin{array}{l} s \vDash \mu Z.f(Z) \text{ iff for some ordinal } \alpha,\ s \vDash \mu_\alpha(f) \\ s \vDash \nu Z.f(Z) \text{ iff for all ordinals } \alpha,\ s \vDash \nu_\alpha(f). \end{array}$$

No state satisfies $\mu_0(f)$, and every state satisfies $\nu_0(f)$.

Definition 6.3.1 (Rank)**.** The μ-sentence $\mu Z.f(Z)$ has *rank* α at state s if α is the least ordinal such that $s \vDash \mu_\alpha(f)$. If there is no ordinal $\alpha < \omega$ such that $s \vDash \mu_\alpha(f)$, then the rank of $\mu Z.f(Z)$ at s is ω. [1]

Example 6.3.1. Consider the following Markov chain:

$$s_1 \xrightarrow{1} s_2 \xrightarrow{1} s_3 \xrightarrow{1} s_4 \xrightarrow{1} \cdots$$

where s_4 satisfies a and s_i satisfies $\sim a$ for $i < 4$. The sentence $\mu Y.(a \vee [\mathsf{X}Y]_{>0})$ has rank 4 at s_1, 3 at s_2, 2 at s_3 etc.

Definition 6.3.2 (Signature)**.** A *signature* is a sequence of ordinals. Let $<$ be the lexicographical ordering on signatures. Over a set of *bounded* length signatures, the lexicographical ordering is total and well defined.

Definition 6.3.3 (μ-height)**.** The *μ-height* of PμTL-sentence f is the nesting depth of closed μ-sub-sentences (including f) of f.

Example 6.3.2. Formula $\mu Z.([\mathsf{X}Z]_{>0} \vee \mu Y.(b \wedge [\mathsf{X}Y]_{>0}))$ has μ-height 2. The μ-height of $\mu Z.(a \vee \mu Y.(b \wedge [\mathsf{X}Z]_{=1} \vee [\mathsf{X}Y]_{>0}))$ is 1, since $\mu Y.(b \wedge [\mathsf{X}Z]_{=1} \vee [\mathsf{X}Y]_{>0})$ is not closed.

Definition 6.3.4. Let f be a PμTL-sentence of μ-height n. Sentence f *has the signature* $\sigma = \alpha_1, \cdots, \alpha_n$ at state s if σ is the (lexicographically) least signature such that $s \vDash f'$ where f' is obtained by replacing every μ-sub-sentence $\mu Z.g$ in f of μ-height i by $\mu_{\alpha_i}(g)$.

Observe that ordinal α_i is used only for least fixed point sentences of μ-height i. Greatest fixed point sentences play no role for this notion.

Example 6.3.3. Let the sentence $f = \mu Z.([\mathsf{X}Z]_{>0} \vee (b \wedge \mu Y.(a \vee [\mathsf{X}Y]_{>0})))$ with μ-height 2. Consider the MC:

$$s_1 \xrightarrow{1} s_2 \xrightarrow{1} s_3 \xrightarrow{1} s_4 \xrightarrow{1} s_5 \xrightarrow{1} \cdots$$

where only s_5 satisfies a and only s_3 satisfies b. Sentence f has signature $(3,3)$ at s_1, $(3,2)$ at s_2, $(3,1)$ at s_3, $(2,\omega)$ at s_4, and $(1,\omega)$ at s_5.

Proposition 6.3.1. *Signatures of* PμTL *sentences satisfy:*

1. *If $f \vee g$ has signature σ at s, then either f or g has signature $\leq \sigma$ at s.*
2. *If $f \wedge g$ has signature σ at s, then f and g both have signatures $\leq \sigma$ at s.*

[1] ω denotes the first infinite ordinal.

3. *If* $[\mathsf{X}g]_{\succ p}$ *has signature* σ *at* s*, then there is a set* H *of successors of* s*, such that* $\sum_{t\in H} P(s,t) \succ p$*, and* g *has a signature* $\leq \sigma$ *at* t*, for every* $t \in H$*.*

4. *If* $\mu Z.f(Z)$ *has signature* σ *at* s*, then* $f(\mu Z.f(Z))$ *has signature* $\sigma' < \sigma$ *at* s*.*

5. *If* $\nu Z.f(Z)$ *has signature* σ *at* s*, then* $f(\nu Z.f(Z))$ *has signature* σ' *with prefix* σ *at* s*.*

Proof. Cases 1 and 2. Suppose $\varphi = f \vee g$ has a signature $\sigma = (\alpha_1, \cdots, \alpha_n)$ at s. Let φ' be the formula obtained by replacing each occurrence of μ-sentence $\mu Z.f(Z)$ of μ-height i by $\mu_{\sigma_i}(f)$. Then $s \vDash \varphi'$. Observe that each μ-sub-sentence of φ belongs either to f or g. Thus the sentence obtained by replacing every μ-sub-sentence $\mu Z.f(Z)$ of either f or g of height j by $\mu_{\sigma_j}(f)$ is also satisfied by s. Thus, either f or g has signature σ' at most σ. Similar arguments hold for $f \wedge g$.

Case 3. Assume $[\mathsf{X}g]_{\succ p}$ has signature $\sigma = (\alpha_1, \cdots, \alpha_n)$. Let g' be the sentence obtained by replacing every occurrence of μ-sentence $\mu Z.f(Z)$ of height i by $\mu_{\sigma_i}(f)$ in g. Then $s \vDash [\mathsf{X}g']_{\succ p}$. This implies that there exists a set H of successors of s such that $\sum_{t\in H} P(s,t) \succ p$ and for each $t \in H$, either $t \vDash g'$, or g has a signature at most σ at t.

Case 4. Let $\varphi = \mu Z.f(Z)$, and the signature at s be $\sigma = (\alpha_1, \cdots, \alpha_n)$. We assume without loss of generality that φ is the only formula with μ-height n. Let ψ be a μ-sub-sentence in $f(\varphi)$. We distinguish:

1. ψ occurs either properly inside φ or does not contain Z. Then the μ-height of ψ in φ and $f(\varphi)$ coincide. Thus, the rank α_i, say, of ψ in φ is the rank of ψ in $f(\varphi)$ too.

2. $\psi = \varphi$. The μ-height of φ in $f(\varphi)$ remains n. By definition of $\mu_\alpha(f)$, the rank of sub-sentence φ (of $f(\varphi)$) is $\alpha_n - 1$.

3. ψ contains Z. Let j be the μ-height of ψ (when occurring) in φ. Then it has μ-height $n+j$ in $f(\varphi)$. The signature of $f(\varphi)$ at s is thus $\sigma' = \{\alpha_1, \cdots, \alpha_{n-1}, (\alpha_n - 1), \alpha'_{n+1}, \cdots \alpha'_{n+k}\}$, where $n+k$ is the μ-height of $f(\varphi)$. Lexicographically, $\sigma' < \sigma$.

Case 5. concerns greatest fixed points, which does not change the signature of any sub-sentence of $f(\nu Z.f(Z))$ whose μ-height is $\leq n$. □

6.4 Pre-Model and derivations

Throughout the rest of this section, we will assume that every sub-sentence of a given sentence is unique.

Definition 6.4.1 (FL closure)**.** The *Fisher-Ladner closure* of $\mathsf{P}\mu\mathsf{TL}$-sentence f is the smallest set $\mathsf{FL}(f)$ satisfying:

1. $f \in \mathsf{FL}(f)$.
2. If $g \vee h \in \mathsf{FL}(f)$ then $g, h \in \mathsf{FL}(f)$.
3. If $g \wedge h \in \mathsf{FL}(f)$ then $g, h \in \mathsf{FL}(f)$.
4. If $[\mathsf{X}g]_{>p} \in \mathsf{FL}(f)$ then $g \in \mathsf{FL}(f)$.
5. If $\eta Z.g \in \mathsf{FL}(f)$ then $g(\eta Z.g) \in \mathsf{FL}(f)$ for $\eta \in \{\mu, \nu\}$.

Example 6.4.1. For $\varphi = \nu Z.(a \wedge [\mathsf{X}\, Z]_{=1})$, we have $\mathsf{FL}(\varphi) = \{\varphi, a \wedge [\mathsf{X}\,(\nu Z.a \wedge [\mathsf{X}\, Z]_{=1})]_{=1}, a, [\mathsf{X}\,(\nu Z.a \wedge [\mathsf{X}\, Z]_{=1})]_{=1}\}$.

Remark ([44]). For every $\mathsf{P}\mu\mathsf{TL}$-formula f, $|\mathsf{FL}(f)| \in O(|f|)$.

We now introduce the notion of pre-model of formula f.

Definition 6.4.2 (Pre-model)**.** A *pre-model* of $\mathsf{P}\mu\mathsf{TL}$ sentence f is an MC $M_f = (S, P, 2^{\mathsf{FL}(f)}, L, s_{in})$ satisfying:

1. $f \in L(s_{in})$.
2. If $f \in L(s)$ then $\sim f \notin L(s)$.
3. If $f \vee g \in L(s)$ then $f \in L(s)$ or $g \in L(s)$.
4. If $f \wedge g \in L(s)$ then $f, g \in L(s)$.
5. If $[\mathsf{X}\, g]_{>p} \in L(s)$ then $\sum_{s' : g \in L(s')} P(s, s') > p$.
6. If $\eta Z.f \in L(s)$ then $f(\eta Z.f) \in L(s)$ with $\eta \in \{\mu, \nu\}$.

Each pre-model defines a specific choice of *derivation* rules.

Definition 6.4.3 (Derivation)**.** The *derivation relation induced* by pre-model $M_f = (S, P, 2^{\mathsf{FL}(f)}, L, s_{in})$ of $\mathsf{P}\mu\mathsf{TL}$ sentence f is defined by:

1. If $\varphi = h \vee g \in L(s)$ and $h \in L(s)$, then φ *derives* h (at s). Similar holds if $g \in L(s)$.

2. If $\varphi = h \wedge g \in L(s)$, then φ *derives* h and g (at s).

3. If $\varphi = [\mathsf{X} g]_{>p} \in L(s)$ and $g \in L(t)$ for some successor t of s, then φ (at s) *derives* g (at t).

4. If $\varphi = \eta Z.h \in L(s)$ with $\eta \in \{\mu, \nu\}$, then φ *derives* $h(\eta Z.h)$ (at s).

The *derivation relation* of f is the union of derivation relations induced by all pre-models of f, i.e., g *derives* h iff for some states s, t of a pre-model of f, g in s derives h in t.

Note that the derivation relation of f only relates sentences in $\mathsf{FL}(f)$. An intuitive way to understand the derivation relation is to consider it as a logical implication. Also note that, not all pre-models of f are models of f. For instance, sentence $\varphi = \mu Z.f(Z)$ could be derived forever (by clause 4 of Def. 6.4.3). For a pre-model to be a model, a μ-sentence cannot be derived over and over again without ever satisfying it. But it is possible that a μ-sub-sentence appears infinitely often in a derivation sequence. So we have to be careful about which derivation sequences we are referring to. We will use the following definition.

Definition 6.4.4 (Regeneration)**.** The μ-sentence φ is *regenerated* by a sequence of derivations, if starting from φ we end up again with φ, and φ is a μ-sub-sentence for every sentence in any intermediate derivation.

Example 6.4.2. Sentence $f = \mu Z.\,(\nu Y.(a \wedge Y) \vee Z)$ derives $\nu Y.(a \wedge Y) \vee \mu Z.(\nu Y.\,(a \wedge Y) \vee Z)$ that contains f, and which can derive $\mu Z.\,(\nu Y.(a \wedge Y) \vee Z)$ which equals f. Thus the μ-sentence f is regenerated (See Fig. 6.4.3, left).

Example 6.4.3. Let $\varphi = \nu Z.\,(\mu Y.(b \vee [\mathsf{X} Y]_{=1}) \wedge [\mathsf{X} Z]_{=1})$. φ derives $\mu Y.(b \vee [\mathsf{X} Y]_{=1}) \wedge [\mathsf{X}\ \varphi]_{=1}$. This derives a formula containing $\mu Y.(b \vee [\mathsf{X} Y]_{=1}) \wedge [\mathsf{X}\varphi]_{=1}$ (see Fig. 6.4.3, right). This in turn derives $b \wedge [\mathsf{X}\varphi]_{=1}$ which derives φ. Though the μ-sentence $\mu Y.(b \vee [\mathsf{X} Y]_{=1})$ is witnessed again. However it is not regenerated, since it is not a sub-sentence of every derived sentence.

Definition 6.4.5 (Well-foundedness)**.** A pre-model M_f of $\mathsf{P}\mu\mathsf{TL}$-sentence f is *well-founded* if every μ-sub-sentence φ of f is regenerated *finitely* often.

Theorem 6.4.1. *Every model of* $\mathsf{P}\mu\mathsf{TL}$*-sentence* f *is a well-founded pre-model of* f*.*

Proof. Let M be a model of f. We first generalize the state-labeling; $\forall f \in \mathsf{FL}(\varphi):\ s \vDash f \iff f \in L(s)$. Note that with the generalized labeling, the model satisfies the conditions in Def. 6.4.2. The rest of the reasoning is as

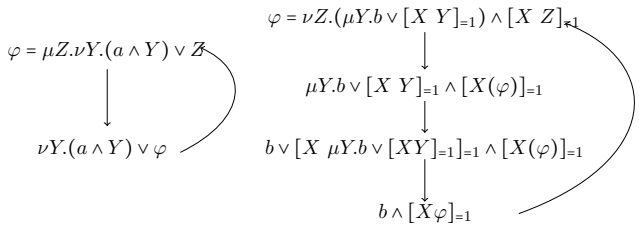


Figure 6.4.3: Derivation sequence for two formulas. The derivation sequence in the upper part shows a regeneration sequence; the derivation in the lower part is not a regeneration sequence.

follows: If $M, s_{in} \vDash \varphi$, then φ has a signature at initial state s_{in}, and we can ensure that every μ-sub-sentence of φ is regenerated finitely often.

Let $\psi = \mu Z.f(Z)$ be a sub-sentence of φ with μ-height n that is regenerated following a sequence of derivations from s to t (s and t can be identical). We will show that the signature $\sigma = (\alpha_1, \cdots, \alpha_n)$ of ψ from s to t decreases. As per definition, the derivation step begins by deriving $f(\mu Z.f(Z))$ from $\mu Z.f(Z)$. By Proposition 6.3.1, the sentence $f(\mu Z.f(Z))$ has lexicographically smaller signature at s. It remains to show that this decrease is not violated by other derivation rules between s and t.

For disjunction $h \vee g$, suppose (w.l.o.g.) $s \vDash g$. By Def. 6.4.3, g is derived. By Proposition 6.3.1, the signature cannot increase. As the derivation relation for a conjunction does not affect the signature, for this case no decrease occurs. The derivation rule for $[\mathsf{X}g]_{>p}$ derives g at some successor state t. If $f(\psi)$ is a sub-sentence, then the signature at t cannot increase; the other case is trivial.

A derivation may involve other fixed point sentences. Derivations of fixed point sentences that are sub-sentences of ψ do not affect the μ-height of $f(\varphi)$. For example, consider $\psi = \mu Z.f(Z, \mu Y.g(Y))$ with μ-height n. Applying the derivation for φ yields $\psi' = f(\varphi, \mu Y.g(Y))$ which has μ-height n too. The derivation steps for $\mu Y.g(Y)$ in $f(\varphi, \mu Y.g(Y))$ give $f(\varphi, g(\mu Y.g(Y)))$. The μ-height of $\mu Z.f(Z, g(\mu Y.g(Y)))$ does not change, hence the ordinal α_n in the signature of $f(\varphi)$ is unaffected.

Now, consider $\phi = \eta Y.g(Y)$ (where $\eta \in \{\mu, \nu\}$) with μ-subsentence $\psi = \mu Z.f(Z)$. Distinguish two cases:

1. The derivation of ϕ does not affect the μ-height of ψ. In such a case, deriving $\mu Y.g(Y)$ decreases the signature (by Proposition 6.3.1). For example, consider the derivation steps for $\varphi = \mu Y.g(Y, \mu Z.f(Z))$. This gives $g(\mu Y.g(Y, \mu Z.f(Z)), \mu Z.f(Z))$. The μ-height of $\mu Z.f(Z)$ is not affected, and the signature of φ decreases.

2. The derivation of ϕ increases the μ-height of ψ. For example, $\phi = \eta Y.g(\psi)$, where $\psi = \mu Z.f(Z, Y)$. Observe that a derivation of ψ can only occur after a derivation of ϕ. That would make ϕ a subsentence of ψ, namely $\mu Z.f(Z, \phi)$. The case where ϕ is a subsentence of ψ has already been considered.

Thus, we have derivations where each μ-sentence reduces its corresponding rank. Since the derivation sequence from φ has bounded length (by Observation 6.4, pp. 102), a regeneration can only happen finitely often. Thus the pre-model is well-founded. □

Theorem 6.4.2. *Each well-founded pre-model of* $\mathsf{P}\mu\mathsf{TL}$*-sentence* f *is a model of* f*.*

Proof. (*sketch*) Let MC M be a well-founded pre-model of f. Then the regeneration relation for every μ-subsentence of f terminates. Each μ-sentence thus has a (finite) rank at every state in M, and hence there exists a signature for $\varphi \in \mathsf{FL}(f)$ at each state. Let $\sigma = \alpha_1, \cdots, \alpha_n$ be the lexicographically smallest such signature. Replace each occurrence of the μ-sentence $\mu Z.f(Z)$ of height i by $\mu_{\alpha_i}(f)$. It follows by structural induction on sentences that $\varphi \in L(s)$ implies $s \vDash \varphi$. □

Theorem 6.4.3. *If* $\mathsf{P}\mu\mathsf{TL}$*-sentence* f *is satisfiable, then it has a model of bounded out-degree at most* $|f|+1$*.*

Proof. Similar to the proof of Proposition 5.2.3 (pp. 82). □

6.5 Decision procedure for satisfiability

This section presents a decision procedure for determining the satisfiability of $\mathsf{P}\mu\mathsf{TL}$ sentence f. The procedure is based on a parity game obtained as cross-product of a game graph and a deterministic parity automaton.

Deterministic parity automaton

We first focus on the parity automaton. The starting point is a Büchi-automaton A_φ for each μ-sentence φ of f. The automaton A_φ accepts the regeneration sequences for φ, i.e., derivation sequences that derive φ infinitely often and for which φ is a sub-sentence of every sentence in the derivation.

Definition 6.5.1 (Büchi automaton for an μ-sentence)**.** Let f be a $\mathsf{P}\mu\mathsf{TL}$-sentence with φ a μ-sentence in $\mathsf{FL}(f)$. The *non-deterministic Büchi automaton (NBA)* A_φ is a quintuple $(Q_\varphi, \Sigma_\varphi, Q_{\varphi,in}, \delta_\varphi, F_\varphi)$ where:

1. $Q_\varphi = \{\psi \in \mathsf{FL}(f) : \varphi \text{ is a } \mu\text{-sentence of } \psi\}$, the state-set
2. $\Sigma_\varphi = 2^{\mathsf{FL}(f)}$, the alphabet
3. $Q_{\varphi,in} = Q_\varphi$, the set of initial states,
4. $\delta_\varphi(q, q') = q'$, if q' is obtained from q by a derivation
5. $F_\varphi = Q_\varphi$, the set of final states.

The transition relation is represented in a compressed form, that is, $\delta_\varphi(q, a) = q'$ implies that for all $A \in \Sigma$ with $a \in A$, $\delta_\varphi(q, A) = q'$.

For PμTL-sentence f, let A_f be a deterministic parity automaton (DPA) which is the complement of the union of the automata A_φ for μ-sentence φ of f:

$$\overline{L(A_f)} = \bigcup_{\varphi \in \mathsf{FL}(f):\varphi \text{ is a } \mu\text{-sentence}} L(A_\varphi)$$

A_f thus accepts all terminating regeneration sequences for every μ-sentence in $\mathsf{FL}(f)$. The union of $L(A_\varphi)$ can be described by an NBA of maximal size in $O(kn)$ where k is the number of μ-sentences in $\mathsf{FL}(f)$ and $|\mathsf{FL}(f)| = n$. The DPA A_f then has size at most $O(2^{(kn)^2})$ [81].

A two-player game

Our next aim is to define a two-player game where player 0 aims to show that PμTL-sentence f is satisfiable, while its opponent wants to refute this claim. The vertices of the game graph are sets of subsets of $\mathsf{FL}(f)$. A vertex v is called *transitive* iff:

- For all $g \vee h \in v$ either $g \in v$ or $h \in v$.
- For all $g \wedge h \in v$, $g, h \in v$.
- For all $\eta X.g(X) \in v$, $g(\eta X.g(X)) \in v$.
- There exists $[\mathsf{X}g]_{\succ p} \in v$.

Definition 6.5.2 (Two-player game)**.** The two-player game G_f for PμTL-sentence f is the triple (V, E, v_0) where $V = V_0 \uplus V_1$ with $V_0 \subseteq 2^{\mathsf{FL}(f)}$ the set of Player 0 vertices, V_1 the set of Player 1 vertices (defined below), $v_0 = \{f\} \in V_0$ the starting vertex, and E is defined by:

1. $(v, v \cup \{g_i\}) \in E$ for i=1, 2, if $g = g_1 \vee g_2 \in v$
2. $(v, v \cup \{g_1, g_2\}) \in E)$ if $g = g_1 \wedge g_2 \in v$

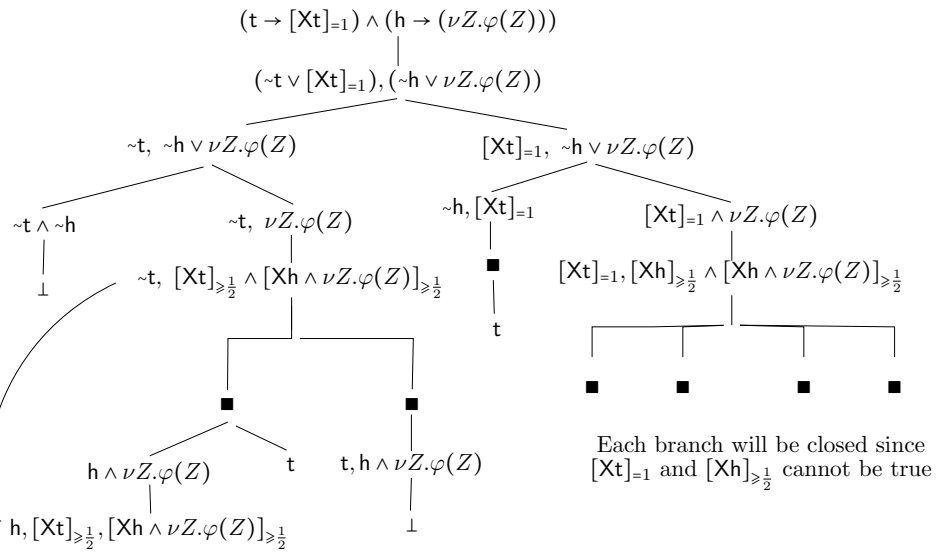


Figure 6.5.4: The satisfiability game for $(\mathsf{t} \to [\mathsf{X}\mathsf{t}]_{=1}) \wedge (\mathsf{h} \to (\nu Z.\varphi(Z)))$, where $\varphi = [\mathsf{X}\mathsf{t}]_{\geqslant\frac{1}{2}} \wedge [\mathsf{X}\mathsf{h} \wedge Z]_{\geqslant\frac{1}{2}}$. The black squares indicate Player 1 vertices, others are Player 0 vertices. Since there are no μ sub-sentences, any infinite path (or ending in no $\bot$-vertex) is winning for Player 0.

3. $(v, v \cup \{g(\mu X.g)\}) \in E$ if $\mu X.g \in v$, and

4. $(v, v_c) \in E$ if v is a transitive vertex and v_c represents weighted cover (c, w) of $V_v = \{g : [\mathsf{X}\, g]_{>p} \in v\}$ with $w(g) > $ for all $g \in V_v$. In addition, $(v_c, v') \in E$ for each $v' \in c$. If no such cover c exists, then $(v, \bot) \in E$. Let $V_1 = \{v_c : (v, v_c) \in E\}$ with v_c as above.

Example 6.5.1. Consider the game in Figure 6.5.4. Let us clarify one of its weighted covers. Consider the vertex $\langle \neg\mathsf{t},\ [\mathsf{X}\,\mathsf{t}]_{\geqslant\frac{1}{2}} \wedge [\mathsf{X}\,\mathsf{h} \wedge \nu Z.\varphi(Z)]_{\geqslant\frac{1}{2}} \rangle$. There are two possible weighted covers: (c, w_1) and (c_2, w_2). Let $c_1 = \{v_1, v_2\}$ and $c_2 = \{v_3\}$, where $v_1 = \langle \mathsf{h} \wedge \nu Z.\varphi(Z) \rangle$, $v_2 = \langle \mathsf{t} \rangle$ and $v_3 = \langle \mathsf{h} \wedge \nu Z.\varphi(Z), \mathsf{t} \rangle$. The weights are $w_1(v_1) = w_1(v_2) = \frac{1}{2}$ and $w_2(v_3) = 1$.

Let $V_\bot$ be the set of vertices which contain propositional contradictions (like a and $\sim a$). The game graph G_f is of size at most $2^{O(n^2)}$ where $n = |\mathsf{FL}(f)|$. Player 0 looses if the finite play reaches $V_\bot$. An infinite play is winning for player 0 if it is accepted by the DPA A_f. To accomplish this, we define the parity game $\mathcal{G}_f$ as the (synchronous) cross product of game G_f and DPA A_f (Definition 6.5.3).

Definition 6.5.3. The cross product between the deterministic parity automaton $A_f = (Q, \Sigma, q_0, \delta, F)$ and game graph $G_f = (V, E, v_0)$ yields the parity game $\mathcal{G}_f = (U, R, u_0, \Omega)$ where

- $U \subseteq V \times Q$
- $u_0 = (v_0, q_0)$
- The transition relation R is defined by:
  - If v is not a transitive vertex then: $(v, q), (v', q') \in R$ iff $q' = \delta(q, v)$.
  - If v is a transition vertex then: $(v, q), (v', \blacktriangle) \in R$ and $(v', \blacktriangle), (v'', q') \in R$ iff $q' = \delta(q, v')$.
- $\Omega : U \to \mathsf{Img}(\mathsf{F})$ such that $\Omega(v, q) = F(q)$. Recall F is a parity condition of A_f.

