Verification of Multi-Objective Markov Models

Tim Quatmann

Verification of Multi-Objective Markov Models

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

Tim Quatmann, M. Sc.
aus Lohne

Berichter: Prof. Dr. Ir. Dr. h.c. Joost-Pieter Katoen
Prof. Dr. Mickael Randour

Tag der mündlichen Prüfung: 5. September 2023

Abstract

Probabilistic systems evolve based on environmental events that occur with a certain probability. For such systems to perform well, we are often interested in multiple objectives, i.e., quantitative performance measures like the probability of a failure or the expected time until task completion. Sometimes, these objectives conflict with each other: minimizing the failure probability possibly means completing the task takes longer. Compromises need to be found.

We consider Markov models—particularly Markov decision processes (MDPs) and Markov Automata (MAs). These state-based modeling formalisms describe a system in its random environment. Starting from an initial state, the transitioning behavior in MDPs is determined by probabilistic and nondeterministic choices. MAs further extend MDPs by exponentially distributed continuous time delays. Rewards can be attached to states or transitions to model system quantities such as energy consumption, productivity, or monetary costs. Objectives are formally specified by a mapping from (infinite) system executions to the value of interest, e.g., the total accumulated costs or the average energy consumption. The expected value of an objective is defined once the nondeterminism is resolved using a strategy—intuitively reflecting the choices of a system controller. Different strategies induce different expected objective values. Multi-objective verification of MDPs and MAs analyzes the interplay between the considered objectives and identifies which trade-offs between expected objective values are possible, i.e., achievable by some strategy.

We study practically efficient methods to compute the set of achievable solutions. For this, we establish a general framework and its instantiation for (undiscounted) total reachability reward objectives, long-run average reward objectives, and reward-bounded objectives. We propagate the errors made by approximative methods, yielding sound under- and over-approximations. We further consider multi-dimensional quantiles that ask under which reward constraints a given objective value is achievable. Finally, we investigate a setting in which the strategies must be simple, i.e., non-randomized and with limited memory access. All presented approaches are integrated into the state-of-the-art probabilistic model checker Storm. An extensive evaluation of this implementation on a broad set of multi-objective benchmarks shows that our approaches scale to large models with millions of states.

Zusammenfassung

Probabilistische Systeme werden von Umgebungsereignissen beeinflusst, welche mit einer gewissen Wahrscheinlichkeit auftreten. Damit solche Systeme einwandfrei funktionieren, sind häufig mehrere Ziele, also quantitative Leistungsmaße wie die Ausfallwahrscheinlichkeit oder die erwartete Zeit bis zur Vervollständigung einer Aufgabe, relevant. Manchmal stehen diese Ziele im Konflikt: Das Minimieren der Ausfallwahrscheinlichkeit kann möglicherweise bedeuten, dass sich die Zeit zur Aufgabenfertigstellung erhöht. Kompromisse müssen gefunden werden.

Wir betrachten Markow-Modelle, insbesondere Markow Entscheidungsprozesse (MDPs) und Markow Automaten (MAs). Diese zustandsbasierten Modellformalismen beschreiben ein System in dessen zufälliger Umgebung. Von einem Initialzustand ausgehend ist das Transitionsverhalten in MDPs durch probabilistische und nichtdeterministische Wahlen gegeben. MAs erweitern MDPs zusätzlich durch exponentiell verteilte, kontinuierliche Verzögerungen. Zuständen und Transitionen können Nutzenwerte zugewiesen werden, sodass verschiedene Systemgrößen wie Energieverbrauch, Produktivität oder monetäre Kosten modelliert werden können. Die Systemziele werden formal spezifiziert, indem (unendliche) Systemläufe auf einen Wert abgebildet werden. Beispielsweise kann dies die Summe der Kosten oder der durchschnittliche Energieverbrauch auf dem jeweiligen Systemlauf sein. Damit der Erwartungswert eines so spezifizierten Zieles wohldefiniert ist, muss der Nichtdeterminismus mithilfe einer Strategie aufgelöst werden, welche die Entscheidungen eines Systemcontrollers widerspiegelt. Verschiedene Strategien führen zu verschiedenen erwarteten Zielwerten. Die Verifikation von MDPs und MAs mit mehreren Zielen analysiert das Zusammenspiel zwischen den betrachteten Zielen und identifiziert, welche Kompromisse zwischen erwarteten Zielwerten möglich, d. h. von einer Strategie realisierbar sind.

Wir untersuchen praktisch effiziente Methoden, um die Menge der realisierbaren Lösungen zu berechnen. Dazu führen wir ein algorithmisches Grundgerüst ein und instanziieren dieses bezüglich verschiedener Ziele, wie die (nicht rabattierte) Gesamtsumme der Nutzenwerte bis zum Erreichen eines Zielzustandes, der langfristige, durchschnittliche Nutzenwert sowie durch Nutzenbedingungen eingeschränkte Ziele. Wir schätzen die Fehler von approximativen Methoden ab, sodass korrekte Unter- und Überapproximationen entstehen. Außerdem betrachten wir multidimensionale Quantile, bei denen die Frage ist, unter welchen Nutzenbedingungen ein gegebener Zielwert ermöglicht werden kann. Schließlich untersuchen wir die Einschränkung auf simple Strategien, d. h. Strategien, die nicht randomisieren und nur eine eingeschränkte Merkfähigkeit haben. Alle präsentierten Ansätze sind im hochmodernen probabilistischen Model Checker Storm integriert. Eine ausführliche Evaluation dieser Implementierung auf einer breiten Menge von Benchmarks mit mehreren Zielen zeigt, dass unsere Ansätze auf große Modelle mit Millionen von Zuständen skalieren.

Acknowledgements

I am extremely grateful for having Joost-Pieter Katoen as my PhD supervisor. Thank you, Joost-Pieter, for all the invaluable advice and feedback I received from you over the years. With the MOVES group, you provide a healthy and fruitful research environment that enabled me to find my own path. I want to thank Mickael Randour for acting as the external examiner. Mickael, thank you for showing so much interest in this thesis and for providing very useful feedback. Special thanks go to Wil van der Aalst and Bastian Leibe for being part of the examination committee.

My PhD journey started with a software project course offered by the MOVES group. During enthusiastic discussions on strategies for a Reversi AI, I was lured into writing my Bachelor thesis with Chris. One semester later, Reversi was replaced by Markov models and Chris had handed me over to Nils—but the enthusiasm remained.

Nils, you saw potential in me that I myself was not aware of. Thank you for sparking my interest in science and for showing me the beauty of probabilistic model checking research. You played an influential role in forming the foundation of my academic career.

Sebastian, you introduced me to multi-objective model checking when you supervised my master thesis. I have learned a lot from you (and still do!), be it writing papers (sometimes at 3 AM), dealing with `C++` and `cmake`, or resolving all kinds of weird situations one can come across as a junior researcher. Thank you for all that!

Chris, you always had good advice for me and your dedication for Storm has had a big impact on me. Maybe this is why I somehow ended up inheriting your legacy of being “Mr. Storm”. Thanks a lot!

Matthias, you are always extremely helpful when discussing and solving all kinds of issues. Also, thanks for giving me a daily chuckle when you had to squeeze between the plant and my open office door.

Alex, thank you for continuing the tradition of plant-door chuckles after Matthias has left. Your passion and excitement for POMDPs is contagious which is one of the many reasons I very much enjoy doing research with you. I am also truly impressed by your ability to soak up random knowledge.

Jip, thank you for your countless attempts to teach me Dutch words and for letting my daughter play with your ridiculously large Lego collection.

Christoph, you have been the best (and only) office mate I had so far. Thanks for all the chats, laughs, rants, and office clean-up sessions we had together.

Kevin, Lutz, and Tobias, I still remember your very first day at RWTH (`Make`!). I am so happy that you somehow also ended up in the MOVES group. Thank you for all the technical and not-so-technical conversations we had over the years.

Arnd, I had the pleasure of collaborating with you on various occasions. Thank you for all the fruitful and efficient discussions and all the lessons I have learned from you.

Thank you to all current and former members of the MOVES group, to all my co-authors, and to everyone who contributed to Storm. You all shaped my academic life and this thesis is what came out of that.

Der größte Dank gilt meiner Familie, welche mich auf diesem Weg begleitet hat und regelmäßig dafür sorgt, dass ich nicht vollständig wahnsinnig werde.

Vielen Dank an meine Eltern, Ulla und Josef, dafür, dass ihr meine Neugier gefördert habt und mir immer mit Rat und Tat zur Seite steht.

Hanna und Mia, ihr schafft es jeden Tag mich zum Lachen zu bringen. Euch langsam aber sicher groß werden zu sehen erfüllt mich mit sehr viel Stolz.

Sara, du gibst mir den nötigen Rückhalt und schaffst es immer, die richtigen Worte zu finden—auch wenn ich sie vielleicht nicht immer hören möchte. Seitdem wir uns kennen, hast du maßgeblich dazu beigetragen, dass ich zu dem Menschen werden konnte, der ich heute bin. Ich danke dir sehr für all deine Unterstützung, Fürsorge und Liebe.

Contents

—Chapter 1—

Introduction

1.1 Motivation

Formal Verification | *Digital systems* are a fundamental aspect of our modern society. They appear everywhere in our daily lives—from driver-assistance systems over home entertainment up to medical devices. These systems are often found in critical environments, where a malfunction causes serious material or personal damage. Prominent examples are the crash of the Ariane 5 missile [Lio96] causing damage worth of more than 370 million US dollars, and the Therac-25 radiation machine that treated at least 6 patients with a serious overdoses [Lev17]. Testing is common practice in software development to find bad system behavior. However, in most applications it is infeasible to test for *every* possible input of a system. Consequently, testing is inadequate for showing the absence of bad behavior [Dij72]. *Formal verification* mathematically proves system correctness to methodically rule out system failure.

Model Checking | *Model checking* [CE81; QS82; BK08; CHVB18] is a popular formal verification technique that is applicable throughout a system's life cycle. Hardware developers such as IBM and Intel commonly use model checking to verify their products [BEGW03; Fix08]. The model checking tool SLAM—developed at Microsoft—verifies device drivers [BLR11]. Model checking has successfully been applied to verify security protocols [BCM18]. [BL17] empirically shows that software model checking of `C` code finds more bugs in less time compared to testing.

The technique works by providing a *model*—a "blueprint" of the system specifying its behavior—and a set of *properties* that are derived from the system's requirements. A *model checker*, i.e., a tool implementing model checking algorithms, then checks if the given model satisfies the given properties and returns that all properties are satisfied or which properties are violated. In the latter case, useful debug information is provided to system designers, helping them understanding and fixing faulty system

behavior. A major reason for the success of model checking is that this process is fully automatic: once model and properties are given, no further user input is needed.

Markov Models | In classical model checking, the model is given as a transition system (also known as Kripke structure). In this formalism, system states are connected via transitions specifying how the system evolves from one state to another. A state can have multiple outgoing transitions which reflect nondeterministic choices to model, e.g., uncertain behavior or concurrency. However, richer formalisms are needed when nondeterministic choices alone do not adequately represent the system.

Markov model
MDP, Markov decision process
MA, Markov automaton

This work considers model checking of *Markov models*, in particular *Markov decision processes (MDPs)* [Put94] and *Markov automata (MAs)* [EHZ10]. These models are similar to transition systems, but besides nondeterminism their transitioning behavior is also influenced by the outcome of random variables. MAs additionally consider random delays, thus combining nondeterministic and probabilistic choices with continuous time aspects. This is realized by distinguishing two types of states: *Markovian states* and *probabilistic states*. In both cases, the successor state is determined by a probability distribution such as the outcome of a coin flip. In a Markovian state, this only happens after an exponentially distributed time delay. In a probabilistic state, an *action* is chosen nondeterministically from a menu of enabled actions and the successor state distribution depends on the selected action. No time passes in probabilistic states. Distinguishing between Markovian and probabilistic states enables a well-defined notion of compositionality for MAs, where components synchronize on common actions at probabilistic states [Tim13]. States and transitions can be equipped with rewards to model various quantities of the system such as energy consumption or productivity. MAs and MDPs—the latter can be seen as MAs without Markovian states—adhere to the Markov property: once nondeterminism is resolved, the system behavior only depends on the current state. We showcase the modeling capabilities of MAs in terms of a motivating example. A more formal treatment is postponed to Section 2.2.

Markovian state
probabilistic state
action

Example 1.1 Figure 1.1 depicts an MA that models a machine with three different configurations. For a given configuration $i \in \{1, 2, 3\}$, the machine is either in a *working* state w_i, in a *failing* state f_i, or in a *repairing* state r_i. Whenever the machine is working, a failure eventually occurs due to wear of its components. However, the time until a failure occurs is not known in advance. Instead, we assume that the delay can adequately be modeled by an exponential distribution with a certain rate. For example, the machine switches from working state w_1 to failing state f_1 with rate $1/8$, i.e., the average time to fail when in configuration $i = 1$ is $1/(1/8) = 8$ time units.
If the machine is in a failing state f_i, a controller of the machine can decide to either repair (rep) the machine in its current configuration i leading to the repair

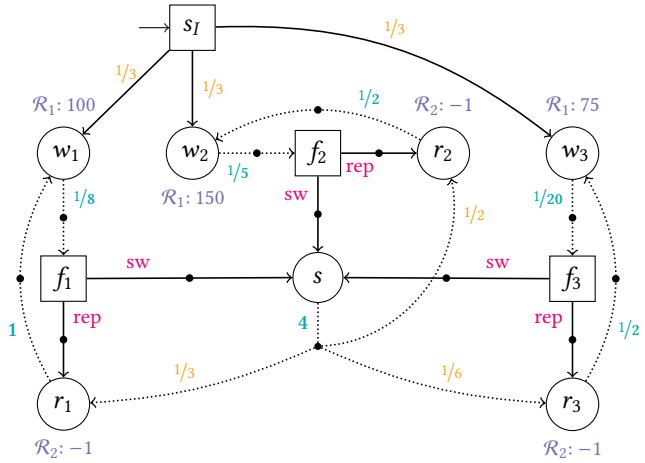


Figure 1.1: Example Markov Automaton (cf. Examples 1.1 and 1.2)

state r_i or to transition (sw) to a dedicated switching state s in which the current configuration i is changed to a potentially different configuration $j \in \{1, 2, 3\}$. The decision of the controller and the transition to either r_i or s are both assumed to be instantaneous, i.e., no time passes. In a repair state r_i, the machine is repaired and ready to work again (w_i) after a certain amount of time—which we also assume to be exponentially distributed. In the switching state s, the next configuration is chosen according to a discrete probability distribution and after an exponentially distributed delay. The initial configuration is chosen uniformly at random without any delay.

Figure 1.1 depicts the MA using circles and dashed lines to indicate Markovian states and their outgoing transitions whereas for probabilistic states we use rectangles and straight lines. We sometimes use an incoming arrow to mark a dedicated initial state—such as the state s_I in Figure 1.1 from which the initial configuration is chosen uniformly at random. Rates of exponential distributions are given in teal, successor probabilities are given in orange, and controller actions are given in pink. We also attach two kinds of rewards to the states of the MA that model the productivity of the machine ($\mathcal{R}_1$) and the repair costs ($\mathcal{R}_2$). For example, in configuration $i = 1$, the machine produces $\mathcal{R}_1[w_1] = 100$ units per time as long as it is in the working state w_1. Repair costs are modeled using negative rewards. We omit probability values when they are 1, reward values when they are 0, and action labels at states with just a single enabled action.

Quantitative Objectives | Model checking verifies the complete absence of bad system behavior. However, such a verification objective might be too ambitious as bad behavior can not always be fully excluded. In those cases, it might be acceptable that, e.g., a system failure is possible, but *unlikely*. Verification of Markov models argues over quantities of the system such as probabilities and rewards to answer quantitative verification objectives such as:

- Is the probability of a crash below 0.001%?
- Is the expected time until task completion at most 8 hours?,
- Is the probability to reach a charging station before running out of battery at least 99%?
- Is the expected average number of produced units per hour at least 15?

strategy

To verify the above objectives, the nondeterminism potentially present in a system needs to be resolved. A *strategy* (also commonly referred to as policy, scheduler, or adversary) of an MA resolves the nondeterminism by specifying which action to choose whenever the system is in a probabilistic state. Generally, the strategy choices may depend on the system execution so far and may also be randomized. System measures are only defined under a given strategy and verification queries typically ask whether an objective holds under *some* or—dually—under *all* strategies. Alternatively, objectives may ask for the actual quantity, e.g., the maximal (w.r.t. all available strategies) probability of a crash.

probabilistic model checking

Probabilistic Model Checking | *Probabilistic model checking* [CY88; FKNP11; Kat16; BAFK18; BHK19] studies automated procedures to verify quantitative objectives for Markov models. Successful application areas include network and security protocols [LMT04; NS06; KNP12a], systems biology [KNP08], planning problems [YLWA05; FWHT15], performance models for microservices [CJR22], and self-adaptive software systems [CGKM12; CLLV+17].

EPMC [FHLS+22], MCSTA [HH14], PRISM [KNP11], and STORM [12] are popular probabilistic model checkers that verify large-scale Markov models with millions, or even billions of states. The models are specified using a high-level description in a domain-specific modeling formalism such as the `Modest` language [HHHK13], the `PRISM` language [KNP11], the `JANI` model exchange format [BDHH+17], `PGCL` specifications for probabilistic programs [GHNR14], dynamic fault trees [SDC99], or generalized stochastic Petri nets [EHKZ13]. Quantitative objectives are formally given based on temporal logics such as LTL [Pnu77], PCTL [HJ94], and CSL [ASSB00; BHHK03] equipped with operators that address probabilities, rewards, and time.

Multi-objective Model Checking | Probabilistic model checking commonly considers each verification objective for a given Markov model independently. This is non-problematic when the objectives are composable. For example, when we want to show that several different objectives hold for *all* strategies, it suffices to verify the objectives one after the other. The interplay between the objectives is not important in such a setting. On the other hand, assume we have shown for some system that, e.g.,

- there is a strategy for which the probability of a crash is below 0.001% and
- there is a strategy for which the expected task completion time is at most 8 hours.

From this, we can *not* infer the existence of a strategy that satisfies *both* objectives. In fact, any strategy with a sufficiently low crashing probability might need a long time to complete the task and—dually—a strategy that induces a low expected time for task completion might take high-risk actions that lead to a higher crashing probability. *Multi-objective model checking* [CMH06; EKVY08] aims to analyze trade-offs between potentially conflicting objectives, allowing to verify queries that address combinations of objectives such as

multi-objective model checking

- Is there a *single* strategy for which both the probability of a crash is below 0.001% *and* the expected task completion time is at most 8 hours?

Pareto Optimality | The above query considers *two* objectives that each specify a *threshold* on the value that is to be achieved. In general, arbitrarily many objectives can be considered and the thresholds can be dropped in which case we want to find strategies that either *maximize* or *minimize* (w.r.t. all available strategies) the value of the individual objectives. Due to potentially conflicting objectives, there usually is not a single strategy that yields optimal values for *all* objectives. We therefore aim for *Pareto optimality* [Mat91]. Intuitively, a strategy is Pareto optimal for a given set of objectives if every other strategy either yields the exact same objective values or a worse objective value for one or more objectives. In that sense, Pareto optimal strategies are *nondominated*: there is no strategy that performs better for one objective without diminishing the value of at least one other objective. The *Pareto front* is the set of objective values that are induced by a Pareto optimal strategy.

Pareto optimal

Example 1.2 Consider the MA from Figure 1.1 and recall from Example 1.1 that the depicted reward assignments model productivity of the machine ($\mathcal{R}_1$) and repair costs ($\mathcal{R}_2$), the latter modeled using negative rewards. We are interested in the trade-offs between average productivity and average costs that can be expected in the long run. Formally, these objectives can be stated as so-called long-run average (LRA) reward objectives for reward assignments $\mathcal{R}_1$ and $\mathcal{R}_2$, denoted by $lra(\mathcal{R}_1)$ and $lra(\mathcal{R}_2)$, respectively.

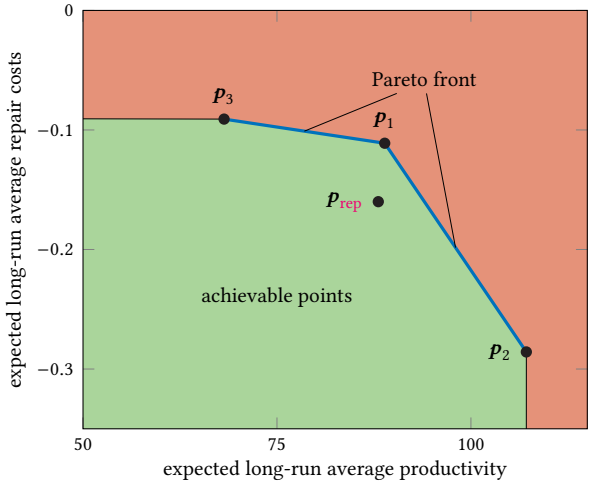


Figure 1.2: Achievable points for two LRA objectives (cf. Example 1.2)

We consider the three strategies σ_1, σ_2, and σ_3 for the machine controller given as follows. For $i \in \{1, 2, 3\}$, the strategy σ_i decides to repair the machine—i.e., select action rep—whenever the system is in the failing state f_i for configuration i. In all other failing states f_j for $j \in \{1, 2, 3\} \setminus \{i\}$, the strategy σ_i decides to switch, i.e., selects action sw. Consequently, when following strategy σ_i, the machine eventually switches to and stays in configuration i almost surely, i.e., with probability 1. In the case of σ_1, this yields an expected LRA productivity of $100 \cdot \frac{8}{8+1} \approx 88.9$ per time unit and an expected LRA repair cost of $-1 \cdot \frac{1}{8+1} \approx -0.111$ per time unit. Figure 1.2 plots these values as point $\boldsymbol{p}_1 \approx \langle 88.9, -0.111 \rangle$ in a 2-dimensional coordinate system. Similarly, we obtain the points $\boldsymbol{p}_2 \approx \langle 107, -0.286 \rangle$ and $\boldsymbol{p}_3 \approx \langle 68.2, -0.091 \rangle$, reflecting the expected productivity and costs when adhering to σ_2 and σ_3, respectively. For $i \in \{1, 2, 3\}$, we say that σ_i *achieves* $\boldsymbol{p}_i$.
Among the three considered strategies, σ_2 yields the highest average productivity but does not perform well in terms of average costs. Vice versa, σ_3 induces a low cost but is also bad in terms of productivity. Strategy σ_1 achieves a compromise between the two objectives. However, without further information, none of the three strategies can be considered "better" than the other.
Other controller strategies are possible. The strategy σ_{rep} which always selects action rep in each failing state f_i, $i \in \{1, 2, 3\}$ yields expected values $\boldsymbol{p}_{\text{rep}} \approx \langle 88.1, -0.16 \rangle$ also indicated in Figure 1.2. However, σ_1 *dominates* σ_{rep} since the

former achieves better values for both productivity and cost. Strategy σ_1 should be preferred over σ_{rep}. On the other hand, there is no strategy that *dominates* σ_1—it is Pareto optimal as all other strategies either yield the same values, less productivity, or more cost. Similarly, σ_2 and σ_3 are Pareto optimal, as well.
The points on the two blue line segments in Figure 1.2 connecting $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ as well as $\boldsymbol{p}_1$ and $\boldsymbol{p}_3$ are achieved by randomized strategies that intuitively flip a (biased) coin and based on its outcome mimic either σ_1 or σ_2 and either σ_1 or σ_3, respectively. The *Pareto front* for the considered MA and the two objectives consists of the two blue line segments including their end points $\boldsymbol{p}_1$, $\boldsymbol{p}_2$, and $\boldsymbol{p}_3$. The green area "below" the Pareto front depicts the set of *achievable points*—points for which a strategy exists that achieves at least the given values (or better values).

1.2 Topics of this Thesis

Approximating Pareto Fronts | Pareto fronts depict in a concise and human-readable way which objective trade-offs the system can possibly achieve. This provides valuable insights, helping decision makers to design system controllers and potentially revealing flaws in early development stages by showing that desired objective values can not simultaneously be achieved with the current design. Our primal goal for this thesis is to establish efficient, fully automized methods for the following task:

> Given a Markov model and a set of objectives, compute the Pareto front.

We tackle this task in a general setting, where an objective is given by a mapping from system executions to real (positive or negative) values. For example, this value can be the passed time until completing a task or the average costs per time unit. We then optimize the *expected values* of multiple such objectives. This framework subsumes several settings considered in related works, e.g., [EKVY08; FKNP+11; FKP12; BBCF+14; RRS17]. In particular, all kinds of probability objectives such as reachability probabilities and LTL specifications are supported by mapping system executions to either 1 if the considered event occurred or 0 otherwise. We lift the approach by [FKP12]—which itself borrows ideas from convex multi-objective optimization [SAC93; RDH11]—towards our general setting. The rough idea is to iteratively refine an approximation of the Pareto front by optimizing weighted sums of the individual objective values for various different combinations of weights. While this framework is independent of the specific kinds of objectives, the subtask of optimizing weighted objective sums requires dedicated methods. We take a closer look at (mixtures of)

- *total reachability reward objectives* such as the probability to reach a bad state, or

the time until task completion,

- *long-run average (LRA) reward objectives* such as the average energy consumption per time unit, and
- *multi-reward bounded objectives* such as the probability to complete a task within a given time and energy budget.

multi-dimensional quantile

Multi-dimensional Quantiles | For the above-mentioned multi-reward bounded objectives, we optimize the objective value for given constraints on the accumulated reward. *Multi-dimensional quantile queries* consider the reverse question: what are valid constraints on the accumulated rewards under which a given expected objective value is achievable? More specifically, this framework deals with questions such as

- How much time and energy is needed to complete a task with probability 0.8?
- How many products can be manufactured with how many workers within an expected time of 8 hours?
- How much energy is needed to complete how many tasks with probability 0.9?

These queries consider trade-offs between the considered resource constraints, e.g., a high energy budget might mean that the task can be completed in less time. We provide automized techniques to compute and visualize these trade-offs.

simple strategy

Restriction to Simple Strategies | Pareto optimal strategies need to make randomized choices to achieve certain trade-offs as we have seen in Example 1.2. Moreover, strategies might need to memorize parts of the execution history, e.g., which tasks have already been completed. Such sophisticated strategies are inadequate in certain scenarios. For example, in a product design application, the final product should not be subject to a random choice; all customers should receive the same product. For ethical reasons, a medical device should not treat patients based on the outcome of a coin flip. Small scale embedded systems might not have the capacity to implement complicated strategies. Finally, a simple strategy is easier to understand, implement, and debug. We also study a variant of the multi-objective verification problem that approximates Pareto fronts when restricted to *simple*, i.e., pure (non-randomized) strategies with limited access to memory.