Note that $\blacktriangle$ is simply used as a placeholder and has no special meaning.

Remark. There are some crucial differences between the game $\mathcal{G}_f$ and the tree-automaton construction for $\mathsf{P}\mu\mathsf{TL}$ in [74]. For vertices with formula $g \wedge h$, the set of formulas is not split into two vertices (one containing g and one containing h); instead they are kept together. The key difference is the distribution of formulas as solutions of weighted covers.

Proposition 6.5.1. *Player 0 has a winning strategy in parity game $\mathcal{G}_f$ for every satisfiable $\mathsf{P}\mu\mathsf{TL}$-sentence f.*

Proof. Let f be satisfiable. By Theorem 6.4.1, f has a well-defined pre-model, say MC $M_f = (S, P, \mathsf{AP}, L, s_{in})$. The proof is by constructing a winning strategy $\pi : V^+ \to V$ in game $\mathcal{G}_f$ for player 0 against any strategy of player 1. This is done using the auxiliary function $\Gamma : V^+ \to S$ that maps finite plays in $\mathcal{G}_f$ onto states of M_f. Define $\Gamma(v_0) = s_{in}$. Consider a finite play ρ of $\mathcal{G}_f$ with $s = \Gamma(\rho)$ and $v = \mathsf{last}(\rho)$. Distinguish the following cases:

- v is a vertex with $g_1 \vee g_2 \in v$. If $g_i \in L(s)$ then let $\pi(\rho) = v_i$ (see Def. 6.5.2) and $\Gamma(\rho \cdot v_i) = s$, for $i \in \{1, 2\}$.
- v is a transitive vertex. Assume s has direct successors $\{t_1, \cdots, t_k\}$. By Theorem 6.4.3, it follows $k \leq |\mathsf{FL}(f)|+1$. Define $\pi(\rho) = v_c$ (i.e., a cover vertex, see Def. 6.5.2) for cover $c = \{L(t_1), \cdots, L(t_k)\}$. (Note that such cover always exists.) If player 1 selects $L(t_i)$, then let $\Gamma(\rho \cdot v_c \cdot L(t_i)) = t_i$.

- in any other case, v has at the most one successor, say v'. Define $\pi(\rho) = v'$ and $\Gamma(\rho{\cdot}v') = s$.

It remains to show that π is a winning strategy. For any strategy σ of Player 1, consider the resulting path ρ from the pair of strategies (π, σ). Path ρ cannot terminate in a $\bot$-vertex, as otherwise the label of $\Gamma(\rho)$ should contain a propositional contradiction. If ρ is infinite, then every regenerating μ-sub-sentence in the vertices of ρ is terminating. Hence $\rho \in L(A_f)$. □

Proposition 6.5.2. *If there exists a winning strategy for player 0 in parity game $\mathcal{G}_f$, then f is satisfiable.*

Proof. Let π be a winning strategy of player 0 in $\mathcal{G}_f$. Applying the strategy π to the game $\mathcal{G}_f$ yields the digraph $\mathcal{G}_f^\pi$. Let Π be the set of all finite paths $\sigma = (\sigma_0 \cdots \sigma_n)$ in $\mathcal{G}_f^\pi$ such that

1. σ_0 is a player 0 configuration, and it is either the initial configuration or has a player 1 configuration as a parent.

2. σ_n is a player 1 configuration.

Consider a path $\sigma \in \Pi$. Observe that each configuration except the last configuration (σ_n) of σ has at most one descendant. Path σ is said to *lead* to σ', if the last configuration of σ has an edge to every configuration of σ'. Let pre-model $M_f = (S, P, \mathsf{AP}, L, s_0)$ be obtained from digraph $\mathcal{G}_f^\pi$ in the following way:

- $S = \{s_\sigma : \sigma \in \Pi\}$.

- $P(s_\sigma, s_{\sigma'}) > 0$, if σ *leads* to σ'. The exact value is the weight defined by the weighted cover σ_n (see also Example 6.5.1).

- $L(s_\sigma) = \bigcup_{i=0}^{k-1} \sigma_i$, where $k = |\sigma|$.

- $s_0 = s_\sigma$, where $\sigma = (\sigma_0, \cdots, \sigma_n)$ such that σ_0 is the initial configuration of the game.

It is easy to see that the pre-model M is well-founded. From Proposition 6.4.2 it follows that f is satisfiable. □

Theorem 6.5.3. *Every satisfiable* $\mathsf{P}\mu\mathsf{TL}$ *sentence f has a model of size exponential in $|f|$.*

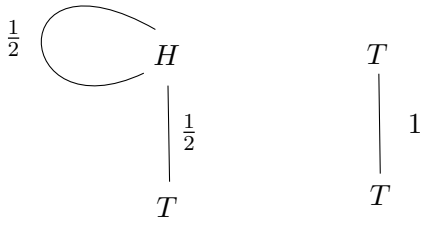


Figure 6.5.5: Two models generated by two different winning strategies of Player 0 in the game given in Fig. 6.5.4

Proof. By Propositions 6.5.1 and 6.5.2 it follows that PμTL-sentence f is satisfiable iff Player 0 has a winning strategy in $\mathcal{G}_f$. It is well-known that if a winning strategy for a parity game exists, then there exists a pure memoryless winning strategy [46]. The size of the well-founded pre-model defined by a pure memoryless strategy is $2^{O((kn)^2)}$. □

Remark. Our last result together with the fact that qualitative PCTL has no finite model property [13] yields that qualitative PCTL and PμTL have incomparable expressive power. The PμTL-formula $\nu Y.\,(a \wedge [\mathsf{X}\,Y]_{>0})$ cannot be expressed in qualitative PCTL. Vice versa, the PCTL-formula $[\mathsf{G}\,([\mathsf{X}\,a]_{>0} \wedge \neg a)]_{>0}$ cannot be expressed in PμTL.

Corollary. *Every satisfiable* PμTL *sentence has a model whose transition probabilities are rational.*

Proof. The weight function for the weighted cover is determined by linear constraints. Thus if it is satisfiable, then it has a rational solution (recall that the probability bounds on the formula are rational). □

6.5.1 Discussion

Compared to the alternating tree automaton approach in [18], our game-based approach has the following advantages:

- It provides a clear separation of recursive sub-sentences and sentences with probability bounds. The probability bounds in the decision procedure only affect the existence of the *cover* vertices (v, v_c) in the game.
- An (ordinary) non-stochastic parity game was used for the satisfiability of a probabilistic logic.

- The game graph of a $\mathsf{P}\mu\mathsf{TL}$-sentence f captures all models of f.

The latter property enables querying some quantitative properties, e.g., the maximum probability of reaching certain states in the models of f. For instance, consider the game graph $G_f = (V, E, v_0)$ in Fig. 6.5.4, where all vertices that are losing for player 0 are omitted. The maximum probability of reaching specific vertices can then be obtained as follows. The cover vertices (player 1 vertices) act as probabilistic vertices whose transition relations are defined by linear equations. Using algorithms for determining reachability probabilities in convex MDPs [86, 21][2], we can calculate the maximum probability of reaching certain vertices.

This observation is a step towards considering an extension of $\mathsf{P}\mu\mathsf{TL}$ with until-modalities (that do neither occur as sub-sentence of another until-modality nor as part of a μ-sentence). Our decision procedure can be extended as follows for $[f_1 \,\mathsf{U}\, f_2]_{\succ p}$. We first extend the transition relation in Def. 6.5.2 by:

5.1 $(v, v') \in E$ if $[f_1 \,\mathsf{U}\, f_2]_{\succ p} \in v$ and $f_2 \vee (f_1 \wedge [\mathsf{X}\,\mathsf{U}^*(f_1, f_2)]_{\succ 0}) \in v'$, where $\mathsf{U}^*(f_1, f_2)$ denotes $[f_1 \,\mathsf{U}\, f_2]_{\succ p'}$ for some p' (which is not relevant here).

5.2 $(v, v') \in E$ if $\mathsf{U}^*(f_1, f_2) \in v$ and $f_2 \vee (f_1 \wedge [\mathsf{X}\,\mathsf{U}^*(f_1, f_2)]_{\succ 0}) \in v'$.

Intuitively, $[f_1 \,\mathsf{U}\, f_2]_{\succ p}$ is dealt as $\mu Z.\,(f_2 \vee (f_1 \wedge [\mathsf{X}\, Z]_{\succ 0}))$. This yields the game graph $G_f = (V, E, v_0)$ and the winning condition is obtained as in Section 6.5. We remove all vertices that are not winning for player 0 and the sentence is satisfiable if for each (remaining) vertex w with $[f_1 \,\mathsf{U}\, f_2]_{\succ p} \in w$, the supremum probability of reaching vertices with f_2 only via vertices containing f_1 is $\succ p$.

This raises the question whether this technique can be extended to nested until-modalities. By a similar mechanism as above we annotate vertices with sentences $[f_1 \,\mathsf{U}\, f_2]_{\succ p}$ with *obligation* $\succ p$. This then amounts to decide the reachability problem for MDPs with *obligations*.[3] To our knowledge, such obligatory games can only be solved under strong structural restrictions. Nonetheless we believe this indicates that tying the satisfiability problem of a recursive probabilistic logic (with unbounded until) to a *finite* obligatory game is a promising avenue.

[2] Even in the presence of non-strict in-equalities [21].

[3] Formal definition of obligatory games is beyond the scope of this paper.

Logic	Finite model	Small model	SAT checking
Px_ω	yes	$O(\|f\|)$	PSPACE-c.
bounded PCTL	yes	$2^{O(\mathsf{size}(f))}$	NEXPTIME EXPTIME-hard
qualitative PCTL	no	–	EXPTIME-c.
PCTL	no	–	?
PμTL	yes	$2^{O(\|f\|)}$	UTIME($2^{O(\|f\|)}$)∩ co-UTIME($2^{O(\|f\|)}$)
μPCTL	no	–	?

Table 6.1: Overview of known satisfiability results (where $\mathsf{size}(f)$ equals $|\mathsf{ord}(f)|+|\mathsf{sub}(f)|$). The first two rows and the fifth row summarise this paper.

6.6 Conclusion

This chapters considered the satisfiability problem of PμTL. The logic possesses the small model property, is shown to also have the rational model property. Our results for PμTL show that PμTL and qualtitative PCTL have incomparable expressive power. The satisfiability problem for PμTL is shown to be in the same complexity class as the satisfiability problem for the modal μ-calculus, i.e., in UTIME($2^{O(|f|)}$)∩co-UTIME($2^{O(|f|)}$). This improves the 2-EXPTIME algorithm recently provided in [74]. Table 6.1 summarises the current situation. The satisfiability of PCTL [53] and μ-PCTL [18] remain open problems.

Chapter 7

P-automata for MDPs

P-automata provide an automata-theoretic approach to probabilistic verification. Similar to alternating tree automata which accept labelled transition systems, p-automata accept labelled Markov chains (MCs). In this chapter we proposes an extension of p-automata that accept the set of all MCs (modulo bisimulation) obtained from a Markov decision process under its schedulers.

7.1 Introduction

Model checking of μ-calculus [66] formulas on a (finite) Kripke structure (also know as labeled translation system) is a well studied verification technique of discrete state systems [41]. The problem entails whether every execution (infinite tree) of a Kripke structure satisfies a given μ-calculus formula. The satisfiability problem for μ-calculus, on the other hand, is to decide whether there exists an infinite tree which satisfies a given μ-calculus formula. Both these problems are algorithmically feasible, and the key method is the translation to alternating tree automata [98].

The notion of p-automata was introduced in [58] to provide a similar automata-theoretical foundation for the verification of probabilistic systems as alternating tree automata provide for Kripke structures. As alternating tree automata describe a complete framework for abstraction with respect to branching-time logic like, μ-calculus, CTL and CTL* [98], p-automata, similarly give a unifying framework for different probabilistic logics.

Every p-automaton defines a set of labeled Markov chains, that is, a p-automaton reads an entire Markov chain as input and it either accepts the Markov chain or rejects it. Analogous to alternating tree automata where acceptance of a Kripke structure is decided by solving 2-player games [98], the acceptance of a labeled Markov chain by a p-automaton is decided by

solving *stochastic* 2-player games. The language of p-automata are sets of labeled Markov chains.

We view a Markov decision process (MDP) as a set of Markov chains defined by different schedulers. In this chapter we revisit p-automata defined by [58] and extend it with a new construct and semantics for representing set of Markov chains defined by the Markov decision processes.

The chapter deals with the following topics:

- We extend the p-automata with a construct that captures the non-determinism in the choice of probability distribution. This allows us to model Markov decision processes as p-automata. We show that the extended p-automata are closed under bisimulation, union and intersection, (though, in contrast to [58], the language is no longer closed under negation).

- We show that the language of the p-automaton obtained from an MDP accepts exactly those Markov chains that are bisimilar to the Markov chains induced by the schedulers of the MDP.

- We define a simulation relation between p-automata, that approximates the language inclusion. The simulation relation is complete in the sense of Segala's [88] simulation relation defined for probabilistic automata.

7.2 Weak P-automata$^{\oplus}$

In this section, we extend p-automata as defined in [58]. In the rest of the thesis, when we refer to p-automata we will assume the extended p-automata (as defined in Definition 7.2.2 below), unless the contrary is explicitly stated.

Definition 7.2.1 (Boolean formulas on T)**.** Let T be any arbitrary set, then $\mathcal{B}^+(T)$ is the set of positive boolean formulas generated by the following syntax:

$$\varphi ::= \ t \mid \mathsf{true} \mid \mathsf{false} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \tag{7.1}$$

where $t \in T$.

The closure of $\varphi \in \mathcal{B}^+(T)$ is defined as $\mathsf{cl}(\varphi)$, where $\varphi \in \mathsf{cl}(\varphi)$ and if $\varphi_1 \circ \varphi_2 \in \mathsf{cl}(\varphi)$ then $\varphi_1, \varphi_2 \in \mathsf{cl}(\varphi)$, for $\circ \in \{\wedge, \vee\}$. Let Q be any set of states, the

following sets are derived from Q:

$$\begin{array}{lcl}
\|Q\|_{>} & = & \{\|q\|_{\bowtie p} \ : \ q \in Q, \bowtie \in \{\geq, >\}, p \in [0,1] \cap \mathbb{Q}\} \\
\|Q\|^{*} & = & \{\star(t_1, \cdots, t_n) \ : \ n \in \mathbb{N}, \forall i, t_i \in \|Q\|_{>}\} \\
\|Q\|^{\vee} & = & \{\vee(t_1, \cdots, t_n) \ : \ n \in \mathbb{N}, \forall i, t_i \in \|Q\|_{>}\} \\
\|Q\|^{\oplus} & = & \{\oplus(r_1, \cdots, r_n) \ : \ n \in \mathbb{N}, \forall i, r_i \in \|Q\|^{*}\} \\
\|Q\| & = & \|Q\|^{*} \cup \|Q\|^{\vee} \cup \|Q\|^{\oplus}
\end{array}$$

We will call the elements of $\|Q\|_{>}$ as *guarded states* and elements of $\|Q\|^{\oplus}$ as *terms*. Note that the closure of a formula $f \in \mathcal{B}^{+}(\|Q\|)$ only in includes elements from $\|Q\|$.

Example 7.2.1. Consider $\varphi \in \mathcal{B}^{+}(\|Q\|)$, where $\varphi = \oplus(t_1, t_2) \wedge \star(\|q\|_{\geq \frac{1}{3}})$, $t_1 = \star(|q_1|_{\geq \frac{1}{3}}, \|q_2\|_{\geq \frac{4}{5}})$ and $t_2 = \star(\|q\|_{\geq \frac{1}{2}}, \|q_3\|_{\geq \frac{2}{3}})$. Then $\mathsf{cl}(\varphi) = \{\oplus(t_1, t_2), \star(\|q\|_{\geq \frac{1}{3}})\}$.

For brevity, we will write $\star(t : t \in X)$ for $\star(t_1, \cdots, t_n)$ where $X = \{t_1, \cdots, t_n\}$, (similarly for $\varphi \in \|Q\|^{\oplus}$ or $\|Q\|^{\vee}$). For $\varphi = \star(\|q_1\|_{\bowtie_1 p_1}, \cdots, \|q_n\|_{\bowtie_n p_n})$ let the set of guarded states be $\mathsf{gs}(\varphi) = \{q_1, \cdots, q_n\}$ (similarly for $\vee(\|q_1\|_{\bowtie_1 p_1}, \cdots, \|q_n\|_{\bowtie_n p_n})$). We can lift the definition to sets of formulas, $\mathsf{gs}(\Phi) = \bigcup_{\varphi \in \Phi} \mathsf{gs}(\varphi)$. If $\varphi = \oplus(r_1, \cdots, r_n)$ then the set of terms is $\mathsf{gt}(\varphi) = \{r_1, \cdots, r_n\}$. In particular, if $|\mathsf{gt}(\varphi)| = 1$ then $\varphi = \oplus(r)$ is the same as r where $r = \star(t_1, \cdots, t_n)$. Thus, $\|Q\|^{*}$ is a special case of $\|Q\|^{\oplus}$.

Definition 7.2.2. A p-automaton A is a tuple $(Q, \Sigma, \delta, \varphi_{in}, \Omega)$, where Q is a finite set of states, Σ is a finite alphabet (2^{AP}), $\delta : Q \times \Sigma \to \mathcal{B}^{+}(Q) \cup \mathcal{B}^{+}(\|Q\|)$ is the transition function, $\varphi_{in} \in \mathcal{B}^{+}(\|Q\|)$ is an initial condition, and $\Omega \subseteq Q$ is an accepting set of states.

Example 7.2.2. Let the p-automaton $A = (Q, \Sigma, \delta, \varphi, \Omega)$ be defined as follows: $Q = \{q_1, \cdots, q_5\}$, $\Sigma = \{a, b, c\}$, $\varphi = \oplus(\star(\|q_1\|_{\geq \frac{1}{2}}, \|q_5\|_{\geq \frac{1}{2}}), \star(\|q_2\|_{\geq 1}))$, $\delta(q_1, a) = \star\|q_3\|_{\geq 1}$, $\delta(q_2, a) = \star\|q_4\|_{\geq 1}$, $\delta(q_3, b) = \star\|q_3\|_{\geq 1}$, $\delta(q_4, c) = \star\|q_4\|_{\geq 1}$, $\delta(q_5, a) = \varphi$ and $\Omega = Q$.

As a convention, p-automata have states, MCs have locations, and weak stochastic games have configurations. We will make the following simplification, from hereon we assume that for each $\varphi \in \|Q\|^{\oplus}$ and $r, r' \in \mathsf{gt}(\varphi)$, if a state $q \in \mathsf{gs}(r)$ and $q \in \mathsf{gs}(r')$ then $r = r'$. That is, two term r and r' in $\mathsf{gt}(\varphi)$ do not have any common guarded state. A p-automaton $A = (Q, \Sigma, \delta, \varphi_{in}, \Omega)$ defines a labeled directed graph $G_A = (Q', E, E_b, E_u)$ (called the *game graph*):

$$\begin{array}{lcl}
Q' & = & Q \cup cl(\delta(Q, \Sigma)) \\
E & = & \{(\varphi_1 \wedge \varphi_2, \varphi_i) : \varphi_i \in Q' \setminus Q, 1 \leq i \leq 2\} \ \cup \ \{(q, \delta(q, \sigma)) \ : \ q \in Q, \sigma \in \Sigma\} \\
 & & \cup \ \{(\varphi_1 \vee \varphi_2, \varphi_i) : \varphi_i \in Q' \setminus Q, 1 \leq i \leq 2\} \\
E_u & = & \{(\varphi \wedge q, q), (q \wedge \varphi, q), (\varphi \vee q, q), (q \vee \varphi, q) \ : \ \varphi \in Q', q \in Q\} \\
E_b & = & \{(\varphi, q) \ : \varphi \in \|Q\|^{\vee}, q \in \mathsf{gs}(\varphi)\} \ \cup \ \{(\varphi, q) \ : \varphi \in \|Q\|^{\oplus}, q \in \mathsf{gs}(\mathsf{gt}(\varphi))\}
\end{array}$$

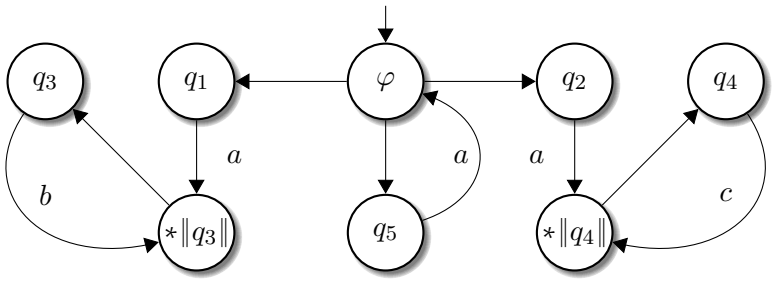


Figure 7.2.1: Game graph G_A without unbounded edges.

where $\delta(Q, \Sigma) = \{\delta(q, \sigma) \ : \ q \in Q \text{ and } \sigma \in \Sigma\} \cup \{\varphi_{in}\}$.

Example 7.2.3. The game graph of p-automaton A defined in example 7.2.2 is shown in Figure 7.2.1 (page 116).

We add markings on the edges to distinguish them. Edges in E_u and E are unmarked and are called *unbounded* and *simple* transitions, respectively. Edge $(\varphi, q) \in E_b$ is called a *bounded* transition and is marked with $\oplus$ if $\varphi \in \|Q\|^{\oplus}$, else it is marked with $\vee$. For example in the Figure 7.2.1, the edges $(\star\|q_3\|_{=1}, q), (\varphi, q_1), (\varphi, q_2), (\varphi, q_5), (\star\|q_4\|_{=1}, q_4)$ are marked $\oplus$.

Two formulas $\varphi, \varphi' \in Q'$ are related as $\varphi \preceq_A \varphi'$ iff there is a path from φ to φ' in G_A. For example in Figure 7.2.1 (page 116), $\varphi \preceq_A q_3$. Let $\equiv_A$ be defined as $\preceq_A \cap \preceq_A^{-1}$. The equivalence class $[\![\varphi]\!]$ of φ with respect to $\equiv_A$ forms a maximal strongly connected component (MSCC) in G_A. In Figure 7.2.1, $\varphi \equiv_A q_5$, but $\varphi \not\equiv_A q_3$. An MSCC is *bounded* iff every edge in an MSCC of G_A, is either in $E \cup E_b$, and an MSCC is *unbounded* iff every edge of the MSCC is in $E \cup E_u$.

Example 7.2.4. Referring to the example 7.2.2 (page 115), we get the following partial order: $\Phi_\setminus \equiv_A = \{[\![q_3]\!], [\![q_4]\!], [\![\varphi]\!], [\![q_1]\!], [\![q_2]\!]\}$, and $\preceq_A$ is the reflexive transitive closure of the relation $\{([\![\varphi]\!], [\![q_1]\!]), ([\![\varphi]\!], [\![q_2]\!]), ([\![q_1]\!], [\![q_3]\!]), ([\![q_2]\!], [\![q_4]\!])\}$.

Definition 7.2.3 (Uniform weak p-automata)**.** A p-automaton A is called *uniform* if:

1. Every MSCC of G_A is either bounded or unbounded.
2. For every bounded MSCC, all marked edges are either marked with $\oplus$ or (exclusively) with $\vee$.
3. The set of equivalence classes $\{[\![\varphi]\!] : \varphi \in Q'\}$ is finite.

A (not necessarily uniform) p-automaton A is called *weak* if for all $q \in Q$, either $[\![q]\!] \cap Q \subseteq F$ or $[\![q]\!] \cap F = \varnothing$.

In the rest of the chapter we will only consider uniform weak p-automata.

7.2.1 Acceptance game of the extended p-automata

Let $A = (Q, \Sigma, \delta, \varphi_{in}, \Omega)$ be a p-automaton and $M = (S, P, L, \mathsf{AP}, s_{in})$ be an Markov chain. The acceptance of M by A depends on the results of a sequence of (stochastic) weak games, called the acceptance games. The acceptance games have configurations from the set $2^S \times \mathsf{cl}(Q, \Sigma)$ or $2^S \times \mathsf{cl}(Q, \Sigma) \times \mathcal{F}^{\odot}$ (where $\odot$ is either $\oplus$ or $\vee$. Meaning of $\mathcal{F}^{\odot}$ will soon become clear). Important point to note is that, each configuration is defined over a set of states of the Markov chain, in contrast to a single state in alternating tree automata (or in [58]).