Soundness | Verification results should be trusted. Computing quantitative values *exactly* (i.e., without any kind of error) however is very challenging for large-scale systems. Probabilistic model checking therefore commonly relies on giving *approximations* of the computed quantities. We provide *sound* approximations by considering error bounds and their propagation throughout the verification procedure.

Implementability | Practical applicability is a major asset of model checking. The presented algorithms are meant to be implemented and executed on real-world applications. This is demonstrated using an implementation in the model checker Storm.

1.3 Contributions

We summarize the core contributions of this work. These results are based on earlier publications of the author (cf. Section 1.6). Contributions made beyond existing works are outlined at the beginning of the corresponding chapters.

A Framework for Every Objective (Chapter 3) | We provide theoretical and practical results for multi-objective Markov models that *apply to all kinds of expected value objectives*. These results include the convexity of the set of achievable points (Theorem 3.5 on page 70) and a lifting of the Pareto front approximation algorithm of [FKP12] towards our general setting (Section 3.4). The latter is achieved by passing on objective-specific computations to optimization queries for weighted objective sums. Depending on the objective types, these *weighted sum optimization* queries can often be solved using single-objective methods for Markov models. Our approaches take error propagations of approximative solutions into account, enabling *correct* approximations for large-scale systems.

Sound Expected Total Rewards (Chapter 4) | We present *sound value iteration*—an extension of the classical value iteration approach for Markov decision processes [Put94]—which provides correct lower- and upper bounds for (single-objective) expected total reachability rewards.

Mixing Total and Long-run Average Rewards (Chapter 5) | We instantiate our general Pareto front approximation algorithm towards mixtures of total- and LRA reward objectives. The crux is a treatment of the so-called end components (Section 5.3).

Multi-Dimensional Reward-Bounds and Quantiles (Chapters 6 and 7) | We show how reward-bounded objectives can efficiently be evaluated by considering the relevant reward epochs *sequentially*—lifting ideas of [BDDK+14; HH16] towards more expressive objectives with multiple reward-bounds (Section 6.2). We further use this sequential approach to lift *quantile queries* as considered in [BDDK+14] to multiple reward dimensions (Section 7.2). We show that such a multi-dimensional setting requires specific care to ensure termination of our procedure.

Restriction to Simple Strategies (Chapter 8) | We argue that deciding achievability under simple (in particular non-randomized) strategies is NP-complete and thus theoretically more complex compared to verification under general strategies when a fixed number of objectives is assumed. We provide *two* mixed integer linear program encodings to find simple Pareto optimal strategies (Section 8.3). The first encoding applies to general total reachability reward objectives. The second encoding is more compact but only applicable to total reward objectives that do not consider goal states. Using these encodings, we present an algorithm for approximating the Pareto front under simple strategies (Section 8.5).

Implementation (Chapter 9) | We evaluate the usability of our approaches in practice. All algorithms given in this thesis have been integrated in the model checker Storm. We present an extensive evaluation on a newly composed benchmark set consisting of 172 multi-objective, 27 (single-objective) multi-reward bounded queries, and 18 quantile queries. Our results show that Storm outperforms competing tools and scales to large multi-objective systems with millions of states.

1.4 Overview of Contents

Figure 1.3 depicts the main chapters of this thesis and how they built on each other.

In Chapter 2 we establish the theoretical background for the remainder of the thesis. This includes an introduction to the necessary concepts from mathematical *geometry* as well as the used notions and formalisms for *Markov models* and their *objectives*.

Chapter 3 introduces our general framework for multi-objective verification. We formally define the *Pareto front* and show various interesting properties. We then provide formal *problem statements* and present *algorithms* to answer those queries. Our algorithms require solving multiple instances of what we call the *weighted sum optimization problem* (WSO). The subsequent Chapters 4 to 6 provide further details on how to answer such WSO queries for various objective types.

Chapter 4 focuses on *total reachability reward objectives*. We reduce WSO queries to classical (single-objective) computations of expected total reachability rewards. Then, we present *sound value iteration*, a practically efficient algorithm that computes sound lower- and upper bounds of the desired value with arbitrary accuracy.

Chapter 5 provides details on how WSO instances over mixtures of total reward objectives and *long-run average reward objectives* can be handled using a synergy between existing (single-objective) total reward and long-run average reward methods.

Chapter 6 considers objectives with additional constraints given by one or more *reward bounds*, i.e., thresholds on the accumulated reward with respect to multiple different reward structures. We first show that such objectives can be tackled by encoding the rewards accumulated so far into the state-space, thus explicitly *unfolding*

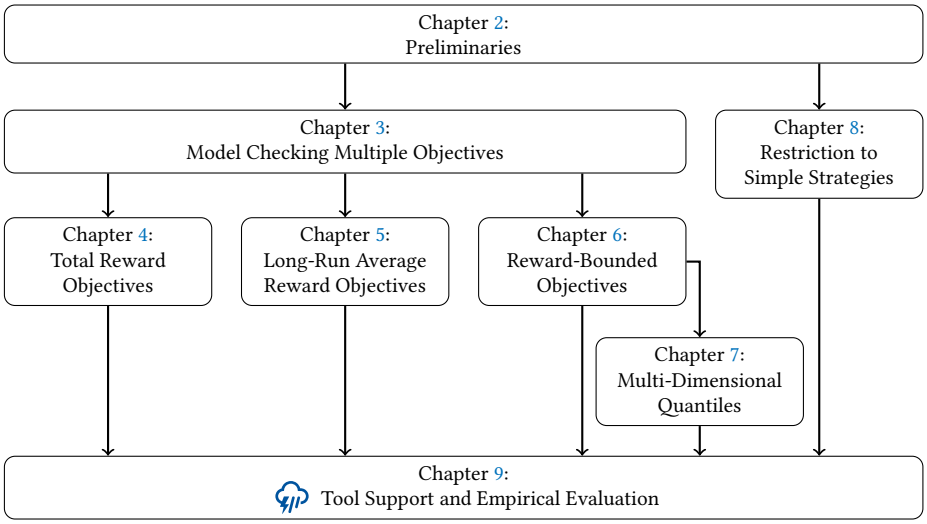


Figure 1.3: Overview of main chapters

the state space of the model. As such a model unfolding can become very large, we also present a *sequential* approach that intuitively exploits regularities of the unfolding to keep the memory footprint small.

Chapter 7 builds upon the sequential approach by providing an algorithm to compute *multi-dimensional quantiles*, where the threshold for the accumulated rewards are not known in advance.

Chapter 8 considers multi-objective verification under *simple strategies*, i.e., strategies with limited memory that are not allowed to perform randomized choices. We provide formal problem statements, discuss theoretical complexity, and present algorithms based on *mixed integer linear programming* (MILP) encodings. Some of the encodings rely on results for *expected visiting times* which are also presented in this chapter.

Chapter 9 discusses practical aspects of our results by presenting and evaluating our *implementation in* Storm on an extensive set of *multi-objective verification benchmarks*.

Chapter 10 wraps up this thesis and provides an outlook to future work.

1.5 Related Work

This section provides an overview of related works on multi-objective Markov models. At the end of Chapters 4 to 8, we further discuss works specifically related to the

contents of those chapters in greater detail.

Research on multi-objective MDPs emerged in operations research and decision theory around the early 1980s [Fur80; Whi82; Hen83]. In a formal verification context, multi-objective model checking of MDPs has first been studied in [CMH06] for discounted total reward objectives, in [Cha07] for LRA reward objectives, and in [EKVY08] for linear time (ω-regular or LTL) objectives. These seminal works show that randomizing strategies are generally required in their setting and provide linear programming-based algorithms for the Pareto front approximation that run in time polynomial in the size of the MDP.

[EKVY08] further presents efficient graph-based algorithms for checking qualitative objectives that ask if a non-zero probability can be achieved or if the objective specification holds almost surely, i.e., with probability 1. [FKNP+11] incorporates total (undiscounted) reward objectives into the framework of [EKVY08] and presents the first implementation in a probabilistic model checker—namely in the tool Prism [KNP11]. One motivation given in this line of research is the application of multi-objective model checking in an assume-guarantee framework for compositional verification of large probabilistic systems.

[BBCF+14; CKK17] extend results of [Cha07] on LRA reward objectives and also consider (mixtures with) so-called satisfaction objectives which concern the probability that the LRA reward stays above a given threshold. An implementation is available in the tool MultiGain [BCFK15]. [RRS17] provides a theoretical framework for percentile queries, which—for the LRA reward case—are similar to the satisfaction objectives of [BBCF+14; CKK17], but also considers percentiles of other measures such as total (discounted or undiscounted) rewards.

The works discussed so far predominantly rely on linear programming. Instead, [FKP12] presents a Pareto front approximation algorithm for total reward and linear time objectives based on value iteration [Put94] and algorithms from convex multi-objective optimization [SAC93; RDH11]. Using an implementation in Prism, they empirically show that this new approach significantly outperforms the linear programming-based method of [FKNP+11] in terms of runtime and scalability. The approach has been applied to multi-objective interval MDPs [HHHL+19] and stochastic games [ACKW+20]. In the context of this work, the approach is lifted towards general objectives for MDPs and MAs.

[WT98] finds Pareto optimal strategies for discounted and long-run average reward objectives based on strategy iteration. [BN08] extends value iteration to multiple objectives by using geometric set representations which is further lifted to stochastic games in [BKTW15; BKW18]. [SBHH17] treats multi-objective interval MDPs using a strategy iteration-based approach to compute a set of mutually non-dominated simple strategies that are likely to be Pareto optimal. In [PW10], Tchebycheff-optimal strategies which minimize the distance to a reference point are obtained via a linear programming approach. [BFRR17; BGMR18; BGR20] consider mixtures of probabilistic

and non-probabilistic objectives, where the latter impose hard constraints on *all* system executions, even those that appear with probability 0. [BDK14] presents efficient methods for non-standard multi-objective queries, including (single-dimensional) quantiles, conditional probabilities, and long-run ratio objectives.

[RVWD13] surveys multi-objective MDPs from an AI perspective. [FWHT15; LPH17] apply multi-objective analysis for controller synthesis in probabilistic planning domains. [LTHS+22] reduces preference-based planning problems to multi-objective LTL queries. [GCCA+21] solves multi-objective MDP queries to synthesize strategies for quality-of-service software requirements using a transformation to parametric Markov chains and evolutionary algorithms.

1.6 Origins

Major contributions of this work originate from preceding publications. Their presentation has been thoroughly revised and put into context. Further explanations, examples, proofs, as well as additional theorems and lemmas have been added, yielding a coherent and—hopefully—approachable thesis.

The coauthored publications whose results are covered in this thesis are listed below, together with a clarification concerning author contributions. The latter is a requirement by the current doctoral degree regulations at the RWTH Aachen Faculty of Mathematics, Computer Science and Natural Sciences. The advice, feedback, ideas, questions, and findings of all coauthors have been invaluable. Ultimately, I consider all listed publications as the result of collaborative research.

A full list of peer-reviewed (co-)authored publications is given on page 331.

T. Quatmann and J.-P. Katoen. "*Sound Value Iteration.*" *CAV (1)*. Volume 10981. LNCS. Springer, 2018, pages 643–661. DOI: 10.1007/978-3-319-96145-3_37

- Technical report available at https://arxiv.org/abs/1804.05001
- Represented in Chapter 4

I contributed the initial idea, its implementation and evaluation, as well as the technical parts of the paper while taking the extremely helpful feedback and advice of my coauthor into account.

A. Hartmanns, S. Junges, J.-P. Katoen, and T. Quatmann. "*Multi-cost Bounded Tradeoff Analysis in MDP.*" *J. Autom. Reason.* 64.7 (2020), pages 1483–1522. DOI: 10.1007/s10817-020-09574-9

- Supersedes conference version [3]
- Represented in Chapters 6 and 7

This research was initiated in a joint discussion of the authors during which I had a significant role. I contributed further technical details, in particular for the

sequential approach and for multi-dimensional quantiles, as well as major parts of the implementation and its evaluation. Both, the conference paper [3] and the above-mentioned journal paper have been written collaboratively with similar efforts from all authors.

T. Quatmann and J.-P. Katoen. "*Multi-objective Optimization of Long-run Average and Total Rewards*." *TACAS (1)*. Volume 12651. LNCS. Springer, 2021, pages 230–249. DOI: 10.1007/978-3-030-72016-2_13

- Technical report available at https://arxiv.org/abs/2010.13566
- Represented in Chapters 3 and 5

I developed and implemented the approach, conducted the evaluation, and wrote major parts of the paper. Throughout this process, I got invaluable feedback and advice from my coauthor.

F. Delgrange, J.-P. Katoen, T. Quatmann, and M. Randour. "*Simple Strategies in Multi-Objective MDPs*." *TACAS (1)*. Volume 12078. LNCS. Springer, 2020, pages 346–364. DOI: 10.1007/978-3-030-45190-5_19

- Technical report available at https://arxiv.org/abs/1910.11024
- Represented in Chapter 8

The research question and our solution approach has emerged from joint discussions (initially also including Sebastian Junges). The details of the MILP encodings and their lifting to Pareto front approximations have mostly been developed, implemented, and evaluated by me—with notable involvement of Florent Delgrange with whom I closely worked together. The paper was initially drafted by Florent and me and brought into shape in a joint effort from all authors.

T. Quatmann, S. Junges, and J.-P. Katoen. "*Markov automata with multiple objectives*." *Formal Methods Syst. Des.* 60.1 (2022), pages 33–86. DOI: 10.1007/s10703-021-00364-6

- Supersedes conference version [2]
- Technical report available at https://arxiv.org/abs/1704.06648
- Represented in Sections 3.2 and 4.1

The foundations of this paper have been laid during my master's thesis [1]. Results from the master's thesis are only stated in this work for the sake of a complete presentation and shall not be attributed as contributions of *this* thesis. This mostly affects Section 4.1. I contributed most of the ideas, results and implementations with advice, help, and feedback from my coauthors who also supervised my master's thesis. The paper was written in a joint effort with equal participation from all authors.

—Chapter 2—

Preliminaries

Outlook | We establish the mathematical foundations relevant for this thesis in Section 2.1, covering *geometry*, *linear programming*, and *probability theory*. Afterwards, we discuss syntax and semantics of *Markov models* with reward assignments and continuous timing aspects in Section 2.2. Finally, we introduce various measures—called *objectives*—on such models in Section 2.3.

Origins | Most of the material presented in this chapter originates from various textbooks, in particular [AD99; BK08; Sol15]. The presented notations for Markov automata are close to [11]. Lemmas 2.2 and 2.3 and their proofs are new.

2.1 Mathematical Background

2.1.1 General Notation

The set of *integers* is denoted by $\mathbb{Z}$. The set of *natural numbers* (including 0) is given by $\mathbb{N}$. For $i, j \in \mathbb{Z}$ we let $\{i..j\} \coloneqq \{i, i+1, \ldots, j\} \subseteq \mathbb{Z}$ denote the set of integers ranging from i to j, where $\{i..j\} = \emptyset$ if $i > j$.

The *power set* of a set A is denoted by $2^A \coloneqq \{A' \subseteq A\}$. We sometimes write $A \uplus B$ for the union of two *disjoint* sets A and B with $A \cap B = \emptyset$. Let $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ be a set of sets $A_1, A_2, \ldots A_n$ with $n \in \mathbb{N}$. The union $\bigcup_{i=1}^{n} A_i = \bigcup_{A \in \mathcal{A}} A$ is sometimes abbreviated by $\bigcup_{\mathcal{A}}$. The intersection $\bigcap_{\mathcal{A}}$ is defined similarly. The *Cartesian product* of sets $A_1, A_2, \ldots, A_n$ is given by

$$A_1 \times A_2 \times \cdots \times A_n \;=\; \bigtimes_{i=1}^{n} A_i \;\coloneqq\; \{\langle a_1, a_2, \ldots, a_n\rangle \mid \forall\, i \in \{1..n\} \colon\; a_i \in A_i\}.$$

We abbreviate $\bigtimes_{i=1}^{n} A$ by A^n. An element $\boldsymbol{a} = \langle a_1, a_2, \ldots a_n\rangle$ of a Cartesian product is called an *n-tuple*, or simply *tuple*. For $i \in \{1..n\}$ we write $\boldsymbol{a}(i) = a_i$ to denote the

tuple

i^{th} entry of tuple $\boldsymbol{a}$. We sometimes also use Cartesian products to describe sets of *sequences* $\{a_1 a_2 \dots a_n \in \bigtimes_{i=1}^{n} A_i\}$.

For a function $f\colon A \to B$ and some $A' \subseteq A$ we write $f\big|_{A'}\colon A' \to B$ for the *restriction* of f to domain A', where $\forall\, a \in A'\colon\ f\big|_{A'}(a) \coloneqq f(a)$. We denote the *preimage* of $b \in B$ by $f^{-1}(b) \coloneqq \{a \in A \mid f(a) = b\}$. *Iverson brackets* map a Boolean expression *expr* to either 0 or 1:

Iverson bracket

$$[\mathit{expr}] \coloneqq \begin{cases} 0 & \text{if } \mathit{expr} \text{ is false} \\ 1 & \text{if } \mathit{expr} \text{ is true.} \end{cases}$$

2.1.2 Geometry

We introduce the notations and concepts from geometry—in particular topology—that occur in later chapters. We keep the descriptions at a high level and forward the interested reader to, e.g., [Zie95; Sol15; TD17].

real number
extended real number

Real Numbers | Let $\mathbb{R}$ be the set of *real numbers*, $\mathbb{R}_{\geq 0} \coloneqq \{x \in \mathbb{R} \mid x \geq 0\}$ be the set of *non-negative* real numbers, $\mathbb{R}_{>0} \coloneqq \mathbb{R}_{\geq 0} \setminus \{0\}$ be the set of *positive* real numbers, and $\overline{\mathbb{R}} \coloneqq \mathbb{R} \cup \{-\infty, +\infty\}$ be the set of *extended* real numbers. For the latter, we sometimes write ∞ instead of $+\infty$ and extend the usual order and arithmetic operators for $\mathbb{R}$ by

$$-\infty < a < +\infty, \qquad a \pm \infty = \infty \pm a = \pm\infty, \qquad \pm\infty \pm \infty = \pm\infty, \qquad \text{and}$$

$$b \cdot (\pm\infty) = (\pm\infty) \cdot b = \begin{cases} \pm\infty & \text{if } b > 0 \\ 0 & \text{if } b = 0 \\ \mp\infty & \text{if } b < 0 \end{cases}$$

where $a \in \mathbb{R}$, $b \in \overline{\mathbb{R}}$, $\pm\infty \in \{-\infty, +\infty\}$, and $\mp\infty \in \{-\infty, +\infty\} \setminus \{\pm\infty\}$. As usual in a measure theoretic context we have, e.g., $0 \cdot \infty = 0$. Arithmetic expressions not covered by the above rules—such as "a/∞" or "$\infty - \infty$"—are undefined. For $a, b \in \overline{\mathbb{R}}$ with $a < b$ we define the *intervals*

interval

$$(a, b) \coloneqq \{x \in \overline{\mathbb{R}} \mid a < x < b\}, \qquad [a, b] \coloneqq (a, b) \uplus \{a, b\}, \qquad [a, a] \coloneqq \{a\},$$
$$[a, b) \coloneqq (a, b) \uplus \{a\}, \quad \text{and} \quad (a, b] \coloneqq (a, b) \uplus \{b\}.$$

Points | For the rest of this section we let $\ell \in \mathbb{N} \setminus \{0\}$. We also refer to an ℓ-tuple $\boldsymbol{p} = \langle p_1, \dots, p_\ell \rangle \in \overline{\mathbb{R}}^{\ell}$ as an ℓ-dimensional *point*. If $\boldsymbol{p} \in \mathbb{R}^{\ell}$ we also use the term *vector*. For a set $\mathcal{I} \subseteq \{1..\ell\}$ we define the *indicator point* $\mathbf{1}_{\mathcal{I}} \in \{0, 1\}^{\ell}$ by

point
vector
indicator point

$$\forall\, i \in \{1..\ell\}\colon\ \mathbf{1}_{\mathcal{I}}(i) \coloneqq [i \in \mathcal{I}].$$

Moreover, we write $\mathbf{0} \coloneqq \mathbf{1}_\emptyset = \langle 0, \dots, 0 \rangle$ for the point consisting of only zeroes, $\mathbf{1} \coloneqq \mathbf{1}_{\{1..\ell\}} = \langle 1, \dots, 1 \rangle$ for the point consisting of only ones, and $\mathbf{1}_i \coloneqq \mathbf{1}_{\{i\}}$ for the point where the i^{th} entry ($i \in \{1..\ell\}$) is one and all other entries are zero.

For $\boldsymbol{p}, \boldsymbol{q} \in \overline{\mathbb{R}}^\ell$ we denote the element-wise sum by

$$\boldsymbol{p} + \boldsymbol{q} \coloneqq \langle \boldsymbol{p}(1) + \boldsymbol{q}(1), \dots, \boldsymbol{p}(\ell) + \boldsymbol{q}(\ell) \rangle \in \overline{\mathbb{R}}^\ell$$

and the dot product by

$$\boldsymbol{p} \cdot \boldsymbol{q} \coloneqq \sum_{i=1}^{\ell} \boldsymbol{p}(i) \cdot \boldsymbol{q}(i) \in \overline{\mathbb{R}}.$$

For $\sim \in \{<, \leq, =, \geq, >\}$ we write $\boldsymbol{p} \sim \boldsymbol{q}$ iff the relation holds element-wise, i.e., $\forall\, i \in \{1..\ell\} \colon\ \boldsymbol{p}(i) \sim \boldsymbol{q}(i)$. Furthermore, we write $\boldsymbol{p} \lneq \boldsymbol{q}$ iff $\boldsymbol{p} \leq \boldsymbol{q}$ and $\boldsymbol{p} \neq \boldsymbol{q}$. The notation $\boldsymbol{p} \gneq \boldsymbol{q}$ is similar.

Definition 2.1 (Distance) The *distance* between two points $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^\ell$ is given by point distance

$$dist(\boldsymbol{p}, \boldsymbol{q}) \coloneqq \sqrt{(\boldsymbol{p} - \boldsymbol{q}) \cdot (\boldsymbol{p} - \boldsymbol{q})} = \sqrt{\textstyle\sum_{j=1}^{\ell} (\boldsymbol{p}(j) - \boldsymbol{q}(j))^2}.$$

The (shortest) distance between two non-empty sets $P, Q \subseteq \mathbb{R}^\ell$ is given by

$$dist(P, Q) \coloneqq \inf \left\{ dist(\boldsymbol{p}, \boldsymbol{q}) \;\middle|\; \boldsymbol{p} \in P, \boldsymbol{q} \in Q \right\}.$$

Furthermore, we set $dist(\emptyset, \emptyset) \coloneqq 0$ and $dist(P, \emptyset) = dist(\emptyset, P) \coloneqq \infty$ for $P \neq \emptyset$.

We may omit the set brackets and write, e.g., $dist(\boldsymbol{p}, Q)$ instead of $dist(\{\boldsymbol{p}\}, Q)$ for $\boldsymbol{p} \in \mathbb{R}^\ell$ and $Q \subseteq \mathbb{R}^\ell$. We have $dist(P, Q) = dist(Q, P)$ for all $P, Q \in \mathbb{R}^\ell$.

Topology | For sets of points $P \subseteq \overline{\mathbb{R}}^\ell$ we sometimes need to formally reason about points that lie on the boundary between P and $\overline{\mathbb{R}}^\ell \setminus P$. This requires notions from topology.

Definition 2.2 (Neighborhoods) The set of *neighborhoods* of some (extended) real value $p \in \overline{\mathbb{R}}$ is given by the family of sets neighborhood

$$\mathcal{N}(p) \coloneqq \begin{cases} \left\{ N \subseteq \overline{\mathbb{R}} \;\middle|\; \exists\, \varepsilon \in \mathbb{R}_{>0} \colon\ N \supseteq (p - \varepsilon,\ p + \varepsilon) \right\} & \text{if } p \in \mathbb{R} \\ \left\{ N \subseteq \overline{\mathbb{R}} \;\middle|\; \exists\, a \in \mathbb{R} \colon\ N \supseteq (a, +\infty] \right\} & \text{if } p = +\infty \\ \left\{ N \subseteq \overline{\mathbb{R}} \;\middle|\; \exists\, a \in \mathbb{R} \colon\ N \supseteq [-\infty, a) \right\} & \text{if } p = -\infty \end{cases}$$

The set of neighborhoods of a point $\boldsymbol{p} \in \overline{\mathbb{R}}^{\ell}$ is given by

$$\mathcal{N}(\boldsymbol{p}) \;\coloneqq\; \Big\{ \bigtimes_{i=1}^{\ell} N_i \;\Big|\; \forall\, i \in \{1..\ell\} \colon\; N_i \in \mathcal{N}(\boldsymbol{p}(i)) \Big\}.$$

The set $\overline{\mathbb{R}}^{\ell}$ can be interpreted as a *topological space* [TD17] using the neighborhoods defined above. Intuitively, each neighborhood of a point $\boldsymbol{p} \in \overline{\mathbb{R}}^{\ell}$ describes an area around $\boldsymbol{p}$. We use this notion to specify when $\boldsymbol{p}$ is *close* to a set $P \in \overline{\mathbb{R}}^{\ell}$.

closure

Definition 2.3 (Closure) Let $P \subseteq \overline{\mathbb{R}}^{\ell}$. A point $\boldsymbol{p} \in \overline{\mathbb{R}}^{\ell}$ is a *point of closure* of P if every neighborhood of $\boldsymbol{p}$ contains a point in P. The *closure* of $P \subseteq \overline{\mathbb{R}}^{\ell}$ is given by the set $cl(P) \subseteq \overline{\mathbb{R}}^{\ell}$ consisting of all points of closure of P, i.e.,

$$cl(P) \;\coloneqq\; \{\boldsymbol{p} \in \overline{\mathbb{R}}^{\ell} \mid \forall\, N \in \mathcal{N}(\boldsymbol{p}) \colon\; N \cap P \neq \emptyset\}.$$

Example 2.1 Consider the 1-dimensional case ($\ell = 1$). For $a, b \in \overline{\mathbb{R}}$ with $a < b$ we have

$$cl(\,(a,b)\,) \;=\; cl(\,(a,b]\,) \;=\; cl(\,[a,b)\,) \;=\; cl([a,b]) \;=\; [a,b]$$

In particular, $cl(\mathbb{R}) = cl(\,(-\infty,+\infty)\,) = [-\infty,+\infty] = \overline{\mathbb{R}}$.
As an example for the 2-dimensional case ($\ell = 2$), we consider the set

$$P \;=\; \big\{\boldsymbol{p} \in \overline{\mathbb{R}}^{2} \;\big|\; \boldsymbol{p}(1) \geq 0 \text{ and } dist(\boldsymbol{p}, \mathbf{0}) < 1\big\}$$

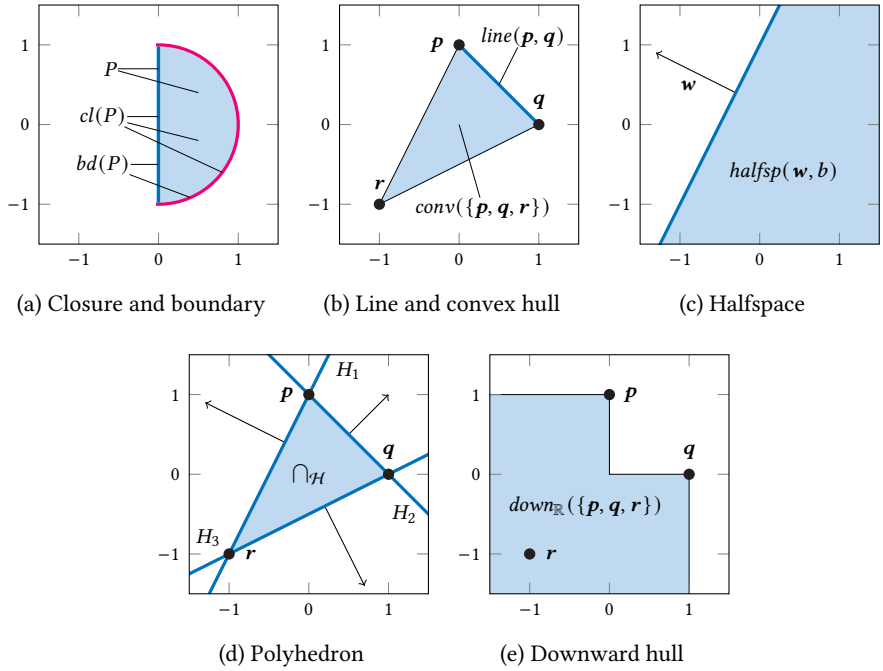
which is depicted by the blue line segment and the light blue area in Figure 2.1a. Each point in $\big\{\boldsymbol{p} \in \overline{\mathbb{R}}^{2} \;\big|\; \boldsymbol{p}(1) \geq 0 \text{ and } dist(\boldsymbol{p}, \mathbf{0}) = 1\big\}$—depicted by the pink semicircle in Figure 2.1a—is a point of closure of P but not contained in P. The closure $cl(P)$ of P is the union of the blue line segment, the light blue area, and the pink semicircle.

Let $P \subseteq \overline{\mathbb{R}}^{\ell}$ be a set of points. It always holds that $P \subseteq cl(P)$ since every neighborhood $N \in \mathcal{N}(\boldsymbol{p})$ of a point $\boldsymbol{p}$ contains $\boldsymbol{p}$ itself. In case P is equal to its closure, i.e., $P = cl(P)$, we say that P is *closed*. However, $cl(P) \subseteq P$ does not hold in general as shown in Example 2.1. Intuitively, $cl(P)$ contains all points in P as well as the points that lie on the boundary between P and $\overline{\mathbb{R}}^{\ell} \setminus P$. Formally, the *boundary* of a set P is given by

boundary

$$bd(P) \;\coloneqq\; cl(P) \cap cl(\overline{\mathbb{R}}^{\ell} \setminus P).$$

Closed sets thus contain all points on their boundary.



(a) Closure and boundary (b) Line and convex hull (c) Halfspace

(d) Polyhedron (e) Downward hull

Figure 2.1: Geometry sets and operations (cf. Examples 2.1 to 2.6)

Example 2.2 The boundary of the set P from Example 2.1 coincides with the blue line segment and the pink semicircle in Figure 2.1a.

Convex Sets | We define the line segment between two points $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^\ell$ as

$$\mathit{line}(\boldsymbol{p}, \boldsymbol{q}) \;:=\; \big\{\boldsymbol{p} \in \mathbb{R}^\ell \;\big|\; \exists\, \lambda \in [0,1] \colon\; \boldsymbol{p} = \lambda \cdot \boldsymbol{p} + (1-\lambda) \cdot \boldsymbol{q}\big\}.$$

Line segments do not straightforwardly generalize to points over the extended reals.

Definition 2.4 (Convex Hull) The *convex hull* of a set $P \subseteq \mathbb{R}^\ell$ is given by

convex hull

$$\mathit{conv}(P) \;:=\; \big\{\boldsymbol{p} \in \mathbb{R}^\ell \;\big|\; \exists\, \boldsymbol{q}, \boldsymbol{r} \in P \colon\; \boldsymbol{p} \in \mathit{line}(\boldsymbol{q}, \boldsymbol{r})\big\}.$$

We say that $P \in \mathbb{R}^\ell$ is *convex* if it is equal to its convex hull, i.e., $P = \mathit{conv}(P)$. A *vertex* of a convex set P is a point $\boldsymbol{p} \in P$ such that $P \neq \mathit{conv}(P \setminus \{\boldsymbol{p}\})$. For any $\boldsymbol{p} \in \mathit{conv}(P)$ there is $\mu \colon P \to [0,1]$ with $\sum_{\boldsymbol{q} \in P} \mu(\boldsymbol{q}) = 1$ such that $\boldsymbol{p} = \sum_{\boldsymbol{q} \in P} \mu(\boldsymbol{q}) \cdot \boldsymbol{q}$.

distribution We say that μ is a *distribution* over P (cf. Definition 2.12).

Example 2.3 The blue line in Figure 2.1b depicts the line segment $line(\boldsymbol{p}, \boldsymbol{q})$ for the two indicated points $\boldsymbol{p} = \langle 0, 1\rangle$ and $\boldsymbol{q} = \langle 1, 0\rangle$. The convex hull $conv(\{\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}\})$ with $\boldsymbol{p}$ and $\boldsymbol{q}$ as before and $\boldsymbol{r} = \langle -1, -1\rangle$ is indicated by the light blue area (including its boundary). $\boldsymbol{p}$, $\boldsymbol{q}$, and $\boldsymbol{r}$ are the vertices of $conv(\{\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}\})$.

halfspace **Definition 2.5 (Halfspace)** An ℓ-dimensional *halfspace* is a set

$$halfsp(\boldsymbol{w}, b) \;:=\; \{\boldsymbol{p} \in \mathbb{R}^\ell \mid \boldsymbol{w} \cdot \boldsymbol{p} \leq b\}$$

for some $\boldsymbol{w} \in \mathbb{R}^\ell \setminus \{\boldsymbol{0}\}$ and $b \in \mathbb{R}$.

Given a halfspace $H = halfsp(\boldsymbol{w}, b)$, we call $\boldsymbol{w}$ the *normal vector* and b the *offset* of H. The boundary of a halfspace $bd(H) = \{\boldsymbol{p} \in \mathbb{R}^\ell \mid \boldsymbol{w} \cdot \boldsymbol{p} = b\}$ is called a *plane*. The normal vector of a halfspace is always orthogonal to its boundary.

Example 2.4 For $\boldsymbol{w} = \langle -1, 0.5\rangle$ and $b = 0.5$, the halfspace $halfsp(\boldsymbol{w}, b)$ is shown in Figure 2.1c. Its boundary $bd(halfsp(\boldsymbol{w}, b))$ is indicated by the blue line.

Lemma 2.1 For a halfspace $H = halfsp(\boldsymbol{w}, b)$ and a point $\boldsymbol{p} \notin H$ we have

$$dist(\boldsymbol{p}, H) \;=\; dist(\boldsymbol{p}, bd(H)) \;=\; \frac{\boldsymbol{w} \cdot \boldsymbol{p} - b}{\sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}}.$$

Proof. The above expression is well-defined: we have $\sqrt{\boldsymbol{w} \cdot \boldsymbol{w}} > 0$ because $\boldsymbol{w} \neq \boldsymbol{0}$ by Definition 2.5. Now let $\boldsymbol{q} := \arg\min_{\boldsymbol{q}' \in H} dist(\boldsymbol{q}', \boldsymbol{p})$ be the point in H with the smallest distance to $\boldsymbol{p}$. The point $\boldsymbol{q}$ must lie on the plane $Q := bd(H) = \{\boldsymbol{q}' \in \mathbb{R}^\ell \mid \boldsymbol{w} \cdot \boldsymbol{q}' = b\}$ and the vector $\boldsymbol{q} - \boldsymbol{p}$ must be orthogonal to Q and thus parallel to $\boldsymbol{w}$, i.e., $\boldsymbol{p} - \boldsymbol{q} = a \cdot \boldsymbol{w}$ for some $a \in \mathbb{R}$. We have

$$\boldsymbol{w} \cdot \boldsymbol{p} - b \;=\; \boldsymbol{w} \cdot \boldsymbol{p} - \boldsymbol{w} \cdot \boldsymbol{q} \;=\; \boldsymbol{w} \cdot (\boldsymbol{p} - \boldsymbol{q}) \;=\; \boldsymbol{w} \cdot (a \cdot \boldsymbol{w}) \;=\; a \cdot (\boldsymbol{w} \cdot \boldsymbol{w}). \tag{2.1}$$

We get that $a > 0$ because $\boldsymbol{p} \notin H$ yields

$$\boldsymbol{w} \cdot \boldsymbol{p} > b \iff \boldsymbol{w} \cdot \boldsymbol{p} - b > 0 \overset{(2.1)}{\iff} a \cdot \underbrace{(\boldsymbol{w} \cdot \boldsymbol{w})}_{>0} > 0$$

The lemma follows with $dist(\boldsymbol{p}, H) = dist(\boldsymbol{p}, Q) = dist(\boldsymbol{p}, \boldsymbol{q})$ and

$$dist(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{(\boldsymbol{p}-\boldsymbol{q})\cdot(\boldsymbol{p}-\boldsymbol{q})} = \sqrt{(a\cdot\boldsymbol{w})\cdot(a\cdot\boldsymbol{w})} = |a|\cdot\sqrt{\boldsymbol{w}\cdot\boldsymbol{w}} = a\cdot\sqrt{\boldsymbol{w}\cdot\boldsymbol{w}}$$
$$= \frac{a\cdot(\boldsymbol{w}\cdot\boldsymbol{w})}{\sqrt{\boldsymbol{w}\cdot\boldsymbol{w}}} \overset{(2.1)}{=} \frac{\boldsymbol{w}\cdot\boldsymbol{p}-b}{\sqrt{\boldsymbol{w}\cdot\boldsymbol{w}}}. \qquad \blacksquare$$

Definition 2.6 (Polyhedron) A *polyhedron* is the intersection $\bigcap_{i=1}^{n} H_i$ of finitely many halfspaces $H_1, \ldots, H_n \subseteq \mathbb{R}^\ell$. polyhedron

Halfspaces and polyhedra are closed and convex. The convex hull $conv(P)$ of a finite set of points $P \subseteq \mathbb{R}^\ell$ is a polyhedron [Sol15, Theorem 9.20]. The Quickhull algorithm [BDH96] efficiently converts a finite set of points P to a finite set of halfspaces $\mathcal{H} = \{H_1, \ldots, H_n\}$ such that $\bigcap_{\mathcal{H}} = \bigcap_{i=1}^{n} H_i = conv(P)$.

Example 2.5 The intersection of the halfspaces $H_1 = halfsp(\langle -1, 0.5\rangle, 0.5)$, $H_2 = halfsp(\langle 0.5, 0.5\rangle, 0.5)$, and $H_3 = halfsp(\langle 0.5, -1\rangle, 0.5)$ yields the polyhedron $\bigcap_{\mathcal{H}}$ with $\mathcal{H} = H_1, H_2, H_3$ as indicated by the light blue area in Figure 2.1d. Observe that $\bigcap_{\mathcal{H}} = conv(\{\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}\})$ for $\boldsymbol{p}$, $\boldsymbol{q}$, and $\boldsymbol{r}$ as in Example 2.3.

Downward Hulls | We study sets of points that intuitively are closed under considering smaller points (w.r.t. the order $\leq$).

Definition 2.7 (Downward Hull) The *downward hull* of $P \subseteq \overline{\mathbb{R}}^\ell$ is given by downward hull

$$down(P) := \{\boldsymbol{p} \in \overline{\mathbb{R}}^\ell \mid \exists\, \boldsymbol{p}' \in P\colon\ \boldsymbol{p} \leq \boldsymbol{p}'\}.$$

The real-valued fragment of a downward hull is denoted by $down_{\mathbb{R}}(P) := down(P) \cap \mathbb{R}^\ell$.

Example 2.6 The set $down_{\mathbb{R}}(\{\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}\})$ for $\boldsymbol{p}$, $\boldsymbol{q}$, and $\boldsymbol{r}$ as in Example 2.3 is shown by the light blue area in Figure 2.1e. The downward hull $down(\{\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}\})$ also contains the points $\langle a, -\infty\rangle$ and $\langle -\infty, a\rangle$ for any $a \leq 1$.

We say that $P \subseteq \overline{\mathbb{R}}^\ell$ is *downward closed* if it is equal to its downward hull, i.e., $P = down(P)$.

Lemma 2.2 If $P \subseteq \overline{\mathbb{R}}^\ell$ is downward closed, then $cl(P)$ is also downward closed.

Proof. $cl(P) \subseteq down(cl(P))$ is obvious. To show $cl(P) \supseteq down(cl(P))$, let $\boldsymbol{p} \in down(cl(P))$, i.e., there is $\boldsymbol{p}' \in cl(P)$ with $\boldsymbol{p} \leq \boldsymbol{p}'$. To show $\boldsymbol{p} \in cl(P)$, we consider

an arbitrary neighborhood N of $\boldsymbol{p}$ and show that N contains a point $\boldsymbol{p}_N \in P$. Let $N = \bigtimes_{i=1}^{\ell} N_i$, where $N_i \subseteq \overline{\mathbb{R}}$ is a neighborhood of $\boldsymbol{p}(i)$ for each $i \in \{1..\ell\}$. We construct the neighborhood $N' = \bigtimes_{i=1}^{\ell} N'_i$ of $\boldsymbol{p}'$ by setting

$$N'_i \;:=\; \begin{cases} (\boldsymbol{p}'(i)-\varepsilon,\ \boldsymbol{p}'(i)+\varepsilon) & \text{if } \boldsymbol{p}(i), \boldsymbol{p}'(i) \in \mathbb{R} \text{ and } N_i \supseteq (\boldsymbol{p}(i)-\varepsilon,\ \boldsymbol{p}(i)+\varepsilon) \\ (\boldsymbol{p}(i),\ \infty] & \text{if } \boldsymbol{p}(i) \in \mathbb{R} \text{ and } \boldsymbol{p}'(i) = \infty \\ N_i & \text{if } \boldsymbol{p}(i) = \boldsymbol{p}'(i) \in \{-\infty, \infty\} \\ (\boldsymbol{p}'(i) - 1,\ \boldsymbol{p}'(i) + 1) & \text{if } \boldsymbol{p}(i) = -\infty \text{ and } \boldsymbol{p}'(i) \in \mathbb{R} \\ (0, \infty] & \text{if } \boldsymbol{p}(i) = -\infty \text{ and } \boldsymbol{p}'(i) = \infty \end{cases}$$

for all $i \in \{1..\ell\}$. Other cases do not occur as $\boldsymbol{p}(i) \leq \boldsymbol{p}'(i)$ for all $i \in \{1..\ell\}$. Since $\boldsymbol{p}' \in cl(P)$, neighborhood N' of $\boldsymbol{p}'$ contains a point $\boldsymbol{p}'_{N'} \in N' \cap P$. We construct the point $\boldsymbol{p}_N \in \overline{\mathbb{R}}^{\ell}$ with

$$\boldsymbol{p}_N(i) \;:=\; \begin{cases} \boldsymbol{p}'_{N'}(i) - \boldsymbol{p}'(i) + \boldsymbol{p}(i) & \text{if } \boldsymbol{p}(i), \boldsymbol{p}'(i) \in \mathbb{R} \\ \boldsymbol{p}'_{N'}(i) & \text{if } \boldsymbol{p}(i) = \boldsymbol{p}'(i) \in \{-\infty, \infty\} \\ \boldsymbol{p}(i) & \text{otherwise} \end{cases}$$

for all $i \in \{1..\ell\}$. We have $\boldsymbol{p}_N \in N$ and $\boldsymbol{p}_N \leq \boldsymbol{p}'_{N'}$. Since P is downward closed and $\boldsymbol{p}'_{N'} \in P$, it follows that $\boldsymbol{p}_N \in P$, i.e., $\boldsymbol{p}_N \in N \cap P$.
In summary, we have thus shown that every neighborhood N of a point $\boldsymbol{p} \in down(cl(P))$ contains a point in P, yielding $\boldsymbol{p} \in cl(P)$. ■

We often consider downward closed sets of the form $down(conv(P))$ for some finite set $P \subseteq \mathbb{R}^{\ell}$. The following lemma allows us to obtain halfspace representations for such sets.

Lemma 2.3 For a finite, non-empty set of points $P \subseteq \mathbb{R}^{\ell}$ consider the set $\widehat{P} := P \cup \{\boldsymbol{p} - \mathbf{1}_i \mid \boldsymbol{p} \in P, i \in \{1..\ell\}\}$ and let $\widehat{\mathcal{H}}$ be some set of halfspaces representing $conv(\widehat{P})$, i.e., $conv(\widehat{P}) = \bigcap_{\widehat{\mathcal{H}}}$. Then $down_{\mathbb{R}}(conv(P))$ is a polyhedron with

$$down_{\mathbb{R}}(conv(P)) \;=\; \bigcap_{\mathcal{H}} \quad \text{where} \quad \mathcal{H} \;=\; \left\{ halfsp(\boldsymbol{w}, b) \in \widehat{\mathcal{H}} \mid \boldsymbol{w} \in (\mathbb{R}_{\geq 0})^{\ell} \right\}.$$

Example 2.7 Consider the set $P = \{\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}\} \subseteq \mathbb{R}^2$ as illustrated in Figure 2.2. Lemma 2.3 yields a representation for $down_{\mathbb{R}}(conv(P))$ as an intersection of halfspaces. First, we obtain a halfspace representation $\widehat{\mathcal{H}}$ for $conv(\widehat{P})$ (light blue area) using, e.g., the Quickhull algorithm [BDH96] on the auxiliary pointset

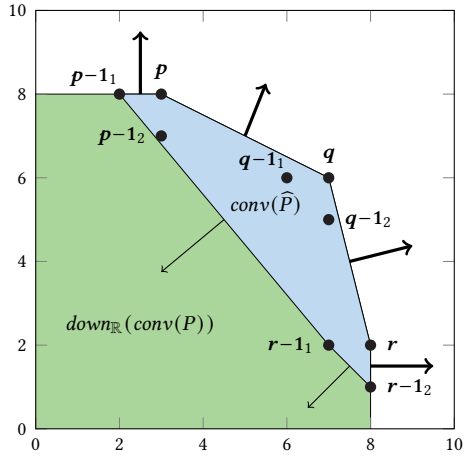


Figure 2.2: Illustration for Lemma 2.3 (cf. Example 2.7)

$\widehat{P} = \bigcup_{\boldsymbol{p}' \in \{\boldsymbol{p},\boldsymbol{q},\boldsymbol{r}\}} \{\boldsymbol{p}', \boldsymbol{p}' - \mathbf{1}_1, \boldsymbol{p}' - \mathbf{1}_2\}$. In the figure, the six halfspaces in $\widehat{\mathcal{H}}$ are indicated by the six arrows representing the corresponding normal vectors. By dropping those halfspaces from $\widehat{\mathcal{H}}$ whose normal vectors have at least one negative entry, we obtain a new set of halfspaces $\mathcal{H}$ (bold faced arrows) that represents the polyhedron $down_{\mathbb{R}}(conv(P))$ (light blue and green area).

Proof (of Lemma 2.3). We proceed in five steps: we first show that $down_{\mathbb{R}}(conv(P))$ is a polyhedron. Secondly, we show that both polyhedra $down_{\mathbb{R}}(conv(P))$ and $conv(\widehat{P})$ can be represented by unique, irreducible sets of halfspaces. The third step is to show that halfspaces used to represent $down_{\mathbb{R}}(conv(P))$ must have non-negative normal vectors. We then use these insights to show that $down_{\mathbb{R}}(conv(P)) \supseteq \bigcap_{\mathcal{H}}$. We conclude the proof by showing $down_{\mathbb{R}}(conv(P)) \subseteq \bigcap_{\mathcal{H}}$.
The proof makes use of the *affine hull* of a set $Q \subseteq \mathbb{R}^\ell$ which is defined by

affine hull

$$affine(Q) := \left\{ \sum_{i=1}^{m} \lambda_i \cdot \boldsymbol{p}_i \;\middle|\; m > 0,\ \boldsymbol{p}_1, \ldots, \boldsymbol{p}_m \in Q,\ \lambda_1, \ldots, \lambda_m \in \mathbb{R},\ \text{and } \sum_{i=1}^{m} \lambda_i = 1 \right\}.$$

Step 1 | We can write $down_{\mathbb{R}}(conv(P))$ as

$$down_{\mathbb{R}}(conv(P)) = \left\{ \boldsymbol{p} + \boldsymbol{q} \mid \boldsymbol{p} \in conv(P), \boldsymbol{q} \in \left\{ \sum_{i=1}^{\ell} a_i \cdot -\mathbf{1}_i \mid a_1, \ldots, a_\ell \in \mathbb{R}_{\geq 0} \right\} \right\}$$

which—using standard terminology as in, e.g., [Zie95; Sol15]—is the Minkowski sum of the convex set $conv(P)$ and the *cone* $\{\sum_{i=1}^{\ell} a_i \cdot -\mathbf{1}_i \mid a_1, \ldots, a_\ell \in \mathbb{R}_{\geq 0}\}$. By [Sol15, Theorem 9.19] it follows that $down_{\mathbb{R}}(conv(P))$ is a polyhedron.

Step 2 | We have $affine(\widehat{P}) = \mathbb{R}^\ell$: for a point $\boldsymbol{q} \in \mathbb{R}^\ell$ consider some $\boldsymbol{p} \in \widehat{P}$ and let $\boldsymbol{v} \coloneqq \boldsymbol{p} - \boldsymbol{q}$. We can write $\boldsymbol{q}$ as an affine combination of the points $\boldsymbol{p}, \boldsymbol{p}-\mathbf{1}_1, \ldots, \boldsymbol{p}-\mathbf{1}_\ell \in \widehat{P}$:

$$\begin{aligned} &\left(1 - \textstyle\sum_{i=1}^{\ell} \boldsymbol{v}(i)\right) \cdot \boldsymbol{p} + \sum_{i=1}^{\ell} \boldsymbol{v}(i) \cdot (\boldsymbol{p} - \mathbf{1}_i) \\ &= \boldsymbol{p} - \textstyle\sum_{i=1}^{\ell} \boldsymbol{v}(i) \cdot \mathbf{1}_i = \boldsymbol{p} - \boldsymbol{v} \;=\; \boldsymbol{p} - \boldsymbol{p} + \boldsymbol{q} = \boldsymbol{q}. \end{aligned}$$

With $\widehat{P} \subseteq conv(\widehat{P}) \subseteq down_{\mathbb{R}}(conv(P))$ we get

$$affine(conv(\widehat{P})) \;=\; affine(down_{\mathbb{R}}(conv(P))) \;=\; \mathbb{R}^\ell.$$

We further note that both sets $conv(\widehat{P})$ and $down_{\mathbb{R}}(conv(P))$ are not equal to $\mathbb{R}^\ell$ since both do not contain—for example—the point $\mathbf{1}_\ell \cdot \max\{\boldsymbol{p}(\ell) + 1 \mid \boldsymbol{p} \in P\} \in \mathbb{R}^\ell$. In terms of [Sol15], it follows that both polyhedra $conv(\widehat{P})$ and $down_{\mathbb{R}}(conv(P))$ are not a plane [Sol15, Theorem 1.47] and halfplanes of their affine hull $\mathbb{R}^\ell$ are halfspaces as in Definition 2.5. Thus, [Sol15, Theorem 9.2] yields that there must be unique, irreducible sets of halfspaces $\widehat{\mathcal{H}'}$ and $\mathcal{H}^*$ representing $conv(\widehat{P})$ and $down_{\mathbb{R}}(conv(P))$, respectively. In other words we have

$$conv(\widehat{P}) \;=\; \textstyle\bigcap_{\widehat{\mathcal{H}'}} \qquad \text{and} \qquad down_{\mathbb{R}}(conv(P)) \;=\; \bigcap_{\mathcal{H}^*}$$

and every set of halfspaces representing $conv(\widehat{P})$ (resp. $down_{\mathbb{R}}(conv(P))$) must be a superset of $\widehat{\mathcal{H}'}$ (resp. $\mathcal{H}^*$). In particular, the set $\widehat{\mathcal{H}}$ considered in the lemma satisfies $\widehat{\mathcal{H}'} \subseteq \widehat{\mathcal{H}}$.

Step 3 | Let $H^* = halfsp(\boldsymbol{w}, b) \in \mathcal{H}^*$ with $\mathcal{H}^*$ as above. We argue that $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$. To see this, assume towards a contradiction that there is $i \in \{1..\ell\}$ with $\boldsymbol{w}(i) < 0$. Then, for $\boldsymbol{p} \in P \subseteq down_{\mathbb{R}}(conv(P))$ we have $\boldsymbol{p} \in H^*$, i.e., $\boldsymbol{w} \cdot \boldsymbol{p} \leq b$ or, equivalently, $b - \boldsymbol{w} \cdot \boldsymbol{p} \geq 0$. Now consider the point

$$\boldsymbol{p}' \;\coloneqq\; \boldsymbol{p} + \mathbf{1}_i \cdot \Big(\underbrace{\frac{b - \boldsymbol{w} \cdot \boldsymbol{p}}{\boldsymbol{w}(i)}}_{\leq 0} - 1 \Big)$$

Clearly, $\boldsymbol{p}' \leq \boldsymbol{p}$ and thus $\boldsymbol{p}' \in \mathit{down}_{\mathbb{R}}(\{\boldsymbol{p}\}) \subseteq \mathit{down}_{\mathbb{R}}(\mathit{conv}(P))$. However, we also have $\boldsymbol{p}' \notin H$ since

$$\begin{aligned} \boldsymbol{w} \cdot \boldsymbol{p}' &= \boldsymbol{w} \cdot \boldsymbol{p} + \boldsymbol{w}(i) \cdot \left(\frac{b - \boldsymbol{w} \cdot \boldsymbol{p}}{\boldsymbol{w}(i)} - 1 \right) \\ &= \boldsymbol{w} \cdot \boldsymbol{p} + (b - \boldsymbol{w} \cdot \boldsymbol{p}) - \boldsymbol{w}(i) \\ &= b - \boldsymbol{w}(i) > b. \end{aligned}$$

This yields a contradiction as $H \supseteq \mathit{down}_{\mathbb{R}}(\mathit{conv}(P))$. The assumption that there is $i \in \{1..\ell\}$ with $\boldsymbol{w}(i) < 0$ must be false.