Let $\Phi = Q \cup \mathsf{cl}(\delta(Q, \Sigma))$ be the set of formulas appearing in the vertices of the game graph G_A. Consider the partial order $(\Phi_{\setminus} \equiv_A, \leq_A)$ defined over the nodes of the game graph of G_A. Consider the following set of sets of states of the Markov chain M.

$$\mathcal{S} = \{T \subseteq S : \forall s, s' \in T,\ L(s) = L(s')\},$$

that is, for each set $T \in \mathcal{S}$, every state in T has the same label. We extend the labeling function as follows; $L : \mathcal{S} \to 2^{\mathsf{AP}}$, where $L(T)$ is $L(s)$ for some $s \in T$. For a formula $\varphi \in \Phi$, $\mathsf{val}(T, \varphi)$ is calculated for each MSCC $[\![\varphi]\!]$ inductively, according to the partial order $\leq_A$. $\mathsf{val}(T, \varphi)$ is the value $\mathsf{val}_0(T, \varphi)$ of Player 0 in the game $\mathsf{G}(M, [\![\varphi]\!]) = (V, E, V_0, V_1, V_p, P, \Omega)$ (defined below). When calculating $\mathsf{val}(T, \varphi)$, the value of $\mathsf{val}(T', \varphi')$ is pre-calculated for every $\varphi' \in [\![\varphi']\!]$, such that $[\![\varphi]\!] \leq_A [\![\varphi']\!]$. Initially, we set $\mathsf{val}(T, \varphi) = \bot$. Depending on the type of MSCC $[\![\varphi]\!]$, we have the following cases:

Case 1. Let $[\![\varphi]\!]$ be a non-trivial *bounded* MSCC where marked edges have marking $\oplus$. For $\varphi = \oplus(r_1, \cdots, r_n)$, let $I_\varphi = \{q : q \in \mathsf{gs}(r), r \in \mathsf{gt}(\varphi)\}$, and $p_{i,q}$ be the probability bound on the state q in the term r_i, i.e., $r_i = *(\|q\|_{\geq p_{i,q}} : q \in \mathsf{gs}(r_i))$. Consider $T \in \mathcal{S}$, and let the label of every state of T be σ. We define the set $R_{T,\varphi}$, which is the set of successor configurations of $\langle T, \varphi \rangle$, and $\mathsf{Val}_{T,\varphi}$, which is the set of possible values of $\mathsf{val}(T, \varphi)$.

$$\begin{aligned} R_{T,\varphi} &= \bigcup_{q \in I_\varphi} \{(T', \varphi') : T' \in \mathsf{succ}(T) \text{ and } \varphi' \in \mathsf{cl}(\delta(q, L(T)))\} \\ \mathsf{Val}_{T,\varphi} &= \{0, 1\} \cup \{\mathsf{val}(T', \varphi') : \langle T', \varphi' \rangle \in R_{T,\varphi}, \mathsf{val}(T', \varphi') \neq \bot\} \end{aligned} \tag{7.2}$$

Where $\mathsf{succ}(T) = \{T' \in \mathcal{S} : T' \subseteq \bigcup_{s \in T} \mathsf{succ}(s)\}$. Thus, $R_{T,\varphi}$ is the set of all Player 0 descendant configurations of $\langle T, \varphi \rangle$ (refer to the table 7.1 for the definition of successors), and $\mathsf{Val}_{T,\varphi}$ is the set of possible values Player 0 can obtain from the configuration $\langle T, \varphi \rangle$. Observe, $R_{T,\varphi}$ is finite and hence

$$
\begin{aligned}
V_0^{T,\varphi} = \quad & \{\langle T,\varphi\rangle\} \ \cup \ \{\langle T',\varphi',v\rangle \in R_{T,\varphi} \times \mathsf{Val}_{T,\varphi} : \bot \neq \mathsf{val}(T',\varphi') < v\} \cup \\
& \{\langle T',\varphi_1 \vee \varphi_2, v\rangle \in R_{T,\varphi} \times \mathsf{Val}_{T,\varphi} : \mathsf{val}(T',\varphi_1 \vee \varphi_2) = \bot\} \\
\\
V_1^{T,\varphi} = \quad & \{\langle T,\varphi,f\rangle : f \in \mathcal{F}^{\oplus}_{T,\varphi}\} \cup \\
& \{\langle T,\varphi,v\rangle \in R_{T,\varphi} \times \mathsf{Val}_{T,\varphi} : \bot \neq \mathsf{val}(T,\varphi) \geq v\} \cup \\
& \{\langle T,\varphi_1 \wedge \varphi_2, v\rangle \in R_{T,\varphi} \times \mathsf{Val}_{T,\varphi} : \mathsf{val}(T,\varphi_1 \wedge \varphi_2) = \bot\} \\
\\
E^{T,\varphi} = \quad & \{(\langle T,\varphi\rangle, \langle T,\varphi,f\rangle) : f \in \mathcal{F}^{\oplus}_{T,\varphi}\} \cup \\
& \{(\langle T',\varphi_1 \circ \varphi_2, v\rangle, \langle T',\varphi_i, v\rangle) : \circ \in \{\wedge,\vee\}, 1 \leq i \leq 2, \\
& \qquad (T',\varphi_1 \circ \varphi_2, v) \in R_{T,\varphi} \times \mathsf{Val}_{T,\varphi}, \mathsf{val}(T',\varphi_1 \circ \varphi_2) = \bot\} \cup \\
& \{(\langle T',\varphi',v\rangle, \langle T',\varphi'\rangle) : T' \in \mathsf{succ}(T), \varphi' \in [\![\varphi]\!], v \in \mathsf{Val}_{T,\varphi}, \mathsf{val}(T,\varphi') = \bot\} \cup \\
& \{(\langle T,\varphi,f\rangle, \langle T',\delta(q,\sigma), f(q,T')\rangle) : T' \subseteq \mathsf{succ}(T), q \in I_\varphi, f(q,T') > 0,\} \cup \\
& \{(\langle T,\varphi,f\rangle, \langle \{s'\},\delta(q,\sigma), f(q,s')\rangle) : s' \in \mathsf{succ}(T), q \in I_\varphi, f(q,T') > 0, \\
& \delta(q,\sigma) \in \mathcal{B}^+(Q)\}
\end{aligned}
$$

Table 7.1: Acceptance game $\mathsf{G}(M,[\![\varphi]\!])$, Case 1. $\sigma = L(T)$, and $f(q,T) = \min_{s \epsilon T} f(q,s)$.

$\mathsf{Val}_{T,\varphi} \subseteq \mathbb{Q}$ is also finite. It is instructive to ask oneself why Player 0 can have only finite number of values at $\langle T,\varphi\rangle$. Observe that the game $\mathsf{G}(M,[\![\varphi]\!])$ has no stochastic branches. Thus any configuration of the game can only take value of some descendant configuration (which is 0 or 1 if all the descendant are in the same MSCC).

Let $\mathcal{F}^{\oplus}_{T,\varphi}$ be a set of functions $I_\varphi \times S \to \mathsf{Val}_{T,\varphi}$ where $f \in \mathcal{F}^{\oplus}_{T,\varphi}$ iff there exists a $\vec{d} \in \mathcal{D}_{\mathsf{gt}(\varphi)}$ and $\{a_{q,s'}\} \in R^{I_\varphi \times S}$ for each $q \in I_\varphi$ and $s' \in \mathsf{succ}(T)$ such that:

$$
\begin{aligned}
& \forall q, \forall s \in T \in I_\varphi : \sum_{s' \epsilon succ(s)} a_{q,s'} f(q,s') P(s,s') \geq p_{i,q} \vec{d}_{r_i}, \\
& \forall s' \in \mathsf{succ}(s) : \sum_{q \in I_\varphi} a_{q,s'} = 1
\end{aligned}
\tag{7.3}
$$

$\vec{d}$ and $\{a_{q,s'}\}$ are called *witness* of the function f. Note that, the set $\mathcal{F}^{\oplus}_{s,\varphi}$ is finite, since for each $f \in \mathcal{F}^{\oplus}_{s,\varphi}$ the domain and the range are finite sets, though its cardinality is exponential in the size of the domain and range. The game $\mathsf{G}(M,[\![\varphi]\!]) = (V, V_0, V_1, V_p, E, P, \Omega)$ is defined as follows:

$$
\begin{aligned}
V_0 &= \bigcup_{T,\varphi' \in [\![\varphi]\!]} V_0^{T,\varphi'} \quad V_1 = \bigcup_{T,\varphi' \in [\![\varphi]\!]} V_1^{T,\varphi'} \quad V_p = \varnothing \\
E &= \bigcup_{T,\varphi' \in [\![\varphi]\!]} E^{T,\varphi'} \quad \Omega = \varnothing \text{ or } V
\end{aligned}
$$

where $V_0^{T,\varphi}, V_1^{T,\varphi}$, and $E^{T,\varphi}$ are defined in Table 7.1, and $\Omega = V$ if for some

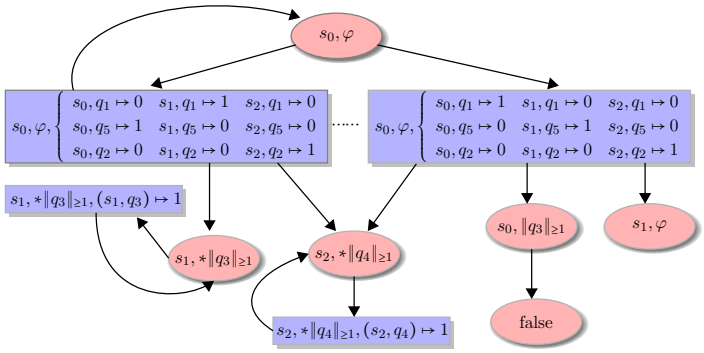


Figure 7.2.2: 2-player game (case 1.) generated by p-automaton A in Example 7.2.2 and MC M in Figure 7.2.3. The oval states are Player 0 states and the rectangle states are Player 1 states. State $\langle \{s_1\}, \varphi \rangle$ belongs to another game and $\mathsf{val}(\{s_1\}, \varphi)$ has been pre-computed.

$q \in [\![\varphi]\!]$, $q \in F$ else $\Omega = \varnothing$.

From the configuration $\langle T, \varphi \rangle$ (where $\varphi \in \|Q\|^{\oplus}$), the game progresses as follows: At $\langle T, \varphi \rangle$, Player 0 selects a function $f \in \mathcal{F}^{\oplus}_{T,\varphi}$ (which implicitly assumes some witnesses $\{a_{q,s'}\}$ and $\vec{d}$), and moves to a Player 1 configuration $\langle T, \varphi, f \rangle$. Player 1 first chooses a $q \in I_{\varphi}$, and then decides to move to a configuration depending on the following cases:

- If $\delta(q, \sigma) \in \mathcal{B}^+(\|Q\|)$. Player 1 can select any subset $T' \in \mathsf{succ}(T)$, and move to a configuration $\langle T', \delta(q, \sigma), v \rangle$, where $v = \min_{s' \in T'} f(s', q)$.
- If $\delta(q, \sigma) \notin \mathcal{B}^+(\|Q\|)$. Player 1 can only select $T' = \{s'\}$, and move to a configuration $\langle T', \delta(q, \sigma), f(s', q) \rangle$.

The intuition is as follows: When $\delta(q, \sigma) \in \mathcal{B}^+(\|Q\|^{\oplus})$ then the descendant configuration belongs to a MSCC that is categorized as *Case 1.* In such a situation, Player 1 has the possibility of selecting a set of states T' (as defined above), and move to $\langle T', \delta(q, \sigma), v \rangle$. When $\delta(q, \sigma) \notin \mathcal{B}^+(\|Q\|)$ then the descendant configuration belongs to a MSCC that is categorized differently (presented below) than case 1. In such a situation Player 1 can only select a singleton set. For convenience, we will write $\langle s, \varphi \rangle$ (or $\langle s, \varphi, v \rangle$) for $\langle \{s\}, \varphi \rangle$ (or $\langle \{s\}, \varphi, v \rangle$, respectively).

Example 7.2.5. Suppose, $\varphi = \oplus(\star\|q_1\|_{\geq \frac{1}{3}}, \star\|q_2\|_{\geq \frac{2}{3}})$., and $\delta(q_1, \sigma) = \varphi' \in \|Q\|^{\oplus}$. Player 0 at configuration $\langle \{s\}, \varphi \rangle$ decided upon $f \in \mathcal{F}^{\oplus}_{\{s\},\varphi}$, where $f(s_1, q_1) = f(s_2, q_2) = 1$, s_1 and s_2 are successors of s. In such a situation,

Player 1 can decide to go to the configuration $\langle\{s_1, s_2\}, \varphi', 1\rangle$. On the other hand, suppose $\delta(q_2, \sigma) = \varphi' \in \mathcal{B}^+(Q)$. Then Player 1 can either move to configurations $\langle\{s_1\}, \varphi', f(s_1, q_2)\rangle$ or $\langle\{s_2\}, \varphi', f(s_2, q_2)\rangle$.

A winning play of the game (see Figure 7.2.2) for Player 0 is determined by the following rules:

1. Consider a finite play that reaches a configuration $\langle T', \varphi', v\rangle$ such that $\mathsf{val}(s', \varphi') \neq \bot$, that is, the value of the configuration $\langle s', \varphi'\rangle$, was already been determined. Recall (Table 7.1, page 118), configuration $\langle T', \varphi', v\rangle$ where $\mathsf{val}(T', \varphi) \neq \bot$ has no successor configurations. Player 0 wins if $v \leq \mathsf{val}(T', \varphi')$ else player 1 wins. This is enforced by making $\langle T', \varphi', v\rangle$ a player 1 configuration if $\bot \neq v \leq \mathsf{val}(T', \varphi)$, and a player 0 configuration if $\bot \neq v > \mathsf{val}(T', \varphi')$.

2. If at $\langle T', \varphi', v\rangle$, $\mathsf{val}(T', \varphi') = \bot$ then the play continues with $\langle T', \varphi'\rangle$. An infinite play is winning if it satisfies the weak acceptance condition Ω. That is, if the second component of an infinite play stays in V (the set of configuration of the game $\mathsf{G}(M, [\![\varphi]\!])$) then player 0 wins if and $V \subseteq \Omega$ else player 1 wins.

Case 2. $[\![\varphi]\!]$ is a non-trivial unbounded MSCC of G_A. The configurations of $\mathsf{G}(M, [\![\varphi]\!])$ are of the type $\langle T, \varphi\rangle$ where $|T| = 1$. So we will write $\langle s, \varphi\rangle$ instead of writing $\langle\{s\}, \varphi\rangle$. Let $[\![\varphi]\!]$ be a nontrivial MSCC such that all the transitions in $[\![\varphi]\!]$ of G_A are in $E_u \cup E$. This gives rise to a weak stochastic game.

$$\begin{array}{ll} V = \{\langle s, \varphi'\rangle \ : \ s \in S \text{ and } \varphi' \in [\![\varphi]\!]\} & V_0 = \{\langle s, \varphi_1 \vee \varphi_2\rangle \in V\} \\ V_p = (S \times Q) \cap V & V_1 = \{\langle s, \varphi_1 \wedge \varphi_2\rangle \in V\} \\ P(\langle s, q\rangle, \langle s', \delta(q, L(s))\rangle) = P(s, s') & \Omega = \varnothing \text{ or } V \end{array}$$

where Ω is V if some q in $[\![\varphi]\!]$ is in F else $\Omega = \varnothing$.

$$\begin{array}{rcl} E & = & \{(\langle s, \varphi_1 \wedge \varphi_2\rangle, \langle s, \varphi_i\rangle) \in V \times V \ : \ 1 \leq i \leq 2\} \cup \\ & & \{(\langle s, \varphi_1 \vee \varphi_2\rangle, \langle s, \varphi_i\rangle) \in V \times V \ : \ 1 \leq i \leq 2\} \cup \\ & & \{(\langle s, q\rangle, \langle s', \delta(q, L(s))\rangle) \in V \times V \ : P(s, s') > 0\} \end{array}$$

By Theorem 2.2.1 (pg 29) a value $\mathsf{val}_0(s, \varphi)$ of any configuration $\langle s, \varphi\rangle \in V$ exists. We set $\mathsf{val}(s, \varphi)$ to this value.

Case 3. Let $[\![\varphi]\!]$ be a nontrivial bounded MSCC with $\vee$ marked edges. We will not make use of this case in this exposition on p-automata. Hence,

$$
\begin{array}{ll}
V_0^{s,\varphi} = & \{\langle s,\varphi\rangle\} \ \cup \ \{\langle s',\varphi',v\rangle \in R_{s,\varphi} \times \mathsf{Val}_{s,\varphi} : \bot \neq \mathsf{val}(s',\varphi') < v\} \ \cup \\
& \{\langle s',\varphi_1 \vee \varphi_2, v\rangle \in R_{s,\varphi} \times \mathsf{Val}_{s,\varphi} : \mathsf{val}(s',\varphi_1 \vee \varphi_2) = \bot\} \\
\\
V_1^{s,\varphi} = & \{\langle s,\varphi,f\rangle : f \in \mathcal{F}^{\vee}_{s,\varphi}\} \ \cup \\
& \{\langle s,\varphi,v\rangle \in R_{s,\varphi} \times \mathsf{Val}_{s,\varphi} : \bot \neq \mathsf{val}(s,\varphi) \geq v\} \ \cup \\
& \{\langle s,\varphi_1 \wedge \varphi_2, v\rangle \in R_{s,\varphi} \times \mathsf{Val}_{s,\varphi} : \mathsf{val}(s,\varphi_1 \wedge \varphi_2) = \bot\} \\
\\
E^{s,\varphi} = & \{(\langle s,\varphi\rangle, \langle s,\varphi,f\rangle) : f \in \mathcal{F}^{\vee}_{s,\varphi}\} \ \cup \\
& \{(\langle s',\varphi_1 \circ \varphi_2, v\rangle, \langle s',\varphi_i,v\rangle) : \circ \in \{\wedge,\vee\}, 1 \leq i \leq 2, \\
& \qquad (s',\varphi_1 \circ \varphi_2, v) \in R_{s,\varphi} \times \mathsf{Val}_{s,\varphi}, \mathsf{val}(s',\varphi_1 \circ \varphi_2) = \bot\} \ \cup \\
& \{(\langle s',\varphi',v\rangle, \langle s',\varphi'\rangle) : s' \in \mathsf{succ}(s), \varphi' \in [\![\varphi]\!], v \in \mathsf{Val}_{s,\varphi}, \mathsf{val}(s,\varphi') = \bot\} \ \cup \\
& \{(\langle s,\varphi,f\rangle, \langle s',\delta(q,\sigma), f(q,s')\rangle) : s' \subseteq \mathsf{succ}(s), q \in I_\varphi, f(q,s') > 0,\} \ \cup \\
& \{(\langle s,\varphi,f\rangle, \langle \{s'\},\delta(q,\sigma), f(q,s')\rangle) : s' \in \mathsf{succ}(s), q \in I_\varphi, f(q,s') > 0, \\
& \delta(q,\sigma) \in \mathcal{B}^+(Q)\}
\end{array}
$$

Table 7.2: Acceptance game $\mathsf{G}(M, [\![\varphi]\!])$, Case 2. $\sigma = L(s)$.

the semantics of the acceptance game is kept identical to [58]. Let $\varphi = \vee(\|q_1\|_{\bowtie_1 p_1}, \cdots, \|q_n\|_{\bowtie_n p_n})$. The sets $R_{s,\varphi}, \mathsf{Val}_{s,\varphi}$ are defined as follows:

$$
\begin{aligned}
R_{s,\varphi} &= \bigcup_{q \in I_\varphi} \{\langle s',\varphi'\rangle : s' \in \mathsf{succ}(s) \text{ and } \varphi' \in \mathsf{cl}(\delta(q, L(s)))\} \\
\mathsf{Val}_{s,\varphi} &= \{0,1\} \cup \{\mathsf{val}(s',\varphi') : \langle s',\varphi'\rangle \in R_{s,\varphi}, \mathsf{val}(s',\varphi') \neq \bot\}
\end{aligned}
\tag{7.4}
$$

$$
\begin{aligned}
V_0 &= \bigcup_{T,\varphi' \in [\![\varphi]\!]} V_0^{T,\varphi'} \quad V_1 = \bigcup_{T,\varphi' \in [\![\varphi]\!]} V_1^{T,\varphi'} \quad V_p = \varnothing \\
E &= \bigcup_{T,\varphi' \in [\![\varphi]\!]} E^{T,\varphi'} \quad \Omega = \varnothing \text{ or } V
\end{aligned}
$$

where $V_0^{T,\varphi}, V_1^{T,\varphi}$, and $E^{T,\varphi}$ are defined in Table 7.2, and $\Omega = V$ if for some $q \in [\![\varphi]\!]$, $q \in F$ else $\Omega = \varnothing$. Observe that the configurations of the game are of the set $S \times \delta(Q,\Sigma) \cup S \times \delta(Q,\Sigma) \times \mathcal{F}^{\vee}$ (as contrast to case 1.)

Another notable difference is the set of functions $\mathcal{F}^{\vee}_{s,\varphi} = I_\varphi \times \mathsf{succ}(s) \to \mathsf{Val}_{s,\varphi}$ (rather than $\mathcal{F}^{\oplus}_{s,\varphi}$) (See Table 7.2). A function $f \in \mathcal{F}^{\vee}_{s,\varphi}$ if there exists $a \in \mathbb{R}^{I_\varphi} \times \mathbb{R}^S$ such that:

- there is a $q \in I_\varphi$ with $\sum_{s' \in \mathsf{succ}(s)} a_{q,s} f(q,s) P(s,s') \geq p_i$ or,
- there is a $s \in \mathsf{succ}(s)$ with $\sum_{q \in I_\varphi} a_{q,s'} \neq 1$.

The winning condition is same as case 1. As mentioned before, we will not need the terms in $\|Q\|^{\vee}$ and present it here only for completeness.

Case 4. Let $[\![\varphi]\!]$ be a trivial MSCC. It is handled as one of the above cases. The value of the configurations $\mathsf{val}(T, \varphi)$ is obtained from the values of the successor configurations, which have already been calculated in $\mathsf{G}(M, [\![\varphi']\!])$.

Definition 7.2.4. *A Markov chain M is accepted by a p-automaton A, iff $\mathsf{val}(\{s_{in}\}, \varphi_{in}) = 1$. The language of A, $\mathcal{L}(A) = \{M : A \text{ accepts } M\}$.*

The p-automata defined here has two notable difference than p-automata in [58]. First is the syntactic difference due to the presence of construct $\oplus(\varphi_1, \cdots, \varphi_n)$. Second is the semantic difference were the configurations of the acceptance game is defined over sets of states of the Markov chains for a bounded MSCC (case 1.). AS we will see presently, this is crucial for proving correctness of Theorem 7.2.3. For unbounded MSCC the description of the acceptance game is same as the original definition.

The number of configurations of the weak game $\mathsf{G}(M, [\![\varphi]\!])$ when φ is a bounded MSCC is exponential in the size of $[\![\varphi]\!]$ and the Markov chain(case 1.). The exponential blowup is due to the different function $f \in \mathcal{F}^{\oplus}_{s,\varphi}$ and the cardinality of $\mathcal{S}$. For the other cases the size of the game is polynomial in the size of the automaton and the Markov chain. Since, weak games can be solved in polynomial time in the size of the game and the weak stochastic game can be solved in $\mathsf{NP} \cap \text{co-}\mathsf{NP}$, the problem whether a finite Markov chain is accepted by a p-automaton can be decided in exponential time.

7.2.2 Properties of p-automata

Closure Properties: We will first show that the language of a p-automaton is closed under probabilistic bi-simulation.

Proposition 7.2.1. *For a p-automaton A and MCs M_1 and M_2 with $M_1 \sim M_2$, $M_1 \in \mathcal{L}(A)$ iff $M_2 \in \mathcal{L}(A)$.*

Proof. Let $M_1 = (S_1, P, L, s_{1,in})$ and $M_2 = (S_2, P, L, s_{2,in})$, with S_1 disjoint from S_2, hence we use the same function P and L for both MCs with impunity. Let A be $(Q, \Sigma, \delta, \varphi_{in}, \Omega)$, G_1 and G_2 be the acceptance game for MCs M_1 and M_2, respectively.

Consider any two configurations $\langle T_1, \varphi\rangle$ and $\langle T_2, \varphi\rangle$ in G_1 and G_2, respectively, such that $\forall s_1 \in T, s_2 \in T_2 : s_1 \sim s_2$. We show that $\mathsf{val}(T_1, \varphi) = \mathsf{val}(T_2, \varphi)$. Equivalently, we construct a wining strategy π_2 for Player 0 in G_2 from the winning strategy π_1 of Player 0 in G_1. By symmetry of the argument (presented below), it also follows that we can construct a wining strategy for Player 0 in G_1 from the winning strategy of Player 0 in G_2.

Consider the case when $[\![\varphi]\!]$ is unbounded MSCC. $\mathsf{G}_i(M_i, [\![\varphi]\!])$ is a stochastic weak game (for $i \in \{1, 2\}$). We start from the configurations $c_1 = \langle s_1, \varphi\rangle$

and $c_2 = \langle s_2, \varphi \rangle$ where $s_1 \in S_1$ and $s_2 \in S_2$ and $s_1 \sim s_2$. The claim is, at each step of any play of the games, we move to configurations $\langle s_1', \varphi' \rangle$ and $\langle s_2', \varphi' \rangle$ in G_1 and G_2 (according to strategy π_1 and π_2), respectively, where $s_1' \sim s_2'$.

When φ is of the form $\varphi_1 \wedge \varphi_2$, c_1 and c_2 are Player 1 configurations. If Player 1 chooses (s_2, φ_i) in G_2 then we make Player 1 in G_1 choose (s_1, φ_i) for $i \in \{1, 2\}$. When φ is of the form $\varphi_1 \vee \varphi_2$, c_1 and c_2 are Player 0 configuration, Player 0 in G_2 follows the choice of Player 0 in G_1, i.e., if Player 0 chose $\langle s_1, \varphi_i \rangle$ in G_1 then Player 0 in G_2 chooses $\langle s_2, \varphi_i \rangle$ in G_2 (for $i \in \{1, 2\}$). For $\varphi = q \in Q$, the play is resolved by a probabilistic choice. We know that $P(s_1, C_1) = P(s_2, C_2)$ where $C_i \subseteq S_i$ (for $i \in \{1, 2\}$) and $C_1 \cup C_2$ is an equivalence class of $\sim$. Thus, for any play that ends in $\langle s_1', \delta(q, \sigma) \rangle$ in G_1, there is a corresponding play in G_2 that ends in $\langle s_2', \delta(q, \sigma) \rangle$, and we have $s_1' \sim s_2'$ where $\sigma = L(s_1) = L(s_2)$. Hence the set of plays that are winning in G_1 have the same probability measure as the set of corresponding play in G_2. Consequently, $\mathsf{val}(s_1, \varphi) = \mathsf{val}(s_2, \varphi)$.

Let $[\![\varphi]\!]$ be a *bounded* MSCC of G_A where the only marked edges have $\oplus$ as markings. Consider $T_1 \subseteq S_1$ and $T_2 \subseteq S_2$, such that for each $s_1 \in T_1$, there exist $s_2 \in T_2$ such that $s_1 \sim s_2$ and vice-versa. We show that if $\mathsf{val}(T_1, \varphi) = 1$ then $\mathsf{val}(T_2, \varphi) = 1$. Disjunction and conjunctions are handled as before. Let $\varphi \in \|Q\|^{\oplus}$, and consider $s_1 \in T_1$ and $s_2 \in T_2$, such that $s_1 \sim s_2$. We have a function $f_1 \in \mathcal{F}^{\oplus}_{s_1,\varphi}$ with witness $\vec{d}$ and $\{a_{q,s'}\}_{q \in I_\varphi, s' \in \mathsf{succ}(s)}$ for the play in G_1. Define $f_2 : I_\varphi \times \mathsf{succ}(s_2) \to [0, 1]$ with $f_2(q, s_2') = f_1(q, s_1')$ for $s_j' \in \mathsf{succ}(s_j)$ $(j \in \{1, 2\})$ for some $s_1' \sim s_2'$. It remains to show that $f_2 \in \mathcal{F}^{\oplus}_{s_2,\varphi}$. That is, we need to find suitable witnesses $\vec{d}'$ and $\{a'_{q,s_2'}\}_{q \in I_\varphi, s_2' \in \mathsf{succ}(s_2)}$ for f_2, that satisfies the equation 7.3 (page 118). Let $\vec{d}' = \vec{d}$ and choose $\{a'_{q,s'}\}$ such that for each $q \in I_\varphi$, whenever $f(q, s_2') = f(q, s_1')$, $a'_{q,s_2'} = a_{q,s_1'}$. This implies that for each equivalence class C:

$$\sum_{s_1' \in C} a_{q,s_1'} P(s_1, s_1') \;=\; \sum_{s_2' \in C} a'_{q,s_2'} P(s_2, s_2') \tag{7.5}$$

There could be many possible solution for $\{a'_{q,s'}\}$, we need to find one solution such that $f_2 \in \mathcal{F}^{\oplus}_{s,\varphi}$. For each $q \in I_\varphi$:

$$\sum_{s_2' \in \mathsf{succ}(s_2)} a'_{q,s_2'} P(s_2, s_2') f_2(q, s_2') \;=\; \sum_{C \in S_2 \cup S_1 \setminus \sim} \left(\sum_{s_2' \in C} a'_{q,s_2'} P(s_2, s_2') f_2(q, s_2') \right)$$

$$= \sum_{C \in S_2 \cup S_1 \setminus \sim} \left(\sum_{s_1' \in C} a_{q,s_1'} P(s_1, s_1') f_1(q, s_1') \right) \;=\; \sum_{s_1' \in \mathsf{succ}(s_1)} a'_{q,s_1'} P(s_1, s_1') f_1(q, s_1') = \; p_{i,q} \vec{d}'(r_i).$$

Thus any value of $\{a'_{q,s_2'}\}$ satisfying the constraint (7.5) also satisfies the first condition of equation (7.3). If now Player 1 in G_2 chooses $\langle T_2', \delta(q, \sigma), v \rangle$,

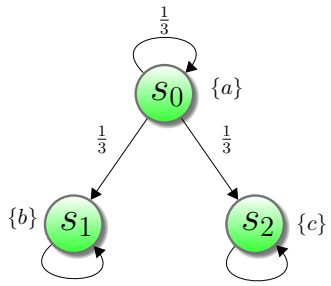


Figure 7.2.3: A Markov chain M, with $S = \{s_0, s_1, s_2\}$ and $P(s_0, s_0) = P(s_0, s_1) = P(s_0, s_2) = \frac{1}{3}$, $P(s_1, s_1) = P(s_2, s_2) = 1$.

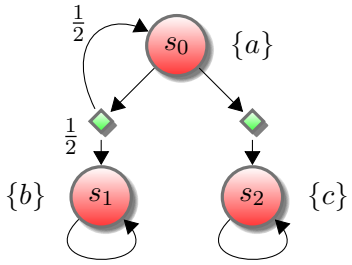


Figure 7.2.4: A Markov decision process D. $\Delta(s_0) = \{\mu_1, \mu_2\}$, where $\mu_1(s_0) = \frac{1}{2}, \mu_1(s_1) = \frac{1}{2}$ and $\mu_2(s_2) = 1$.

Player 1 in G_1 is made to choose T_1, such that for each s_2' in T_2' there exist $s_1' \in T_1$, such that $s_1' \sim s_2'$. □

Proposition 7.2.2. *Language of p-automata are closed under union and intersection.*

Proof. Follows trivially from the construction of p-automata. □

Embedding of MDPs:

Definition 7.2.5 (p-automata for an MDP)**.** The p-automaton $A_D = (Q, \Sigma, \delta, \varphi_{in}, \Omega)$ is defined as follows: [1]

$$
\begin{aligned}
&Q = S \times S \quad ; \quad \Omega = Q \quad ; \\
&\delta((s,s'), L(s)) = \varphi_{s'} \text{ and } \delta((s,s'), \sigma) = \mathsf{false} \qquad \text{if } \sigma \neq L(s) \\
&\varphi_{in} = \oplus(r_i \mid \mu_i \in \Delta(s_{in}), r_i = *(\|(s_{in}, \mu_i, s')\|_{\geq \mu_i(s')} \mid \mu_i(s') > 0)) \\
&\varphi_s = \oplus(r_i \mid \mu_i \in \Delta(s) \text{ and } r_i = *(\|(s, \mu_i, s')\|_{\geq \mu_i(s')} \mid \mu_i(s') > 0))
\end{aligned}
$$

Example 7.2.6. The MDP in the Figure 7.2.4 is embedded in the automaton A defined in the Example 7.2.2 and the MC of Figure 7.2.3 is induced by a scheduler of the MDP and is accepted by A.

Theorem 7.2.3. *Let D be an MDP and A_D be its p-automaton.*

1. *For every scheduler η, the induced Markov chain $D_\eta \in \mathcal{L}(A_D)$.*
2. *For every MC $M \in \mathcal{L}(A_D)$ there exists a $\eta \in \mathsf{HR}(D)$ such that $M \sim D_\eta$.*

[1] It could be the case that there is some state $q \in Q$ which a guarded state of more than one term of a formula $\varphi \in \|Q\|^{\oplus}$. This can be resolved by renaming and introducing new states.

Proof. **1**.) Let the MDP D be $(S, \Delta, \Sigma, L, s_{in})$. We will first show that for any scheduler $\eta \in \mathsf{HR}(D)$, $D_\eta = (S^+, \Sigma, P', L, s_{in})$ is in $\mathcal{L}(A_D)$. The acceptance game of D_η is a weak game and any configuration of the game has value either 0 or 1. We fix the following strategy for Player 0. At $\langle\{w\}, \varphi_{w\downarrow}\rangle$ (where w is a state of D_η and a finite path of D), Player 0 chooses a function $f : I_{\varphi_{w\downarrow}.w} \times \mathsf{succ}(w) \to \{0,1\}$ with witnesses $\{a_{q,w\cdot s'}\}_{q\in I_{w,\varphi_{w\downarrow}}, w\cdot s'\in \mathsf{succ}(w)}$ and $\vec{d}$ such that, $f(q, w\cdot s') = 1$, $a_{q,w\cdot s'} = 1$ iff $q = (w\downarrow, s')$ and $\vec{d}(i) = \eta(\mu_i)$, for $\mu_i \in \Delta(w\downarrow)$. It follows immediately, that equation (7.3) is satisfied. That is, for all $q = (w\downarrow, s') \in I_{\varphi_{w\downarrow}}$: $a_{q,w\cdot s'} f(q, w\cdot s') P'(w, w\cdot s') = \mu_i(s')\eta(\mu_i)$. Note that there is exactly one successor of w, say $u = w\cdot s$, such that $f(u, q) = 1$. Thus Player 1 can only move to configurations of the type $\langle\{w\cdot s'\}, \varphi_{s'}, 1\rangle$, and Player 0 can always continue from $\langle\{w\}, \varphi_{w\downarrow}\rangle$ to $\langle\{w\cdot s'\}, \varphi_{s'}\rangle$. Since the set of accepting configurations Ω is Q, any finite play according to the chosen strategy can be extended indefinitely, and hence is winning for Player 0.

2.) Suppose MC M' is not accepted by A. This implies, that a finite path $\langle T_0, \varphi_{s_0}\rangle, \cdots, \langle T_n, \varphi_{s_n}\rangle$ is winning for Player 1, with $T_0 = \{t_0\}$ which is the initial state of M'. Since every infinite path is winning for Player 0. Hence at $\langle T_n, \varphi_{s_n}\rangle$ it is not the case that Player 0 can find witnesses
bd and $\{a_{q,s'}\}$ such that,

1. $\forall r_i \in \mathsf{gt}(\varphi_{s_n})\ \forall q \in {}_>(r_i)\ \forall s \in T_n : \sum_{s'\in \mathsf{succ}(s)} a_{q,s'} P(s, s') f(q, s') = p_{i,q}\vec{d}_i$

2. for each $q \in {}_>(\mathsf{gt}(\varphi_{s_n}))$ and any set $T' \subseteq \mathsf{succ}(T_n)$, where $\forall s' \in T' : f(q, s') = 1$, $\langle T', \varphi_{s_n}\rangle$ is winning for Player 0.

Take any other (arbitrary) play $\langle T'_0, \varphi_{s_0}\rangle, \cdots, \langle T'_n, \varphi_{s_n}\rangle$ (with $T_0 = T'_0 = \{t_0\}$). Then $\langle T_0 \cup T'_0, \varphi_{s_0}\rangle, \cdots, \langle T_n \cup T'_n, \varphi_{s_n}\rangle$ is also winning for Player 1. So Player 1 plays rationally, she will choose T'_i as large as possible.

Let $M = (T, \Sigma, P, L, t_0)$, and $M \in \mathcal{L}(A_D)$. The value of configuration $\langle\{t_0\}, \varphi_{s_0}\rangle$ is 1, and assume Player 1 plays optimally, i.e., she chooses a set as large as possible. We will construct a map $\eta^\star \subseteq (S^+ \times \mathcal{D}_{\mathcal{D}_s})$. For any possible finite run, $\rho_n = \langle T_0, \varphi_{s_0}\rangle, \cdots, \langle T_n, \varphi_{s_n}\rangle$, with $T_0 = \{s_0\}$, $(s_0, \cdots, s_n, \vec{d}) \in \eta^\star$, where $\vec{d}$ is the distribution chosen by Player 0 at $\langle T_n, \varphi_{s_n}\rangle$. Since, Player 1 plays optimally, it cannot be the case that two distinct play $\rho_n = \langle T_0, \varphi_{s_0}\rangle, \cdots, \langle T_n, \varphi_{s_n}\rangle$ and $\rho'_n = \langle T'_0, \varphi_{s_0}\rangle, \cdots, \langle T'_n, \varphi_{s_n}\rangle$ exists. Thus, we see that $\eta^\star \in \mathsf{HR}(D)$.

Now consider an unrolling of M (recall this gives us a probabilistic tree pg 79). Thus, states of M are subsets of T^+. It suffices to show a bisimulation relation between, $D_{\eta^\star}$ and the unrolled M. Let $R \subseteq (T^+ \cup S^+) \times (T^+ \cup S^+)$ be

the smallest transitive, reflexive and symmetric relation with the following property:

- $t_0 R s_0$.
- For each play $\rho_n = \langle T_0, \varphi_{s_0}\rangle, \cdots, \langle T_n, \varphi_{s_n}\rangle\langle T_{n+1}, \varphi_s\rangle$, all $t_{n+1} \in T_{n+1}$, vRw, where $v = t_0, \cdots, t_{n+1}$, such that $t_{i+1} \in \mathsf{succ}(t_i)$ for all $0 \geq i \leq n$ and $w = s_0, \cdots, s_n$.

We will show that R is a bi-simulation relation.

- If uRw then $L(u) = L'(w)$. If $L(u) \neq L'(w)$ then $\langle T, \varphi_{w\downarrow}\rangle$ cannot be winning for Player 0, where $u\downarrow \in T$. [Recall for a sequence $\rho = (a_0, \cdots, a_n)$, $\rho\downarrow$ is the last element a_n.]
- Let $t = u\downarrow$. For each $q \in I_{\varphi_{w\downarrow}}$, we know, $\sum_{t' \in \mathsf{succ}(t)} P(t,t') a_{q,t'} f(q,t') = p_{q,i}\vec{d_i}$. Let C be the set of successors of u and w such that $C \in (T^+ \cup S^+) \setminus R$. From this we can deduce,

$$\sum_{u \cdot t' \in C} P(u, u\cdot t') = \sum_{w \cdot s' \in C} P'(w, w\cdot s').$$

where P' is the probability distribution function of the Markov chain $D_{\eta^\star}$. For each equivalence class C, let $Q(C) = \{q = (w, \mu, w\cdot s') : w\cdot s' \in C\}$. Now consider any arbitrary $q \in Q(C)$ where $q = (w, \mu, w\cdot s')$. We know for each $q \in I_{\varphi_{w\downarrow}}$:

$$\sum_{u' \in \mathsf{succ}(u)} P(u,u') a_{q,u'} f(q,u') = P'(w, w\cdot s')$$

If $u \notin C$ then $f(u', q) = 0$. Thus, we can be rewrite as:

$$\sum_{u' \in \mathsf{succ}(u) : u \in C} P(u,u') a_{q,u'} f(q,u') = P'(w, w\cdot s')$$

Summing over all $q \in Q(C)$ gives us:

$$\sum_{q \in Q(C)} \sum_{u' \in \mathsf{succ}(u) : u \in C} P(u,u') a_{q,u'} f(q,u') = \sum_{q \in Q(C)} P'(w, w\cdot s')$$

Changing the order of summation:

$$\sum_{u' \in \mathsf{succ}(u) : u \in C} P(u,u') \left(\sum_{q \in Q(C)} a_{q,u'} f(q,u') \right) = \sum_{u' \in \mathsf{succ}(u) : u \in C} P(u,u')$$

As $f(q,u') = 0$ for $q \notin C$, we can deduce:

$$\sum_{u' \in \mathsf{succ}(u) : u \in C} P(u,u') = \sum_{q \in Q(C)} P'(w, w\cdot s').$$

centering

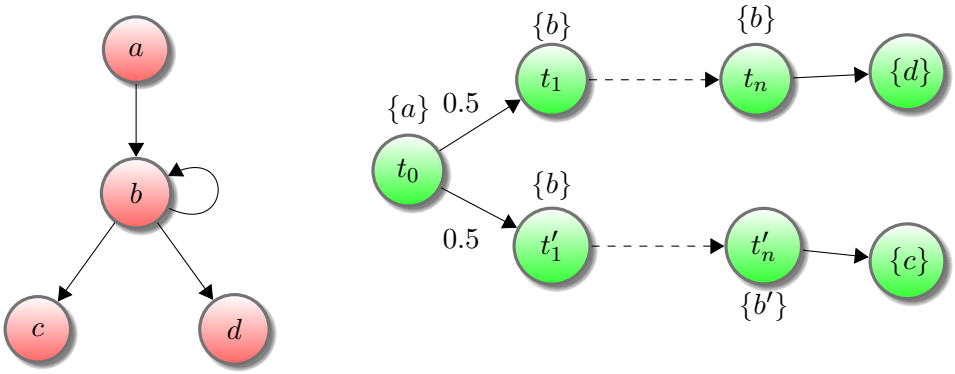


Figure 7.2.5: The MDP (left) does not induces the Markov chain (right).

Thus, R is a bi-simulation relation , and $M \sim D_\eta$. □

The embedding of MDP relies on the construct $\varphi \in \|Q\|^{\oplus}$. Consider the MDP in Figure 7.2.3. At the state s_0 there are two choices of distribution. If we limit the definition of the p-automata to [58] then we have only disjunction (or conjunction) to define the non-determinism at the state s_0 and we cannot accept the MC in Figure 7.2.3.

We also keep track of the subset of states T that were *induced* by the same $q \in I_\varphi$. This is the crucial feature of the extended p-automata. Refer to the Figure 7.2.5, observe that, at t_n and t'_n the formula φ_b is satisfied. Player 0 can select with probability 1 the d branch for t_n and with probability 1 the c branch for t'_n. But the resulting Markov chain will not be induced by any scheduler of MDP. Thus, we need to remember that states t_1 and t'_1 were induced by the same distribution. So, at $\langle\{t_0\}, \varphi_a\rangle$ Player 1 can change the current configuration of the acceptance game to $\langle\{t_1, t'_1\}, \varphi_b\rangle$. This will eventually lead to a configuration $\langle\{t_n, t'_n\}, \varphi_b\rangle$. At this point Player 0 cannot find a distribution such that both t_n and t'_n satisfy φ_b and looses.

Embedding of PCTL formulas: PCTL can be embedded into p-automata. That is, given a PCTL formula f over AP, there exists a p-automaton A, such that for any Markov chain M, f satisfies M if and only if M is accepted by A. The algorithm for this translation from PCTL to p-automata was provided in [58]. In this section we recall the construction.

Definition 7.2.6. From PCTL formula φ over APin *negative normal form*, we construct a p-automaton $A_\varphi = (Q, \Sigma, \delta, \varphi_{in}, F)$ as follows:

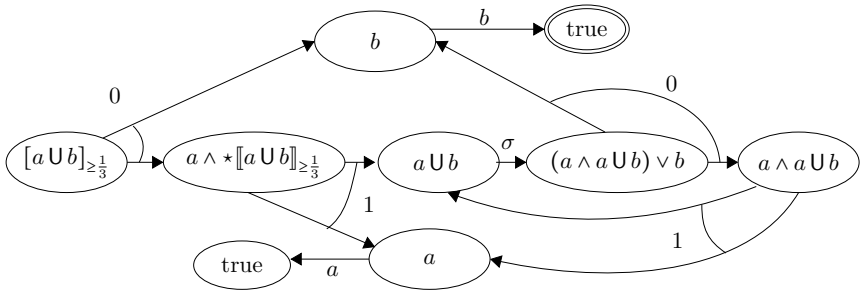


Figure 7.2.6: The game graph of p-automaton embedding $[a\,\mathsf{U}\,b]_{>\frac{1}{3}}$.

1. Let $\mathsf{cl}_p(\varphi)$ be the set of path sub-formula of φ. $Q = \mathsf{cl}_p(\varphi) \cup \mathsf{AP}$.
2. F consists of AP, their negations, and all $\psi \in \mathsf{cl}_p(\varphi)$ *not* of the form $\varphi_1\,\mathsf{U}\,\varphi_2$.
3. The alphabet $\Sigma = 2^{\mathsf{AP}}$.
4. The definition of the translation relation δ needs a little preparation. We define the function τ:

$$\begin{array}{lcl}
\tau(a) & = & a \text{ where } a \in \mathsf{AP}, \\
\tau(\sim a) & = & \sim a, \\
\tau(\psi_1 \circ \psi_2) & = & \tau(\psi_1) \circ \tau(\psi_2) \text{ where } \circ \in \{\wedge, \vee\}, \\
\tau([\mathsf{X}\psi]_{\bowtie p}) & = & *(\|\mathsf{X}\psi\|_{\bowtie p}) \\
\tau([\psi_1\,\mathsf{U}\,\psi_2]_{\bowtie p}) & = & (\ \tau(\psi_1) \wedge *(\|\psi_1\,\mathsf{U}\,\psi_2\|_{\bowtie p})\ ) \vee \tau(\psi_2) \\
\tau([\psi_1\,\mathsf{W}\,\psi_2]_{\bowtie p}) & = & (\ \tau(\psi_1) \wedge *(\|\psi_1\,\mathsf{W}\,\psi_2\|_{\bowtie p})\ ) \vee \tau(\psi_2)
\end{array}$$

   δ is defined as follows:

$$\begin{array}{lcl}
\delta(a, \sigma) & = & (a \in \sigma) \text{ where } a \in \mathsf{AP} \text{ and } \sigma \in \Sigma. \\
\delta(\sim a, \sigma) & = & (a \notin \Sigma) \\
\delta(\mathsf{X}\psi, \sigma) & = & \tau(\psi) \text{ where } \tau(\psi) \text{ is defined above.} \\
\delta(\psi_1\,\mathsf{U}\,\psi_2, \sigma) & = & (\ \tau(\psi_1) \wedge \psi_1\,\mathsf{U}\,\psi_2\ ) \vee \tau(\psi_2) \\
\delta(\psi_1\,\mathsf{W}\,\psi_2, \sigma) & = & (\ \tau(\psi_1) \wedge \psi_1\,\mathsf{W}\,\psi_2\ ) \vee \tau(\psi_2)
\end{array}$$

5. The initial formula $\varphi_{in} = \tau(\varphi)$.

Example 7.2.7. Figure 7.2.6 shows the game graph of the p-automaton that encodes the PCTLformula $[a\,\mathsf{U}\,b]_{>\frac{1}{3}}$.

Observe that in the game graph of any p-automaton obtained from a PCTL formula by the definition 7.2.6, all bounded MSCCs are trivial. Thus the acceptance game of such a p-automaton is same as in [58]. The following theorem establishes the correctness of the translation.

Theorem 7.2.4 ([58]). *For any MC M and PCTL formula φ, $M \vDash \varphi$ if and only if $M \in \mathcal{L}(A_\varphi)$.*

7.2.3 Simulation game

In the previous section we have seen how Markov chain induced by an MDP can be captured by a p-automaton. To accomplish this, acceptance game of the extended p-automata have configurations which keep track of sets of states of the input Markov chain. This increase in power comes at the cost that emptiness and hence language inclusion cannot be decidable. This follows from [12] where the scheduler synthesis problem (recall pg 75) for PCTL winning condition was shown to be undecidable.

In this section we present a simulation relation that resembles Roberto Segala's [88] simulation relation on probabilistic automata. We will consider *simulation game* $\mathsf{G}_{\preceq}$ for two p-automata where game graphs of the p-automata do not have unbounded MSCCs marked with $\vee$ marked edges. The technique reuses the method presented in [58], the only notable difference being the formulas in $\|Q\|^{\oplus}$.

Let G_1 and G_2 be the game graphs of the p-automata $A_1 = (Q, \Sigma, \delta, \varphi_{in}, F)$ and $A_2 = (U, \Sigma, \delta, \psi_{in}, F)$ (Q is disjoint from U), and $\preceq_1$ and $\preceq_2$ be the partial orders for their respective game graphs. The partial order $\preceq \subseteq Q' \times U'$, where $Q' = Q \cup \mathsf{cl}(Q, \Sigma)$ and $U' = U \cup \mathsf{cl}(U, \Sigma)$, is defined as the lexicographical ordering on $\preceq_1$ and $\preceq_2$. Formally, $(\varphi, \psi) \preceq (\varphi', \psi')$ if either $\varphi \preceq_1 \varphi'$ or $\varphi = \varphi'$ and $\psi \preceq_2 \psi'$. The equivalence relation $\preceq \cap \preceq^{-1}$ is denoted by $\equiv$. A formula φ is said to be simulated by ψ, if the value of the configuration $\langle \varphi, \psi \rangle$ in the game $\mathsf{G}_{\preceq}(\llbracket \varphi \rrbracket, \llbracket \psi \rrbracket)$ (defined below) is 1.

Similar to the acceptance games, the values of configurations in $(\llbracket \varphi \rrbracket, \llbracket \psi \rrbracket)$ are calculated after the value of all configurations in $(\llbracket \varphi' \rrbracket, \llbracket \psi' \rrbracket)$ has already been calculated, where $(\llbracket \varphi \rrbracket, \llbracket \psi \rrbracket) \preceq (\llbracket \varphi' \rrbracket, \llbracket \psi' \rrbracket)$. We have the following cases:

Case 1. Let $\llbracket \varphi \rrbracket$ and $\llbracket \psi \rrbracket$ are non-trivial unbounded and bounded MSCCs, respectively. We set $\mathsf{val}(\varphi, \psi) = 0$, formulas from unbounded MSCCs cannot be simulated by formulas from bounded MSCCs.

Case 2. Let $\llbracket \varphi \rrbracket$ and $\llbracket \psi \rrbracket$ be unbounded MSCCs. $\mathsf{G}_{\preceq}(\llbracket \varphi \rrbracket, \llbracket \psi \rrbracket)$ is defined in Table 7.3. The winning condition $\Omega = V$ if $\llbracket \varphi \rrbracket \cap Q \subseteq F$ implies $\llbracket \psi \rrbracket \cap U \subseteq F$ else $\Omega = \varnothing$.

Case 3. Let $\llbracket \varphi \rrbracket$ and $\llbracket \psi \rrbracket$ be bounded MSCCs. Suppose $\varphi \in \|Q\|^{\oplus}$ and

$$
\begin{array}{lll}
V = & \{\langle \varphi', \psi' \rangle \ : \ \varphi \preceq_1 \varphi', \psi \preceq_2 \psi'\} & V_p \ = \ \{\} \\
V_0 = & \cup \ \{\langle \varphi_1 \wedge \varphi_2, \psi_1 \vee \psi_2 \rangle \ : \ \varphi_i \in [\![\varphi]\!], \psi_i \in [\![\psi]\!]\} & \\
& \cup \ \{\langle \varphi_1 \wedge \varphi_2, u' \rangle \ : \ u' \in U, \varphi_i \in [\![\varphi]\!]\} \ \cup \ \{\langle q', \psi_1 \vee \psi_2 \rangle \ : \ q' \in Q, \psi_i \in [\![\psi]\!]\} & \\
& & \\
V_1 = & \{\langle q', u' \rangle \ : \ q; \in Q, u' \in U\} \ \cup \ \{\langle \varphi_1 \vee \varphi_2, \psi' \rangle \ : \ \varphi_i \in [\![\varphi]\!], \psi' \in [\![\psi]\!]\} & \\
& \cup \ \{\langle \varphi', \psi_1 \wedge \psi_2 \rangle \ : \ \varphi' \in [\![\varphi]\!], \psi_i \in [\![\psi]\!]\} & \\
& & \\
E = & \{(\langle \varphi_1 \wedge \varphi_2, \psi_1 \vee \psi_2 \rangle, \langle \varphi_i, \psi_j \rangle) \in V \times V \ : \ 1 \leq i, j \leq 2\} \cup & \\
& \{(\langle q', \psi_1 \vee \psi_2 \rangle, \langle q', \psi_i \rangle) \ : \ 1 \leq i \leq 2\} \cup & \\
& \{(\langle \varphi_1 \wedge \varphi_2, u' \rangle, \langle \varphi_i, u' \rangle) \in V \times V \ : \ 1 \leq i \leq 2\} \cup & \\
& \{(\langle q', u' \rangle, \langle \delta(q', \sigma), \delta(u', \sigma) \rangle) \in V \times V \ : \ a \in \Sigma\} \cup & \\
& \{(\langle \varphi_1 \vee \varphi_2, \psi \rangle, \langle \varphi_i, \psi \rangle) \ : \ 1 \leq i \leq 2\} \ \cup \ \{(\langle \varphi, \psi_1 \wedge \psi_2 \rangle, \langle \varphi_i, \psi \rangle) \ : \ 1 \leq i \leq 2\} &
\end{array}
$$

Table 7.3: Simulation game $\mathsf{G}_{\preceq}([\![\varphi]\!], [\![\psi]\!])$, where $[\![\varphi]\!]$ and $[\![\psi]\!]$ are unbounded MSCCs (Case 2).