Step 4 | Using $\mathcal{H}^*$ as in the second step, we show $\mathcal{H}^* \subseteq \mathcal{H}$ from which we get

$$\bigcap\nolimits_{\mathcal{H}} \subseteq \bigcap\nolimits_{\mathcal{H}^*} = \mathit{down}_{\mathbb{R}}(\mathit{conv}(P)).$$

Let $H^* = \mathit{halfsp}(\boldsymbol{w}, b) \in \mathcal{H}^*$. Since we already have $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$, it remains to show $H^* \in \widehat{\mathcal{H}}'$ with $\widehat{\mathcal{H}}' \subseteq \widehat{\mathcal{H}}$ as in the second step. Consider the sets

$$\begin{aligned} B &:= \mathit{bd}(H) = \{\boldsymbol{p} \in \mathbb{R}^\ell \mid \boldsymbol{w} \cdot \boldsymbol{p} = b\}, \\ Q^* &:= B \cap \mathit{down}_{\mathbb{R}}(\mathit{conv}(P)), \quad \text{and} \quad \widehat{Q} := B \cap \mathit{conv}(\widehat{P}). \end{aligned}$$

Q^* is a so-called *facet* of $\mathit{down}_{\mathbb{R}}(\mathit{conv}(P))$, i.e., Q^* has dimension $\ell - 1$, $\mathit{down}_{\mathbb{R}}(\mathit{conv}(P)) \setminus Q^*$ is convex, and $Q^* = \mathit{affine}(Q^*) \cap \mathit{down}_{\mathbb{R}}(\mathit{conv}(P))$ [Sol15, Theorem 7.2, Theorem 9.12]. We show similar properties for $\widehat{P}$ with respect to $\mathit{conv}(\widehat{P})$ from which we can infer that $\widehat{P}$ is a facet of $\mathit{conv}(\widehat{P})$.
Let $\boldsymbol{p} \in Q^*$. We show $\boldsymbol{p} \in \mathit{affine}(\widehat{Q})$ by distinguishing two cases.

- If $\boldsymbol{p} \in \mathit{conv}(P)$ then also $\boldsymbol{p} \in \mathit{conv}(\widehat{P})$. With $\boldsymbol{p} \in Q^* \subseteq B$ we get

$$\boldsymbol{p} \in B \cap \mathit{conv}(\widehat{P}) = \widehat{Q} \subseteq \mathit{affine}(\widehat{Q}).$$

- If $\boldsymbol{p} \in \mathit{down}_{\mathbb{R}}(\mathit{conv}(P)) \setminus \mathit{conv}(P)$ then there must be $\boldsymbol{p}' \in \mathit{conv}(P)$ with $\boldsymbol{p} \leq \boldsymbol{p}'$. Since $\boldsymbol{p} \in B$, $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$, and $\boldsymbol{p}' \in \mathit{down}_{\mathbb{R}}(\mathit{conv}(P)) \subseteq H^*$ we get

$$b = \boldsymbol{w} \cdot \boldsymbol{p} \leq \boldsymbol{w} \cdot \boldsymbol{p}' \leq b$$

  yielding $\boldsymbol{w} \cdot \boldsymbol{p} = \boldsymbol{w} \cdot \boldsymbol{p}'$ and thus $\boldsymbol{p}' \in B$. Moreover, $\forall\, i \in \{1..\ell\}\colon \boldsymbol{p}(i) \neq \boldsymbol{p}'(i)$ implies $\boldsymbol{w}(i) = 0$. Let $\mathcal{I}_0 := \{i \in \{1..\ell\} \mid \boldsymbol{w}(i) = 0\}$ be the set of indices where $\boldsymbol{w}$ is 0. We have $\forall\, i \in \mathcal{I}_0\colon \boldsymbol{p}' - \mathbf{1}_i \in B \cap \mathit{conv}(\widehat{P}) = \widehat{Q}$. Let $\boldsymbol{v} := \boldsymbol{p}' - \boldsymbol{p}$. Using a similar construction as in Step 2, we can express $\boldsymbol{p}$ as the following affine

combination of points from $\widehat{Q}$, yielding $\boldsymbol{p} \in \mathit{affine}(\widehat{Q})$:

$$\begin{aligned}&\big(1 - \textstyle\sum_{i \in \mathcal{I}_0} \boldsymbol{v}(i)\big) \cdot \boldsymbol{p}' + \sum_{i \in \mathcal{I}_0} \boldsymbol{v}(i) \cdot (\boldsymbol{p}' - \mathbf{1}_i)\\ &= \boldsymbol{p}' - \textstyle\sum_{i \in \mathcal{I}_0} \boldsymbol{v}(i) \cdot \mathbf{1}_i = \boldsymbol{p}' - \boldsymbol{v} = \boldsymbol{p}' - \boldsymbol{p}' + \boldsymbol{p} = \boldsymbol{p}.\end{aligned}$$

We thus have $\widehat{Q} \subseteq Q^* \subseteq \mathit{affine}(\widehat{Q})$ yielding $\mathit{affine}(\widehat{Q}) = \mathit{affine}(Q^*)$. On the one hand, we can infer that $\widehat{Q}$ has the same dimension as Q^*, i.e., both have dimension $\ell - 1$. On the other hand, with $\mathit{conv}(\widehat{P}) = \mathit{down}_{\mathbb{R}}(\mathit{conv}(P)) \cap \mathit{conv}(\widehat{P})$ we get

$$\begin{aligned}\mathit{affine}(\widehat{Q}) \cap \mathit{conv}(\widehat{P}) &= \mathit{affine}(Q^*) \cap \mathit{conv}(\widehat{P})\\ &= \mathit{affine}(Q^*) \cap \mathit{down}_{\mathbb{R}}(\mathit{conv}(P)) \cap \mathit{conv}(\widehat{P})\\ &= Q^* \cap \mathit{conv}(\widehat{P})\\ &= B \cap \mathit{down}_{\mathbb{R}}(\mathit{conv}(P)) \cap \mathit{conv}(\widehat{P})\\ &= B \cap \mathit{conv}(\widehat{P}) = \widehat{Q}.\end{aligned}$$

Moreover, we have that $\mathit{conv}(\widehat{P}) \setminus \widehat{Q}$ is convex: let $\boldsymbol{p}, \boldsymbol{q} \in \mathit{conv}(\widehat{P}) \setminus \widehat{Q}$. From $\mathit{conv}(\widehat{P}) \subseteq \mathit{down}_{\mathbb{R}}(\mathit{conv}(P)) \subseteq H^*$ we get $\boldsymbol{w} \cdot \boldsymbol{p} < b$ and $\boldsymbol{w} \cdot \boldsymbol{q} < b$. Thus, for $\lambda \in [0, 1]$:

$$\boldsymbol{w} \cdot \big(\lambda \cdot \boldsymbol{p} + (1 - \lambda) \cdot \boldsymbol{q}\big) = \lambda \cdot \underbrace{\boldsymbol{w} \cdot \boldsymbol{p}}_{<b} + (1 - \lambda) \cdot \underbrace{\boldsymbol{w} \cdot \boldsymbol{q}}_{<b} < b.$$

This yields $\mathit{line}(\boldsymbol{p}, \boldsymbol{q}) \subseteq \mathit{conv}(\widehat{P}) \setminus \widehat{Q}$.
Using [Sol15, Theorem 7.2, Theorem 9.12], we conclude that $\widehat{P}$ is a facet of $\mathit{conv}(\widehat{P})$. By [Sol15, Definition 9.11], that must mean that $\widehat{\mathcal{H}}'$ contains a halfspace with boundary B, i.e., $H^* \in \widehat{\mathcal{H}}'$.

Step 5 | The last step is to show that $\mathit{down}_{\mathbb{R}}(\mathit{conv}(P)) \subseteq \bigcap_{\mathcal{H}}$. For $\boldsymbol{p} \in \mathit{down}_{\mathbb{R}}(\mathit{conv}(P))$ there is $\boldsymbol{p}' \in \mathit{conv}(P)$ with $\boldsymbol{p} \leq \boldsymbol{p}'$. Since $P \subseteq \widehat{P}$ and $\mathcal{H} \subseteq \widehat{\mathcal{H}}$ we have

$$\boldsymbol{p}' \in \mathit{conv}(P) \subseteq \mathit{conv}(\widehat{P}) = \textstyle\bigcap_{\widehat{\mathcal{H}}} \subseteq \bigcap_{\mathcal{H}}.$$

Now let $H = \mathit{halfsp}(\boldsymbol{w}, b) \in \mathcal{H}$. It follows that $\boldsymbol{p}' \in H$. Due to $\boldsymbol{p} \leq \boldsymbol{p}'$ and $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$ we get

$$\boldsymbol{w} \cdot \boldsymbol{p} \leq \boldsymbol{w} \cdot \boldsymbol{p}' \leq b$$

and thus $\boldsymbol{p} \in H$ for any $H \in \mathcal{H}$. Hence, $\boldsymbol{p} \in \bigcap_{\mathcal{H}}$. ■

2.1.3 Linear Programming

Problem 2.1: Linear Programming (LP)

Input: $\ell \in \mathbb{N}$, a set of ℓ-dimensional halfspaces $\mathcal{H}$, $\boldsymbol{v} \in \mathbb{R}^\ell$

Output:
$$\begin{cases} \text{`INFEASIBLE'} & \text{if } \bigcap_{\mathcal{H}} = \emptyset \\ \text{`UNBOUNDED'} & \text{if } \sup_{\boldsymbol{p} \in \bigcap_{\mathcal{H}}} (\boldsymbol{v} \cdot \boldsymbol{p}) = +\infty \\ \boldsymbol{p}_{\text{opt}} \in \arg\max_{\boldsymbol{p} \in \bigcap_{\mathcal{H}}} (\boldsymbol{v} \cdot \boldsymbol{p}) & \text{otherwise} \end{cases}$$

LP, linear programming

For an LP instance as above, each of the given halfspaces $\mathit{halfsp}(\boldsymbol{w}, b) \in \mathcal{H}$ describes a linear inequality $\boldsymbol{w} \cdot \boldsymbol{x} \leq b$—also called *constraint*—over a set of ℓ real-valued variables $\boldsymbol{x} = \langle x_1, \ldots, x_\ell \rangle$. Similarly, we can interpret the given vector $\boldsymbol{v}$ as the coefficients of a linear *optimization function* $\boldsymbol{v} \cdot \boldsymbol{x}$. We can thus describe instances for LP by providing a linear optimization function and a finite number of linear constraints over a given set of variables.

constraint

An LP instance $\langle \ell, \mathcal{H}, \boldsymbol{v} \rangle$ is called *feasible* iff there is some assignment

$$\mathit{val} \colon \{x_1, \ldots, x_\ell\} \to \mathbb{R}$$

of variables to values that satisfies all constraints simultaneously. The optimization function can be omitted if we are only interested in feasibility. We further say that the instance is *bounded* iff the constraints imply an upper bound on the largest possible value of the optimization function. For a variable assignment $\mathit{val} \colon \{x_1, \ldots, x_\ell\} \to \mathbb{R}$ we use the notation $\mathit{val}(\boldsymbol{x}) \coloneqq \langle \mathit{val}(x_1), \ldots, \mathit{val}(x_\ell) \rangle$. Solving a feasible and bounded LP instance corresponds to finding an assignment val that maximizes the value $\boldsymbol{v} \cdot \mathit{val}(\boldsymbol{x})$ of the optimization function such that all constraints $\boldsymbol{w} \cdot \mathit{val}(\boldsymbol{x}) \leq b$ for $\mathit{halfsp}(\boldsymbol{w}, b) \in \mathcal{H}$ are satisfied. We call such an assignment val a *solution* of the LP.

We also study a variant of LP where some variables are restricted to the integer domain.

Problem 2.2: Mixed Integer Linear Programming (MILP)

Input: $\ell, m \in \mathbb{N}$, a set of $(\ell + m)$-dimensional halfspaces $\mathcal{H}$, $\boldsymbol{v} \in \mathbb{R}^{\ell+m}$

Output:
$$\begin{cases} \text{`INFEASIBLE'} & \text{if } Q = \emptyset \\ \text{`UNBOUNDED'} & \text{if } \sup_{\boldsymbol{p} \in Q} (\boldsymbol{v} \cdot \boldsymbol{p}) = +\infty \\ \boldsymbol{p}_{\text{opt}} \in \arg\max_{\boldsymbol{p} \in Q} (\boldsymbol{v} \cdot \boldsymbol{x}) & \text{otherwise} \end{cases}$$
using $Q \coloneqq \bigcap_{\mathcal{H}} \cap (\mathbb{Z}^\ell \times \mathbb{R}^m)$

MILP, mixed integer linear programming

LP instances can be solved in polynomial time [Sch99, Theorem 13.4] whereas MILP is NP-complete (see [Sch99, Theorem 18.1] for hardness and [PDM17, Proposition 3]

for containedness). There are several tools such as CPLEX [Stu21], GLPK [GNU20], GUROBI [Gur22], and SCIP [GABC+20] that can solve instances of LP and MILP. In both cases, algorithms typically employ the *simplex method* which—while having an exponential worst-case runtime—is known to be very efficient for large practical instances[1]. More information on linear programming can be found in, e.g., [Sch99].

2.1.4 Probability Theory

We briefly recap the concepts of probability theory that are most relevant for this thesis. A more detailed treatment is given in, e.g., , [AD99; Neu10].

measurable space

Definition 2.8 (σ-algebra, Measurable Space) A *measurable space* is a pair $\langle \Omega, \mathcal{F} \rangle$ where

- Ω is a non-empty set of *outcomes* and
- $\mathcal{F} \subseteq 2^\Omega$ is a σ*-algebra* on Ω such that
  - $\Omega \in \mathcal{F}$,
  - $\forall A \subseteq \Omega\colon\ A \in \mathcal{F}$ implies $\Omega \setminus A \in \mathcal{F}$, and
  - $\forall A_1, A_2, \dots \subseteq \Omega\colon\ (\forall i \geq 1\colon\ A_i \in \mathcal{F})$ implies $\bigcup_{i \geq 1} A_i \in \mathcal{F}$.

σ-algebra

Intuitively, a measurable space defines the possible outcomes of a random experiment as well as the sets of outcomes for which we are able to measure, e.g., the probability of. Given a measurable space $\langle \Omega, \mathcal{F} \rangle$, we say that $A \subseteq \Omega$ is a *measurable set* if $A \in \mathcal{F}$. Moreover, for two measurable spaces $\langle \Omega_1, \mathcal{F}_1 \rangle$ and $\langle \Omega_2, \mathcal{F}_2 \rangle$ we call $f\colon \Omega_1 \to \Omega_2$ a *measurable function* if for each measurable set $A_2 \in \mathcal{F}_2$, the set of outcomes (from Ω_1) that are mapped to an element of A_2 is again measurable, i.e., $\{a_1 \in \Omega_1 \mid f(a_1) \in A_2\} \in \mathcal{F}_1$. The product of $n \in \mathbb{N}$ σ-algebras $\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_n$ is given by the σ-algebra

measurable function

$$\mathcal{F}_1 \otimes \mathcal{F}_2 \otimes \dots \otimes \mathcal{F}_n = \bigotimes_{i=1}^{n} \mathcal{F}_i := \left\{ \bigtimes_{i=1}^{n} A_i \mid \forall i \in \{1..n\}\colon\ A_i \in \mathcal{F}_i \right\}.$$

As for the Cartesian product, we may write $\mathcal{F}^n$ instead of $\bigotimes_{i=1}^{n} \mathcal{F}_i$ for $n \in \mathbb{N}$.

Borel σ-algebra

Definition 2.9 (Borel σ-algebra on $\overline{\mathbb{R}}$) The *Borel σ-algebra* on $\overline{\mathbb{R}}$, denoted by $\mathfrak{B}(\overline{\mathbb{R}})$ is the smallest σ-algebra containing all open intervals on $\overline{\mathbb{R}}$, i.e., $\forall a, b \in \overline{\mathbb{R}}$ with $a < b\colon\ (a, b) \in \mathfrak{B}(\overline{\mathbb{R}})$.

Since $\mathfrak{B}(\overline{\mathbb{R}})$ is a σ-algebra, we conclude that also half-open and closed intervals are contained in $\mathfrak{B}(\overline{\mathbb{R}})$. When discussing measurable sets or functions, we usually assume

[1] [ST04] uses smoothed analysis to show that the simplex method usually runs in polynomial time.

the measurable space $\langle \overline{\mathbb{R}}, \mathfrak{B}(\overline{\mathbb{R}}) \rangle$ for $\overline{\mathbb{R}}$. The Borel σ-algebras $\mathfrak{B}(A)$ for measurable subsets $A \subseteq \overline{\mathbb{R}}$ are defined similarly.

Definition 2.10 (Probability Measure, Probability Space) A *probability space* is a triple $\langle \Omega, \mathcal{F}, \Pr \rangle$ where probability space

- $\langle \Omega, \mathcal{F} \rangle$ is a measurable space and
- $\Pr \colon \mathcal{F} \to [0,1]$ is a *probability measure* on $\langle \Omega, \mathcal{F} \rangle$ such that
  - $\Pr(\Omega) = 1$ and
  - $\forall A_1, A_2, \dots \in \mathcal{F} \colon (\forall\, 1 \leq i < j \colon A_i \cap A_j = \emptyset)$ implies $\Pr(\bigcup_{i \geq 1} A_i) = \sum_{i \geq 1} \Pr(A_i)$.

A probability measure assigns a probability to each measurable set. Given two sets $A, B \in \mathcal{F}$ with $\Pr(B) \neq 0$, the *conditional probability* is defined as

$$\Pr(A \mid B) \coloneqq \frac{\Pr(A \cap B)}{\Pr(B)}.$$

Definition 2.11 (Expected Value) Let $\langle \Omega, \mathcal{F}, \Pr \rangle$ be a probability space and let $f \colon \Omega \to \overline{\mathbb{R}}$ be a measurable function such that $\Pr(\{\omega \in \Omega \mid f(\omega) = +\infty\}) = 0$ or $\Pr(\{\omega \in \Omega \mid f(\omega) = -\infty\}) = 0$. The *expected value* of f for $\Pr$ is given by the Lebesque integral expected value

$$\mathrm{Ex}(f) \coloneqq \int_{\omega \in \Omega} f(\omega) \, \mathrm{d}\Pr(\omega).$$

Expected values of functions $f \colon \Omega \to \overline{\mathbb{R}}$ with $\Pr(\{\omega \in \Omega \mid f(\omega) = +\infty\}) > 0$ and $\Pr(\{\omega \in \Omega \mid f(\omega) = -\infty\}) > 0$ are undefined.

Definition 2.12 (Finite Probability Distribution) A *finite probability distribution* (or simply *distribution*) over a finite set Ω is a function $\mu \colon \Omega \to [0,1]$ such that $\sum_{\omega \in \Omega} \mu(\omega) = 1$. The set of all finite probability distributions over Ω is denoted by $\mathit{Dist}(\Omega)$. distribution

Let $\mu \in \mathit{Dist}(\Omega)$ be a distribution over finite Ω. μ can be associated to the probability space $\langle \Omega, 2^\Omega, \Pr \rangle$ which is uniquely given by $\Pr(\{\omega\}) \coloneqq \mu(\omega)$ for all $\omega \in \Omega$. The support of distribution μ is defined as $\mathit{supp}(\mu) \coloneqq \{\omega \in \Omega \mid \mu(\omega) > 0\}$. We sometimes depict a distribution μ by listing the probabilities of all (supported) outcomes, i.e., $\mu = \{\omega_1 \mapsto p_1, \dots, \omega_n \mapsto p_n\}$ where $\mathit{supp}(\mu) \subseteq \{\omega_1, \dots, \omega_n\} \subseteq \Omega$ and $\forall\, i \in \{1..n\} \colon p_i = \mu(\omega_i)$. A *Dirac* distribution μ satisfies $|\mathit{supp}(\mu)| = 1$. We let $\mathit{dirac}^\Omega(\omega) \in \mathit{Dist}(\Omega)$ (or $\mathit{dirac}(\omega)$ if Ω is clear from context) denote the Dirac distribution $\{\omega \mapsto 1\}$ over Ω. Furthermore, $\mathit{unif}(\Omega) \in \mathit{Dist}(\Omega)$ is the *uniform* distribution given by $\forall\, \omega \in \Omega \colon \mathit{unif}(\Omega)(\omega) \coloneqq \frac{1}{|\Omega|}$. Dirac distribution

exponential distribution

Definition 2.13 (Exponential Distribution) An *exponential distribution* with rate $\lambda \in \mathbb{R}_{>0}$ is described by the unique probability measure $\Pr_\lambda$ on $\langle \mathbb{R}_{\geq 0}, \mathfrak{B}(\mathbb{R}_{\geq 0})\rangle$, where for interval $(a, b) \subseteq \mathbb{R}_{\geq 0}$ and Euler's number e we have

$$\Pr_\lambda(\,(a,b)\,) \;=\; \int_a^b \lambda \cdot \mathrm{e}^{-\lambda x}\,\mathrm{d}x \;=\; \mathrm{e}^{-\lambda a} - \mathrm{e}^{-\lambda b}.$$

Exponential distributions are commonly used to model random timings such as the time until a new job arrives at a workstation or the time until a radioactive particle decays. The expected value of $f_c : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with $f_c(x) = c \cdot x$ for $\Pr_\lambda$ as in Definition 2.13 and constant $c \in \mathbb{R}$ is $\mathrm{Ex}(f_c) = \frac{c}{\lambda}$. Furthermore, exponential distributions are *memoryless*: given $t_1, t_2 \in \mathbb{R}_{\geq 0}$, we have

$$\Pr_\lambda(\,(t_1{+}t_2, \infty) \mid (t_1, \infty)\,) \;=\; \Pr_\lambda(\,(t_2, \infty)\,).$$

2.2 Markov Models

We introduce the core modeling formalisms that are studied in this thesis: *Markov decision processes* and *Markov automata*. Since the former is a subclass of the latter, we focus our attention on Markov automata and briefly explain how the introduced concepts carry over to Markov decision processes.

2.2.1 Markov Automata

Markov automata provide an expressive formalism that allows to model exponentially distributed delays, nondeterminism, probabilistic branching, and instantaneous (undelayed) transitions.

MA, Markov automaton

Definition 2.14 (Markov Automaton [EHZ10; DH13]) A *Markov automaton (MA)* is a tuple $\mathcal{M} = \langle S, Act, \Delta, \mathbf{P}\rangle$, where

state

- S is a finite set of *states*,

action

- Act is a finite set of *actions*,

transition function
Markovian state
probabilistic state

- $\Delta : S \to \mathbb{R}_{>0} \uplus (2^{Act} \setminus \{\emptyset\})$ is a *transition function* assigning exit rates to *Markovian states* $MS^{\mathcal{M}} := \{s \in S \mid \Delta(s) \in \mathbb{R}_{>0}\}$ and non-empty sets of enabled actions to *probabilistic states* $PS^{\mathcal{M}} := \{s \in S \mid \emptyset \neq \Delta(s) \subseteq Act\}$, and
- $\mathbf{P} : MS^{\mathcal{M}} \uplus SA^{\mathcal{M}} \to Dist(S)$ with $SA^{\mathcal{M}} := \{\langle s, \alpha\rangle \in PS^{\mathcal{M}} \times Act \mid \alpha \in \Delta(s)\}$ is a *probability function* that assigns a distribution over possible successor states for each Markovian state and for each enabled state-action pair.

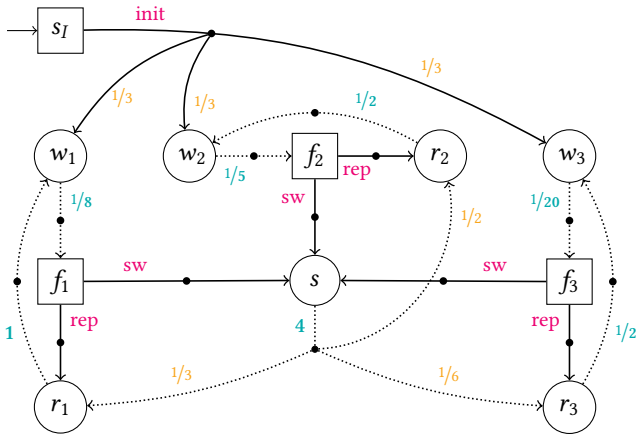


Figure 2.3: Example Markov Automaton $\mathcal{M}$

For the rest of this section we fix an MA $\mathcal{M} = \langle S, Act, \Delta, \mathbf{P} \rangle$. If $\mathcal{M}$ is clear from the context, we may omit the superscript from $MS^{\mathcal{M}}$, $PS^{\mathcal{M}}$, $SA^{\mathcal{M}}$, and similar notations defined below.

Intuitively, the time MA $\mathcal{M}$ stays in a Markovian state $s \in MS$ is governed by an *exponential distribution* with rate $\Delta(s) \in \mathbb{R}_{>0}$, i.e., the probability to take some transition from s within $t \in \mathbb{R}_{\geq 0}$ time units is $\Pr_{\Delta(s)}([0,t)) = 1 - e^{-\Delta(s)\cdot t}$ and the average time to take a transition is $\frac{1}{\Delta(s)}$. Upon taking a transition, a successor state $s' \in S$ is randomly chosen according to the distribution $\mathbf{P}(s)$, i.e., $\mathbf{P}(s)(s')$ is the probability that the transition leads to $s' \in S$. Probabilistic states $\hat{s} \in PS$ are left immediately (without any delay) by nondeterministically choosing an enabled action $\alpha \in \Delta(\hat{s}) \subseteq Act$ and then transitioning to a successor state determined according to distribution $\mathbf{P}(\langle \hat{s}, \alpha \rangle)$. Nondeterminism is thus only possible at probabilistic states. A precise definition of the semantics of an MA is postponed to Section 2.2.4.

exponential distribution

Example 2.8 Consider again the machine with its three different configurations from Example 1.1 on page 2 as well as the corresponding MA model from Figure 1.1 on page 3, also depicted in Figure 2.3. Recall that we use circles and dashed lines to indicate Markovian states and their outgoing transitions whereas for probabilistic states we use rectangles and straight lines. Rates of exponential distributions are given in teal, successor probabilities are given in orange, and controller actions are given in pink. For the sake of completeness, Figure 2.3 explicitly depicts the (unique) action init at state s_I.

Formally, the MA is given by $\mathcal{M} = \langle S, Act, \Delta, \mathbf{P} \rangle$ with

$$S = \underbrace{\{s, w_1, r_1, w_2, r_2, w_3, r_3\}}_{=MS} \cup \underbrace{\{s_I, f_1, f_2, f_3\}}_{=PS}, \quad Act = \{\text{init}, \text{rep}, \text{sw}\},$$

and Δ and $\mathbf{P}$ as in the figure, e.g.,

- $\Delta(s_I) = \{\text{init}\}$, $\Delta(f_1) = \{\text{rep}, \text{sw}\}$, $\Delta(w_1) = 1/8$,
- $\mathbf{P}(\langle s_I, \text{init} \rangle)(w_1) = 1/3$, $\mathbf{P}(w_1)(f_1) = 1$, and $\mathbf{P}(s)(r_3) = 1/6$.