$\psi \in \|U\|^{\oplus}$.

$$R_{\varphi,\psi} \ = \ \bigcup_{q \in I_\varphi} \bigcup_{u \in I_\psi} \bigcup_{\sigma \in \Sigma} \{\langle \alpha, \beta \rangle : \alpha \in \mathsf{cl}(\delta(q, \sigma)), \beta \in \mathsf{cl}(\delta(u, \sigma))\}$$

$$\mathsf{Val}_{\varphi,\psi} \ = \ \{0, 1\} \cup \{\mathsf{val}(\alpha, \beta) : \langle \alpha, \beta \rangle \in R_{\varphi,\psi}, \mathsf{val}(\alpha, \beta) \neq \bot\}$$

Let $\varphi = \oplus(r_1, \cdots, r_n)$ and $\psi = \oplus(t_1, \cdots, t_m)$, where $r_i \in \|Q\|^*$ and $t_l \in \|U\|^*$. Let $\mathcal{F}^{\oplus}_{r_i,\psi}$ be a set of functions $I_{r_i} \times I_\psi \to \mathsf{Val}_{\varphi,\psi}$. $f \in \mathcal{F}^{\oplus}_{r_i,\psi}$ iff there exist $\{a_{q,u}\}_{q \in I_{r_i}, u \in I_\psi}$ and $\vec{d} \in \mathcal{D}_{\mathsf{gt}(\psi)}$ such that:

- for all $u \in I_\psi$, $\sum_{q \in I_{r_i}} a_{q,u} p_{i,q} f(q, u) \geq \vec{d}_{t_j} p_{j,u}$,
- For all $q \in I_{r_i}$, $\sum_{u \in I_\psi} a_{q,u} = 1$.

The game is defined in the Table 7.4, where $\alpha, \alpha_i \in [\![\varphi]\!]$ and $\beta, \beta_i \in [\![\psi]\!]$, and γ and ϵ belong to $[\![\varphi]\!] \cap \|Q\|^{\oplus}$ and $[\![\psi]\!] \cap \|U\|^{\oplus}$, respectively. The winning condition $\Omega = V$ if $[\![\varphi]\!] \cap Q \subseteq F$ implies $[\![\psi]\!] \cap U \subseteq F$ else $\Omega = \varnothing$.

Most of the transitions are similar to Table 7.1 (page 118). The important difference is at configuration $\langle \gamma, \epsilon \rangle$, where $\gamma \in \|Q\|^{\oplus}$ and $\epsilon \in \|U\|^{\oplus}$. Player 1 in such a configuration selects a letter σ and a term $r \in \mathsf{gt}(\gamma)$, and Player 0 tries to win from the configuration $\langle r, \epsilon, \sigma \rangle$. Intuitively, the idea is reminiscent of Segala' simulations on probabilistic automata [88], where state s is simulated by state s', iff every *probabilistic transition* from s is simulated by a combined (weighted) probabilistic transition of the other.

$$
\begin{array}{ll}
V = & \{\langle \varphi', \psi', \sigma\rangle \ : \ \varphi' \in [\![\varphi]\!], \psi' \in [\![\psi]\!], \sigma \in \Sigma\} \cup \\
& \{\langle r, \epsilon, \sigma, f\rangle \ : \ r \in \mathsf{gt}(\gamma), t \in \mathsf{gt}(\epsilon), \sigma \in \Sigma, f \in \mathcal{F}^{\oplus}_{r,\epsilon}\} \cup \\
& \{\langle \varphi', \psi'\rangle, \langle \varphi', \psi', v\rangle \ : \ \varphi \preceq_1 \varphi', \psi \preceq_2 \psi', v \in \mathsf{Val}_{\varphi',\psi'}\} \\
\\
V_0 = & \{\langle \varphi_1 \wedge \varphi_2, \psi_1 \vee \psi, v\rangle, \langle \varphi_1 \wedge \varphi_2, \epsilon, v\rangle, \langle \gamma, \psi_1 \vee \psi_2, v\rangle\} \cup \\
& \{\langle \alpha, \beta, v\rangle \ : \ \bot \neq \mathsf{val}(\alpha, \beta) < v\} \cup \{\langle r, \epsilon, \sigma\rangle \ : \ r \in \mathsf{gt}(\gamma), \sigma \in \Sigma\} \\
\\
V_1 = & \{\langle \alpha, \beta, v\rangle \ : \ \bot \neq \mathsf{val}(\alpha, \beta) \geq v\} \cup \{\langle r, \epsilon, \sigma, f\rangle \ : \ r \in \mathsf{gt}(\gamma), f \in \mathcal{F}^{\oplus}_{r,\epsilon}\} \\
& \cup \{\langle \gamma, \epsilon\rangle, \langle \gamma, \epsilon, \sigma\rangle, \langle \alpha_1 \vee \alpha_2, \beta, v\rangle, \langle \alpha, \beta_1 \wedge \beta_2, v\rangle\} \\
\\
E = & \{(\langle \alpha_1 \wedge \alpha_2, \beta_1 \vee \beta_2, v\rangle, \langle \alpha_i, \beta_j, v\rangle \ : 1 \leq i, j \leq 2\} \cup \\
& \{(\langle \alpha_1 \wedge \alpha_2, \epsilon, v\rangle, \langle \alpha_i, \epsilon, v\rangle) \ : \ 1 \leq i \leq 2\} \cup \{(\langle \gamma, \beta_1 \vee \beta_2\rangle, \langle \gamma, \beta_i\rangle) \ : \ 1 \leq i \leq 2\} \\
& \cup \{(\langle \gamma, \epsilon, v\rangle, \langle \gamma, \epsilon\rangle) \ : \ \mathsf{val}(\gamma, \epsilon) = \bot\} \cup \ \{(\langle \gamma, \epsilon\rangle, \langle \gamma, \epsilon, \sigma\rangle)\} \cup \\
& \{(\langle \gamma, \epsilon, \sigma\rangle, \langle r, \epsilon, \sigma\rangle) \ : \ r \in \mathsf{gt}(\gamma)\} \cup \{(\langle r, \epsilon, \sigma\rangle, \langle r, \epsilon, \sigma, f\rangle) \ : f \in \mathcal{F}^{\oplus}_{r,\epsilon}\} \cup \\
& \{(\langle r, \epsilon, \sigma, f\rangle, \langle \delta(q, \sigma), \delta(u, \sigma), f(q, u)\rangle) \ : f(q, u) > 0\} \cup \\
& \{(\langle \alpha_1 \vee \alpha_2, \beta, v\rangle, \langle \alpha_i, \beta, v\rangle) \ : \ 1 \leq i \leq 2\} \cup \\
& \{(\langle \alpha, \beta_1 \wedge \beta_2, v\rangle, \langle \alpha, \beta_i, v\rangle) \ : \ 1 \leq i \leq 2\}
\end{array}
$$

Table 7.4: Simulation game $\mathsf{G}_{\leq}([\![\varphi]\!], [\![\psi]\!])$, where $[\![\varphi]\!]$ and $[\![\psi]\!]$ are bounded MSCCs (Case 3).

$$
\begin{array}{ll}
V = & \{\langle \varphi', \psi'\rangle \ : \ \varphi \preceq_1 \varphi', \psi \preceq_2 \psi'\} \ \cup \{\langle r, u, \sigma\rangle \ : \ \exists \gamma \in [\![\varphi]\!] \cap \|Q\|^{\oplus} : r \in \mathsf{gt}(\gamma)\} \\
& \cup \{\langle \gamma, u, \sigma\rangle \ : \ \gamma \in \|Q\|^{\oplus} \cap [\![\varphi]\!], u \in U \cap [\![\psi]\!], \sigma \in \Sigma\} \\
\\
V_0 = & \{\langle \alpha_1 \wedge \alpha_2\rangle, \langle \alpha_1 \wedge \alpha_2, u\rangle, \langle \gamma, \beta_1 \vee \beta_2\rangle\} \\
\\
V_1 = & \{\langle \alpha_1 \vee \alpha_2, \beta\rangle, \langle \alpha, \beta_1 \wedge \beta_2\rangle\} \ \cup \ \{\langle \gamma, u\rangle \ : \ \gamma \in \|Q\|^{\oplus} \cap [\![\varphi]\!], u \in U \cap [\![\psi]\!]\} \\
& \cup \{\langle \gamma, u, \sigma\rangle \ : \ \gamma \in \|Q\|^{\oplus} \cap [\![\varphi]\!], u \in U \cap [\![\psi]\!], \sigma \in \Sigma\} \\
\\
V_p = & (\|Q\|^* \times U \times \Sigma) \cap V \\
\\
E = & \{(\langle \alpha_1 \wedge \alpha_2, \beta_1 \vee \beta_2\rangle, \langle \alpha_i, \alpha_j\rangle) \ : \ 1 \leq i, j \leq 2\} \cup \\
& \{(\langle \alpha_1 \wedge \alpha_2, u\rangle, \langle \alpha_i, u\rangle) \ : \ 1 \leq i \leq 2\} \cup \\
& \{(\langle \gamma, \beta_1 \vee \beta_2\rangle, \langle \gamma, \beta_i\rangle) \ : \ 1 \leq i \leq 2\} \ \cup \{(\langle \alpha, \beta_1 \wedge \beta_2\rangle, \langle \alpha, \beta_i\rangle) \ : \ 1 \leq i \leq 2\} \\
& \{(\langle \gamma, u\rangle, \langle \gamma, u, \sigma\rangle) \ : \ \gamma \in \|Q\|^{\oplus}, \sigma \in \Sigma\} \cup \\
& \{(\langle r, u, \sigma\rangle, \langle \delta(q, \sigma), \delta(u, \sigma)\rangle) \ : \|q_{\epsilon}\|_{\bowtie_{\epsilon} p_{\epsilon}} \mathsf{gt}(r)\} \cup \\
& \{(\langle \gamma, u, \sigma\rangle, \langle r, u, \sigma\rangle) \ : \ \gamma \in \|Q\|^{\oplus}, r \in {}_{>}(\gamma), a \in \Sigma\}
\end{array}
$$

Table 7.5: Simulation game $\mathsf{G}_{\leq}([\![\varphi]\!], [\![\psi]\!])$, where $[\![\varphi]\!]$ is bounded and $[\![\psi]\!]$ is unbounded MSCCs (Case 4).

Case 4. Let $[\![\varphi]\!]$ be a bounded MSCC of G_1 and $[\![\psi]\!]$ be unbounded MSCC of G_2. The game $\mathsf{G}_{\leq}([\![\varphi]\!], [\![\psi]\!])$ is a stochastic weak game, it is defined in the Table 7.5. The probability distribution is defined as: $P(\langle r_i, u, \sigma\rangle, \langle \delta(q,\sigma), \delta(u,\sigma)\rangle) =$

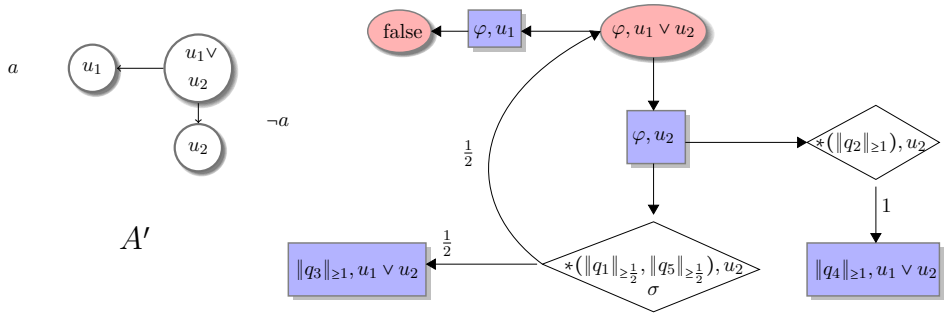


Figure 7.2.7: Simulation game of the p-automaton A of example 7.2.2 by automaton A' (on the left). The MSCCs of A' are unbounded and the MSCCs (Figure 7.2.1) of A are bounded. $\mathsf{val}(\varphi, u_1) = 0$ is pre-calculated.

$p_{i,q}$, where $r_i \in \mathsf{gt}(\gamma)$ for some $\gamma \in \|Q\|^{\oplus} \cap [\![\varphi]\!]$. The wining condition $\Omega = V$ if $[\![\varphi]\!] \cap Q \subseteq F$ implies $[\![\psi]\!] \cap U \subseteq F$ else $\Omega = \varnothing$. Conjunctions and disjunctions are handled as in previous cases. Player 1 at configuration $\langle \gamma, u\rangle$ selects (similar to case 3.) a letter σ and a term $r \in \mathsf{gt}(\gamma)$ and move to a probabilistic configuration $\langle r, u, \sigma\rangle$. The bounds on the guarded states of r determine the distribution on the configurations of the game (Figure 7.2.7). If the bounds do not add to 1, sink states are added, which are losing for Player 0.

Case 5. The case when one or both $[\![\varphi]\!]$ and $[\![\psi]\!]$ are trivial MSCC is subsumed in at least one of the above cases.

Automaton $A_2 = (U, \Sigma, \delta, \psi_{in}, F)$ simulates automaton $A_1 = (Q, \Sigma, \delta, \varphi_{in}, F)$ (also denoted by $A_1 \leq A_2$) if the value of the configuration $\langle \varphi_{in}, \psi_{in}\rangle$ computed by the simulation game is 1.

The weak game generated by bounded MSCC $[\![\varphi]\!]$ and $[\![\psi]\!]$, of the game graphs, can be exponential in size of the graphs. This is due to the exponential number of functions $f \in \mathcal{F}^{\oplus}_{r,\psi}$ ($r \in \mathsf{gt}(\varphi)$). But the size stochastic games is polynomial in the size of the game graph. Thus, the simulation procedure can be implemented by an exponential time algorithm.

Theorem 7.2.5. *Let A_1 and A_2 be p-automata. Then:*

$$A_1 \leq A_2 \text{ implies } \mathcal{L}(A_1) \subseteq \mathcal{L}(A_2).$$

Proof. Let $M = (S, P, \Sigma, L, s_{in})$ be an arbitrary MC and A_1, A_2 be p-automata $(Q, \Sigma, \delta, \varphi_{in}, F)$, $(U, \Sigma, \delta, \psi_{in}, F)$, respectively. We assume that Q and U are disjoint and hence use the same symbol for the transition relations and final states for the two automata.

We show that, if $\mathsf{val}(\{s_{in}\}, \varphi_{in}) = 1$ in the acceptance game of M by A_1 and $\mathsf{val}(\varphi_{in}, \psi_{in}) = 1$ in the simulation game of A_1 by A_2, then $\mathsf{val}(\{s_{in}\}, \psi_{in}) = 1$ in the acceptance game of M by A_2. Let the acceptance games of M by A_1 and A_2 be G_1 and G_2, respectively, and the simulation game of A_1 by A_2 be $\mathsf{G}_{\leq}$. Equivalently, we show that the *claim:* $\mathsf{val}(T, \varphi) \cdot \mathsf{val}(\varphi, \psi) \leq \mathsf{val}(T, \psi)$, is true for any arbitrary $\varphi \in Q \cup \mathsf{cl}(\delta(Q, \Sigma))$, $\psi \in U \cup \mathsf{cl}(\delta(U, \Sigma))$, and $T \in \mathcal{S}$.

A triplet of configurations c_1, c_2 and c_3 is said to be *matching*, where c_1, c_2 and c_3 are configurations of the game $\mathsf{G}_1, \mathsf{G}_{\leq}$ and G_2, respectively, if the first component of c_1 is equal to the first component of c_3, the second component of c_1 is equal to the second component of c_2 and the second component of c_2 is equal to the second component of c_3 (i.e., $c_1 = \langle T, \varphi \rangle, c_2 = \langle \varphi, \psi \rangle, c_3 = \langle T, \psi \rangle$) .

We proceed by induction on the partial order $\preceq$, and when considering configurations in $(\llbracket \varphi \rrbracket, \llbracket \psi \rrbracket)$, we assume that the *claim* holds for every configuration in the pair $(\llbracket \varphi' \rrbracket, \llbracket \psi' \rrbracket)$, where $(\llbracket \varphi \rrbracket, \llbracket \psi \rrbracket) \preceq (\llbracket \varphi' \rrbracket, \llbracket \psi' \rrbracket)$. Effectively, we construct a winning strategy for Player 0 in G_2 from the strategies of the Players in G_1 and $\mathsf{G}_{\leq}$. We have the following cases:

Case 1. If $\varphi \in Q$ and $\psi \in \|U\|^{\oplus}$ then $\mathsf{val}(\varphi, \psi) = 0$, and the claim follows trivially.

Case 2. Let $\llbracket \varphi \rrbracket$ and $\llbracket \psi \rrbracket$ be unbounded MSCCs, where $\mathsf{G}_1(M, \llbracket \varphi \rrbracket)$ and $\mathsf{G}_2(M, \llbracket \psi \rrbracket)$ are weak stochastic game and $\mathsf{G}_{\leq}(\llbracket \varphi \rrbracket, \llbracket \psi \rrbracket)$ is stochastic game. The configurations of game $\mathsf{G}_1(M, \llbracket \varphi \rrbracket)$ and $\mathsf{G}_2(M, \psi)$ are of the form $\langle T, \varphi' \rangle$ and $\langle T, \psi' \rangle$, respectively, where T is a singleton set. As before, we will write $\langle \{s\}, \varphi' \rangle$ as $\langle s, \varphi' \rangle$. Consider three matching configurations $c_1 = \langle s, \alpha \rangle$, $c_2 = \langle \alpha, \beta \rangle$ and $c_3 = \langle s, \beta \rangle$, such that $\alpha \in \llbracket \varphi \rrbracket$ and $\beta \in \llbracket \psi \rrbracket$.

1. If $\alpha = \alpha_1 \wedge \alpha_2$ and β is not a conjunction then c_2 is a Player 0 configuration. Suppose Player 0 at c_2 chose $\langle \alpha_i, \beta \rangle$, then Player 1 at c_1 is made to choose $\langle s, \alpha_i \rangle$. Else if $\beta = \beta_1 \wedge \beta_2$ then c_3 is a Player 1 configuration and if he chose $\langle s, \beta_i \rangle$ then Player 1 at c_2 chooses $\langle \alpha, \beta_i \rangle$.

2. If $\alpha = \alpha_1 \vee \alpha_2$ then c_1 is a Player 0 configuration, and if she chose $\langle s, \alpha_i \rangle$ at c_1 then Player 1 at c_2 chooses $\langle \alpha_i, \beta \rangle$. If $\beta = \beta_1 \vee \beta_2$ and Player 0 chooses $\langle \alpha, \beta_i \rangle$ at c_2 then Player 0 in c_3 chooses $\langle s, \beta_i \rangle$.

3. If $\alpha = q$ and $\beta = u$ then c_1 and c_3 are stochastic configurations and c_2 is a Player 1 configuration. Player 1 is made to select the action $\sigma = L(s)$ and reach a configuration $\langle \delta(q,\sigma), \delta(u,\sigma)\rangle$ and next configuration in G_1 and G_2 is $\langle s', \delta(q,\sigma)\rangle$ and $\langle s', \delta(u,\sigma)\rangle$, respectively.

Note that these choices of moves always ensures that we move from one matching triplet to the next. Consider three matching paths in the games G_1, $\mathsf{G}_{\leq}$ and G_2. If the path in $\mathsf{G}_{\leq}$ is infinite then, and the corresponding path in G_1 is winning, then by the winning condition of $\mathsf{G}_{\leq}$, the respective path in G_2 is also winning. If it is finite then the triplet of paths end in configuration $(\langle s'', \alpha'\rangle, \langle \alpha', \beta'\rangle, \langle s'', \beta'\rangle)$, where $\langle \alpha', \beta'\rangle \notin (\llbracket\varphi\rrbracket, \llbracket\psi\rrbracket)$. Since, $(\llbracket\varphi\rrbracket, \llbracket\psi\rrbracket)$ is a weak game $\mathsf{val}(\alpha', \beta') \geq \mathsf{val}(\alpha, \beta)$. By assumption $\mathsf{val}(s'', \alpha') \cdot \mathsf{val}(\alpha', \beta')$ $\leq \mathsf{val}(s'', \beta')$ or $\mathsf{val}(s'', \alpha') \cdot \mathsf{val}(\alpha, \beta) \leq \mathsf{val}(s'', \beta')$. The inequality holds for every matching paths in all three games thus, $\mathsf{val}(s, \alpha) \cdot \mathsf{val}(\alpha, \beta) \leq \mathsf{val}(s, \beta)$.

Case 3. Suppose $\llbracket\varphi\rrbracket$ and $\llbracket\psi\rrbracket$ are bounded MSCCs, $G_1(M, \llbracket\varphi\rrbracket)$, $G_{\leq}(\llbracket\varphi\rrbracket, \llbracket\psi\rrbracket)$ and $G_2(M, \llbracket\psi\rrbracket)$ are all weak games. Consider a triplet of configurations $(\langle T, \alpha\rangle, \langle \alpha, \beta\rangle, \langle T, \beta\rangle)$. We assume $\mathsf{val}(T, \alpha) = 1$ and $\mathsf{val}(\alpha, \beta) = 1$, else $\mathsf{val}(T, \alpha) \cdot \mathsf{val}(\alpha, \beta) \leq \mathsf{val}(T, \beta)$ follows immediately.

The cases of conjunctions and disjunctions are handled as in *case 2*. The interesting case is when $\alpha \in \|Q\|^{\oplus}$ and $\beta \in \|U\|^{\oplus}$, where $\alpha = \oplus(r_1, \cdots, r_n)$ and $\beta = \oplus(t_1, \cdots, t_m)$. It follows that $\langle T, \alpha\rangle$ and $\langle T, \beta\rangle$ are Player 0 configurations and $\langle \alpha, \beta\rangle$ is a Player 1 configuration. Suppose Player 0 at $\langle T, \alpha\rangle$ selects a function f with witness $\vec{d}$ and $\{a_{q,s'}\}$, such that:

$$\forall r_i \in \mathsf{gt}(\alpha), q \in \mathsf{gs}(r_i) \sum_{s' \in \mathsf{succ}(s)} a_{q,s'} f(s', q) P(s, s') \geq p_{i,q} \vec{d}_{r_i}$$
$$\forall s' \in \mathsf{succ}(s) \sum_{q \in I_\alpha} a_{q,s'} = 1$$

We make Player 1 at $\langle \alpha, \beta\rangle$ (of $G_{\leq}$) choose an action $\sigma = L(T)$ and move to Player 0 configuration $\langle \alpha, \beta, \sigma\rangle$. Configuration $\langle r_i, \beta, \sigma\rangle$ is winning for Player 0, for each $r_i \in \mathsf{gt}(\alpha)$, in the game $\mathsf{G}_{\leq}$. Let $f^i \in \mathcal{F}^{\oplus}_{r_i,\beta}$ be the function chosen by Player 0 in $\mathsf{G}_{\leq}$ with witnesses $\{a^i_{q,u}\}_{q \in I_{r_i}, u \in I_\beta}$ and $\vec{c}^{\,i} \in \mathcal{D}_{\mathsf{gt}(\beta)}$. Thus for each r_i we have:

- $\forall u \in I_\beta \sum_{q \in I_{r_k}} a_{q,u} f^i(q, u) p_{i,q} \geq \vec{c}^{\,i}_{t_j} \cdot p_{j,u}$,
- $\forall q \in I_{r_i} \sum_{u \in I_\beta} a_{q,u} = 1$.

Player 0 in game G_2 selects a function f'' such that $f''(u, s')$ is then the minimum value in $\mathsf{Val}_{T,\beta}$ that is at least $\max_{q \in \mathsf{gs}(r_i), r_i \in \mathsf{gt}(\alpha)} f(q, s') f^i(q, u)$.

The reason for choosing such a f'' will soon become clear. The witness of f'' are as follows: $a_{u,s'} = \sum_{r_i \in \mathsf{gt}(\alpha)} \sum_{q \in \mathsf{gs}(r_i)} a_{q,s'} a^i_{q,u}$ and for each $t_j \in \mathsf{gt}(\beta)$, $\vec{d}'_{t_j} = \sum_{r_i \in \mathsf{gt}(\alpha)} \vec{d}_{r_i} \cdot \vec{c}^{\,i}_{t_j}$. Intuitively, Player 0 in G_1, Player 0 gives the distribution $\vec{d}$ on the guarded terms $r_k \in \mathsf{gt}(\alpha)$ and in the game $\mathsf{G}_{\leq}$ gives the distribution to simulate each r_i by β. This determines the distribution $\vec{d}'$ for the game G_2.

We will now show that $f'' \in \mathcal{F}^{\oplus}_{s,\beta}$. For each $s' \in \mathsf{succ}(s)$:

$$\sum_{u \in I_\beta} a_{u,s'} = \sum_{r_i \in \mathsf{gt}(\alpha)} \sum_{q \in \mathsf{gs}(r_i)} a_{q,s'} \left(\sum_{u \in I_\beta} a_{q,u} \right) = \sum_{q \in I_\alpha} a_{q,s'} = 1.$$

Consider any $u \in \mathsf{gs}(t_j)$, where $t_j \in \mathsf{gt}(\beta)$:

$$\begin{aligned}
& \sum_{s' \in \mathsf{succ}(s)} a_{u,s'} f''(u,s') P(s,s') \\
= & \sum_{s' \in \mathsf{succ}(s)} \left(\sum_{r_i \in \mathsf{gt}(\alpha)} \sum_{q \in \mathsf{gs}(r_i)} a_{q,s'} a^i_{q,u} \right) f''(u,s') P(s,s') \\
= & \sum_{s' \in \mathsf{succ}(s)} \left(\sum_{r_i \in \mathsf{gt}(\alpha)} \sum_{q \in \mathsf{gs}(r_i)} a_{q,s'} a^i_{q,u} \right) \max_{q \in \mathsf{gs}(r_i), r_i \in \mathsf{gt}(\alpha)} f(u,s') f^i(q,u) P(s,s') \\
\geq & \sum_{s' \in \mathsf{succ}(s)} \sum_{r_i \in \mathsf{gt}(\alpha)} \sum_{q \in \mathsf{gs}(r_i)} a_{q,s'} a^i_{q,u} f(q,s') f^i(q,u) P(s,s') \\
= & \sum_{r_i \in \mathsf{gt}(\alpha)} \sum_{q \in \mathsf{gs}(r_i)} a^i_{q,u} f^i(q,u) \sum_{s' \in \mathsf{succ}(s)} a_{q,s'} f(q,s') P(s,s') \\
\geq & \sum_{r_i \in \mathsf{gt}(\alpha)} \sum_{q \in \mathsf{gs}(r_i)} a^i_{q,u} f^i(q,u) \vec{d}_{r_i} p_{i,q} \\
= & \sum_{r_i \in \mathsf{gt}(\alpha)} \vec{d}_{r_i} \sum_{q \in \mathsf{gs}(r_i)} a_{q,u} f'(q,u) p_{i,q} \\
\geq & \sum_{r_i \in \mathsf{gt}(\alpha)} \vec{d}_{r_i} \vec{c}^{\,i}_{t_j} p_{j,u} \\
= & \vec{d}'_{t_j} p_{j,u}
\end{aligned}$$

If Player 1 in G_2 chooses $\langle T', \delta(u,\sigma), v_1 \rangle$, where $v_1 = \min_{s' \in T} f(q,s'))$, we make Player 1 in G_1 choose a state $q \in I_\beta$ (and hence a term r_i such that $q \in \mathsf{gs}(r_i)$) such that $\min_{s' \in T'} f(q,s') f^i(q,u)$ is maximal and move to $\langle T', \delta(q,\sigma), v_1 \rangle$. Correspondingly, we make Player 1 in $G_{\leq}$ to move to $\langle \delta(q,\sigma), \delta(u,\sigma), f^i(q,u) \rangle$.