The set of successor states of $c \in MS \cup SA$ is given by

$$post^{\mathcal{M}}(c) := \{s \in S \mid \mathbf{P}(c)(s') > 0\}.$$

Similarly, the set of predecessors of $s \in S$ is given by

$$pre^{\mathcal{M}}(s) := \{c \in MS \cup SA \mid \mathbf{P}(c)(s) > 0\}.$$

To improve readability, we often write $\mathbf{P}(s, s')$ instead of $\mathbf{P}(s)(s')$, $\mathbf{P}(\langle \hat{s}, \alpha \rangle, s')$ or simply $\mathbf{P}(\hat{s}, \alpha, s')$ instead of $\mathbf{P}(\langle \hat{s}, \alpha \rangle)(s')$, and $post(s, \alpha)$ instead of $post(\langle s, \alpha \rangle)$. For $s \in PS$, we set $post(s) := \bigcup_{\alpha \in \Delta(s)} post(s, \alpha)$. A state $s \in S$ is called *absorbing* if $post(s) = \{s\}$. If $\forall\, s \in PS\colon\ |\Delta(s)| = 1$, the enabled action is always unique and can be dropped from the notations. In this case, the MA is called *deterministic*. In our notations, we sometimes use the dedicated symbol ▲ to indicate actions at Markovian states.

absorbing state

deterministic MA

Remark 2.1 (Open and Closed MAs) The literature (e.g., [EHZ10; Tim13; GTHR+14]) often distinguishes between *open* and *closed* MA.
For open MAs, it is assumed that there is some (synchronous or asynchronous) interaction between the MA and other system components. As a consequence, decisions at probabilistic states might be delayed in order to wait, e.g., for another component to synchronize. To enable more flexible modeling, open MAs usually allow states to be both Markovian and probabilistic giving rise to so-called *hybrid* states. This allows for a straightforward notion of compositionality for (open) MAs—a property that does not carry over to the related *continuous time Markov decision processes (CTMDP)* [How60].
On the other hand, closed MA assume that all relevant interactions are already encoded in the model. Typically, a system model arises from the parallel composition of various open MAs which then yields a single closed MA describing the complete system. For closed MAs, *maximal progress* is assumed, meaning that probabilistic (and thus also hybrid) states are left instantaneously—without

any delay [DH13]. As a consequence, exponentially delayed transitions at hybrid states will never be enabled and can thus be dropped entirely, essentially allowing us to convert hybrid states to (pure) probabilistic states.
To summarize, open MAs are useful for system modeling whereas closed MAs simplify model analysis. Since our focus lies on the latter, we refrain from cluttering the notations and immediately introduce MAs in Definition 2.14 as—in the above terms—closed MAs under maximal progress assumptions.

2.2.2 Paths

Definition 2.15 (Path) An *(infinite) path* in MA $\mathcal{M}$ is a sequence $\pi = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \dots$, where for each $i \in \mathbb{N}$ either

- $s_i \in MS$, $\kappa_i \in \mathbb{R}_{\geq 0}$, and $s_{i+1} \in post(s_i)$ or
- $s_i \in PS$, $\kappa_i \in \Delta(s_i) \subseteq Act$, and $s_{i+1} \in post(s_i, \kappa_i)$.

A *finite path* $\hat{\pi} = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \dots \xrightarrow{\kappa_{n-1}} s_n$ is a finite prefix of an infinite path.

path

The sets of infinite and finite paths of $\mathcal{M}$ starting in a state $s \in S$ are given by $Paths^{\mathcal{M}}_{\text{inf}}(s)$ and $Paths^{\mathcal{M}}_{\text{fin}}(s)$, respectively. We set $Paths^{\mathcal{M}}_{\text{inf}} := \cup_{s \in S} Paths^{\mathcal{M}}_{\text{inf}}(s)$ and $Paths^{\mathcal{M}}_{\text{fin}} := \cup_{s \in S} Paths^{\mathcal{M}}_{\text{fin}}(s)$.

We fix a path π with finite prefix $\hat{\pi}$ as in Definition 2.15. Intuitively, if s_i is Markovian, $\kappa_i \in \mathbb{R}_{\geq 0}$ reflects the time we have stayed in s_i until transitioning to s_{i+1}. If s_i is probabilistic, $\kappa_i \in Act$ is the performed action via which we transition to s_{i+1}. For convenience, we define

$$dur(\kappa) := \begin{cases} \kappa & \text{if } \kappa \in \mathbb{R}_{>0} \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \qquad act(\kappa) := \begin{cases} \kappa & \text{if } \kappa \in Act \\ \blacktriangle & \text{otherwise.} \end{cases}$$

We set $last(\hat{\pi}) := s_n$ and $|\hat{\pi}| := n$ for finite $\hat{\pi}$ and $|\pi| := \infty$ for infinite π. For (finite or infinite) $\bar{\pi} \in \{\pi, \hat{\pi}\}$ let $dur(\bar{\pi}) := \sum_{i=0}^{|\bar{\pi}|-1} dur(\kappa_i)$ be the total duration of $\bar{\pi}$. If $\bar{\pi}$ is infinite and $dur(\bar{\pi}) < \infty$, the path is called *Zeno*. We let $pref(\bar{\pi}) = \bigcup_{k=0}^{|\bar{\pi}|} \{pref(\bar{\pi}, k)\}$ be the set of prefixes of $\bar{\pi}$, where $pref(\bar{\pi}, k)$ denotes the prefix length $k \in \mathbb{N}$, i.e.,

Zeno

$$pref(\bar{\pi}, k) := \begin{cases} s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \dots \xrightarrow{\kappa_{k-1}} s_k & \text{if } k \leq |\bar{\pi}| \\ \bar{\pi} & \text{otherwise.} \end{cases}$$

For states $s, s' \in S$ we say that s' is *reachable* from s iff $\exists\, \hat{\pi} \in Paths^{\mathcal{M}}_{\text{fin}}(s) \colon last(\hat{\pi}) = s'$.

reachable state

Example 2.9 The sequence

$$\pi \;=\; s_I \to w_1 \xrightarrow{1} f_1 \xrightarrow{\text{rep}} r_1 \xrightarrow{1} w_1 \xrightarrow{1} f_1 \xrightarrow{\text{rep}} r_1 \xrightarrow{1} w_1 \xrightarrow{1} \ldots$$

is an infinite path in the MA $\mathcal{M}$ from Figure 2.3 and

$$\hat{\pi} = \mathit{pref}(\pi, 4) = s_I \to w_1 \xrightarrow{1} f_1 \xrightarrow{\text{rep}} r_1 \xrightarrow{1} w_1$$

is a finite path in $\mathcal{M}$. In both cases we dropped the action label init for the outgoing transition at s_I from the notation as the action is unique and of no further relevance. We have $\mathit{dur}(\pi) = \infty$, $\mathit{dur}(\hat{\pi}) = 2$, $\mathit{last}(\hat{\pi}) = w_1$, and

$$\mathit{pref}(\hat{\pi}) \;=\; \left\{ s_I,\; s_I \to w_1,\; s_I \to w_1 \xrightarrow{1} f_1,\; s_I \to w_1 \xrightarrow{1} f_1 \xrightarrow{\text{rep}} r_1,\; \hat{\pi} \right\} \subsetneq \mathit{pref}(\pi).$$

The path

$$\pi' \;=\; w_1 \xrightarrow{1} f_1 \xrightarrow{\text{rep}} r_1 \xrightarrow{1/2} w_1 \xrightarrow{1/4} f_1 \xrightarrow{\text{rep}} r_1 \xrightarrow{1/8} w_1 \xrightarrow{1/16} \ldots$$

is Zeno since $\mathit{dur}(\pi') = \sum_{i=0}^{\infty} (1/2)^i = 2 < \infty$.

2.2.3 Strategies

When the control flow of an MA $\mathcal{M}$ reaches a probabilistic state $\hat{s} \in PS$, an enabled action $\alpha \in \Delta(\hat{s})$ needs to be selected. On system level, such a nondeterministic choice can reflect, for instance, the selection of a job that is to be processed or the route that a robot takes through a maze. To resolve the nondeterminism present in $\mathcal{M}$, a *strategy* (also known as *policy*, *scheduler*, or *adversary*) for $\mathcal{M}$ unambiguously specifies which action is chosen in which situation. In general, the choice made by a strategy depends on the execution history observed so far—given by a finite path of the MA—and the outcome of a random experiment—given by a finite probability distribution over actions enabled at the current state.

strategy

Definition 2.16 (Strategy) A (general) *strategy* for MA $\mathcal{M} = \langle S, \mathit{Act}, \Delta, \mathbf{P} \rangle$ is a function $\sigma \colon \mathit{Paths}_{\text{fin}} \to \mathit{Dist}(\mathit{Act} \cup \{\blacktriangle\})$ such that for $\hat{\pi} \in \mathit{Paths}_{\text{fin}}$ we have

$$\sigma(\hat{\pi}) \;\in\; \begin{cases} \mathit{Dist}(\Delta(\mathit{last}(\hat{\pi}))) & \text{if } \mathit{last}(\hat{\pi}) \in PS \\ \mathit{Dist}(\{\blacktriangle\}) & \text{if } \mathit{last}(\hat{\pi}) \in MS. \end{cases}$$

memoryless strategy

A strategy σ is called *memoryless* if the choice only depends on the current state, i.e., $\forall\, \hat{\pi}, \hat{\pi}' \in \mathit{Paths}_{\text{fin}} \colon \mathit{last}(\hat{\pi}) = \mathit{last}(\hat{\pi}')$ implies $\sigma(\hat{\pi}) = \sigma(\hat{\pi}')$. A strategy σ is

called *pure* if all assigned distributions are Dirac, i.e., $\forall\, \hat{\pi} \in \mathit{Paths}_{\text{fin}} \colon\ |\sigma(\hat{\pi})| = 1$. A strategy is *simple* if it is memoryless and pure. Let $\Sigma^{\mathcal{M}}$, $\Sigma^{\mathcal{M}}_{\text{P}}$, and $\Sigma^{\mathcal{M}}_{\text{PM}}$ denote the set of general, pure, and pure memoryless strategies of $\mathcal{M}$, respectively. For simplicity, we often interpret $\sigma \in \Sigma^{\mathcal{M}}_{\text{PM}}$ as a function $\sigma \colon S \to \mathit{Act} \cup \{\blacktriangle\}$. A finite or infinite path $\bar{\pi} = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \dots$ is *σ-consistent* if $\forall\, \hat{\pi} \in \mathit{pref}(\bar{\pi}) \colon\ \mathit{act}(\kappa_{|\hat{\pi}|}) \in \mathit{supp}(\sigma(\hat{\pi}))$.

pure strategy

simple strategy

consistent path

Example 2.10 The strategy σ for $\mathcal{M}$ as in Figure 2.3 that always chooses action rep at states f_i, $i \in \{1,2,3\}$—that is $\sigma(\hat{\pi}) = \mathit{dirac}(\text{rep})$ for all $\hat{\pi} \in \mathit{Paths}_{\text{fin}}$ with $\mathit{last}(\hat{\pi}) \in \{f_1, f_2, f_3\}$—is both pure and memoryless, i.e., $\sigma \in \Sigma^{\mathcal{M}}_{\text{PM}}$.
A strategy $\sigma' \in \Sigma^{\mathcal{M}}$ with $\sigma(w_1 \xrightarrow{t} f_1) = \{\text{rep} \mapsto \mathrm{e}^{-t},\ \text{sw} \mapsto 1 - \mathrm{e}^{-t}\}$ is neither pure nor memoryless as the decision at f_1 depends on the time we spent in state w_i and is subject to a non-Dirac probability distribution.

Remark 2.2 (Generalized Strategy Notions) We refer to strategies from Definition 2.16 as *general* strategies to make a clear distinction to memoryless or pure strategies. However, the literature also uses the term *behavioral strategy* for strategies as defined above and considers more general notions, e.g., , strategies as distributions over behavioral strategies [Aum64; MR22].

2.2.4 Probability Spaces

We define a probability space induced by an MA $\mathcal{M} = \langle S, \mathit{Act}, \Delta, \mathbf{P}\rangle$, an initial state $s_I \in S$ and a strategy $\sigma \in \Sigma^{\mathcal{M}}$. This allows us to measure the probability of sets of infinite paths of $\mathcal{M}$. Following standard constructions [AD99; Neu10], we proceed in three steps:

- define a probability space for single transition steps,
- lift this to a probability space for paths of a fixed length $n \in \mathbb{N}$, and
- construct a probability space for infinite paths.

To simplify these definitions, we sometimes consider a less restrictive notion of paths compared to Definition 2.15.

Definition 2.17 (Permissive Path) An infinite *permissive* path in $\mathcal{M}$ is a sequence $s_0\kappa_0 s_1\kappa_1 \dots$ with $s_i \in S$ and $\kappa_i \in \mathit{Act} \cup \mathbb{R}_{\geq 0}$ for all $i \in \mathbb{N}$. A finite permissive path is a prefix $s_0\kappa_0 \dots \kappa_{n-1}s_n$ of an infinite permissive path.

permissive path

We identify each non-permissive (finite or infinite) path $s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \dots$ as in Definition 2.15 with the corresponding permissive path $s_0\kappa_0 s_1\kappa_0 \dots$. However, note

that not every permissive path adheres to the restrictions imposed in Definition 2.15. Without loss of generality, we assume that $Act \cap \mathbb{R}_{\geq 0} = \emptyset$.

Let $\hat{\pi} \in Paths_{\text{fin}}$ be a non-permissive finite path. We consider the probability space $\langle \Omega^{Step}, \mathcal{F}^{Step}, \Pr^{Step}_{\sigma,\hat{\pi}} \rangle$ for single (Markovian or probabilistic) transition steps assuming history $\hat{\pi}$, where

$$\Omega^{Step} \;\coloneqq\; (Act \uplus \mathbb{R}_{\geq 0}) \times S, \quad \mathcal{F}^{Step} \;\coloneqq\; \{K \subseteq Act \uplus \mathbb{R}_{\geq 0} \mid K \cap \mathbb{R}_{\geq 0} \in \mathfrak{B}(\mathbb{R}_{\geq 0})\} \otimes 2^S,$$

and for $T \in \mathcal{F}^{Step}$, $last(\hat{\pi}) = s$ we have

$$\Pr^{Step}_{\sigma,\hat{\pi}}(T) \;\coloneqq\; \begin{cases} \sum_{\alpha \in \Delta(s)} \sigma(\hat{\pi})(\alpha) \cdot \sum_{\langle \alpha, s' \rangle \in T} \mathbf{P}(s, \alpha, s') & \text{if } s \in PS \\ \int_{t \in \mathbb{R}_{\geq 0}} \Delta(s) \cdot \mathrm{e}^{-\Delta(s) \cdot t} \cdot \sum_{\langle t, s' \rangle \in T} \mathbf{P}(s, s')\, \mathrm{d}t & \text{if } s \in MS. \end{cases}$$

Example 2.11 Consider $\hat{\pi} = s_I \longrightarrow w_1$ for the MA $\mathcal{M}$ from Figure 2.3. The probability to take a transition to f_1 within 2 time units is

$$\begin{aligned} \Pr^{Step}_{\sigma,\hat{\pi}}([0,2] \times \{f_1\}) \;&=\; \int_{t \in \mathbb{R}_{\geq 0}} \Delta(w_1) \cdot \mathrm{e}^{-\Delta(w_1) \cdot t} \cdot [0 \leq t \leq 2] \cdot \mathbf{P}(w_1, f_1)\, \mathrm{d}t \\ &=\; \int_0^2 {}^1\!/_8 \cdot \mathrm{e}^{-1/8 \cdot t} \cdot \mathrm{d}t \;=\; 1 - \mathrm{e}^{-2/8} \;\approx\; 0.221. \end{aligned}$$

Observe that this value does not depend on the considered strategy $\sigma \in \Sigma^{\mathcal{M}}$. For $\hat{\pi}' = s_I \longrightarrow w_1 \xrightarrow{1} f_1$ we get

$$\Pr^{Step}_{\sigma,\hat{\pi}'}(\{\mathrm{sw}, s\}) \;=\; \sigma(\hat{\pi}')(\mathrm{rep}) \cdot 0 + \sigma(\hat{\pi}')(\mathrm{sw}) \cdot \mathbf{P}(f_1, \mathrm{sw}, s) \;=\; \sigma(\hat{\pi}')(\mathrm{sw}).$$

Next, we define the probability space $\langle \Omega^n, \mathcal{F}^n, \Pr^n_{\sigma,s_I} \rangle$ for sequences of exactly $n \in \mathbb{N}$ transition steps, i.e., permissive paths of length n, where

$$\Omega^n \;\coloneqq\; S \times (\Omega^{Step})^n, \quad \mathcal{F}^n \;\coloneqq\; 2^S \otimes (\mathcal{F}^{Step})^n,$$

and $\Pr^n_{\sigma,s_I}$ is defined recursively such that for $S' \in \mathcal{F}^0 = 2^S$ we have $\Pr^0_{\sigma,s_I}(S') \coloneqq [s_I \in S']$, and for $n > 0$, $\Pi \times T \in \mathcal{F}^n$ such that $\Pi \in \mathcal{F}^{n-1}$ and $T \in \mathcal{F}^{Step}$ we have

$$\Pr^n_{\sigma,s_I}(\Pi \times T) \;\coloneqq\; \int_{\hat{\pi} \in \Pi \cap Paths_{\text{fin}}} \Pr^{Step}_{\sigma,\hat{\pi}}(T)\, \mathrm{d}\Pr^{n-1}_{\sigma,s_I}(\hat{\pi}).$$

Recall that paths $s_0 \xrightarrow{\kappa_0} \ldots \xrightarrow{\kappa_{n-1}} s_n \in Paths_{\text{fin}}$ are identified with permissive paths $s_0 \kappa_0 \ldots \kappa_{n-1} s_n \in \Omega^n$. In particular, $\Omega^n \cap Paths_{\text{fin}} = Paths_{\text{fin}}$.

Finally, the probability space $\langle \Omega^{\mathcal{M}}, \mathcal{F}^{\mathcal{M}}, \Pr^{\mathcal{M}}_{\sigma,s_I} \rangle$ for infinite paths of $\mathcal{M}$ is defined

where $\Omega^{\mathcal{M}}$ is the set of infinite permissive paths and $\mathcal{F}^{\mathcal{M}}$ is the smallest σ-algebra on $\Omega^{\mathcal{M}}$ containing all *cylinder sets* of all measurable $\Pi \in \bigcup_{n=0}^{\infty} \mathcal{F}^n$, where for $n \in \mathbb{N}$ the cylinder set of $\Pi \in \mathcal{F}^n$ is given by

cylinder set

$$Cyl(\Pi) \coloneqq \{s_0\kappa_0 s_1\kappa_1 \cdots \in \Omega^{\mathcal{M}} \mid s_0\kappa_0 s_1\kappa_1 \ldots \kappa_{n-1}s_n \in \Pi\}.$$

$\Pr^{\mathcal{M}}_{\sigma,s_I}$ is the unique probability measure on $\langle \Omega^{\mathcal{M}}, \mathcal{F}^{\mathcal{M}} \rangle$ satisfying

$$\Pr^{\mathcal{M}}_{\sigma,s_I}(\, Cyl(\Pi)\,) \;=\; \Pr^{n}_{\sigma,s_I}(\Pi)$$

for all $\Pi \in \mathcal{F}^n$, $n \in \mathbb{N}$. The set of non-permissive infinite paths $Paths_{\text{inf}}$ is measurable, i.e., $Paths_{\text{inf}} \in \mathcal{F}^{\mathcal{M}}$ and we have $\Pr^{\mathcal{M}}_{\sigma,s_I}(Paths_{\text{inf}}) = 1$. It is therefore reasonable, to restrict our attention to non-permissive paths $\pi \in Paths_{\text{inf}}$ as given in Definition 2.15.

The *expected value* of some measurable function $\varphi\colon Paths_{\text{inf}} \to \overline{\mathbb{R}}$ for $\Pr^{\mathcal{M}}_{\sigma,s_I}$ is given by

expected value

$$\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\varphi) \coloneqq \int_{\pi \in Paths_{\text{inf}}} \varphi(\pi)\, \mathrm{d}\Pr^{\mathcal{M}}_{\sigma,s_I}(\pi).$$

Recall from Section 2.1.4 that the expected value is undefined if both

$$\Pr^{\mathcal{M}}_{\sigma,s_I}(\{\pi \mid \varphi(\pi) = -\infty\}) > 0 \quad \text{and} \quad \Pr^{\mathcal{M}}_{\sigma,s_I}(\{\pi \mid \varphi(\pi) = +\infty\}) > 0$$

hold. We sometimes drop the MA $\mathcal{M}$ or the initial state s_I from the notations $\Pr^{\mathcal{M}}_{\sigma,s_I}$ and $\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}$ if they are clear from the context. Furthermore, we also drop the (unique) strategy σ if $\mathcal{M}$ is deterministic, i.e., $\forall\, s \in PS\colon\ |\Delta(s)| = 1$. For measurable Π we use the abbreviations $\Pr^{\mathcal{M}}_{\max,s_I}(\Pi) \coloneqq \sup_{\sigma \in \Sigma^{\mathcal{M}}} \Pr^{\mathcal{M}}_{\sigma,s_I}(\Pi)$ and $\Pr^{\mathcal{M}}_{\min,s_I}(\Pi) \coloneqq \inf_{\sigma \in \Sigma^{\mathcal{M}}} \Pr^{\mathcal{M}}_{\sigma,s_I}(\Pi)$. The notations $\mathrm{Ex}^{\mathcal{M}}_{\max,s_I}$ and $\mathrm{Ex}^{\mathcal{M}}_{\min,s_I}$ are similar.

2.2.5 Reward Assignments

MAs can be equipped with *rewards* or, equivalently, *costs* to model various quantities like, e.g., energy consumption or the number of produced units. We distinguish between *transition* rewards that are collected when transitioning from one state to another and *state* rewards that are collected over time while residing in a state. A reward assignment combines the two notions.

reward

Definition 2.18 (Reward Assignment) A *reward assignment* for MA $\mathcal{M}$ is a pair $\mathcal{R} = \langle \mathcal{R}^{\text{trans}}, \mathcal{R}^{\text{state}} \rangle$, where

reward assignment

- $\mathcal{R}^{\text{trans}}\colon (MS \cup SA) \times S \to \mathbb{R}$ is a transition reward assignment, and
- $\mathcal{R}^{\text{state}}\colon S \to \mathbb{R}$ is a state reward assignment.

For a reward assignment $\mathcal{R} = \langle \mathcal{R}^{\text{trans}}, \mathcal{R}^{\text{state}} \rangle$, $s, s' \in S$, and $\kappa \in Act \cup \mathbb{R}_{\geq 0}$ we set $\mathcal{R}[s] := \mathcal{R}^{\text{state}}(s)$ and

$$\mathcal{R}[s, \kappa, s'] := \begin{cases} \kappa \cdot \mathcal{R}^{\text{state}}(s) + \mathcal{R}^{\text{trans}}(s, s') & \text{if } s \in MS \text{ and } \kappa \in \mathbb{R}_{\geq 0} \\ \mathcal{R}^{\text{trans}}(\langle s, \kappa \rangle, s') & \text{if } \langle s, \kappa \rangle \in SA \\ 0 & \text{otherwise} \end{cases}$$

Since no time passes in probabilistic states, the assigned state rewards $\mathcal{R}^{\text{state}}(\hat{s})$ for $\hat{s} \in PS$ are not relevant. We lift the notation to finite paths: for $\hat{\pi} = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \ldots \xrightarrow{\kappa_{n-1}} s_n \in Paths_{\text{fin}}$ we define the *accumulated reward* for $\mathcal{R}$ along $\hat{\pi}$ as

$$\mathcal{R}[\hat{\pi}] := \sum_{i=0}^{n-1} \mathcal{R}[s_i, \kappa_i, s_{i+1}].$$

We stress that an MA can be annotated with several different reward assignments.

Example 2.12 We annotate the states and transitions of the MA $\mathcal{M}$ from Figure 2.3 with three different reward assignments $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$. State- and transition rewards that are not depicted are assumed to be zero. For example, at state w_1 we have state rewards $\mathcal{R}_1[w_1] = 100$ and $\mathcal{R}_2[w_1] = \mathcal{R}_3[w_1] = 0$. When moving from f_1 to s via action sw we collect $\mathcal{R}_1[f_1, \text{sw}, s] = \mathcal{R}_2[f_1, \text{sw}, s] = 0$ and $\mathcal{R}_3[f_1, \text{sw}, s] = 1$ reward. For the path

$$\hat{\pi} = s_I \to w_1 \xrightarrow{3} f_1 \xrightarrow{\text{rep}} r_1 \xrightarrow{1} w_1 \xrightarrow{2} f_1 \xrightarrow{\text{sw}} s \xrightarrow{1/2} r_2$$

the accumulated rewards for $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$ along $\hat{\pi}$ are $\mathcal{R}_1[\hat{\pi}] = 3 \cdot 100 + 2 \cdot 100 = 500$, $\mathcal{R}_2[\hat{\pi}] = 1 \cdot (-1) = -1$, and $\mathcal{R}_3[\hat{\pi}] = 1$, respectively.
Recall from Example 2.8 that the depicted MA models a machine with three different configurations $i \in \{1, 2, 3\}$. The assigned rewards are interpreted as follows.

- $\mathcal{R}_1$ models the productivity of the machine: when in state w_i, the machine produces $\mathcal{R}_1[w_i]$ units per time. The three machine configurations thus do not only differ in their failure- and repair rates, but also in their productivity.
- $\mathcal{R}_2$ yields the repair costs that is accumulated while in a repairing state r_i. We model this cost as a negative reward.
- $\mathcal{R}_3$ reflects the number of times the machine controller switches to the state s in order to change the configuration.

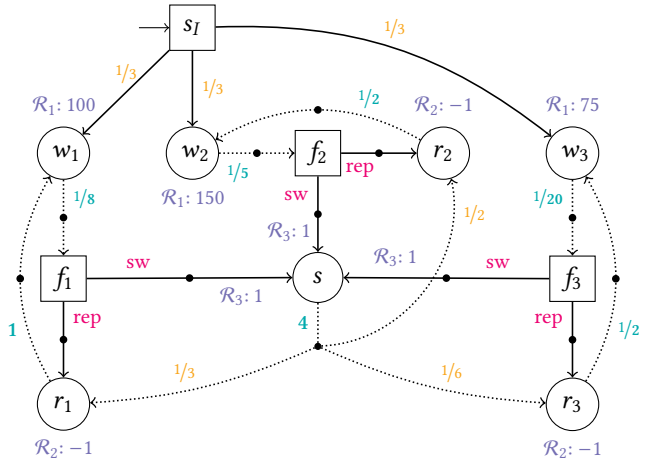


Figure 2.4: MA $\mathcal{M}$ from Figure 2.3 with three reward assignments $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$ (cf. Example 2.12)

For constant $a \in \mathbb{R}$ and reward assignments $\mathcal{R}$ and $\mathcal{R}'$ we define scaling $a \cdot \mathcal{R}$ and addition $\mathcal{R} + \mathcal{R}'$ by carrying out the respective operation to the declared reward values. Furthermore, we introduce the notations $-\mathcal{R}$, $\mathcal{R}^+$, and $\mathcal{R}^-$ which consider the negation of $\mathcal{R}$, the non-negative part of $\mathcal{R}$, and the non-positive part of $\mathcal{R}$, respectively. For the reward at state $s \in S$ we thus have

$$(a \cdot \mathcal{R})[s] = a \cdot \mathcal{R}[s], \qquad (\mathcal{R} + \mathcal{R}')[s] = \mathcal{R}[s] + \mathcal{R}'[s], \qquad (-\mathcal{R})[s] = -\mathcal{R}[s],$$
$$(\mathcal{R}^+)[s] = \max(\mathcal{R}[s], 0), \quad \text{and} \quad (\mathcal{R}^-)[s] = \min(\mathcal{R}[s], 0).$$

Transition-based rewards are defined similarly. Furthermore, for a relation $\sim \in \{<, \leq, =, \geq, >\}$ we use the shortcut $\mathcal{R} \sim 0$ to denote that all rewards assigned by $\mathcal{R}$ are ~ 0, i.e.,

$$\mathcal{R} \sim 0 \quad \text{iff} \quad \forall \langle s, \kappa \rangle \in SA \cup (MS \times \mathbb{R}_{\geq 0}) \colon \forall s' \in S \colon \mathcal{R}[s, \kappa, s'] \sim 0.$$

2.2.6 Components

A *component* of $\mathcal{M}$ is a non-empty set $C \subseteq MS \cup SA$. We denote the state of a component element $c \in MS \cup SA$ by $state(c)$, where $state(c) := c$ if $c \in MS$ and $state(c) := s$ if $c = \langle s, \alpha \rangle \in SA$. The set of states occurring in a component C is given by $states(C) := \{state(c) \mid c \in C\}$. Moreover, we let $exits(C) := \{c \in MS \cup SA \mid state(c) \in states(C) \text{ and } post(c) \not\subseteq C\}$. A component C is *closed* if

component

closed component

bottom component

$\forall c \in C\colon supp(\mathbf{P}(c)) \subseteq states(C)$. C is a *bottom component* if it is closed and $exits(C) = \emptyset$. We sometimes overload notations for components by using sets of states $B \subseteq S$ as alias for the component $\{c \in MS \cup SA \mid state(c) \in B\}$.

SCC, strongly connected component

A *strongly connected component (SCC)* is a component C such that for all pairs of states $s, s' \in states(C)$ there is a path $\hat{\pi} = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \dots \xrightarrow{\kappa_{n-1}} s_n \in Paths_{\text{fin}}(s)$ with $last(\hat{\pi}) = s'$ and for each $i \in \{0..(n-1)\}$ either $s_i \in C \cap MS$ or $\langle s_i, \kappa_i\rangle \in C \cap SA$. An SCC C is *trivial* if $states(C)$ is a singleton $\{s\}$ and $\forall c \in C\colon s \notin supp(\mathbf{P}(c))$. An SCC is *maximal* if it is not a proper subset of another SCC. Each state belongs to exactly one maximal SCC. The set of all maximal SCCs of $\mathcal{M}$ is denoted by $SCC(\mathcal{M})$, while $BSCC(\mathcal{M})$ denotes the maximal *bottom* SCCs (BSCCs) of $\mathcal{M}$. $SCC(\mathcal{M})$ and $BSCC(\mathcal{M})$ can be computed efficiently [Tar72].

BSCC, bottom strongly connected component

EC, end component

Definition 2.19 (End Component) An *end component (EC)* is a strongly connected and closed component.

MEC, maximal end component

A *maximal* end component (MEC) is an EC that is not a proper subset of another EC. The sets of all ECs and all MECs of $\mathcal{M}$ is denoted by $EC(\mathcal{M})$ and $MEC(\mathcal{M})$, respectively. Finite MAs (as in Definition 2.14) must have at least one EC. Since ECs may overlap, the number of ECs of $\mathcal{M}$ can be exponential in $|MS|+|SA|$. MECs, however, are always disjoint, i.e., each $c \in MS \cup SA$ belongs to *at most* one MEC $c \in C \in MEC(\mathcal{M})$. The set $MEC(\mathcal{M})$ can be computed efficiently [CH14].

Example 2.13 Consider the components $C_i = \{w_i, \langle f_i, \text{rep}\rangle, r_i\}$ of the MA from Figure 2.3 for $i \in \{1,2,3\}$. We have $states(C_i) = \{w_i, f_i, r_i\}$ and $exits(C_i) = \{\langle f_i, \text{sw}\rangle\}$. C_i is closed, strongly connected, but not bottom. Furthermore, the components $C_i \cup \{\langle f_i, \text{sw}\rangle\}$ and $C_i \cup \{\langle f_i, \text{sw}\rangle, s\}$ are both strongly connected but not closed. The component $C_i \setminus \{r_i\}$ is not strongly connected.
There are two maximal SCCs for $\mathcal{M}$ given by the trivial SCC $\{\langle s_I, \alpha\rangle\}$ (assuming that α is the action performed at s_I) and the BSCC

$$\{s\} \cup \bigcup_{i\in\{1,2,3\}} C_i \cup \{\langle f_i, \text{sw}\rangle\}.$$

The latter is also the (only) *maximal* EC of $\mathcal{M}$. The other—not maximal—ECs of $\mathcal{M}$ are given by C_1, C_2, and C_3.

ECs characterize the states and transitions of $\mathcal{M}$ that—depending on the strategy—can occur infinitely often with non-zero probability. Formally, for $\pi = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \dots \in Paths_{\text{inf}}^{\mathcal{M}}$ let $inf(\pi)$ denote the set of Markovian states $s \in MS$ and state-action pairs $\langle s, \alpha\rangle \in SA$ that occur infinitely often in π, i.e.,

$$inf(\pi) \coloneqq \Big\{c \in MS \cup SA \;\Big|\; \forall j \geq 0\colon \exists i \geq j\colon \ (c = s_i \in MS) \text{ or } (c = \langle s_i, \kappa_i\rangle \in SA)\Big\}.$$

The following lemma states that $inf(\pi)$ is almost surely an EC. Hence, the probability to infinitely often observe some $c \in MS \cup SA$ with $c \notin \bigcup_{EC(\mathcal{M})}$ under an arbitrary strategy must be zero.

Lemma 2.4 For all $s \in S$ and $\sigma \in \Sigma^{\mathcal{M}}$ we have

$$\Pr^{\mathcal{M}}_{\sigma,s}\big(\{\pi \in Paths^{\mathcal{M}}_{\inf}(s) \mid inf(\pi) \in EC(\mathcal{M})\}\big) \;=\; 1.$$

Proof. The lemma is a straightforward generalization of [BK08, Theorem 10.120] to MA. ∎

Submodels and Quotients | ECs provide an important concept for the analysis of MAs and the subsumed Markov decision processes (MDPs, cf. Section 2.2.8). Approaches such as e.g., [Alf97; BCCF+14; GHHK+14; ACDK+17; BBDG+18; HM18] often inspect the (maximal) ECs in isolation and/or consider a quotient model in which ECs are collapsed. We now provide the required building blocks for such methods.

Definition 2.20 (Submodel) The *submodel* of $\mathcal{M}$ induced by a closed component C is given by $\mathcal{M}[\![C]\!] = \langle states(C), Act, \Delta_C, \mathbf{P}|_C\rangle$, where for $s \in states(C)$ submodel

$$\Delta_C(s) \;:=\; \begin{cases} \Delta(s) & \text{if } s \in C \cap MS^{\mathcal{M}} \\ \{\alpha \in \Delta(s) \mid \langle s, \alpha\rangle \in C\} & \text{otherwise.} \end{cases}$$

Submodels restrict the MA to the states and transitions considered in the inducing component C. As C is required to be closed, we obtain a valid MA $\mathcal{M}[\![C]\!]$ in which no states $s \notin states(C)$ can be reached.

Example 2.14 The submodel of $\mathcal{M}$ as in Figure 2.3 induced by component $C_1 = \{w_1, \langle f_1, \text{rep}\rangle, r_1\}$ is shown in Figure 2.5a.

We also consider submodels to restrict the possible actions of an MA. In particular, the *induced submodel* for $\mathcal{M}$ and some pure memoryless strategy $\sigma \in \Sigma^{\mathcal{M}}_{\text{PM}}$ is given by $\mathcal{M}[\![\sigma]\!] := \mathcal{M}[\![\, MS \cup \{\langle s, \sigma(s)\rangle \mid s \in PS\}\,]\!]$. induced submodel

Next, we define the quotient of an MA $\mathcal{M}$ with respect to a given EC C. Essentially, this construction collapses all states in $states(C)$ into a single, freshly introduced state. Transitions that—in $\mathcal{M}$—lead to or emerge from the EC are redirected to the new state. Intuitively, the quotient thus abstracts away the internals of the EC and only depicts the incoming- and outgoing behavior. To represent the possibility of never exiting the EC we also add a transition to a dedicated sink state $s_\perp$.

quotient

Definition 2.21 (Quotient) The *quotient* of $\mathcal{M}$ with respect to an EC C is the MA $\mathcal{M}_{\setminus C} = \langle S_{\setminus C}, Act_{\setminus C}, \Delta_{\setminus C}, \mathbf{P}_{\setminus C}\rangle$, where

- $S_{\setminus C} = \big(S \setminus states(C)\big) \uplus \{C, s_\bot\}$,
- $Act_{\setminus C} = Act \uplus exits(C) \uplus \{\bot\}$,
- $\forall\, s \in S_{\setminus C}\colon\ \Delta_{\setminus C}(s) = \begin{cases} \Delta(s) & \text{if } s \in S \\ exits(C) \cup \{\bot\} & \text{if } s = C \\ 1 & \text{if } s = s_\bot, \end{cases}$
- and $\mathbf{P}_{\setminus C}$ is defined as follows: let $\mathbf{P}_{\text{aux}}\colon MS^{\mathcal{M}} \cup SA^{\mathcal{M}} \to Dist(S_{\setminus C})$ be given for $c \in MS^{\mathcal{M}} \cup SA^{\mathcal{M}}$ by
  - $\mathbf{P}_{\text{aux}}(c)(C) = \sum_{s' \in states(C)} \mathbf{P}(c)(s')$,
  - $\mathbf{P}_{\text{aux}}(c)(s') = \mathbf{P}(c)(s')$ with $s' \in S \setminus states(C)$, and
  - $\mathbf{P}_{\text{aux}}(c)(s_\bot) = 0$.

  Then, we set
  - $\mathbf{P}_{\setminus C}(c) = \mathbf{P}_{\text{aux}}(c)$ for $c \in (MS^{\mathcal{M}_{\setminus C}} \cap MS^{\mathcal{M}}) \cup (SA^{\mathcal{M}_{\setminus C}} \cap SA^{\mathcal{M}})$,
  - $\mathbf{P}_{\setminus C}(\langle C, \langle s, \alpha\rangle\rangle) = \mathbf{P}_{\text{aux}}(\langle s, \alpha\rangle)$ for $\langle s, \alpha\rangle \in exits(C)$, and
  - $\mathbf{P}_{\setminus C}(\langle C, \bot\rangle) = \mathbf{P}_{\setminus C}(\langle s_\bot, \bot\rangle) = \{s_\bot \mapsto 1\}$.

For a set of ECs $\mathfrak{C} = \{C_1, \ldots, C_n\}$ with $\forall\, i, j \in \{1..n\}\colon\ i \neq j \implies states(C_i) \cap states(C_j) = \emptyset$ (i.e., the ECs in $\mathfrak{C}$ are pairwise disjoint) we also consider the quotient that arises after collapsing *all* ECs in $\mathfrak{C}$:

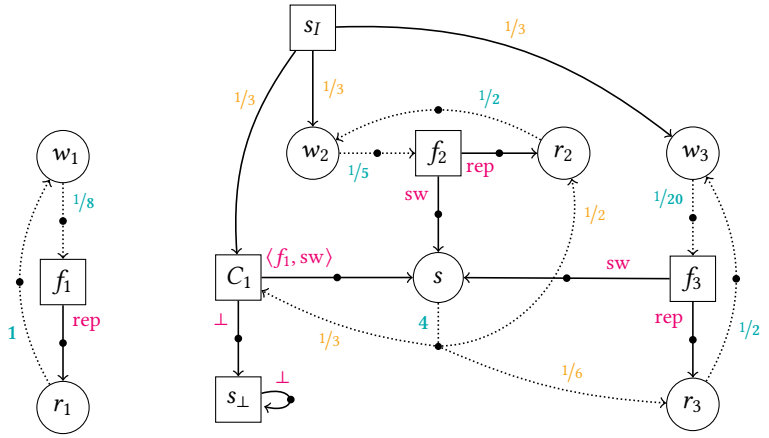
$$\mathcal{M}_{\setminus \mathfrak{C}} \;\coloneqq\; \mathcal{M}_{\setminus C_1 \setminus \cdots \setminus C_n}$$

Considering $\mathcal{M}_{\setminus MEC(\mathcal{M})}$ is always valid since the maximal ECs of an MA are always disjoint. To simplify the notation, we write $\mathcal{M}_{\setminus MEC}$ instead of $\mathcal{M}_{\setminus MEC(\mathcal{M})}$.

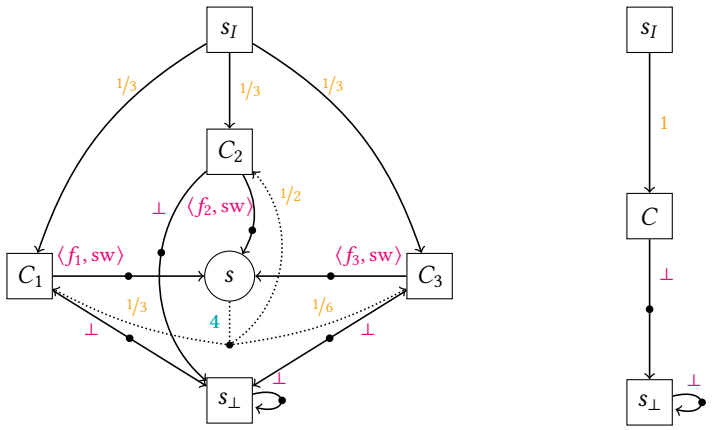
Example 2.15 Consider the MA $\mathcal{M}$ from Figure 2.3 and ECs $C_i = \{w_1, \langle f_1, \text{rep}\rangle, r_1\}$ as in Example 2.13. Let $\mathfrak{C} = \{C_1, C_2, C_3\}$. Figures 2.5b to 2.5d show the quotients $\mathcal{M}_{\setminus C_1}$, $\mathcal{M}_{\setminus \mathfrak{C}}$, and $\mathcal{M}_{\setminus MEC(\mathcal{M})}$, respectively.

Components with Rewards | Let $C \subseteq MS \cup SA$ be a component of $\mathcal{M}$. The sub-assignment $\mathcal{R}[\![C]\!]$ of reward assignment $\mathcal{R} = \langle \mathcal{R}^{\text{trans}}, \mathcal{R}^{\text{state}}\rangle$ w.r.t. C restricts the assigned rewards to the elements of the component, i.e.,

$$\mathcal{R}[\![C]\!] \coloneqq \Big\langle \mathcal{R}^{\text{trans}}\big|_{C \times states(C)},\ \mathcal{R}^{\text{state}}\big|_{states(C)} \Big\rangle.$$



(a) $\mathcal{M}[\![C_1]\!]$ (b) $\mathcal{M}_{\setminus C_1}$



(c) $\mathcal{M}_{\setminus \mathfrak{C}} = \mathcal{M}_{\setminus C_1 \setminus C_2 \setminus C_3}$ (d) $\mathcal{M}_{\setminus MEC(\mathcal{M})}$

Figure 2.5: Submodels and quotients of the MA $\mathcal{M}$ from Figure 2.3 (cf. Examples 2.14 and 2.15)

If C is closed, $\mathcal{R}[\![C]\!]$ is a reward assignment for the submodel $\mathcal{M}[\![C]\!]$ of $\mathcal{M}$.

Similarly, if C is an EC, $\mathcal{R}$ can be transformed to a reward assignment $\mathcal{R}_{\setminus C} \coloneqq \langle \mathcal{R}^{\text{trans}}_{\setminus C}, \mathcal{R}^{\text{state}}_{\setminus C} \rangle$ for the quotient model $\mathcal{M}_{\setminus C} = \langle S_{\setminus C}, Act_{\setminus C}, \Delta_{\setminus C}, \mathbf{P}_{\setminus C} \rangle$. Here, we drop all the rewards that can potentially be collected while staying inside C. More precisely, let $\mathcal{R}^{\text{trans}}_{\text{aux}} \colon (MS^{\mathcal{M}} \cup SA^{\mathcal{M}}) \times S_{\setminus C} \to \mathbb{R}$ be given for $c \in (MS^{\mathcal{M}} \cup SA^{\mathcal{M}})$ and $s' \in S_{\setminus C}$ by

$$\mathcal{R}^{\text{trans}}_{\text{aux}}(c, s') \coloneqq \begin{cases} \mathcal{R}^{\text{trans}}(c, s') & \text{if } s' \in S \\ \sum_{s'' \in states(C)} \mathbf{P}(c, s'') \cdot \mathcal{R}^{\text{trans}}(c, s'') & \text{if } s' = C \\ 0 & \text{otherwise.} \end{cases}$$

We then set for $c \in (MS^{\mathcal{M}_{\setminus C}} \cup SA^{\mathcal{M}_{\setminus C}})$ and $s' \in S_{\setminus C}$

$$\mathcal{R}^{\text{trans}}_{\setminus C}(c, s') \coloneqq \begin{cases} \mathcal{R}^{\text{trans}}_{\text{aux}}(c, s') & \text{if } c \in (MS^{\mathcal{M}} \cup SA^{\mathcal{M}}) \text{ and } s' \in S \\ \mathcal{R}^{\text{trans}}_{\text{aux}}(\langle s, \alpha \rangle, s') & \text{if } c = \langle C, \langle s, \alpha \rangle \rangle \text{ for } \langle s, \alpha \rangle \in exits(C) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{and} \quad \mathcal{R}^{\text{state}}_{\setminus C}(s') \coloneqq \begin{cases} \mathcal{R}^{\text{state}}(s') & \text{if } s' \in S \\ 0 & \text{otherwise.} \end{cases}$$

As before, we lift the notation to sets of pairwise disjoint ECs $\mathfrak{C} = \{C_1, \ldots, C_n\}$, where

$$\mathcal{R}_{\setminus \mathfrak{C}} \coloneqq \mathcal{R}_{\setminus C_1 \setminus \cdots \setminus C_n}.$$

To avoid notational clutter, we usually apply the reward assignment $\mathcal{R}$ for $\mathcal{M}$ directly to the MAs $\mathcal{M}[\![C]\!]$ and $\mathcal{M}_{\setminus C}$—assuming an implicit conversion to $\mathcal{R}[\![C]\!]$ and $\mathcal{R}_{\setminus C}$, respectively.

As in, e.g., [BBDG+18], we also consider end components in which no non-zero reward can possibly be collected.

0-end component

Definition 2.22 (0-End Component) A 0-EC of $\mathcal{M}$ and $\ell \in \mathbb{N}$ reward assignments $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$ is an EC C of $\mathcal{M}$ with $\forall\, j \in \{1..\ell\} \colon \mathcal{R}_j[\![C]\!] = 0$.

A *maximal* 0-EC (0-MEC) is a 0-EC that is no proper subset of another 0-EC. The set of all 0-MECs of $\mathcal{M}$ and $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$ is denoted by $MEC_0(\mathcal{M}, \{\mathcal{R}_1, \ldots, \mathcal{R}_\ell\})$ and can be computed efficiently, as outlined in Algorithm 2.1.

Input: MA $\mathcal{M}$, reward assignments $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$
Output: $MEC_0(\mathcal{M}, \{\mathcal{R}_1, \ldots, \mathcal{R}_\ell\})$

// Only consider parts of $\mathcal{M}$ that yield zero reward
1 $C_0 \leftarrow \{c \in (MS \cup SA) \mid \forall\, j \in \{1..\ell\} \colon \mathcal{R}_j[\![\{c\}]\!] = 0\}$
// Remove elements from C_0 until C_0 is closed
2 **while** $\exists\, c \in C_0 \colon post(c) \not\subseteq states(C_0)$ **do** $C_0 \leftarrow C_0 \setminus \{c\}$
// The 0-MECs of $\mathcal{M}$ are the MECS of the submodel $\mathcal{M}[\![C_0]\!]$
3 **if** $C_0 = \emptyset$ **then return** $\emptyset$ **else return** $MEC(\mathcal{M}[\![C_0]\!])$

Algorithm 2.1: Computing maximal 0-ECs

2.2.7 Zeno Behavior

We say that an $\mathcal{M}$ has *Zeno* behavior if for some initial state and some strategy there is a positive probability for a Zeno path, i.e., Zeno

$$\exists\, s_I \in S \colon \exists\, \sigma \in \Sigma^{\mathcal{M}} \colon \Pr^{\mathcal{M}}_{\sigma,s_I}\big(\{\pi \in Paths_{\text{inf}} \mid dur(\pi) < \infty\}\big) > 0.$$

Intuitively, Zeno behavior reflects the possibility to perform infinitely many transitions in finite time. Such behavior is often considered unrealistic and complicates model analysis. It is commonly assumed that MAs that are subject to verification do *not* have Zeno behavior, see e.g., [HH12; GHHK+14; BWH17; 2]. In this work, we only forbid Zeno behavior in contexts where this is relevant—in particular when analyzing long-run average objectives discussed in Section 2.3.2 and Chapter 5. Checking whether a given MA has Zeno behavior can be done efficiently by considering its end components.

Lemma 2.5 An MA $\mathcal{M}$ has Zeno behavior iff $\exists\, C \in MEC(\mathcal{M}) \colon states(C) \cap MS = \emptyset$.

We stress that by our definitions MDPs always have Zeno behavior since they do not contain any Markovian state. For those models, we usually consider (discrete) transition *steps* instead of time.

2.2.8 Markov Decision Processes and Markov Chains

A Markov decision process is an MA without the continuous timing aspects.

Definition 2.23 (Markov Decision Process [Put94]) A *Markov decision process (MDP)* is an MA M with $MS^M = \emptyset$. MDP, Markov decision process

All notions above directly apply also to MDPs. However, MDPs do not contain any timing information. In particular, all paths of an MDP have duration zero and state reward assignments are not relevant. Due to the latter, we usually drop state rewards

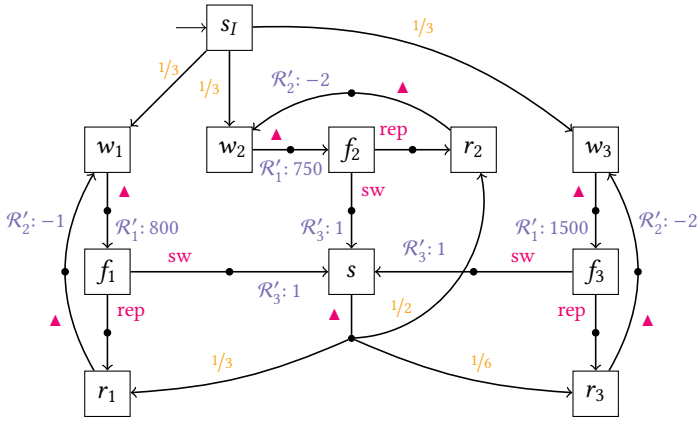


Figure 2.6: Underlying MDP $MDP(\mathcal{M})$ with reward assignments for MA from Figure 2.4 (cf. Example 2.16)

from our notations and identify a reward assignment for an MDP M with a transition reward assignment $\mathcal{R}\colon SA^M \times S \to \mathbb{R}$.

An MA $\mathcal{M}$ can be converted to an MDP to abstract away from all timing information.

underlying MDP

Definition 2.24 (Underlying MDP) The *underlying MDP* of an MA $\mathcal{M} = \langle S, Act, \Delta, \mathbf{P}\rangle$ is given by the MDP $MDP(\mathcal{M}) = \langle S, Act \cup \{\blacktriangle\}, \Delta', \mathbf{P}'\rangle$ where for $s \in S$ and $\alpha \in \Delta'(s)$

$$\Delta'(s) := \begin{cases} \Delta(s) & \text{if } s \in PS^{\mathcal{M}} \\ \{\blacktriangle\} & \text{if } s \in MS^{\mathcal{M}} \end{cases} \quad \text{and} \quad \mathbf{P}'(\langle s, \alpha\rangle) := \begin{cases} \mathbf{P}(\langle s, \alpha\rangle) & \text{if } s \in PS^{\mathcal{M}} \\ \mathbf{P}(s) & \text{if } s \in MS^{\mathcal{M}}. \end{cases}$$

We may also convert a reward assignment $\mathcal{R} = \langle \mathcal{R}^{\text{trans}}, \mathcal{R}^{\text{state}}\rangle$ for MA $\mathcal{M} = \langle S, Act, \Delta, \mathbf{P}\rangle$ to a reward assignment $MDP(\mathcal{R})$ for $MDP(\mathcal{M})$. To this end, state rewards $\mathcal{R}^{\text{state}}(s)$ for Markovian states $s \in MS^{\mathcal{M}}$ are multiplied with the expected time we stay in s, i.e., $\frac{1}{\Delta(s)}$. More precisely, we set

$$MDP(\mathcal{R})[s, \alpha, s'] := \begin{cases} \mathcal{R}[s, \alpha, s'] & \text{if } \langle s, \alpha\rangle \in SA^{\mathcal{M}} \\ \mathcal{R}[s, \frac{1}{\Delta(s)}, s'] & \text{if } s \in MS^{\mathcal{M}}, \alpha = \blacktriangle. \end{cases}$$

Example 2.16 Figure 2.6 shows the underlying MDP $MDP(\mathcal{M})$ and the converted reward assignments $\mathcal{R}'_1 := MDP(\mathcal{R}_1)$, $\mathcal{R}'_2 := MDP(\mathcal{R}_2)$, and $\mathcal{R}'_3 :=$

$MDP(\mathcal{R}_3)$ of MA $\mathcal{M}$ and $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$ as in Figure 2.4.

Besides MDPs, MA also subsume discrete- and continuous-time Markov chains.

Definition 2.25 (Markov Chain [Nor98]) A *discrete-time Markov chain (DTMC)* is a deterministic MA $D = \langle S, Act, \Delta, \mathbf{P}\rangle$ without Markovian states, i.e., $\forall\, s \in PS^D = S\colon\ |\Delta(s)| = 1$ and $MS^D = \emptyset$.
A *continuous-time Markov chain (CTMC)* is an MA $C = \langle S, Act, \Delta, \mathbf{P}\rangle$ without probabilistic states, i.e., $PS^C = \emptyset$.