Consider a triplet of matching paths from G_1, $\mathsf{G}_{\leq}$ and G_2. Suppose the play continues inside of the MSCC pair of $\mathsf{G}_{\leq}$, indefinitely. Then the play in G_1 is winning because the play is according to a winning strategy of Player 0 in G_1, for the same reason the play in $\mathsf{G}_{\leq}$ is winning. Because of the winning condition of $\mathsf{G}_{\leq}$, the corresponding play in G_2 is also winning.

Suppose the plays in $\mathsf{G}_{\leq}$ reach a triplet of configurations $(\langle T'', \alpha'', v_1 \rangle$, $\langle \alpha'', \beta'', v_2 \rangle$, $\langle T'', \beta'', v_3 \rangle)$, where $([\![\alpha'']\!], [\![\beta'']\!]) \neq ([\![\varphi]\!], [\![\psi]\!])$. We have $\mathsf{val}(T'', \alpha'')$.

$\mathsf{val}(\alpha'', \beta'') \leq \mathsf{val}(T'', \beta'')$ from the induction hypothesis. We have to show $\mathsf{val}(T'', \beta'') \geq v_3$. Let the triplet of configurations $(\langle T', \alpha' \rangle, \langle \alpha', \beta' \rangle, \langle T', \beta' \rangle)$ be the last configurations such that $\langle \alpha', \beta' \rangle$ is inside the MSCC pair $([\![\varphi]\!], [\![\psi]\!])$ and $\alpha' \in \|Q\|^{\oplus}$, and $\beta' \in \|U\|^{\oplus}$. Player 1 in G_1 at configuration $\langle T', \alpha' \rangle$ chooses a q such that $f(q, T'')f^i(q, u)$ is maximum, thus $v_3 \geq v_1 \cdot v_2$. As the plays in G_1 and $\mathsf{G}_{\leq}$ are winning for Player 0, $\mathsf{val}(T'', \alpha'') \geq v_1$ and $\mathsf{val}(\alpha'', \beta'') \geq v_2$. This makes $\mathsf{val}(T'', \beta'') \geq v_1 v_2$. Observe that v_3 is the minimum value in $\mathsf{Val}_{T', \beta'}$ which is at least $\max_{q \in I'_\alpha} f^i(q, u) f(q, T'')$ thus $\max_{q \in I_{\alpha'}} f^i(q, u) f(q, T'') \leq v_3 \leq x \in \mathsf{Val}_{s', \beta'}$. Since, $\mathsf{Val}_{T', \beta'}$ includes $\mathsf{val}(T'', \beta'')$, therefore, $\mathsf{val}(T'', \beta'') \geq v_3$.

Case 4. Let $[\![\varphi]\!]$ is bounded and $[\![\psi]\!]$ is unbounded MSCCs. $\mathsf{G}_1(M, [\![\varphi]\!])$ is a weak game, whereas $\mathsf{G}_{\leq}([\![\varphi]\!], [\![\psi]\!])$ and $\mathsf{G}_2(M, [\![\psi]\!])$ are weak stochastic games. Recall that configurations of $\mathsf{G}_2(M, [\![\psi]\!])$ are of the form $\langle T, \psi' \rangle$, where T is a singleton set. In order to reason with matching triplets of configurations we restrict Player 1 in $\mathsf{G}_1(M, [\![\varphi]\!])$ to configurations $\langle T, \varphi' \rangle$, where T is singleton.

Consider a matching triplet of configurations $(\langle s, \varphi \rangle, \langle \varphi, \psi \rangle, \langle s, \psi \rangle)$. The interesting case is when $\mathsf{val}(s, \varphi) = 1$, else the claim follows trivially.

1. If ψ is a conjunction of the form $\psi_1 \wedge \psi_2$, then Player 1 in G_2 chooses the next configuration $\langle s, \psi_i \rangle$, then Player 1 in $G_{\leq}$ chooses $\langle \varphi, \psi_i \rangle$ in $\mathsf{G}_{\leq}$.

2. Suppose $\psi = \psi_1 \vee \psi_2$. If $\varphi = \varphi_1 \wedge \varphi_2$, then $\langle \varphi, \psi \rangle$ is Player 0 configuration in $G_{\leq}$. If Player 0 chooses $\langle \varphi_i, \psi \rangle$ then Player 1 in G_1 is made to choose $\langle s, \varphi_i \rangle$. If $\varphi = \varphi_1 \vee \varphi_2$, and Player 0 in G_1 moves to configuration $\langle s, \varphi_i \rangle$, then Player 1 in $G_{\leq}$ moves to $\langle \varphi_i, \psi \rangle$. If $\varphi \in \|Q\|^{\oplus}$ then $\langle \varphi, \psi \rangle$ is a Player 0 configuration. If she chooses (φ, ψ_i) then Player 0 in G_2 chooses (s, ψ_i).

The remaining case is $\varphi \in \|Q\|^{\oplus}$ and $\psi \in U$, where $\varphi = \oplus(r_1, \cdots, r_n)$ and $\psi = u$. $\langle s, \varphi \rangle$ in G_1 is Player 0 configuration and she chooses a function $f \in \mathcal{F}^{\oplus}_{s,\varphi}$ with witness $\vec{d} \in \mathcal{D}_{\mathsf{gt}(\varphi)}$ and $\{a_{q,s'}\}_{q \in I_\varphi, s' \in \mathsf{succ}(s)}$ and moves to the configuration $\langle s, \varphi, f \rangle$. Player 1 at configuration $\langle \varphi, u \rangle$ in $\mathsf{G}_{\leq}$ chooses the action $\sigma = L(s)$ and then chooses a configurations $\langle r_k, u \rangle$ such that $\mathsf{val}(\varphi, u)$ is minimum (i.e., he plays his best possible move).

$\langle s, \varphi, f \rangle$ in the game G_1 is a Player 1 configuration and there could be more than one way for the game to evolve to the next matching triplet configurations. For example, if $f(q, s') > 0$ and $f(q', s') > 0$, $q, q' \in I_\varphi$, then it possible to have the next matching triplets as $\langle s, \delta(q, \sigma), f(q, s') \rangle$,

$\langle \delta(q,\sigma), \delta(u,\sigma)\rangle$, and $\langle s', \delta(u,\sigma)\rangle$ or $\langle s, \delta(q',\sigma), f(q',s')\rangle$, $\langle \delta(q',\sigma), \delta(u,\sigma)\rangle$, and $\langle s', \delta(u,\sigma)\rangle$. We prove that the claim holds for any of the matching triplets arising from different choice of $q \in I_\varphi$, i.e., it holds in the worst case. Equivalently, we show that the claim holds for a *choice of* q for which the value $\mathsf{val}(s,\alpha)\cdot\mathsf{val}(\alpha,\beta)$ is maximum. Consider the triplet of matching plays (where the configurations are step wise matching) from matching configurations $\langle s,\alpha\rangle$, $\langle \alpha,\beta\rangle$ and $\langle s,\beta\rangle$. We have the following cases:

4.a. The triplet of configurations $\langle s,\alpha\rangle$, $\langle \alpha,\beta\rangle$ and $\langle s,\beta\rangle$ where $\langle \alpha,\beta\rangle$ is not in the pair of equivalence classes $([\![\varphi]\!],[\![\psi]\!])$. The claim follows from induction hypothesis $\mathsf{val}(s,\alpha)\cdot\mathsf{val}(\alpha,\beta) \leq \mathsf{val}(s,\beta)$.

4.b. For every choice of matching of triplets during the evolution of the game, every play from $\langle \alpha,\beta\rangle$ stays in $([\![\varphi]\!],[\![\psi]\!])$ and are winning for Player 0 in $\mathsf{G}_{\leq}$. If the matching play in G_1 starting from $\langle s,\alpha\rangle$ is winning, then the matching play in G_2 from $\langle s,\beta\rangle$ are also winning for Player 0. Suppose this is not the case and there is a play from $\langle s,\beta\rangle$ that is not winning. Consider any corresponding matching play in $\mathsf{G}_{\leq}$, together they define a matching play in G_1. If the play is not winning in G_2 then the matching play in G_1 is also not winning, which cannot happen as $\mathsf{val}(s,\alpha) = 1$ and G_1 is a weak game.

4.c. For every choice of matching of triplets the play stays in $([\![\varphi]\!],[\![\psi]\!])$ and are not winning for Player 0 in $\mathsf{G}_{\leq}$. Then $\mathsf{val}(\alpha,\beta) = 0$ and the claim follows trivially.

4.d. The triplet of configurations $\langle s,\alpha\rangle$, $\langle \alpha,\beta\rangle$ and $\langle s,\beta\rangle$ such that not all the plays in $([\![\varphi]\!],[\![\psi]\!])$ are winning for Player 0 in $\mathsf{G}_{\leq}$ but probability of the set of winning plays is greater than zero. Here we explicitly assume that the MC M and automata A_1, A_2 are finite. Every time a configuration $\langle s,\alpha\rangle$ is revisited, the same function $f \in \mathcal{F}^{\oplus}_{s,\alpha}$ is chosen. Hence, the number of different matching configurations is finite.

We show that the claim, $\mathsf{val}(s,\alpha)P(\alpha,\beta) \leq P(s,\beta)$, holds, where $P(s,\beta)$ and $P(\alpha,\beta)$ is the worst case probability of reaching one of the three types of configurations covered in the previous three cases. Let $P_n(\alpha,\beta)$ and $P_n(s,\beta)$ be the probability of reaching one of the three types of configurations (defined in case 4.a, 4.b and 4.c) in n steps from $\langle \alpha,\beta\rangle$ and $\langle s,\beta\rangle$, respectively by matching paths, when matching triplet are chosen such that, $P_n(\alpha,\beta)\cdot\mathsf{val}(s,\alpha)$ is maximum. We show that $P_n(s,\beta) \geq P_n(\alpha,\beta)\cdot\mathsf{val}(s,\alpha)$

for any n. We proceed by induction on n.

If $\langle s,\alpha\rangle, \langle \alpha,\beta\rangle, \langle s,\beta\rangle$ is one of the three configurations from case 4.a, 4.b and 4.c then $P_0(\alpha,\beta) = \mathsf{val}(\alpha,\beta)$ and $P_0(s,\beta) = \mathsf{val}(s,\beta)$, else zero. From the definition (case 4.a, 4.b, 4.c) the claim holds for $n = 0$. We will prove the hypothesis for $n = 1$. Consider the triplet $\langle s,\alpha\rangle$, $\langle \alpha,u\rangle$ and $\langle s,u\rangle$, where $\alpha = \oplus(r_1,\cdots,r_n)$ and $P_0(\alpha,u) = 0$ and $P_0(s,u) = 0$, i.e., the triplet belongs to case 4.d. But for some successor $s' \in \mathsf{succ}(s)$ and $q \in I_\alpha$, $P_0(s',\delta(u,\sigma)) > 0$ and $P_0(\delta(q,\sigma),\delta(u,\sigma)) > 0$ ($\sigma = L(s)$). Let $f \in \mathcal{F}^{\oplus}_{s,\alpha}$ be the function chosen by Player 0 at $\langle s,\alpha\rangle$ with witnesses $\{a_{q,s'}\}_{q\in I_\alpha, s'\in S}$ and $\vec{d}$. We have:

$$P_1(s,u) \;=\; \sum_{s'\in\mathsf{succ}(s)} P(s,s')\cdot P_0(s',\delta(u,\sigma)). \tag{7.6}$$

And,

$$\begin{aligned} P_1(\alpha,u) &= \min_{r_k\in\mathsf{gt}(\alpha)} \sum_{q\in\mathsf{gs}(r_k)} P_0(\delta(q,\sigma),\delta(u,\sigma))p_{k,q} \\ &\le \sum_{r_i\in\mathsf{gt}(\alpha)} \vec{d}_{r_i} \sum_{q\in\mathsf{gs}(r_i)} P_0(\delta(q,\sigma),\delta(u,\sigma))p_{i,q} \end{aligned} \tag{7.7}$$

For each $s' \in \mathsf{succ}(s)$ let $q_{s'}$ be the choice of q such that $\mathsf{val}(s',\delta(q_{s'},\sigma))\cdot \mathsf{val}(\delta(q_{s'},\sigma),\delta(u,\sigma))$ is maximum. By construction, $P_0(s',\delta(u,\sigma)) \ge P_0(\delta(q,\sigma),\delta(u,\sigma))\cdot \mathsf{val}(s',\delta(q,\sigma))$, since $\mathsf{val}(s',\delta(u,\sigma)) \ge \mathsf{val}(\delta(q,\sigma),\delta(u,\sigma))\mathsf{val}(s',\delta(q,\sigma))$. From Equation 7.6.

$$P_1(s,u) \;\ge\; \sum_{s'\in\mathsf{succ}(s)} P(s,s')P_0(\delta(q,\sigma),\delta(u,\sigma))\mathsf{val}(s',\delta(q,\sigma)) \tag{7.8}$$

Since $f \in \mathcal{F}^{\oplus}_{s,\alpha}$, $\sum_{r_i\in\mathsf{gt}(\alpha)}\sum_{q\in\mathsf{gs}(r_i)} a_{q,s'} = 1$. Therefore:

$$P_1(s,u) \ge \sum_{s'\in\mathsf{succ}(s)} P(s,s')\left(\sum_{r_i\in\mathsf{gt}(\alpha)}\sum_{q\in\mathsf{gs}(r_i)} a_{q,s'}\right)\cdot P_0(\delta(q,\sigma),\delta(u,\sigma))\cdot\mathsf{val}(s',\delta(q,\sigma))$$

Since the configuration $\langle s',\delta(q,\sigma)\rangle$ is winning for Player 0, $\mathsf{val}(s',\delta(q,\sigma)) \ge f(q,s')$.

$$P_1(s,u) \ge \sum_{s'\in\mathsf{succ}(s)} P(s,s')\left(\sum_{r_i\in\mathsf{gt}(\alpha)}\sum_{q\in\mathsf{gs}(r_i)} a_{q,s'}\right)\cdot P_0(\delta(q_{s'},\sigma),\delta(u,\sigma))\cdot f(s',q_{s'})$$

We can distribute $a_{q,s'}$ according to the following:

$$\begin{aligned} P_1(s,u) &\ge \sum_{s'\in\mathsf{succ}(s)}\sum_{r_i\in\mathsf{gt}(\alpha)}\sum_{q\in\mathsf{gs}(r_i)} P_0(\delta(q,\sigma),\delta(u,\sigma))\cdot P(s,s')a_{q,s'}f(s',q) \\ &\ge \sum_{r_i\in\mathsf{gt}(\alpha)}\sum_{q\in\mathsf{gs}(r_i)} P_0(\delta(q,\sigma),\delta(u,\sigma))\cdot \sum_{s'\in\mathsf{succ}(s)} P(s,s')a_{q,s'}f(s',q) \\ &\ge \sum_{r_i\in\mathsf{gt}(\alpha)}\sum_{q\in\mathsf{gs}(r_i)} P_0(\delta(q,\sigma),\delta(u,\sigma))\cdot p_{i,q}\vec{d}_{r_i} \\ &\ge\; P_1(\alpha,u) \end{aligned}$$

Assume now that the claim holds for every matching triplets, for n steps. Consider the matching triplets $\langle s, \alpha\rangle$, $\langle \alpha, u\rangle$ and $\langle s, \beta\rangle$ where $\alpha = \oplus(r_1, \cdots, r_n)$ and $u \in U$. As before, let $f \in \mathcal{F}^{\oplus}_{\alpha,s}$ be the function chosen by Player 0 at the configuration $\langle s, \alpha\rangle$, with witnesses $\{a_{q,s'}\}_{q\in I_\alpha, s'\in S}$ and $\vec{d} \in \mathcal{D}_{\mathsf{gt}(\alpha)}$. (For configurations with conjunction and disjunction, the matching paths are determined in their respective game by the strategies defined before.) We have:

$$\begin{aligned} P_{n+1}(s,u) &= \sum_{s'\in \mathsf{succ}(s):\exists q: a_{q,s'}>0} P(s,s')\cdot P_n(s', \delta(u,\sigma)) \\ P_{n+1}(\alpha,u) &= \min_{r_i\in \mathsf{gt}(\alpha)} \sum_{q\in \mathsf{gs}(r_i)} p_{i,q} P_n(\delta(q,\sigma), \delta(u,\sigma)) \\ &\leq \sum_{r_i\in \mathsf{gt}(\alpha)} \sum_{q\in \mathsf{gs}(r_i)} \vec{d}_{r_i} p_{i,q} P_n(\delta(q,\sigma), \delta(u,\sigma)) \end{aligned} \tag{7.9}$$

By induction hypothesis :

$$P_n(s', \delta(u,\sigma)) \geq P_n(\delta(q_{s'},\sigma), \delta(u,\sigma)) \cdot \mathsf{val}(s', \delta(q_{s'},\sigma)) \tag{7.10}$$

From equation 7.9:

$$P_{n+1}(s,u) \geq \sum_{s'\in \mathsf{succ}(s)} P(s,s')\mathsf{val}(s', \delta(q_{s'},\sigma)) P_n(\delta(q_{s'},\sigma), \delta(u,\sigma))$$

Choose $q_{s'}$ for each s' such that $\mathsf{val}(s', \delta(q_{s'},\sigma)) P_n(\delta(q_{s'},\sigma), \delta(u,\sigma)$ is maximum.

$$\begin{aligned} & P_{n+1}(s,u) = \\ & \sum_{s'\in \mathsf{succ}(s)} P(s,s')\Big(\sum_{r_i\in \mathsf{gt}(\alpha)} \sum_{q\in \mathsf{gs}(r_i)} a_{q,s'}\Big)\mathsf{val}(s', \delta(q_{s'},\sigma)) P_n(\delta(q_{s'},\sigma), \delta(u,\sigma)) \\ & \geq \sum_{s'\in \mathsf{succ}(s)} P(s,s')\Big(\sum_{r_i\in \mathsf{gt}(\alpha)} \sum_{q\in \mathsf{gs}(r_i)} a_{q,s'}\Big) f(q_{s'}, s') P_n(\delta(q_{s'},\sigma), \delta(u,\sigma)) \\ & \geq \sum_{r_i\in \mathsf{gt}(\alpha)} \sum_{q\in \mathsf{gs}(r_i)} P_n(\delta(q,\sigma), \delta(u,\sigma)) \sum_{s'\in \mathsf{succ}(s)} P(s,s') a_{q,s'} f(q,s') \\ & \geq \sum_{r_i\in \mathsf{gt}(\alpha)} \sum_{q\in \mathsf{gs}(r_i)} P_n(\delta(q,\sigma), \delta(u,\sigma)) p_{i,q} \vec{d}(r_i) \\ & = P_{n+1}(\alpha,u) \end{aligned}$$

This concludes the proof. □

7.3 Conclusion

We have presented an extension of p-automata [58], and used it to represent the set of MCs which are bisimilar to the MCs induced by the schedulers of an MDP. We have seen that the languages of the p-automata are closed

	p-Automata (Huth et. al.)	p-Automta$^{\oplus}$
Accepts	Markov Chains	Markov Chains
Complexity of acceptance	EXP-TIME	EXP-TIME ^{2}
Inclusion	Not known	Σ_1^0-hard
Emptiness	Not known	Σ_1^0-hard

Table 7.6: Summary of p-automata

under bi-simulation (union and intersection). We have addressed the issue of non-determinism of the probability distribution, and shown that the extended p-automata are powerful enough to represent various probabilistic systems and logics. One of the salient aspect in which the extended p-automata differs from the original p-automata is that the number of configurations has become exponential in the size of the Markov chain (though the acceptance problem is still EXPTIME). Furthermore, the emptiness and hence language inclusion problem have become undecidable. In the wake of this undecidability results, we have provided a simulation relation which resembles the simulation relation of probabilistic automata. A comparison of the extended p-automata with the p-automata of [58] is presented in the Table 7.6.

Chapter 8

Conclusions and Discussion

Probabilistic logics embellish temporal logics with mechanism that can reason about probabilities of events. Probabilistic logics allow us to specify complex temporal properties over the probability space defined by the system under scrutiny. Probabilistic computation tree logic (or PCTL), one of the most widely recognized probabilistic logic, has been successfully used in expressing properties of probabilistic systems and efficient model checking algorithm have been developed in literature. But we still know next to nothing about what it means for a logical sentence to be true. Consequently we have no decision procedure for determining whether the specifications expressed in the logic are collectively contradictory, or whether one specifications implies the other, etc. This gap in our knowledge springs from our lack of understanding of the satisfiability problem (or the validity problem) of probabilistic logics. This is indeed a very hard problem and we are at the very primal stage of understanding its various intricacy.

In this thesis we have made attempts at tackling the satisfiability problem by considering various simple fragments of various probabilistic logics. One of the major challenges was to identify important fragments of probabilistic logics which fashion themselves to known techniques used for solving the validity problem. In that endeavour, we have focused our attention on *bounded* PCTL and PμTL. The relative expressive power of these logics is represented by the Hesse diagram in Figure 8.0.1. Bounded PCTL is a restriction on PCTL, where *until* formulas are replaced by bounded *untils*, whereas PμTL extends modal μ-calculus with probabilistic next operators.

- Bounded PCTL: The satisfiability problem is in the class NEXPTIME. The problem is at least EXPTIME-hard. Since the formulas are closed under negation, this implies that the problem is in

  NEXPTIME$\cap$co-NEXPTIME.

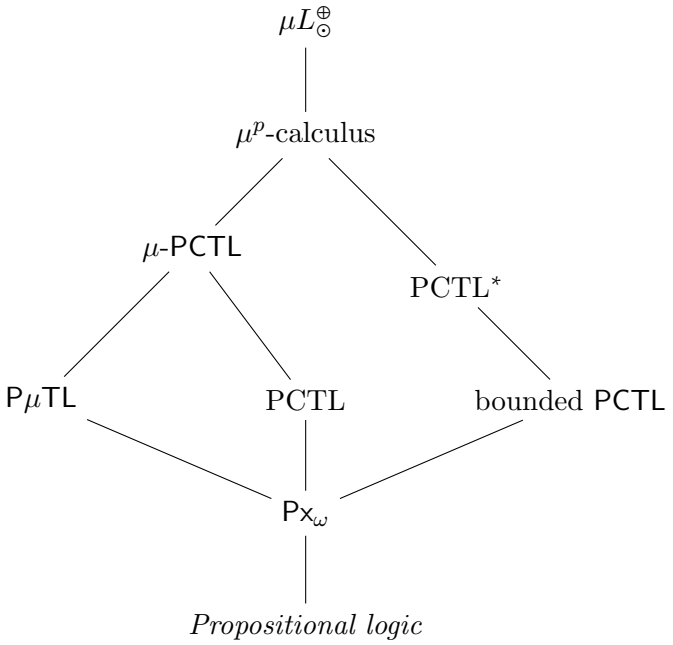


Figure 8.0.1: The relative expressive power of various logics.

- PμTL : The satisfiability problem is in the class

$$\mathsf{UTIME}(2^{O(n)}) \cap \mathsf{co\text{-}UTIME}(2^{O(n)}).$$

Chapter 5 (page 5) also considers other fragments of PCTL based on the classification such as *safety* or *co-safety*.

In a model-theoretical framework a sentence of a logic can be viewed as a set of models that satisfies the sentence. A decision procedure, generally involves devising a finite representation of these sets. Tableau method and subset construction are the usual methods that follow this paradigm of reasoning. In our investigation into the tableau method it was instructive to consider the tableau (which literally means a collection of models) as an Markov decision process, and the models of the sentences defined by the tableau as the Markov chains induced by different schedulers of the MDP. This lead to further investigation of model checking of convex MDPs with open intervals (Chapter 4, page 61).

Unfortunately, deciding whether an arbitrary MDP D has a scheduler η, such that the induced Markov chain D_η satisfies a given PCTL formula is equivalent to deciding the existence of a winning strategy for an $1\frac{1}{2}$-player game with PCTL winning objective. This problem cannot be solved by model checking (as discussed in Chapter 4) and is highly undecidable.

In the face of the above observation, it is natural to ask, if constructing tableau for probabilistic logics leads to instances of $1\frac{1}{2}$-player games with PCTL winning objective, then can any instance of the $1\frac{1}{2}$-player game be encoded as a satisfiability problem for some probabilistic logic? In other words, can we capture the set of Markov chains induced by an MDP as the satisfiability set of a sentence of some probabilistic logic?

The investigation into the representation of the set of Markov chains induced by various schedulers of an MDP lead to the study of p-automata. Automata theoretic approach to logics has proved itself quite useful in ascertaining decidability of various temporal logics. In the same spirit, p-automata were developed to provide automata theoretical understanding for probabilistic logics. In a recent work [19], an equivalence between p-automata and μ^p-calculus was formally established. However we were unsuccessful in capturing the sets of MDPs by any known logics (figure 8.0.1) and p-automata. In Chapter 7 (page 113) we were able to model MDPs, only after considerable tweaking of the semantics of p-automata.

Thus, the decidability of probabilistic logics is still very much an open problem, we were able to give definite answer only for few fringe cases. The subject has immense potential for new discoveries. For example, in the concluding remark of Chapter 6, we talked about the relation between the logic and obligatory games. The consequence of different restriction on obligatory games on the decidability of determining winning strategy would give us a handle on the decidability of corresponding probabilistic logics [28]. Recent workn in [68] fleshes out the difficulty in many of these decidability problems.

In a related research direction, there is quantitative logic [50], which is use to reason about minimal and maximal values of the weights of transitions in a weighted transition system. A complete axiomatization of the logics is also created, which is carried to PμTL in [71, 51]. A good introduction to the application of qualitative verification for Aerospace Systems can be found in [11].

Appendices

Appendix A

Parametric (0/1)-counter automata

In this chapter we improve the undecidability result of [3], where the reachability problem for automata with three (0/1) parametric counters was shown to be undecidable. Closer scrutiny of the proof reveals that the number of parameters (six in total) plays a crucial part in the encoding to the halting problem of a two counter machine. We can improve this result by showing that the reachability problem is still undecidable with three (0/1) counters and *one* parameter and two (0/1) parametric counters which can only be compared with a non-parametric (0/1) counter.

1.1 Preliminaries

Let $\mathcal{C}$ be a set of counters. The counters will be denoted by capital letters $\mathcal{C} := \{C_1, \cdots, C_k\}$ and their values by small letters. For example, if C_1 is a counter then its value is c_1, a non-negative integer. A tuple of counter values is denoted by a vector, $\vec{c} := (c_1, c_2, \cdots, c_k)$ and c_i is the counter value of the i^{th} counter in $\mathbf{c}$.

The set of operations and guards are denoted by O and G, respectively. We consider the following types of operations on a counter C, $O := \{+a(C), 0(C), \varnothing\}$. The operation $+a(C)$ adds a positive integer a to the current value of the counter C, $0(C)$ resets the value of the counter to zero and $\varnothing$ keeps the value of the counter unchanged. The set of guards $G := \{C \sim a, C \sim x, C \sim C'\}$, where a is a positive integer, x is a parameter and $\sim \in \{<, =, >\}$.

A *(0/1) counter automaton* A is a tuple $(S, s_0, \mathcal{C}, O, G, \delta, F)$, where S is a set of states, $s_0 \in S$ is the initial state, F is the set of accepting states ($F \subseteq S$) and $\mathcal{C}$ is the set of counters. The operation set $O = \{+a(C), 0(C), \varnothing\}$, the

guard set is $G = \{C \sim a\}$, where a is a positive integer and $\sim \in \{<, =, >\}$. δ is the transition relation, $\delta \subseteq S \times O \times G \times S$.

A *configuration* of the automaton is a tuple $(s, \vec{c})$, where s is a state and $\vec{c}$ is a vector of counter values. A *computation* (or a path) is a sequence of configurations $\pi : (s_0, \mathbf{c}^0) \to (s_1, \mathbf{c}^1) \to \cdots \to (s_n, \mathbf{c}^n)$, where for each $0 < i < n$, $(s_i, o, g, s_{i+1}) \in \delta$, $c^i_j = o(c^{i-1}_j)$ and $g(c^i_j)$ is true for each $j \leq |\mathcal{C}|$. π_i is the i^{th} configuration $(s_i, \mathbf{c}^i)$. The length of π is denoted by $|\pi|$. The reachability predicate $\mathsf{R}^A(s, \mathbf{c}, t, \mathbf{c}')$ defines the relation $\exists\ \pi : (s_0, \mathbf{c}^0) \to (s_1, \mathbf{c}^1) \to \cdots \to (s_n, \mathbf{c}^n)$ where $(s, \mathbf{c}) = (s_0, \mathbf{c}^0)$ and $(t, \mathbf{c}') = (s_0, \mathbf{c}^n)$ (More succinctly $\exists\ \pi : (s, \mathbf{c}) \to^* (t, \mathbf{c}')$). M_A be the maximum value of the constants in the guard. If $H \subseteq \delta$ then $A \setminus H$ is the automaton obtained from A after removing H from the set of transitions.

A *parametric (0/1) counter automaton* A is a tuple $(S, s_0, \mathcal{C}, O, G, \delta, F)$, where all definitions are the same as the (0/1) counter automaton except for the guard set $G = \{C \sim a, C \sim x\}$ where $a \in \mathbb{N}$, x is a parameter and $\sim \in \{=, <, >\}$. A counter C is called a *parametric counter* if A has a guard of the type $C \sim x$, else it is *non-parametric*. A valuation $\vec{v}$ assigns to each parameter an integer value. $A_{\vec{v}}$ is the counter automaton obtained from A by giving values to every parameter according to $\vec{v}$. A has a computation under the valuation $\vec{v}$, denote as $(s_0, \mathbf{c}^0) \to^*_{\vec{v}} (s_n, \mathbf{c}^n)$, if there is a computation $(s_0, \mathbf{c}^0) \to^* (s_n, \mathbf{c}^n)$ in $A_{\vec{v}}$. Given two configurations $(s, \mathbf{c})$ and $(t, \mathbf{c}')$, the reachability predicate $\mathsf{R}^A(s, \mathbf{c}, t, \mathbf{c}')$ is defined as $\exists \vec{v}, \pi : (s, \mathbf{c}) \to^*_{\vec{v}} (t, \mathbf{c}')$.

1.2 Three Parametric (0/1) Counter Automata

Theorem 1.2.1. *Let A be a counter automaton with three parametric (0/1) counters and a single parameter p. The reachability problem for A is undecidable.*

Proof. The undecidability of the reachability problem is shown by reducing a two counter machine U to an instance of a counter automaton A with three parametric (0/1) counters, two non-parametric (0/1)-counters and a single parameter p. Let $U := (S, \delta, s_0, \mathcal{C}, s_f)$, where S is the set of states, s_0 and s_f are the initial and final states, respectively, the set of counters $\mathcal{C} := \{C_1, C_2\}$. δ is the transition relation, where $\delta \subseteq S \times O \times G \times S$ and $s_0 \in S$ is the starting state and $s_f \in S$ is the final state. Since U is a two counter machine C_1, C_2 can increase as well as decrease. On the other hand, the counters of A (defined shortly) can only increase or reset to 0.

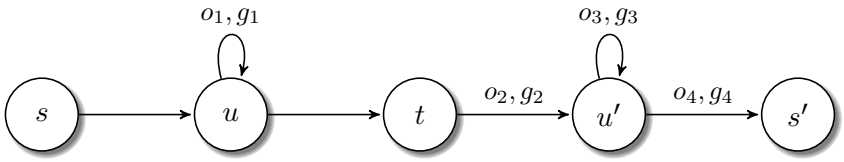
Consider the (0/1)-counter automaton $A = (S', \delta', s_0, \mathcal{C}', s_f)$, where $\mathcal{C}' = \{C'_1, C''_1, C''', C'_2, C''_2\}$ are (0/1) counters and a single parameter x. The set

o_1	$+1(C'_j)$, $+1(C''_j)$, $+1(C''')$	g_1	true
o_2	$0(C''')$	g_2	$(C''' = x)$, (If $a = \{C_j = 0\}$ then $(C''_j = x)$)
o_3	$+1(C'_j), +1(C''_j), +1(C''')$	g_3	true
o_4	$0(C''_j)$	g_4	(If $a = +1(C_j)$ then $g_4 = \{C''_j = x{+}1\}$) (If $a = -1(C_j)$ then $g_4 = \{C''_j = x{-}1\}$)

Table A.1: For each transition $s \xrightarrow{a} s'$ we have the gadget as shown above, where a can be $+1(C_j)$, $-1(C_j)$, or $C_j = 0$, for $j \in \{1, 2\}$.

S' contains a state for each state in S. To simplify notation, we use the same symbol for states $s \in S$ and the corresponding state in S'. S' contains other states as well for book keeping, which will be mentioned as we use them.

Each transition of U is mimicked by a gadget of A. Take a transition $s \xrightarrow{a} s'$ of U, where a stands for either $(+1(C_j))$, $(-1(C_j))$ or $(C_j = 0)$ for $j = 1, 2$. The corresponding part of A is given by the following gadget:



Depending on a, the description of o_i and g_i for $i \leq 4$ and counter C_j, $j = 1, 2$ is presented below.

The construction gives us two disjoint subsets of S', namely $Q, T \subseteq S'$. States in set Q has one-to-one mapping with the set of states S of U and the states in T are the t states added in the construction of the gadget. It follows that every sequence of states $s_0, s_1, \cdots, s_k$ of U has a corresponding sequence of states $s_0 \cdots t_0 \cdots s_1 \cdots t_1 \cdots\cdots s_{k-1} \cdots t_{k-1} \cdots s_k$ of A, where $s_i \in Q$, $t_i \in T$ for $0 \leq i \leq k$ and vice-versa. Assume $\rho : (s_0, \mathbf{c}_0) \to (s_1, \mathbf{c}_1) \to^* (s_k, \mathbf{c}_k)$ is a valid computation of U. We show that there exists a computation $\rho' : (s_0, \mathbf{c}_0) \to^*_p (t_0, \mathbf{c}'_0) \to^*_p (s_1, \mathbf{c}'_1) \to^*_p (s_k, \mathbf{c}'_k)$ of A for some value p of the parameter.

We will use the following functions: $\mathbf{n}_s : \mathcal{C} \times \mathbb{N} \to \mathbb{N}$, gives the value of the counter C_j at state s_i in ρ. Similarly, $\mathbf{n}_q : \mathcal{C}' \times \mathbb{N} \to \mathbb{N}$ and $\mathbf{n}_t : \mathcal{C}' \times \mathbb{N} \to \mathbb{N}$ give the value of the counter C'_j at the states s_i and t_i in the computations ρ', respectively. We have $\mathbf{n}_r(C, 0) = 0$ for all $r \in \{s, q, t\}$ and $C \in \mathcal{C} \cup \mathcal{C}' \setminus \{C'_1, C'_2\}$ and $\mathbf{n}_q(C'_j, 0) = p$ for $j \in \{1, 2\}$. (That is we start with initial value p for C'_1, C'_2.)

The value of C'_j in ρ' mimics the rise and fall of the value of the counter C_j in ρ, for $j = 1, 2$. When the value of C_j increases by 1, the value of C'_j in the corresponding gadget increases by p+1. When the value of C_j decreases by 1, the value of C'_j increases by p-1 and C'_j increases by p when C_j remains unchanged. The value counter C''' is p at every t state.
For an appropriate value p the following relation is imposed by the construction for $i \geq 0$:

$$\begin{array}{lcl} \mathbf{n}_q(C'_j, i) & = & \sum_{k=0}^{i}(\mathbf{n}_s(C_j, k) + p), \\ \mathbf{n}_q(C''_j, i) & = & 0, \\ \mathbf{n}_q(C''', i) & = & \mathbf{n}_q(C'_j, i) \mod p. \end{array} \tag{A.1}$$

We show that the above equation holds for every i^{th} step of a computation $0 \leq i \leq k$. For $i = 0$ the statement follows from the initial values of the counters. We have the following cases:

(i) The counter C_j is incremented in the transition $s_i \xrightarrow{+1(C_j)} s_{i+1}$ in ρ. The corresponding part of ρ' is $s_i \rightarrow^* t_i \rightarrow^* s_{i+1}$. Let the value of C'_j at s_i be n, thus value $\mathbf{n}_q(C''_j, i) = 0$ and $\mathbf{n}_q(C''', i) = n \mod p$ (by equation A.1). At state t_i, the value of counter C''' is $\mathbf{n}_t(C''', i) = p$, owing to the guard g_2. Hence the values of other counters are $\mathbf{n}_t(C''_j, i) = p-(n \mod p)$ and $\mathbf{n}_t(C'_j, i) = n+p-(n \mod p)$. By the construction, the value of the counter C''_j just before arriving at the state s_{i+1} is p+1 (it is reseted after reaching s_{i+1}). This is imposed by the guard g_4 and hence the value of the counters C'_j, C''' at s_{i+1} is $\mathbf{n}(C'_j, i+1) = p+1+n$ and $\mathbf{n}'(C''', i+1) = p+1-((p-n) \mod p) \equiv (n+1) \mod p$. Thus the equation (A.1) holds at s_{i+1}.

(ii) Similar argument holds for the transition where the counter value decreases or remains unchanged.

(iii) The value of the counter is compared with zero ($s_i \xrightarrow{C_j=0} s_{i+1}$). When the value of the counter C_j in the computation ρ becomes zero at the state s_i, i.e., $\mathbf{n}_s(C_j, i) = 0$, the value of C'_j at the state s_i in ρ', is a multiple of p, i.e., $\mathbf{n}_q(C'_j, i) \equiv 0 \mod p$ (equation A.1). This is deduced by checking $g_2 : (C''_i = p)$ at state t_i.

Finally, observe that the counters C'_1 and C'_2 are non-parametric and can be removed (similar to region construction for timed automata). This gives us the automaton A with 3 counters such that each halting computation of U has a corresponding halting computation of A and vice-versa. □

Corollary. *Let A be counter automaton with one non-parametric counter (whose value can increase or decrease) and two parametric (0/1) counters with a parameter p. The reachability problem for A is undecidable.*

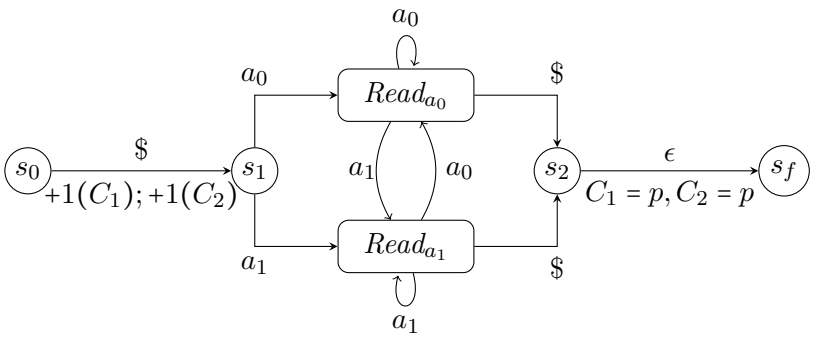


Figure 1.2.1: Counter automaton that solves PCP for two symbols a_0 and a_1.

Proof. Let C_1', C_2' be the parametric (0/1) counters of A and C_3' be the non-parametric counter (whose value can increase or decrease). Consider any two counter machine with counters C_1, C_2. Using similar construction as described in proof of the Theorem 1.2.1, the (0/1) counters C_1', C_2' along with a parameter p can mimic the behaviour of C_1. C_3' can mimic the behaviour of C_2. □

We conclude this chapter by showing that the reachability problem for a counter automaton with two (0/1) parametric counters which can only be compared with a non-parametric (0/1) counter is also undecidable with one parameter.

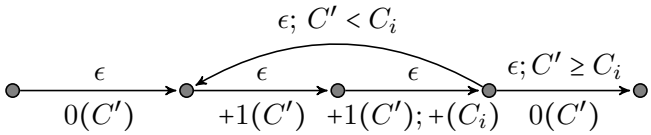
Theorem 1.2.2. *Let A be counter automaton with two parametric (0/1) counters C_1, C_2 and one non-parametric (0/1) counter C', with a guard set $G := \{C' < C_1, C' < C_2, C_1 = p, C_2 = p\}$. The language emptiness problem for A is undecidable.*

Proof. We provide a reduction from *Post's correspondence problem* [84]. Given two maps, $\phi_1 : \Sigma \to [0,1]^*$ and $\phi_2 : \Sigma \to [0,1]^*$, the problem is to decide whether there exists a word $w \in \Sigma^*$, such that $\phi_1(w) = \phi_2(w)$, where $\phi_i(w.a) = \phi_i(w) \cdot \phi_i(a)$ (for $i \in \{1,2\}$). Define l_a^i to be the length of the string $\phi_i(a)$. An equivalent formulation is as follows: Whether there exists a word $w := \$a_0 \cdot a_1 \cdots a_k\$$, such that $\phi_1(a_0).\phi_1(a_1) \cdots \phi_1(a_k) = \phi_2(a_0).\phi_2(a_1) \cdots \phi_2(a_k)$.

The counter automaton A behaves the following way. For an input string $\$w\$$, the automaton adds 1 to the counter C_1 and C_2 with input \$ and then stores the decimal representation of $\phi_i(w)$ in the counter C_i (for $i \in \{1,2\}$). E.g. consider $w := a_0a_1$ and $\phi_i(a_0) := 010, \phi_i(a_1) := 01$. At the end of the execution, the value of C_i should be 41 which is equivalent to 1 010 01 in binary (a leading 1 is introduced by the first \$ symbol).

The automaton is constructed the following way. Whenever a letter $a \in \Sigma$ is read by the automaton, the current value in the counter C_i is shifted to the left to accommodate $\phi_i(a)$. This is accomplished by shifting the bits of C_i by l_a^i places to the left or equivalently, the value of C_i is multiplied with $2^{l_a^i}$ and then $\phi_i(a)$ is added to it.

The gadget, $Read_a$ for the letter $a \in \Sigma$, does exactly this. It is always entered with an symbols a and all its internal symbols are ϵ. The following part of the gadget shifts one bit of the value stored in C_i by doubling it.



The counter C' starts with value 0. Each time the loop is executed, the value of C' increases by 2 and C_i increases by 1. Suppose the value of C_i before entering the loop was k, hence the loop is exited after k times. Thus the value of C_i is $2k$ when the execution exists the loop. The $Read_a$ first repeats this process l_a^i times for each C_i, then it adds the constant $(\phi_i(a))_{10}$ (the decimal value of $\phi_i(a)$) to each C_i (for $i = 1, 2$).

The automaton A for two symbols a_0 and a_1 is presented in Figure 1.2.1. When the symbol \$ is read for the second time, the automaton checks whether the two counter values are equal with a help of the parameter p. □

Appendix B

EXPTIME-hardness of bounded PCTL

Proposition 2.0.1. *The satisfiability of bounded* PCTL*formula is EXPTIME-hard in the encoding of the formula.*

Proof. We will show EXPTIME-hardness by encoding computations of an *alternating Turing machine* in bounded PCTL. Similar technique was also used in [44] to show EXPTIME harness for PDL. An *alternating Turing machine* (ATM) [25] is just like a non-deterministic Turing machine except there is a function in the specification of the machine called type. The function type tells us whether a state is an *and-state* or an *or-state*. An ATM with only or-states behaves exactly like a non-deterministic Turing machine. Formally, an ATM is a seven tuple $A = (Q, \Theta, \Gamma, \delta, q_0, \mathsf{type}, F)$. Q is a finite set of states, Θ is a finite set of input symbols, Γ is a finite set of tape symbols ($\Theta \subseteq \Gamma$), $\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$ is a transition relation, $q_0 \in Q$ is a initial state, $\mathsf{type} : Q \to \{\wedge, \vee\}$, $F \subseteq Q$ is the set of accepting states.

Configurations $\sigma = xqay \in \Gamma^* \times Q \times \Gamma^+$, where the tape content is $xay = \mathsf{tape}(\sigma) \in \Gamma^+$ with $a \in \Gamma$, the head is at position $|x| + 1$, presently reading input a and the current state is $q = \mathsf{state}(\sigma)$. A configuration σ is an *and-configuration (or-configuration)* iff type of $\mathsf{state}(\sigma)$ is $\wedge$ ($\vee$, resp.). σ is *accepting* iff $\mathsf{state}(\sigma) \in F$. For $\sigma = xqay$ the *next* configuration $\sigma' = x'q'a'y'$ is defined as follows:

- If $(q, a, q', b, L) \in \delta$ then $x'a' = x$ and $y' = by$.
- If $(q, a, q', b, R) \in \delta$ then $x' = xb$ and $y = a'y'$.

A *trace* (or a computation) C of A for an input x_{in} is a set of configuration such that, $q_0 x_{in} \in C$ and for every $\sigma \in C$ with $\mathsf{state}(\sigma) \notin F$, if $\mathsf{type}(\mathsf{state}(\sigma)) =$

$\vee$ then one of the *next* configurations σ' of σ is in C, if $\mathsf{type}(\mathsf{state}(\sigma)) = \wedge$ then every next configuration of σ is in C. Pictorially, C is a tree where each node is a configuration and edges are defined by the *next* relation. A trace C is accepting for an input x if C is finite and only configurations without a next configuration in C are accepting. Let:

$$L(A) = \{x \in \Theta^* \mid \text{ there exists an accepting trace } C \text{ for } x \}$$

For some function $S : \mathbb{N} \to \mathbb{N}$, an ATM A is in $\mathsf{ASPACE}(S(n))$ iff for every input $x \in \Theta^*$, and every configuration of every trace of x requires at most $S(|x|)$ space. Furthermore, we assume that no configuration is repeated in any trace C of x. This is ensured by enumerating every reachable configuration and the numbering can be encoded into $S(|x|)$ cells of the tape. Thus, the number of steps is less than $|\Gamma|^{2S(n)}$ or in $2^{O(S(n))}$, where $n = |x|$. We will need the following identity [25]:

$$\mathsf{ASPACE}(S(n)) = \bigcup_k \mathsf{DTIME}(2^{kS(n)}). \tag{B.1}$$

Now consider an input x of length n to an ATM $A \in \mathsf{ASPACE}(S(n))$, where $m = S(n) + 2$ and the maximum number of steps needed by the machine to accept (or reject) is $k = 2^m$. Observe that k can be encoded in m space. We will construct a bounded PCTLformula from A and x such that every model of the formula encodes a computation of A with input x iff $x \in L(A)$. Each node of the model will encode a configuration of the computation and the relation *next* will be simulated by $\Box f$ or $\Diamond f$ (i.e., $[\mathsf{X}f]_{=1}$ or $[\mathsf{X}f]_{>0}$, respectively). We will use the following set of propositions AP:

1. Cell proposition: for each $a \in \Gamma$ and $0 \le i \le m$, $C_{a,i} \in \mathsf{AP}$.
2. State proposition: for each $q \in Q$, $Q_q \in \mathsf{AP}$.
3. Head proposition: for each $0 \le i \le n$, $H_i \in \mathsf{AP}$.

Intuitively, $C_{a,i}$ denotes that the i^{th} cell of the tape contains symbol a, Q_q denotes that the current state is q and H_i denotes that the head is on the i^{th} cell. We will use the following formula to correctly capture the behaviour of A.

- One state proposition Q_q is true at every node of the model:

$$g_1 := \bigvee_{q \in Q} \left(Q_q \wedge \bigwedge_{q' \in Q \setminus \{q\}} \sim Q_{q'} \right)$$

- One cell proposition is true for any particular $i \leq m$.

$$g_2 \;\coloneqq\; \bigwedge_{i=0}^{m} \bigvee_{a \in \Gamma} \left(C_{a,i} \wedge \bigwedge_{a' \in \Gamma \setminus \{a\}} \sim C_{a',i} \right)$$

- One head proposition is true at any node of the model. Head cannot cross the first and the last cells.

$$g_3 \;\coloneqq\; \bigvee_{i=1}^{m-1} \left(H_i \wedge \bigwedge_{j \neq i} \sim H_j \right) \wedge \sim H_0 \wedge \sim H_m$$

- Unread cell propositions remain unchanged in the next node of the model.

$$g_4 \;\coloneqq\; \bigwedge_{i=0}^{m} \bigwedge_{a \in \Gamma} \left(\sim H_i \wedge C_{a,i} \rightarrow \Box C_{a,i} \right)$$

- Transition relation for and-states.

$$g_5 \coloneqq \bigwedge_{i=1}^{m-1} \bigwedge_{a \in \Gamma} \bigwedge_{\mathsf{type}(q)=\wedge} \left(H_i \wedge C_{a,i} \wedge Q_q \rightarrow \bigwedge_{(q,a,q',b,R) \in \delta} \Diamond (H_{i+1} \wedge C_{i,b} \wedge Q_{q'}) \wedge \bigwedge_{(q,a,q',b,L) \in \delta} \Diamond (H_{i-1} \wedge C_{i,b} \wedge Q_{q'}) \right)$$

- Transition relation for or-states.

$$g_6 \coloneqq \bigwedge_{i=1}^{m-1} \bigwedge_{a \in \Gamma} \bigwedge_{\mathsf{type}(q)=\vee} \left(H_i \wedge C_{a,i} \wedge Q_q \rightarrow \bigvee_{(q,a,q',b,R) \in \delta} \Diamond (H_{i+1} \wedge C_{i,b} \wedge Q_{q'}) \vee \bigvee_{(q,a,q',b,L) \in \delta} \Diamond (H_{i-1} \wedge C_{i,b} \wedge Q_{q'}) \right)$$

- The accepting nodes of the model satisfy the following formula:

$$g_F \;\coloneqq\; \bigvee_{q \in F} Q_q$$

- Let the input $x = a_0, \cdots, a_n$, and b be the symbol for blank space. The initial configuration is defined as follows:

$$g_{in} \;\coloneqq\; Q_{q_0} \wedge H_1 \wedge \bigwedge_{i \neq 1} \sim H_i \wedge \bigwedge_{i=1}^{n} C_{a_i,i} \wedge C_{b,0} \wedge \bigwedge_{i=n+1}^{m} C_{b,i}$$

Let $g = \bigwedge_{i=1}^{6} g_i$. Thus, the required bounded PCTL formula is defined as follows:

$$f \;\coloneqq\; g_{in} \wedge [g \,\mathsf{U}^{\,k}\, g_F]_{=1}.$$

The correctness of the translation can be checked by inspection, since there is a one-to-one correspondence between the models of f and computations of A on input x. Observe that the encoding takes $O((|\Gamma| + |Q| + |\delta|)S(n))$ time. If $S(n)$ is a polynomial function then f is constructed in polynomial time of the size of A and x. Furthermore, if $S(n)$ is polynomial in n then $A \in \mathsf{EXPTIME}$ (equation (B.1)). This leads to the Proposition 2.0.1. □

Appendix C

Variable elimination

Consider the ring of polynomials $D[X]$ in the *integral domain* D, where X is the set of indeterminates (or variables). A polynomial $p(x_1, \cdots, x_n)$, with variables $x_1, \cdots, x_n \in X$, is seen as a sum of products with nonzero coefficients in D, where each $x_1^{d_1} \cdots x_n^{d_n}$ is called a *term*; together with its coefficient it is called a *monomial*; the *degree* of the term $x_1^{d_1} \cdots x_n^{d_n}$ is $d_1 + \cdots + d_n$; *degree* of a polynomial is the maximum degree of its terms. A polynomial is multivariate if $|X| > 1$. The ring of multivariate polynomials $D[X]$ can be viewed as a ring of univariate polynomials $D[X \setminus \{x\}][x]$ with coefficients in the integral domain $D[X \setminus \{x\}]$ ([10] page 63, Theorem 2.). Particularly, the degree of a term of a polynomial in $D[X \setminus \{x\}][x]$ is the power of x in that term.