DTMC, discrete-time Markov chain

CTMC, continuous-time Markov chain

A DTMC is also a (deterministic) MDP. The absence of probabilistic states in CTMCs already implies that CTMCs are deterministic. Since actions are not relevant for DTMCs and CTMCs, we usually omit them from the notation. Moreover, for DTMCs, the transition function Δ is unique. This allows us to write DTMCs as pairs $D = \langle S, \mathbf{P}\rangle$ with $\mathbf{P}\colon S \times S \to [0,1]$ and CTMCs as triples $C = \langle S, \Delta, \mathbf{P}\rangle$ with $\Delta\colon S \to \mathbb{R}_{>0}$ and $\mathbf{P}$ as for DTMC.

For an MDP $M = \langle S, Act, \Delta, \mathbf{P}\rangle$ and a pure memoryless strategy $\sigma \in \Sigma^M_{\mathrm{PM}}$, the induced submodel $M[\![\sigma]\!] = M[\![\{\langle s, \sigma(s)\rangle \mid s \in S\}]\!]$ is a DTMC—also referred to as the DTMC *induced* by M and σ. The probability measures $\Pr^M_{\sigma,s_I}$ and $\Pr^{M[\![\sigma]\!]}_{s_I}$ coincide.

induced DTMC

2.3 Objectives

Objectives describe quantitative measures of the system that are relevant for verification, e.g.,

- the probability to reach an unsafe state of the system,
- the expected time until a job is competed,
- the expected long-run average repair costs, or
- the probability to finish a task within a fixed energy budget.

Formally, we define an objective as a function that assigns a value to each execution of an MA.

Definition 2.26 (Objective) An *objective* for MA $\mathcal{M}$ is a measurable function $\varphi\colon Paths^{\mathcal{M}}_{\mathrm{inf}} \to \overline{\mathbb{R}}$. An objective φ is called *well-defined* (for $\mathcal{M}$) if $\Pr^{\mathcal{M}}_{\max}(\{\pi \mid \varphi(\pi) = -\infty\}) = 0$ or $\Pr^{\mathcal{M}}_{\max}(\{\pi \mid \varphi(\pi) = +\infty\}) = 0$.

objective

We fix an MA $\mathcal{M} = \langle S, Act, \Delta, \mathbf{P}\rangle$ and an initial state $s_I \in S$. For two objectives φ and φ' for $\mathcal{M}$, a constant $a \in \mathbb{R}$, and $\circ \in \{+, -, \cdot\}$ we let $\varphi \circ \varphi'$, $a \circ \varphi$, and $-\varphi$ denote the

objectives for $\mathcal{M}$ given by

$$(\varphi \circ \varphi')(\pi) \coloneqq \varphi(\pi) \circ \varphi'(\pi)\,, \quad (a \circ \varphi)(\pi) \coloneqq a \circ \varphi(\pi)\,, \quad \text{and} \quad (-\varphi)(\pi) \coloneqq -\varphi(\pi)$$

for $\pi \in \mathit{Paths}^{\mathcal{M}}_{\mathrm{inf}}$.

An objective φ for $\mathcal{M}$ is evaluated for a strategy $\sigma \in \Sigma$ by considering (or: computing) the expected value $\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\varphi)$. Well-definedness of an objective ensures that its expected value as given in Section 2.2.4 is defined for every strategy σ. From now on, we assume without further mention that all considered objectives are well-defined. We lift the notation for expected values to tuples of objectives, where for tuple $\Phi \coloneqq \langle \varphi_1, \ldots, \varphi_\ell \rangle$ we set

$$\mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) \;\coloneqq\; \langle \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\varphi_1), \ldots, \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\varphi_\ell) \rangle \in \overline{\mathbb{R}}^{\ell}.$$

Lemma 2.6 $\forall \boldsymbol{w} \in \mathbb{R}^{\ell}\colon\ \boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) = \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\boldsymbol{w} \cdot \Phi)$.

Proof.

$$\begin{aligned} \boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\Phi) \;&=\; \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \int_{\pi \in \mathit{Paths}_{\mathrm{inf}}} \varphi(\pi)\, \mathrm{dPr}^{\mathcal{M}}_{\sigma}(\pi) \\ &=\; \int_{\pi \in \mathit{Paths}_{\mathrm{inf}}} \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \varphi(\pi)\, \mathrm{dPr}^{\mathcal{M}}_{\sigma}(\pi) \;=\; \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\boldsymbol{w} \cdot \Phi). \end{aligned}$$ ∎

For a measurable set $\Pi \subseteq \mathit{Paths}^{\mathcal{M}}_{\mathrm{inf}}$ we let $[\![\Pi]\!]\colon \mathit{Paths}^{\mathcal{M}}_{\mathrm{inf}} \to \{0,1\}$ denote the objective that assigns value 1 to paths in Π, i.e., $[\![\Pi]\!](\pi) \coloneqq [\![\pi \in \Pi]\!]$ for $\pi \in \mathit{Paths}^{\mathcal{M}}_{\mathrm{inf}}$. For any $\sigma \in \Sigma^{\mathcal{M}}$ we have $\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}([\![\Pi]\!]) = \Pr^{\mathcal{M}}_{\sigma,s_I}(\Pi)$. We therefore also refer to an objective of the form $[\![\Pi]\!]$ as *probability objective*.

probability objective

By convention, we assume that our optimization goal is to obtain *high* expected values for the objectives. More concretely, if we are concerned with just a single objective φ, we are interested in (an approximation of) $\mathrm{Ex}^{\mathcal{M}}_{\max}(\varphi)$ and—if it exists—an inducing strategy $\sigma_{\max} \in \arg\max_{\sigma \in \Sigma^{\mathcal{M}}} \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\varphi)$. If there are $\ell > 1$ objectives $\varphi_1, \varphi_2, \ldots, \varphi_\ell$ we want to find a single strategy that induces a preferably high value for all objectives. The potential trade-offs that arise when some of these objectives conflict with each other are discussed in Chapter 3.

Preferring *high* objective values is an arbitrary choice that is without loss of generality: if for some objective φ a low value is preferred, we can always consider the negative objective $-\varphi$ instead of φ since for all $\sigma \in \Sigma^{\mathcal{M}}$ we have

$$\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\varphi) \;=\; -\int_{\pi \in \mathit{Paths}_{\mathrm{inf}}} -\varphi(\pi)\, \mathrm{dPr}^{\mathcal{M}}_{\sigma}(\pi) = -\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(-\varphi).$$

In particular, $\mathrm{Ex}^{\mathcal{M}}_{\min}(\varphi) = -\mathrm{Ex}^{\mathcal{M}}_{\max}(-\varphi)$. All classes of objectives introduced below are closed under considering negative objectives (cf. Lemmas 2.8, 2.11 and 2.12).

In the following, we present the various kinds of objectives that are relevant for this thesis. For now, our focus lies on defining and understanding the various objective types. A discussion on potential analysis methods is postponed to subsequent chapters.

2.3.1 Total Reachability Reward Objectives

Total reachability reward objectives accumulate rewards given by some reward assignment $\mathcal{R}$ along the considered path. Accumulation of rewards stops as soon as the path visits a state within a given set of goal states $G \subseteq S$. If no goal state is visited, reward is accumulated along the infinite path, potentially approaching $-\infty$ or $+\infty$.

total reachability reward objective

Definition 2.27 (Total Reachability Reward Objective) The *total reachability reward objective* for MA $\mathcal{M}$, a reward assignment $\mathcal{R}$, and goal states G is given by $tot(\mathcal{R}, G) \colon Paths^{\mathcal{M}}_{\inf} \to \overline{\mathbb{R}}$ where for $\pi = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \cdots \in Paths^{\mathcal{M}}_{\inf}$

$$tot(\mathcal{R}, G)(\pi) \coloneqq \begin{cases} \mathcal{R}[pref(\pi, n)] & \text{if } s_n \in G \text{ and } \forall\, i < n \colon s_i \notin G \\ \liminf_{n\to\infty} \mathcal{R}[pref(\pi, n)] & \text{if } \forall\, i \in \mathbb{N} \colon s_i \notin G. \end{cases}$$

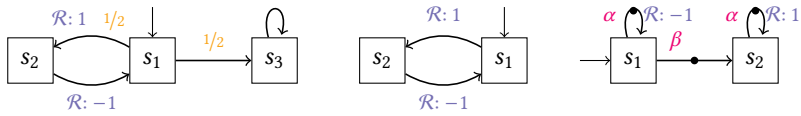
$tot(\mathcal{R}, G)$ is called *convergent* if

$$\forall\, s \in S \colon \mathrm{Ex}^{\mathcal{M}}_{\min,s}\big(tot(\mathcal{R}^-, G)\big) > -\infty \quad \text{or} \quad \forall\, s \in S \colon \mathrm{Ex}^{\mathcal{M}}_{\max,s}\big(tot(\mathcal{R}^+, G)\big) < +\infty.$$

Example 2.17 Recall from Example 2.12 that the reward assignment $\mathcal{R}_1$ for the MA $\mathcal{M}$ depicted in Figure 2.4 reflects the number of produced units per time. The total reachability reward objective $tot(\mathcal{R}_1, \{f_1, f_2, f_3\})$ thus yields the total number of produced units until the very first machine failure. For an infinite path π with prefix $s_I \to w_i \xrightarrow{t} f_i$ we have $tot(\mathcal{R}_1, \{f_1, f_2, f_3\})(\pi) = t \cdot \mathcal{R}_1[w_i]$. The expected value for initial state s_I is—in this case—independent of the considered strategy $\sigma \in \Sigma$ and can be computed by adding up for each $i \in \{1, 2, 3\}$ the product of

- the probability to move from s_I to w_i,
- the expected time to stay at w_i, and
- the reward accumulated while being in w_i.

The expected number of produced units until the first machine failure is thus



(a) $\mathcal{M}_1$ (b) $\mathcal{M}_2$ (c) $\mathcal{M}_3$

Figure 2.7: Example MAs illustrating total reachability rewards

given by

$$\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}\big(tot(\mathcal{R}_1,\{f_1,f_2,f_3\})(\pi)\big) \;=\; \frac{1}{3}\cdot 8\cdot 100 + \frac{1}{3}\cdot 5\cdot 150 + \frac{1}{3}\cdot 20\cdot 75 \;=\; \frac{3050}{3}.$$

We take the limit inferior in Definition 2.27 as the limit $\lim_{n\to\infty}\mathcal{R}[pref(\pi,n)]$ might not exist for all paths π as seen in Example 2.18 below. Alternatively, we can consider objectives $tot^{\sup}(\mathcal{R},G)$ that use the limit superior instead. However, throughout this thesis *we only consider convergent total reachability reward objectives.* In this case, the limit exists for *almost* all paths, i.e., the two variants coincide.

Lemma 2.7 $tot(\mathcal{R},G)$ is convergent iff $tot^{\sup}(\mathcal{R},G)$ is convergent. Furthermore, for convergent objectives we have for all $\sigma\in\Sigma^{\mathcal{M}}_{\mathrm{PM}}$:

$$\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(tot(\mathcal{R},G)) \;=\; \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(tot^{\sup}(\mathcal{R},G)).$$

Proof (sketch). For a given path $\pi\in Paths_{\mathrm{inf}}$ that does not visit a state in G, it holds that either

- from some point on only non-positive reward is accumulated,
- from some point on only non-negative reward is accumulated, or
- infinitely many positive *and* negative reward is accumulated.

For convergent objectives, the last case occurs with probability zero. For the remaining cases, $tot(\mathcal{R},G)$ and $tot^{\sup}(\mathcal{R},G)$ yield the same value. ■

Non-convergent total reachability reward objectives for MDPs are discussed in [BBDG+18].

Example 2.18 Consider the path $\pi = s_1 \to s_2 \to s_1 \to s_2 \to \ldots$ in the MA $\mathcal{M}_1$

from Figure 2.7a. For the depicted reward assignment $\mathcal{R}$ we get

$$tot(\mathcal{R}, \{s_3\}) \;=\; \liminf_{n\to\infty} \mathcal{R}[pref(\pi, n)] \;=\; \liminf_{n\to\infty} \sum_{k=0}^{n} (-1)^k \;=\; 0.$$

The series $1 - 1 + 1 - 1 + \ldots$ diverges and thus does not have a limit. However, the probability of alternating between s_1 and s_2 infinitely often in $\mathcal{M}_1$ is zero. When computing the expected value we can thus ignore such paths, i.e., s_3 is reached almost surely. The objective $tot(\mathcal{R}, \{s_3\})$ is convergent and we have $\mathrm{Ex}_{s_1}^{\mathcal{M}_1}(tot(\mathcal{R}, \{s_3\})) = 0$ and $\mathrm{Ex}_{s_2}^{\mathcal{M}_1}(tot(\mathcal{R}, \{s_3\})) = -1$.
The MA $\mathcal{M}_2$ in Figure 2.7b only has a single infinite path alternating between s_1 and s_2. The objective $tot(\mathcal{R}, \emptyset)$ is *not* convergent since

$$\mathrm{Ex}_{s_1}^{\mathcal{M}_2}(tot(\mathcal{R}^-, \emptyset)) \;=\; 0 - 1 + 0 - 1 + \ldots \;=\; -\infty \quad \text{and}$$
$$\mathrm{Ex}_{s_1}^{\mathcal{M}_2}(tot(\mathcal{R}^+, \emptyset)) \;=\; 1 + 0 + 1 + 0 + \ldots \;=\; +\infty.$$

However, $tot(\mathcal{R}, \emptyset)$ is still well-defined with $\mathrm{Ex}_{s_1}^{\mathcal{M}_2}(tot(\mathcal{R}, \emptyset)) = 0$ and $\mathrm{Ex}_{s_2}^{\mathcal{M}_2}(tot(\mathcal{R}, \emptyset)) = 1$.
Finally, the objective $tot(\mathcal{R}, \emptyset)$ for $\mathcal{M}_3$ in Figure 2.7c is neither convergent nor well-defined: a pure memoryless strategy σ yields value $-\infty$ at s_1 if $\sigma(s_1) = \alpha$ and value $+\infty$ if $\sigma(s_1) = \beta$. In fact, the expected value obtained for the (general) strategy σ' with $\sigma'(s_1) = unif(\{\alpha, \beta\})$ and $\sigma'(\hat{\pi}) = dirac(\alpha)$, $\hat{\pi} \in Paths_{\mathrm{fin}}^{\mathcal{M}_3} \setminus \{s_1\}$ is not defined.

The expected value of the negation of a convergent total reachability reward objective coincides with the expected value of a convergent total reachability reward objective that considers the negation of the reward assignment.

Lemma 2.8 For MA $\mathcal{M}$, $s_I \in S$, $\sigma \in \Sigma^{\mathcal{M}}$: if $tot(\mathcal{R}, G)$ is convergent, then

$$\mathrm{Ex}_{\sigma,s_I}^{\mathcal{M}}(-tot(\mathcal{R}, G)) \;=\; \mathrm{Ex}_{\sigma,s_I}^{\mathcal{M}}(tot(-\mathcal{R}, G)).$$

In the following, we discuss several types of objectives considered in the literature that can be seen as a special case of a total reachability reward objective.

total reward objective

Total reward objectives [Put94; FKNP+11] are total reachability reward objectives with an empty set of goal states. To simplify the notations, we also denote a total reward objective $tot(\mathcal{R}, \emptyset)$ for reward assignment $\mathcal{R}$ by $tot(\mathcal{R})$. Total reward objectives coincide with *discounted reward objectives* [Put94; BWH18] with discount factor 1.

reachability objective

Reachability objectives [BK08; FKNP11] are concerned with the probability to eventually reach a given set of goal states. Formally, the reachability objective for $G \subseteq S$ is

given by the probability objective $[\![\Diamond G]\!]$, where

$$\Diamond G \;\coloneqq\; \left\{ s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \cdots \in Paths^{\mathcal{M}}_{\text{inf}} \;\middle|\; \exists\, n \in \mathbb{N}\colon\; s_n \in G \right\}.$$

A reachability objective $[\![\Diamond G]\!]$ can also be seen as a (convergent) total reachability reward objective $tot(\mathcal{R}_G, G)$ where the reward assignment R_G assigns no state rewards and reward 1 for any transition that enters a goal state, i.e., $\mathcal{R}_G[s, \kappa, s'] \coloneqq [\![s' \in G]\!]$. More precisely, for $\sigma \in \Sigma^{\mathcal{M}}$ and $s \in S$ we have

$$\text{Ex}^{\mathcal{M}}_{\sigma,s}([\![\Diamond G]\!]) \;=\; \begin{cases} 1 & \text{if } s \in G \\ \text{Ex}^{\mathcal{M}}_{\sigma,s}(tot(\mathcal{R}_G, G)) & \text{if } s \notin G. \end{cases}$$

We remark that $\text{Ex}^{\mathcal{M}}_{\sigma,s}(tot(\mathcal{R}_G, G)) = 0$ if $s \in G$, i.e., this case requires special treatment when translating reachability objectives to total reachability rewards. Similarly, *constrained* reachability objectives $[\![H \,\mathcal{U}\, G]\!]$ with $H, G \subseteq S$ and

$$H \,\mathcal{U}\, G \;\coloneqq\; \left\{ s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \cdots \in Paths^{\mathcal{M}}_{\text{inf}} \;\middle|\; \exists\, n \in \mathbb{N}\colon\; s_n \in G \text{ and } \forall\, i < n\colon\; s_i \in H \right\}$$

can be restated as $tot(\mathcal{R}_G, G \cup (S \setminus H))$ with $\mathcal{R}_G$ as above.

Reachability time objectives [GHHK+14] consider the time to reach a set of goal states. Formally, a reachability time objective for a set of goal states $G \subseteq S$ is given by $tot(\mathcal{R}_{\text{time}}, G)$ for reward assignment $\mathcal{R}_{\text{time}}$ without transition rewards and $\mathcal{R}_{\text{time}}[s] = 1$ for all states $s \in S$. For MDPs, a reachability time objective always evaluates to 0.

> **Remark 2.3 (Alternative Definitions for Total Reachability Rewards)**
> Our definition of total reachability rewards (Definition 2.27) is in line with notions defined in, e.g., [GTHR+14] for MA. However, the literature [BK08; FKNP11] often considers a slightly different definition where paths that do not visit a goal state always get assigned a value of $+\infty$. In fact, this alternative definition is more established than our definition and implemented in probabilistic model checkers such as Prism and Storm. We argue that in our setting it is still reasonable to deviate from the standard definition.
> Firstly, in contrast to most works, we consider mixtures of positive and negative rewards. Assigning value $+\infty$ to paths that do not reach a goal state appears to be arbitrary in this setting.
> Secondly, as mentioned above, our definition allows us to treat various types of objectives—in particular reachability objectives—as a special case of total reachability reward objectives.
> Finally, we mention that the two definitions only differ in cases where the set of goal states $G \subseteq S$ is not reached with probability 1. In a multi-objective setting it

can be reasonable to add the probability objective $[\Diamond G]$ as a separate objective.

2.3.2 Long-Run Average Objectives

Long-run average objectives concern the average reward per time unit that is collected on an infinite path of an MA. They are also commonly referred to as mean-payoff or steady-state objectives.

LRA reward objective

Definition 2.28 (Long-run Average Reward Objective) The *long-run average (LRA) reward objective* for MA $\mathcal{M}$ and reward assignment $\mathcal{R}$ is given by $lra(\mathcal{R}) : Paths^{\mathcal{M}}_{\text{inf}} \to \overline{\mathbb{R}}$ where for $\pi = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \cdots \in Paths^{\mathcal{M}}_{\text{inf}}$

$$lra(\mathcal{R})(\pi) \;:=\; \liminf_{n\to\infty} \frac{\mathcal{R}[pref(\pi,n)]}{dur(pref(\pi,n))} \,.$$

Example 2.19 For the MA $\mathcal{M}$ and reward assignments as in Figure 2.4 and Example 2.12, $lra(\mathcal{R}_1)$ yields the long-run average number of units produced per time, whereas $lra(\mathcal{R}_2)$ corresponds to the long-run average repair costs per time. Let $\sigma_1 \in \Sigma^{\mathcal{M}}_{\text{PM}}$ be the pure memoryless strategy with $\sigma_1(f_1) = \text{rep}$ and $\sigma_1(f_2) = \sigma_1(f_3) = \text{sw}$. Under σ_1 the machine will from some time on always be in configuration 1. For σ_1 we observe that—on average—the machine will spent 8 out of $8 + 1 = 9$ time units in the working state w_1. The remaining time is spent in r_1. It follows that the expected average number of produced units per time unit is $\text{Ex}^{\mathcal{M}}_{\sigma_1,s_I}(lra(\mathcal{R}_1)) = {}^8\!/_9 \cdot 100 \approx 88.89$. Similarly, we obtain $\text{Ex}^{\mathcal{M}}_{\sigma_1,s_I}(lra(\mathcal{R}_2)) = {}^1\!/_9 \cdot -1 \approx -0.11$.

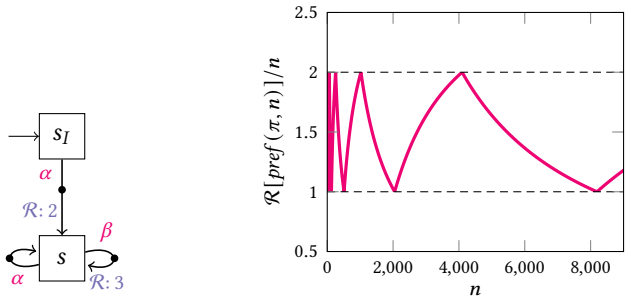
For MAs without Zeno behavior, long-run average reward objectives are always well-defined and their expected value is a real number.

Lemma 2.9 If $\mathcal{M}$ has no Zeno behavior, $\text{Ex}^{\mathcal{M}}_{\sigma}\big(lra(\mathcal{R})\big) \in \mathbb{R}$ for all $\sigma \in \Sigma^{\mathcal{M}}$.

Proof (sketch). The average number of transition steps performed by $\mathcal{M}$ per time unit can be bounded from above using

- the minimal average residence time (i.e., $\min_{s \in MS} {}^1\!/_{\Delta(s)}$) and
- the maximal expected number of steps until a Markovian state is reached from a probabilistic state.

For non-Zeno MA, both measures are finite numbers. As it is—on average—impossible to perform an infinite amount of steps per time unit, only a finite amount



(a) MDP M (b) average value after n steps

Figure 2.8: Example MDP and average value (cf. Example 2.21)

of reward can be accumulated per time unit. Thus, the expected LRA reward must be finite. ■

For MDP, measuring the average reward per time unit is not reasonable, as all paths in an MDP have duration 0. We therefore also introduce the long-run average reward per transition *step*.

step-based LRA reward objective

Definition 2.29 (Step-based Long-run Average Reward Objective) The *step-based LRA reward objective* for MA $\mathcal{M}$ and reward assignment $\mathcal{R}$ is given by $lra_{\mathrm{st}}(\mathcal{R})\colon Paths^{\mathcal{M}}_{\mathrm{inf}} \to \overline{\mathbb{R}}$ where for $\pi = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \cdots \in Paths^{\mathcal{M}}_{\mathrm{inf}}$

$$lra_{\mathrm{st}}(\mathcal{R})(\pi) \;:=\; \liminf_{n\to\infty} \frac{\mathcal{R}[pref(\pi,n)]}{n}\,.$$

Example 2.20 We consider the MDP $M = MDP(\mathcal{M})$ with $\mathcal{R}'_1$ and $\mathcal{R}'_2$ as in Figure 2.4. With the strategy σ_1 that—as in Example 2.19—from some point on keeps cycling between the states w_1, f_1, and r_1, we get $\mathrm{Ex}^{M}_{\sigma_1,s_I}(lra(\mathcal{R}_1)) = {}^{800}/_{3} \approx 266.7$ and $\mathrm{Ex}^{M}_{\sigma_1,s_I}(lra(\mathcal{R}_2)) = {}^{-1}/_{3} \approx -0.33$.

We consider the limit inferior in Definitions 2.28 and 2.29. The objectives $lra^{\mathrm{sup}}(\mathcal{R})$ and $lra^{\mathrm{sup}}_{\mathrm{st}}(\mathcal{R})$ that use the limit superior instead are defined similarly. The following example shows that these objectives generally do not coincide with their limit inferior variant. A similar example is given in [Put94, Example 8.1.1].

Example 2.21 Let M be the MDP from Figure 2.8a and let $\sigma \in \Sigma^{M}_{\mathrm{P}}$ be the pure

strategy such that for $\hat{\pi} \in Paths^{\mathcal{M}}_{\text{fin}}$ with $last(\hat{\pi}) = s$:

$$\sigma(\hat{\pi}) \;=\; \begin{cases} dirac(\alpha) & \text{if } \lfloor \log_2(|\hat{\pi}|) \rfloor \text{ is even} \\ dirac(\beta) & \text{if } \lfloor \log_2(|\hat{\pi}|) \rfloor \text{ is odd.} \end{cases}$$

Upon reaching s, σ first selects α once, then β twice, α four times, β eight times, and so fourth. For the single σ-consistent path $\pi \in Paths^{\mathcal{M}}_{\text{inf}}$, Figure 2.8b plots the average reward accumulated per step against the considered number of steps n. As the average reward oscillates between 1 and 2, we get

$$\text{Ex}^{\mathcal{M}}_{\sigma,s_I}(lra_{\text{st}}(\mathcal{R})) \;=\; 1 \quad \text{and} \quad \text{Ex}^{\mathcal{M}}_{\sigma,s_I}(lra^{\text{sup}}_{\text{st}}(\mathcal{R})) \;=\; 2.$$

Maximal expected LRA reward values $\text{Ex}^{\mathcal{M}}_{\max}(lra(\mathcal{R}))$ and $\text{Ex}^{\mathcal{M}}_{\max}(lra_{\text{st}}(\mathcal{R}))$ are attained by a pure memoryless strategy [Put94, Theorem 9.1.8]). The following result shows that under such strategies the values for $lra(\mathcal{R})$ and $lra^{\text{sup}}(\mathcal{R})$ as well as $lra_{\text{st}}(\mathcal{R})$ and $lra^{\text{sup}}_{\text{st}}(\mathcal{R})$ coincide.

Lemma 2.10 Let $\sigma \in \Sigma^{\mathcal{M}}_{\text{PM}}$. $\text{Ex}^{\mathcal{M}}_{\sigma,s_I}(lra(\mathcal{R})) = \text{Ex}^{\mathcal{M}}_{\sigma,s_I}(lra^{\text{sup}}(\mathcal{R}))$ holds for non-Zeno $\mathcal{M}$. Furthermore, $\text{Ex}^{\mathcal{M}}_{\sigma,s_I}(lra_{\text{st}}(\mathcal{R})) = \text{Ex}^{\mathcal{M}}_{\sigma,s_I}(lra^{\text{sup}}_{\text{st}}(\mathcal{R}))$ holds for arbitrary $\mathcal{M}$.