$E(D[X])$ is the set of (in)equations (e.g $x_1^2 - x_2 \geq 0.4$) where the left hand side (lhs) is a polynomial (e.g. $x_1^2 - x_2$) in $D[X]$ and the right hand side (e.g. 0.4) is in D. A variable x is *independent* of $H \subseteq E(D[X])$ iff $H = H \cap E(D[X \setminus \{x\}])$ else it is *dependent*. The *quotient domain* $\mathcal{Q}(D)$ is the rational form of the type $\frac{f}{g}$ where $f, g \in D$.

A weighted tree T is a triple (V, E, w), where V is the set of vertices, $E \subseteq V \times V$ is the set of edges and w is an injective weight function from $E \to \mathcal{V}$, where $\mathcal{V}$ is a set of variables. Let $X = \mathsf{img}(w)$. Define relations next and parent as follows; for $x, y \in X$, $v, v', v_1, v_2 \in V$, with $w^{-1}(x) = (v_1, v)$ and $w^{-1}(y) = (v', v_2)$, $(x, y) \in \mathsf{next}$ iff $v = v'$, and $(x, y) \in \mathsf{parent}$ iff $v_1 = v'$. next^+ is the transitive closure of next. Consider a term $\sigma = x_1 \cdots x_k$ such that for every $1 \leq i < k$, $(x_i, x_{i+1}) \in \mathsf{next}$. Define $\mathsf{head}(\sigma) = x_1$, $\mathsf{tail}(\sigma) = x_k$ and $x_i \cdots x_k$ as a suffix of σ, for $1 \leq i \leq k$. Let $H \subseteq E(\mathbb{Q}[X])$ be a set of

(in)equations with the following properties. For each $\xi \in H$:

$P1$. For all $x \in X$, $\mathsf{lhs}(\xi) \in \mathbb{Q}[X \setminus \{x\}][x] \rightarrow \mathsf{degree}(\xi) \leq 1$
$P2$. For each term $\sigma = x_1 \cdots x_k$ in ξ, $(x_i, x_{i+1}) \in \mathsf{next}$.
$P3$. If $\mathsf{lhs}(\xi) = a_1\sigma_1 + \cdots + a_k\sigma_k$, where $a_i \in \mathbb{Q}$ and σ_i are terms, then for all $1 \leq i, j \leq k$, $(\mathsf{head}(\sigma_i), \mathsf{head}(\sigma_j)) \in \mathsf{parent}$.

Suppose $H \subseteq E(\mathbb{Q}[X])$ satisfies properties *P1, P2* and *P3* and let n be the number of variables and m be the number of (in)equations in H. We only consider positive variable valuations. Thus for every variable x we have the in-equation $x > 0$ in H. We present a non-deterministic algorithm to decide whether H is satisfiable. We begin by setting $H_0 = H$ and at each iteration i, we eliminate a (particular) variable, say x and transform the set of equations from $H_i \subseteq E(\mathbb{Q}[X])$ to $H_{i+1} \subseteq E(\mathbb{Q}[X \setminus \{x\}])$. We consider comparisons $\bowtie$ to be of the type $\{\geq, =, \leq\}$. (Strict inequalities can be removed by adding very small positive quantity ϵ. For example $f < g$ can be transformed to $f + \epsilon \leq g$.) The algorithm proceeds in the following steps:

1. If H_i is independent of all variables, then each (in)equation, involves only rational numbers (and $\epsilon \rightarrow^+ 0$)[1]. Return true iff each (in)equality in H_i is true.

2. Choose a variable x such that every variable y with $(x, y) \in \mathsf{next}^+$, is independent of H_i.

3. H_x is the largest subset of H_i such that every formula in H_x is dependent on x. If H_x is empty then $H_{i+1} = H_i$. Suppose H_x is not empty, every inequation $\xi \in H_x$ can be transformed to a form $(\sigma x \bowtie a_0 + a_1\sigma_1 + \cdots + a_k\sigma_k)$, where $\sigma, \sigma_1, \cdots, \sigma_k$ are terms in $\mathbb{Q}[X \setminus \{x\}]$ and $a_0, \cdots, a_k \in \mathbb{Q}$. We will denote this form by $f \cdot x \bowtie g$. Set $H_{i+1} = H_i \setminus H_x$.

4. Define $\Lambda_\bowtie \subseteq \mathcal{Q}(\mathbb{Q}[X \setminus \{x\}])$, for $\bowtie \in \{\leq, =, \geq\}$ as follows:

$$\begin{array}{ll} \Lambda_\leq := \{\frac{g}{f} \mid (f \cdot x \leq g) \in H_x\} \cup \{1\}, & \text{quotients that are at least as large as } x \\ \Lambda_= := \{\frac{g}{f} \mid (f \cdot x = g) \in H_x\}, & \text{quotients that are equal } x \\ \Lambda_\geq := \{\frac{g}{f} \mid (f \cdot x \geq g) \in H_x\} \cup \{\epsilon\} & \text{quotients that are at least as small as } x, \end{array}$$

where $g = a_0 + a_1\sigma_1 + \cdots + a_k\sigma_k$ and $f = \sigma$.

[1] ϵ tends to 0 from the positive side.

5. Non-deterministically choose an ordering of elements in $\Lambda_{\leq}$ and $\Lambda_{\geq}$. Then we have the following set of (in)equations:

$$\frac{g_1}{f_1} \leq \cdots \leq \frac{g_{n_1}}{f_{n_1}} \leq \frac{g_{n_1+1}}{f_{n_1+1}} = \cdots = \frac{g_{n_2}}{h_{n_2}} \leq \frac{g_{n_2+1}}{f_{n_2+1}} \leq \cdots \leq \frac{g_{n_3}}{f_{n_3}} \tag{C.1}$$

where, $\frac{g_i}{f_i}$ is in $\Lambda_{\leq}$ for $1 \leq i \leq n_1$, in $\Lambda_{=}$ for $n_1 + 1 \leq i \leq n_2$ and in $\Lambda_{\geq}$ for $n_2 + 1 \leq i \leq n_3$.

6. For each $1 \leq j \leq n_3$, we have $\xi_j := (g_j f_{j+1} \bowtie g_{j+1} f_j)$. ξ'_j is obtained from ξ_j by canceling variables that are common divisors of the polynomials in the left hand side and in the right hand side of ξ_j. Add ξ'_j to H_{i+1} for each ξ_j $(1 \leq j \leq n_3)$. Go to step 1.

First we will show that H_{i+1} created in step 6, satisfies *P1*, *P2* and *P3*. Consider,

$$\frac{a_0 + a_1\sigma_1 + \cdots + a_k\sigma_k}{\sigma} \bowtie \frac{b_0 + b_1\sigma'_1 + \cdots + b_l\sigma'_l}{\sigma'} \tag{C.2}$$

Let $\xi := (\sigma{\cdot}x \bowtie a_0 + a_1\sigma_1 + \cdots + a_k\sigma_k)$, $\xi' := (\sigma'{\cdot}x \bowtie b_0 + b_1\sigma'_1 + \cdots + b_l\sigma'_l)$ and $\xi, \xi' \in H_i$ satisfy *P1*, *P2* and *P3*. From the choice of the variable x (step 2), it is evident that either $\sigma|\sigma'$ or $\sigma'|\sigma$ ($a|b$ means a divides b). W.l.o.g let us assumed $\sigma''\sigma' = \sigma$. The crucial observation is that if $\sigma'|\sigma$ then σ' is a suffix of σ, lest there should exist a variable y, such that $(x, y) \in \mathsf{next}$ and y is not independent of H_i.

Therefore, equation (C.2) can be rewritten as:

$$a_0 + a_1\sigma_1 + \cdots + a_k\sigma_k \;\bowtie\; b_0\sigma'' + b_1\sigma''\sigma'_1 + \cdots + b_l\sigma''\sigma'_l. \tag{C.3}$$

P3 holds for equation (C.3), this follows trivially, as $\mathsf{head}(\sigma) = \mathsf{head}(\sigma_i) = \mathsf{head}(\sigma'')$ for $1 \leq i \leq k$. $(\mathsf{tail}(\sigma''), \mathsf{head}(\sigma')) \in \mathsf{next}$, since $\sigma = \sigma''\sigma'$ and $(\mathsf{head}(\sigma'_i), \mathsf{head}(\sigma'_j)) \in \mathsf{parent}$ for all $1 \leq i, j \leq l$. Thus, the new equations added to H_{i+1} (after canceling common variables) also satisfy $P1$ and $P2$ (cancellation is valid since variables can only take positive value).
Correctness of the algorithm is due to the following arguments:

1. Suppose H_i is feasible and let ν be a satisfying valuation of the variables. Then there exists some order among the rational numbers obtained by substituting the values of the variables in the quotients $\{\frac{g(x_1,\cdots,x_n)}{f(x_1,\cdots,x_n)}\}$ present in $\Lambda_{\leq}$ and $\Lambda_{\geq}$. If we choose this order as the ordering in the equation (C.1) and obtain H_{i+1} subsequently, then ν is also a satisfying valuation for (in)equations H_{i+1}.

2. If H_{i+1} is satisfiable then the (in)equations (C.1) are true for some value of $X \setminus \{x\}$. If $\Lambda_{=}$ is not empty then set $x = \frac{g_{n_2}}{f_{n_2}}$, else choose a

value for x such that $\frac{g_{n_1}}{f_{n_1}} \leq x \leq \frac{g_{n_2+1}}{f_{n_2+1}}$. The value thus chosen is strictly greater than 0, since $\epsilon \in \Lambda_{\geq}$.(Hence, rational form and cancellation of variables defined in step 5 and step 6, respectively is valid.) This gives us a satisfying valuation of H_i.

Observe that at each iteration i, the size of H_i is $O(|H|)$ and in every iteration we remove one variable and spend $O(mn)$ in obtaining H_{i+1} (modulo division of rational numbers). The maximum number of iteration is n and total time complexity of the non-deterministic algorithm is $O(mn^2)$. Thus satisfiability of a set of polynomial equation with properties *P1*, *P2* and *P3* is in NP.

Bibliography

[1] Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.

[2] Rajeev Alur, Kousha Etessami, Salvatore La Torre, and Doron Peled. Parametric temporal logic for "model measuring". *ACM Trans. Comput. Log.*, 2(3):388–407, 2001.

[3] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *In Proceedings of the 25th Annual Symposium On Theory Of Computing*, pages 592–601. ACM Press, 1993.

[4] Rajeev Alur and Salvatore La Torre. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Log.*, 5(1):1–25, 2004.

[5] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[6] Mordechai Ben-Ari, Amir Pnueli, and Zohar Manna. The temporal logic of branching time. *Acta Inf.*, 20:207–226, 1983.

[7] Michael Benedikt, Rastislav Lenhardt, and James Worrell. LTL model checking of interval Markov chains. In *TACAS*, volume 7795 of *LNCS*, pages 32–46. Springer, 2013.

[8] Nathalie Bertrand, John Fearnley, and Sven Schewe. Bounded satisfiability for PCTL. In *CSL*, volume 16 of *LIPIcs*, pages 92–106. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[9] Andrea Bianco and Luca de Alfaro. Model checking of probabalistic and nondeterministic systems. In *FSTTCS*, volume 1026 of *LNCS*, pages 499–513. Springer, 1995.

[10] Garrett Birkhoff and Saunders Mac Lane. *A Survey of Modern Algebra*. AKP classics. A.K. Peters, 1997.

[11] Marco Bozzano, Harold Bruintjes, Alessandro Cimatti, Joost-Pieter Katoen, Thomas Noll, and Stefano Tonetta. *Formal Methods for Aerospace Systems*, pages 133–159. Springer Singapore, Singapore, 2017.

[12] T. Brazdil, V. Brozek, V. Forejt, and A. Kucera. Stochastic games with branching-time winning objectives. In *21st Annual IEEE Symposium on Logic in Computer Science*, pages 349–358, 2006.

[13] Tomás Brázdil, Vojtech Forejt, Jan Kretínský, and Antonín Kucera. The satisfiability problem for probabilistic CTL. In *LICS*, pages 391–402. IEEE CS, 2008.

[14] Daniel Bundala and Joel Ouaknine. Advances in parametric real-time reasoning. *Mathematical Foundations of Computer Science*, 2014.

[15] J.Richard Büchi. On a decision method in restricted second order arithmetic. In Saunders Mac Lane and Dirk Siefkes, editors, *The Collected Works of J. Richard Büchi*, pages 425–435. Springer New York, 1962.

[16] Benoît Caillaud, Benoît Delahaye, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, and Andrzej Wasowski. Constraint Markov chains. *Theoretical Computer Science*, 412(34):4373 – 4404, 2011.

[17] John Canny. Some algebraic and geometric computations in PSPACE. In *STOC*, pages 460–467. ACM, 1988.

[18] Pablo F. Castro, Cecilia Kilmurray, and Nir Piterman. Tractable probabilistic mu-calculus that expresses probabilistic temporal logics. In *STACS*, volume 30 of *LIPIcs*, pages 211–223. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[19] Claudia Cauli and Nir Piterman. Equivalence of probabilistic mu-calculus and p-automata. In *Implementation and Application of Automata*, pages 64–75. Springer International Publishing, 2017.

[20] Souymodip Chakraborty and Joost-Pieter Katoen. Parametric LTL on markov chains. In *Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*, pages 207–221, 2014.

[21] Souymodip Chakraborty and Joost-Pieter Katoen. Model checking of open interval Markov chains. In *ASMTA*, volume 9081 of *LNCS*, pages 30–42. Springer, 2015.

[22] Souymodip Chakraborty and Joost-Pieter Katoen. P-automata for markov decision processes. In *3rd International Workshop on Strategic Reasoning, St Catherine's College, University of Oxford, Sep 17*, 2015.

[23] Souymodip Chakraborty and Joost-Pieter Katoen. On the satisfiability of some simple probabilistic logics. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 56–65, 2016.

[24] Souymodip Chakraborty, Joost-Pieter Katoen, Falak Sher, and Martin Strelec. Modelling and statistical model checking of a microgrid. *STTT*, 17(4):537–554, 2015.

[25] Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

[26] Krishnendu Chatterjee, Tom Henzinger, and Koushik Sen. Model-checking omega-regular properties of interval Markov chains. In Roberto M. Amadio, editor, *Foundations of Software Science and Computation Structure (FoSSaCS) 2008*, pages 302–317, March 2008.

[27] Krishnendu Chatterjee, Marcin Jurdziński, and Thomas A. Henzinger. Quantitative stochastic parity games. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 121–130, 2004.

[28] Krishnendu Chatterjee and Nir Piterman. Obligation blackwell games and p-automata. *J. Symb. Log.*, 82(2):420–452, 2017.

[29] Edmund M. Clarke, Jr., Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999.

[30] Anne Condon. The complexity of stochastic games. *Information and Computation*, 96:203–224, 1992.

[31] Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, July 1995.

[32] J.-M. Couvreur, N. Saheb, and G. Sutre. An optimal automata approach to LTL model checking of probabilistic systems. In *LPAR*, volume 2850 of *LNCS*, pages 361–375. Springer, 2003.

[33] Conrado Daws. Symbolic and parametric model checking of discrete-time Markov chains. In *ICTAC*, volume 3407 of *LNCS*, pages 280–294. Springer, 2005.

[34] Luca de Alfaro and Rupak Majumdar. Quantitative solution of omega-regular games. *J. Comput. Syst. Sci.*, 68(2):374–397, 2004.

[35] Christian Dehnert, Sebastian Junges, Nils Jansen, Florian Corzilius, Matthias Volk, Harold Bruintjes, Joost-Pieter Katoen, and Erika Ábrahám. Prophesy: A probabilistic parameter synthesis tool. In *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*, pages 214–231, 2015.

[36] E. Allen Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 995–1072. 1990.

[37] E. Allen Emerson and Edmund M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Sci. Comput. Program.*, 2(3):241–266, 1982.

[38] E. Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *J. Comput. Syst. Sci.*, 30(1):1–24, 1985.

[39] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 368–377, 1991.

[40] E. Allen Emerson and Richard J. Trefler. Parametric quantitative temporal reasoning. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, pages 336–343, 1999.

[41] E.Allen Emerson, Charanjit S. Jutla, and A.Prasad Sistla. On model checking for the μ-calculus and its fragments. *Theoretical Computer Science*, 258(1–2):491 – 522, 2001.

[42] Harald Fecher, Michael Huth, Nir Piterman, and Daniel Wagner. Hintikka games for PCTL on labeled markov chains. In *Fifth International Conference on the Quantitative Evaluaiton of Systems (QEST 2008), 14-17 September 2008, Saint-Malo, France*, pages 169–178, 2008.

[43] Diana Fischer, Erich Grädel, and Lukasz Kaiser. Model checking games for the quantitative μ-calculus. *Theory Comput. Syst.*, 47(3):696–719, 2010.

[44] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. of Comput. and Syst. Sci.*, 18(2):194 – 211, 1979.

[45] Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal basis of fairness. In *Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages, Las Vegas, Nevada, USA, January 1980*, pages 163–173, 1980.

[46] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *STOC*, pages 60–65. ACM, 1982.

[47] Ernst Moritz Hahn, Tingting Han, and Lijun Zhang. Synthesis for PCTL in parametric Markov decision processes. In *NFM*, volume 6617 of *LNCS*, pages 146–161. Springer, 2011.

[48] Ernst Moritz Hahn, Holger Hermanns, and Lijun Zhang. Probabilistic reachability for parametric Markov models. *STTT*, 13(1):3–19, 2011.

[49] Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Approximate parameter synthesis for probabilistic time-bounded reachability. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 173–182. IEEE Computer Society, 2008.

[50] Mikkel Hansen, Kim Guldstrand Larsen, Radu Mardare, Mathias Ruggaard Pedersen, and Bingtian Xue. A complete approximation theory for weighted transition systems. In Martin Fränzle, Deepak Kapur, and Naijun Zhan, editors, *Dependable Software Engineering: Theories, Tools, and Applications*, pages 213–228, Cham, 2016. Springer International Publishing.

[51] Mikkel Hansen, Kim Guldstrand Larsen, Radu Mardare, Mathias Ruggaard Pedersen, and Bingtian Xue. Reasoning about bounds in weighted transition systems. *CoRR*, abs/1703.03346, 2017.

[52] Hans Hansson and Bengt Jonsson. A framework for reasoning about time and reliability. In *Proceedings of the Real-Time Systems Symposium - 1989, Santa Monica, California, USA, December 1989*, pages 102–111, 1989.

[53] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[54] Hans A. Hansson. *Time and Probability in Formal Design of Distributed Systems*. Elsevier Science Inc., New York, NY, USA, 1994.

[55] Sergiu Hart and Micha Sharir. Probabilistic propositional temporal logics. *Inf. and Control*, 70(2/3):97–155, 1986.

[56] Jaakko Hintikka. Game-theoretical semantics: insights and prospects. *Notre Dame Journal of Formal Logic*, 23(2):219–241, 1982.

[57] Michael Huth and Marta Z. Kwiatkowska. Comparing CTL and PCTL on labeled markov chains. In *Programming Concepts and Methods, IFIP TC2/WG2.2,2.3 International Conference on Programming Concepts and Methods (PROCOMET '98) 8-12 June 1998, Shelter Island, New York, USA*, pages 244–262, 1998.

[58] Michael Huth, Nir Piterman, and Daniel Wagner. p-automata: New foundations for discrete-time probabilistic verification. *Performance Evaluation*, 69(7–8):356 – 378, 2012. Selected papers from {QEST} 2010.

[59] David Janin and Igor Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In *CONCUR '96, Concurrency Theory, 7th International Conference, Pisa, Italy, August 26-29, 1996, Proceedings*, pages 263–277, 1996.

[60] Bengt Jonsson and Kim Guldstrand Larsen. Specification and refinement of probabilistic processes. In *LICS*, pages 266–277. IEEE Computer Society, 1991.

[61] N.N. Katerinochkina. Sets containing a maximal number of pairwise incomparable n-dimensional k-ary sets. *Mathematical notes of the Academy of Sciences of the USSR*, 24(3):696–700, 1978.

[62] Joost-Pieter Katoen, Annabelle McIver, Larissa Meinicke, and Carroll C. Morgan. Linear-invariant generation for probabilistic programs. In *Static Analysis Symposium (SAS)*, volume 6337 of *LNCS*, pages 390–406. Springer, 2010.

[63] Joost-Pieter Katoen, Lei Song, and Lijun Zhang. Probably safe or live. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 55:1–55:10, 2014.

[64] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[65] Dexter Kozen. Results on the propositional μ-calculus. In *Automata, Languages and Programming, 9th Colloquium, Aarhus, Denmark, July 12-16, 1982, Proceedings*, pages 348–359, 1982.

[66] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333 – 354, 1983. Special Issue Ninth International Colloquium on Automata, Languages and Programming (ICALP) Aarhus, Summer 1982.

[67] Dexter C. Kozen. *Theory of Computation*. Springer, 2006.

[68] Jan Kretínský and Alexej Rotar. The satisfiability problem for unbounded fragments of probabilistic CTL. *CoRR*, abs/1806.11418, 2018.

[69] Saul Kripke. A completeness theorem in modal logic. *J. Symb. Log.*, 24(1):1–14, 1959.

[70] Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480, 1977.

[71] Kim G. Larsen, Radu Mardare, and Bingtian Xue. Probabilistic Mu-Calculus: Decidability and Complete Axiomatization. In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2016)*, volume 65 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:18, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[72] Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1 – 28, 1991.

[73] Daniel Lehmann and Saharon Shelah. Reasoning with time and chance. *Information and Control*, 53(3):165 – 198, 1982.

[74] Wanwei Liu, Lei Song, Ji Wang, and Lijun Zhang. A simple probabilistic extension of modal mu-calculus. In *IJCAI*, pages 882–888. AAAI Press, 2015.

[75] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems - specification*. Springer, 1992.

[76] Panagiotis Manolios and Richard J. Trefler. Safety and liveness in branching time. In *16th Annual IEEE Symposium on Logic in Computer Science, Boston, Massachusetts, USA, June 16-19, 2001, Proceedings*, pages 366–374, 2001.

[77] Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102(2):pp. 363–371, 1975.

[78] Annabelle McIver and Carroll Morgan. Games, probability and the quantitative μ-calculus qmμ. In *Logic for Programming, Artificial Intelligence, and Reasoning, 9th International Conference, LPAR 2002, Tbilisi, Georgia, October 14-18, 2002, Proceedings*, pages 292–310, 2002.

[79] Annabelle McIver and Carroll Morgan. Results on the quantitative mu-calculus qmu. *CoRR*, cs.LO/0309024, 2003.

[80] Matteo Mio. Probabilistic modal μ-calculus with independent product. In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures*, volume 6604 of *Lecture Notes in Computer Science*, pages 290–304. Springer Berlin Heidelberg, 2011.

[81] Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3), 2007.

[82] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 46–57, 1977.

[83] Sally Popkorn. *First Steps in Modal Logic*. Cambridge University Press, 1994.

[84] Emil L. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52(4):264–268, 04 1946.

[85] A. N. Prior. *Time and Modality*. Oxford, 1957.

[86] Alberto Puggelli, Wenchao Li, Alberto Sangiovanni-Vincentelli, and Sanjit Seshia. Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In *CAV*, volume 8044 of *LNCS*, pages 527–542. Springer, 2013.

[87] James 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. *Journal of Symbolic Computation*, 13(3):255 – 299, 1992.

[88] Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic J. of Computing*, 2(2):250–273, June 1995.

[89] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Model-checking Markov chains in the presence of uncertainties. In Holger Hermanns and Jens Palsberg, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 3920 of *Lecture Notes in Computer Science*, pages 394–410. Springer Berlin Heidelberg, 2006.

[90] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985.

[91] Robert S. Streett and E.Allen Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81(3):249 – 264, 1989.

[92] Wolfgang Thomas. Handbook of formal languages, vol. 3. chapter Languages, Automata, and Logic, pages 389–455. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[93] Michael Ummels and Christel Baier. Computing quantiles in Markov reward models. In *FoSSaCS*, volume 7794 of *LNCS*, pages 353–368. Springer, 2013.

[94] Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, volume 1043 of *LNCS*, pages 238–266. Springer, 1996.

[95] Moshe Y. Vardi. Why is modal logic so robustly decidable? In *Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop, January 14-17, 1996, Princeton University*, pages 149–184, 1996.

[96] Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1–37, 1994.

[97] Peter Walley. Measure of uncertainty in expert systems. In *Artificial Intelligence, 83.1*, pages 1–58, 1996.

[98] Thomas Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bull. Belg. Math. Soc. Simon Stevin*, 8(2):359–391, 2001.

[99] Martin Zimmermann. Optimal bounds in parametric LTL games. *Theor. Comput. Sci.*, 493:30–45, 2013.

[100] Damjan Škulj. Discrete time Markov chains with interval probabilities. *International Journal of Approximate Reasoning*, 50(8):1314 – 1329, 2009. Special Section on Interval/Probabilistic Uncertainty.

Prior Publications

- **On the Satisfiability of Some Simple Probabilistic Logic**. Souymodip Chakraborty and Joost-Pieter Katoen. In Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016, pages 56–65, 2016.

- **P-automata for markov decision processes**. Souymodip Chakraborty and Joost-Pieter Katoen. In 3rd International Workshop on Strategic Reasoning, St Catherine's College, University of Oxford, Sep 17, 2015.

- **Model Checking of Open Interval Markov Chains**. Souymodip Chakraborty and Joost-Pieter Katoen. In ASMTA, volume 9081 of LNCS, pages 30–42. Springer, 2015.

- **Parametric LTL on Markov Chains**. Souymodip Chakraborty and Joost-Pieter Katoen. n Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings, pages 207–221, 2014.

- **Modeling and statistical model checking of a micro-grid**. Souymodip Chakraborty, Joost-Pieter Katoen, Falak Sher, and Mar- tin Strelec. STTT, 17(4):537–554, 2015.