Proof (sketch). In the limit, only the end components (ECs) are relevant due to Lemma 2.4. If the induced submodel $\mathcal{M}[\![\sigma]\!]$ has no Zeno behavior, we can eliminate the probabilistic states in its ECs while preserving long-run average values. The ECs in the resulting model coincide with bottom strongly connected components of a finite continuous time Markov chain. The well-known Ergodic theorem (see, e.g., [Nor98, Theorem 1.10.2]) yields that the limit

$$\lim_{n\to\infty} \frac{\mathcal{R}[pref(\pi,n)]}{dur(pref(\pi,n))}$$

exists for almost all paths, yielding $\text{Ex}^{\mathcal{M}[\![\sigma]\!]}_{s_I}(lra(\mathcal{R})) = \text{Ex}^{\mathcal{M}[\![\sigma]\!]}_{s_I}(lra^{\text{sup}}(\mathcal{R}))$. A similar result for step-based LRA objectives is shown in [Put94, Proposition 8.1.1]. ■

Following, e.g., [BBCF+14], we limit our attention to the limit inferior variants, i.e., $lra(\mathcal{R})$ and $lra_{\text{st}}(\mathcal{R})$. The following dualities hold.

Lemma 2.11 For $\sigma \in \Sigma^{\mathcal{M}}$, we have $\text{Ex}^{\mathcal{M}}_{\sigma,s_I}(-lra(\mathcal{R})) \;=\; \text{Ex}^{\mathcal{M}}_{\sigma,s_I}(lra^{\text{sup}}(-\mathcal{R}))$ if $\mathcal{M}$ has no Zeno behavior, and $\text{Ex}^{\mathcal{M}}_{\sigma,s_I}(-lra_{\text{st}}(\mathcal{R})) \;=\; \text{Ex}^{\mathcal{M}}_{\sigma,s_I}(lra^{\text{sup}}_{\text{st}}(-\mathcal{R}))$ for arbitrary $\mathcal{M}$.

Long-run average time objectives [GHHK+14] query the average time the execution spends in a set of states $G \subseteq S$. They can be formulated as an LRA reward objective

$lra(\mathcal{R}_G)$ where $\mathcal{R}_G$ assigns reward 0 to all transitions and $\mathcal{R}_G[s] = [\![s \in G]\!]$ for $s \in S$. The corresponding notions for step-based measures are defined similarly.

2.3.3 Reward-Bounded Objectives

We consider objectives where constraints on the accumulated reward with respect to one or more reward assignments can be imposed. For this type of objective we mostly restrict ourselves to MDPs. MAs are addressed briefly in Remark 2.4. Let us fix an MDP $M = \langle S, Act, \Delta, \mathbf{P} \rangle$.

reward bound

Definition 2.30 (Reward Bound) A *(reward) bound* for MDP M is a structure $(\mathcal{R} \sim b)$ with

- reward assignment $\mathcal{R}$ for M,
- relation $\sim \in \{<, \leq, \geq, >\}$, and
- value $b \in \mathbb{R}$.

Intuitively, we use reward bounds to constrain the fragment of a path that is relevant for an objective. For a bound $(\mathcal{R} \sim b)$ and a path $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots \in Paths^M_{\text{inf}}$ we say that the bound is *active* at position $n \in \mathbb{N}$ iff $\mathcal{R}[pref(\pi, n)] \sim b$, i.e., the reward accumulated until step n relates to b according to $\sim$. The idea is to only consider the state s_n and transition step $s_{n-1} \xrightarrow{\alpha_{n-1}} s_n$ of π if *all* given reward bounds are active at position n. In the following, we make this notion more concrete by defining *bounded reward objectives* and *bounded reachability objectives*. Both use multiple reward bounds in conjunction. To simplify the notations when multiple objectives (potentially sharing the same bounds) are present, we fix a tuple

$$\mathcal{B} \;=\; \big\langle (\mathcal{R}_1 \sim_1 b_1), \ldots, (\mathcal{R}_d \sim_d b_d) \big\rangle$$

of $d > 0$ reward bounds and use sets of indices $\mathcal{I} \subseteq \{1..d\}$ to refer to the set of bounds $\mathcal{B}(\mathcal{I}) := \{\mathcal{B}(i) \mid i \in \mathcal{I}\}$.

For a reward assignment $\mathcal{R}$, an index set $\mathcal{I} \subseteq \{1..d\}$, and a finite path $\hat{\pi} = s_0 \xrightarrow{\alpha_0} \ldots \xrightarrow{\alpha_{n-1}} s_n \in Paths^M_{\text{fin}}$ we define the *accumulated reward* for $\mathcal{R}$ under $\mathcal{B}(\mathcal{I})$ along $\hat{\pi}$ as

$$\mathcal{R}^{\mathcal{B}(\mathcal{I})}[\hat{\pi}] \;:=\; \sum_{k=1}^{n} \Big(\mathcal{R}[s_{k-1}, \alpha_{k-1}, s_k] \cdot \big[\!\big[\forall\, i \in \mathcal{I}\colon\ \mathcal{R}_i[pref(\hat{\pi}, k)] \sim_i b_i \big]\!\big] \Big)$$

Intuitively, the accumulated reward under $\mathcal{B}(\mathcal{I})$ only accumulates rewards on fragments of the path where *all* bounds in $\mathcal{B}(\mathcal{I})$ are active. Bounded reward objectives lift this notion to infinite paths.

Definition 2.31 (Bounded Reward Objective) Let $\mathcal{B}$ be a d-tuple of reward bounds. The *bounded reward objective* for MDP M, reward assignment $\mathcal{R}$, and index set $\mathcal{I} \subseteq \{1..d\}$ is given by $bnd^{\mathcal{B}}(\mathcal{R}, \mathcal{I}) \colon Paths^{M}_{\text{inf}} \to \overline{\mathbb{R}}$ where for $\pi \in Paths^{M}_{\text{inf}}$

$$bnd^{\mathcal{B}}(\mathcal{R}, \mathcal{I})(\pi) \;:=\; \liminf_{n\to\infty} \mathcal{R}^{\mathcal{B}(\mathcal{I})}[pref(\pi, n)].$$

$bnd^{\mathcal{B}}(\mathcal{R}, \mathcal{I})$ is *convergent* if

$$\forall\, s \in S \colon \mathrm{Ex}^{M}_{\min,s}\big(bnd^{\mathcal{B}}(\mathcal{R}^{-}, \mathcal{I})\big) > -\infty \quad \text{or} \quad \forall\, s \in S \colon \mathrm{Ex}^{M}_{\max,s}\big(bnd^{\mathcal{B}}(\mathcal{R}^{+}, \mathcal{I})\big) < \infty.$$

Similar to total reachability reward objectives (cf. Definition 2.27), a convergent bounded reward objective is also well-defined and the limit in Definition 2.31 exists for almost all paths.

A total reachability reward objective $tot(\mathcal{R}, G)$ can also be restated as a bounded reward objective $bnd^{\mathcal{B}}(\mathcal{R}, \{i\})$, where $\mathcal{B}(i) = \llparenthesis \mathcal{R}_G \leq 0 \rrparenthesis$ and for $\langle s, \alpha \rangle \in SA^{M}$ and $s' \in S$: $\mathcal{R}_G[s, \alpha, s'] := [\![s' \in G]\!]$.

Example 2.22 Consider the MDP $M = MDP(\mathcal{M})$ with reward assignments $\mathcal{R}'_1$, $\mathcal{R}'_2$, and $\mathcal{R}'_3$ as in Figure 2.6. We assume a similar interpretation for M and the reward assignments as discussed in Example 2.12 for the original MA $\mathcal{M}$. However, the measures and values below do not carry over to the MA model, cf. Remark 2.4. Let

$$\mathcal{B} \;=\; \big\langle \llparenthesis \mathcal{R}'_2 \geq -2 \rrparenthesis,\ \llparenthesis \mathcal{R}'_3 \leq 0 \rrparenthesis \big\rangle.$$

The bounded reward objective $bnd^{\mathcal{B}}(\mathcal{R}'_1, \{1, 2\})$ asks for the number of units that are produced until either

- more than 2 repair costs are incurred (here, repair costs are given as negative rewards) or
- a switching action (sw) has been performed.

To maximize the expected value, we consider the strategy that never chooses action sw. Under this strategy, if we reach w_1 we may collect $3 \cdot 800$ reward for $\mathcal{R}'_1$ before the accumulated reward for $\mathcal{R}'_2$ exceeds the threshold -2. Similarly, we get $2 \cdot 750$ and $2 \cdot 1500$ when we initially move to w_2 and w_3, respectively. This yields

$$\mathrm{Ex}^{M}_{\max,s_I}\big(bnd^{\mathcal{B}}(\mathcal{R}'_1, \{1, 2\})\big) \;=\; \frac{1}{3} \cdot 2400 + \frac{1}{3} \cdot 1500 + \frac{1}{3} \cdot 3000 \;=\; 2300.$$

Next, we consider bounded reachability objectives where reward bounds can be equipped with a set of goal states.

bounded goal

Definition 2.32 (Bounded Goal) A *bounded goal* for MDP M over reward bounds $\mathcal{B}$ as above is a pair $\langle I, G\rangle$, where $I \subseteq \{1..d\}$ is a set of indices and $G \subseteq S$ is a set of goal states.

The interpretation of a bounded goal $\langle I, G\rangle$ is that we only account for visits of goal states in G if *all* reward bounds $\mathcal{B}(i)$ for $i \in I$ are active. For a path $\pi \in Paths^M_{\text{inf}}$ we write

$$\pi \models \langle I, G\rangle \quad \text{iff} \quad \exists\, \hat{\pi} \in pref(\pi)\colon\ last(\hat{\pi}) \in G \text{ and } \forall\, i \in I\colon\ \mathcal{R}_i[\hat{\pi}] \sim_i b_i.$$

We fix a tuple of $m > 0$ bounded goals over $\mathcal{B}$

$$\mathcal{G} \;=\; \big\langle \langle I_1, G_1\rangle, \ldots, \langle I_m, G_m\rangle \big\rangle.$$

Similar as for $\mathcal{B}$, we simplify notations when multiple objectives are considered by using sets of indices $\mathcal{J} \subseteq \{1..m\}$ to refer to the set of bounded goals $\mathcal{G}(\mathcal{J}) := \{\mathcal{G}(j) \mid j \in \mathcal{J}\}$.

bounded reachability objective

Definition 2.33 (Bounded Reachability Objective) Let $\mathcal{B}$ and $\mathcal{G}$ be as above. The *bounded reachability objective* for MDP M, scaling value $c \in \mathbb{R}$, and index set $\mathcal{J} \subseteq \{1..m\}$ is given by $bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})\colon Paths^M_{\text{inf}} \to \overline{\mathbb{R}}$ where for $\pi \in Paths^M_{\text{inf}}$

$$bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})(\pi) \;:=\; c \cdot \big[\forall\, j \in \mathcal{J}\colon\ \pi \models \langle I_j, G_j\rangle\big].$$

The expected value $\mathrm{Ex}^M_{\sigma,s_I}(bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J}))$ coincides with the probability that *all* bounded goals in $\mathcal{G}(\mathcal{J})$ are satisfied (from $s_I \in S$ under $\sigma \in \Sigma^M$) multiplied by the constant factor c.

Example 2.23 Consider again the MDP M with reward assignments as in Figure 2.6 and Example 2.22. Furthermore, we consider reward bounds $\mathcal{B}$ and bounded goals $\mathcal{G}$ with

$$\mathcal{B} \;=\; \big\langle [\![\mathcal{R}'_1 \geq 800]\!],\ [\![\mathcal{R}'_1 \leq 1500]\!],\ [\![\mathcal{R}'_1 < 1000]\!],\ [\![\mathcal{R}'_2 \leq -2]\!] \big\rangle$$
$$\mathcal{G} \;=\; \big\langle \langle \{1,2\}, \{s\}\rangle,\ \langle \{3,4\}, \emptyset\rangle \big\rangle.$$

The bounded goal $\mathcal{G}(1)$ asks to reach the state s when at least 800 and at most 1500 units have been produced. The bounded goal $\mathcal{G}(2)$ asks to produce less than 1000 units while 2 or more repair costs are incurred.

Every path that satisfies *both* bounded goals has one of the following prefixes:

$$s_I \to w_1 \xrightarrow{\blacktriangle} f_1 \xrightarrow{\text{sw}} s \xrightarrow{\blacktriangle} r_2 \xrightarrow{\blacktriangle} w_2$$
$$s_I \to w_1 \xrightarrow{\blacktriangle} f_1 \xrightarrow{\text{sw}} s \xrightarrow{\blacktriangle} r_3 \xrightarrow{\blacktriangle} w_3$$
$$s_I \to w_2 \xrightarrow{\blacktriangle} f_2 \xrightarrow{\text{rep}} r_2 \xrightarrow{\blacktriangle} w_2 \xrightarrow{\blacktriangle} f_2 \xrightarrow{\text{sw}} s$$
$$s_I \to w_2 \xrightarrow{\blacktriangle} f_2 \xrightarrow{\text{sw}} s \xrightarrow{\blacktriangle} r_2 \xrightarrow{\blacktriangle} w_2 \xrightarrow{\blacktriangle} f_2 \xrightarrow{\text{sw}} s$$

To maximize the expected value for the bounded reachability objective $bnd^{\mathcal{B}}_{\mathcal{G}}(1, \{1,2\})$, we consider the strategy that selects rep at the first visit of f_2 and sw in all other cases. We obtain the resulting value by summing up the probabilities for the four path prefixes above:

$$\mathrm{Ex}^{M}_{\max,s_I}\big(bnd^{\mathcal{B}}_{\mathcal{G}}(1,\{1,2\})\big) \;=\; \frac{1}{3}\cdot\frac{1}{2} + \frac{1}{3}\cdot\frac{1}{6} + \frac{1}{3} + 0 \;=\; \frac{5}{9}.$$

The classes of bounded reward objectives and bounded reachability objectives are closed under taking their negation.

Lemma 2.12 For MDP M, $s_I \in S$, $\sigma \in \Sigma^M$, and convergent $bnd^{\mathcal{B}}(\mathcal{R}, \mathcal{I})$:

$$\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(-bnd^{\mathcal{B}}(\mathcal{R},\mathcal{I})) \;=\; \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(bnd^{\mathcal{B}}(-\mathcal{R},\mathcal{I})) \text{ and}$$
$$\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(-bnd^{\mathcal{B}}_{\mathcal{G}}(c,\mathcal{J})) \;=\; \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(bnd^{\mathcal{B}}_{\mathcal{G}}(-c,\mathcal{J})).$$

Furthermore, reward bounds are not affected by scaling with a constant factor.

Lemma 2.13 For $\mathcal{B} = \langle \llbracket \mathcal{R}_1 \sim_1 b_1 \rrbracket, \ldots, \llbracket \mathcal{R}_d \sim_d b_d \rrbracket \rangle$ and $a_1, \ldots, a_d \in \mathbb{R} \setminus \{0\}$ let $\mathcal{B}'$ be the d-tuple with

$$\mathcal{B}'(i) \;=\; \llbracket (a_i \cdot \mathcal{R}_i) \sim'_i (a_i \cdot b_i) \rrbracket, \quad \text{where} \quad \sim'_i \;=\; \begin{cases} \sim_i & \text{if } a_i < 0 \\ \sim_i^C & \text{if } a_i < 0 \end{cases}$$

and $\sim_i^C$ is the *converse relation* of $\sim_i$, e.g., $\leq^C = \geq$. Then, for objectives $bnd^{\mathcal{B}}(\mathcal{R},\mathcal{I})$ and $bnd^{\mathcal{B}}_{\mathcal{G}}(c,\mathcal{J})$ we have

$$bnd^{\mathcal{B}}(\mathcal{R},\mathcal{I}) \;=\; bnd^{\mathcal{B}'}(\mathcal{R},\mathcal{I}) \quad \text{and} \quad bnd^{\mathcal{B}}_{\mathcal{G}}(c,\mathcal{J}) \;=\; bnd^{\mathcal{B}'}_{\mathcal{G}}(c,\mathcal{J}).$$

Proof. When observing a finite path $\hat{\pi} \in Paths^{M}_{\text{fin}}$, bound $\llbracket \mathcal{R}_i \sim_i b_i \rrbracket$ is active iff

bound $\wr(a_i \cdot \mathcal{R}_i) \sim_i' (a_i \cdot b_i)\wr$ is active, i.e.,

$$\mathcal{R}_i[\hat{\pi}] \sim_i b_i \quad \text{iff} \quad (a_i \cdot \mathcal{R}_i)[\hat{\pi}] \sim_i' a_i \cdot b_i.$$

The lemma follows straightforwardly from the definition of bounded reward and bounded reachability objectives. ■

The following theorem yields that optimal expected values are not computable in general. Assumption 6.1 on page 172 imposes restrictions on the reward bounds that enable computational tractability. In particular, these restrictions enforce that reward bounds may not refer to mixtures of positive and negative rewards.

Theorem 2.14 Given $a \in \mathbb{R}$ and objectives $bnd^{\mathcal{B}}(\mathcal{R}, \mathcal{I})$ and $bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})$, it is undecidable whether $\mathrm{Ex}^{M}_{\max,s_I}(bnd^{\mathcal{B}}(\mathcal{R}, \mathcal{I})) \geq a$ and $\mathrm{Ex}^{M}_{\max,s_I}(bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})) \geq a$, respectively.

Proof. [RRS17, Theorem 12] shows—using a reduction from the halting problem for two-counter machines—that it is undecidable whether $\Pr^{M}_{\max}(\{\pi \models \langle \mathcal{I}, G\rangle\}) = 1$ holds for bounded goals $\langle \mathcal{I}, G\rangle$ with $|\mathcal{I}| \geq 4$ reward bounds and an absorbing set of goal states G. Theorem 2.14 follows with

$$\begin{aligned}\Pr^{M}_{\max}(\{\pi \models \langle \mathcal{I}, G\rangle\}) = 1 \ \text{ iff } \ & \mathrm{Ex}^{M}_{\max,s_I}(bnd^{\mathcal{B}}(\mathcal{R}_G, \mathcal{I})) \geq 1 \\ \text{iff } \ & \mathrm{Ex}^{M}_{\max,s_I}(bnd^{\mathcal{B}}_{\mathcal{G}}(1, \{1\})) \geq 1,\end{aligned}$$

where $\mathcal{R}_G[s, \alpha, s'] \coloneqq [\![s' \in G]\!]$ and $\mathcal{G} = \left\langle \langle \mathcal{I}, G\rangle \right\rangle$.

Remark 2.4 (Reward-bounded objectives on MA) Definitions 2.31 and 2.33 do not apply for MA as they do not reflect timing aspects. More precisely, for a path $\pi = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \ldots$ of an MA, and reward bounds $\mathcal{B} = \langle \wr \mathcal{R}_1 \sim_1 b_1 \wr, \wr \mathcal{R}_2 \sim_2 b_2 \wr \rangle$ we can have $\mathcal{R}_1[pref(\pi, n)] \nsim_1 b_1$ and $\mathcal{R}_2[pref(\pi, n+1)] \nsim_2 b_2$ but since reward can be accumulated while staying in s_n it might be the case that $\mathcal{R}[pref(\pi, n)] + t \cdot \mathcal{R}[s_n] \sim_i b_i$ for $i \in \{1, 2\}$ and some $t \leq dur(\kappa_n)$, i.e., both reward bounds can be active some time between the n^{th} and the $(n+1)^{\text{th}}$ step of π.

If we only allow transition rewards in reward bounds (i.e., all assigned state rewards are set to 0), reward-bounded objectives for MA can be treated just like for MDP. More general reward-bounded objectives on MA are outside the scope of this thesis.

1-dimensional time-bounded reachability objectives concerning the probability to

reach a set of goal states within a given time interval have been treated in [HH12; GHHK+14; BHHK15; BF19] for the single-objective case and in [2] for the multi-objective case. Bounded reward objectives on MA have been considered in a single-objective setting for a single time-bound [GTHR+14; BFHW+15] and for a single reward-bound [HWBF+17].

—Chapter 3—

Model Checking Multiple Objectives

Outlook | We consider the verification of Markov automata against $\ell > 0$ objectives. The objectives can be arbitrary measurable functions (cf. Definition 2.26 on page 47), i.e., the results presented in this chapter are applicable to a broad spectrum of objectives that go beyond the kinds of objectives presented in Section 2.3—including multiple ω-regular [FKNP+11] and percentile [RRS17] objectives.

Section 3.1 introduces the notions of *achievability* and *Pareto optimality* which are both defined on sets of points in $\overline{\mathbb{R}}^{\ell}$. We investigate properties of such sets of points in Section 3.2 and define the different kinds of multi-objective model checking queries that we want to solve in Section 3.3. Finally, Section 3.4 presents an algorithmic framework based on the approach of [FKP12] to answer multi-objective model checking queries efficiently.

Origins | The treatment of arbitrary combinations of objectives (each given by some measurable function) originates from [11] which we extend towards more elaborative proofs and explanations as well as a more careful consideration of cases involving infinite values. As a consequence, the majority of this chapter is new—with the notable exceptions of Theorem 3.5 and Lemma 3.11 and the corresponding proofs which are adapted from [13, Proposition 1 and Theorem 2]. The framework of [11] itself takes strong inspiration from [FKP12] which—by transitivity—also carries over to this chapter.

Set-up | We fix an MA $\mathcal{M} = \langle S, Act, \Delta, \mathbf{P} \rangle$, an initial state $s_I \in S$, and a tuple $\Phi := \langle \varphi_1, \ldots, \varphi_\ell \rangle$ of $\ell > 0$ well-defined objectives $\varphi_1, \ldots, \varphi_\ell \colon Paths^{\mathcal{M}}_{\text{inf}} \to \overline{\mathbb{R}}$.

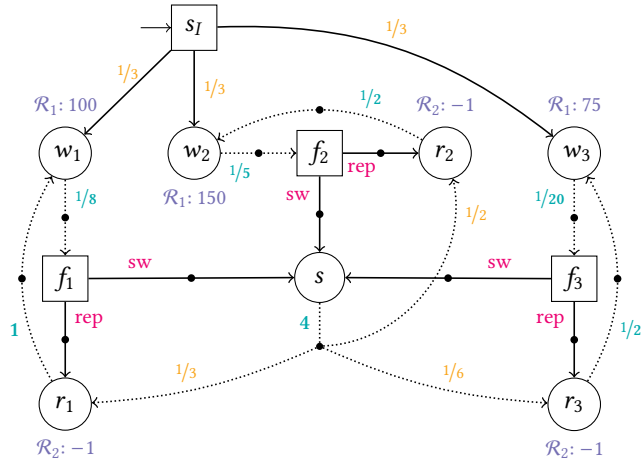


Figure 3.1: Markov automaton $\mathcal{M}$ with reward assignments $\mathcal{R}_1, \mathcal{R}_2$.

3.1 Achievability and Pareto Optimality

As mentioned in Section 2.3, we assume without loss of generality that the objectives φ_j ($j \in \{1..\ell\}$) are given such that strategies σ inducing high expected values $\mathrm{Ex}^{\mathcal{M}}_{\sigma}(\varphi_j)$ are preferred[1]. More concretely, if for two strategies $\sigma_1, \sigma_2 \in \Sigma^{\mathcal{M}}$ we have $\mathrm{Ex}^{\mathcal{M}}_{\sigma_1}(\varphi_j) > \mathrm{Ex}^{\mathcal{M}}_{\sigma_2}(\varphi_j)$, strategy σ_1 should be considered "better" than σ_2—at least with respect to the individual objective φ_j. However, for another objective φ_k ($k \in \{1..\ell\} \setminus \{j\}$) we might have $\mathrm{Ex}^{\mathcal{M}}_{\sigma_1}(\varphi_k) < \mathrm{Ex}^{\mathcal{M}}_{\sigma_2}(\varphi_k)$, i.e., σ_2 is "better" than σ_1. The crux in multi-objective model checking is that all objectives shall be evaluated under the same strategy. This potentially introduces trade-offs between the objectives since an optimal strategy for one objective is not necessarily optimal for another.

achievable point

Definition 3.1 (Achievable Point) A point $\boldsymbol{p} \in \overline{\mathbb{R}}^{\ell}$ is called *achievable* for objectives Φ iff $\exists\, \sigma \in \Sigma^{\mathcal{M}}\colon \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\Phi) \geq \boldsymbol{p}$. The set of achievable points for Φ is denoted by $Ach^{\mathcal{M}}_{s_I}(\Phi) \subseteq \overline{\mathbb{R}}^{\ell}$.

For an achievable point $\boldsymbol{p}$ there is one strategy σ that, when evaluated for the individual objectives, induces expected values that are at least the values given by $\boldsymbol{p}$. In this case we also say that σ *achieves* $\boldsymbol{p}$. If clear from context, we may drop the MA $\mathcal{M}$ or the initial state s_I from the notation $Ach^{\mathcal{M}}_{s_I}(\Phi)$.

[1] If low expected values are preferred, the negated objective $-\varphi_j$ can be considered instead, cf. Lemmas 2.8, 2.11 and 2.12 in Section 2.3.

Example 3.1 Having established the formal background, we now revisit Example 1.2 on page 5. The MA $\mathcal{M}$ with reward assignments from Figure 1.1 on page 3 is again shown in Figure 3.1. Furthermore, we consider the two long-run average reward objectives $\Phi = \langle lra(\mathcal{R}_1), lra(\mathcal{R}_2)\rangle$. For $i \in \{1,2,3\}$ let $\sigma_i \in \Sigma^{\mathcal{M}}_{\text{PM}}$ be the pure memoryless strategy with $\sigma_i(f_i) = \text{rep}$ and $\sigma_i(f_j) = \text{sw}$, $j \in \{1,2,3\} \setminus \{i\}$. With a similar reasoning as in Example 2.19 on page 53 we get

$$\begin{aligned}
\text{Ex}^{\mathcal{M}}_{\sigma_1,s_I}(lra(\mathcal{R}_1)) &\approx 88.89 & \text{Ex}^{\mathcal{M}}_{\sigma_1,s_I}(lra(\mathcal{R}_2)) &\approx -0.11\\
\text{Ex}^{\mathcal{M}}_{\sigma_2,s_I}(lra(\mathcal{R}_1)) &\approx 107.1 & \text{Ex}^{\mathcal{M}}_{\sigma_2,s_I}(lra(\mathcal{R}_2)) &\approx -0.29\\
\text{Ex}^{\mathcal{M}}_{\sigma_3,s_I}(lra(\mathcal{R}_1)) &\approx 68.18 & \text{Ex}^{\mathcal{M}}_{\sigma_3,s_I}(lra(\mathcal{R}_2)) &\approx -0.09
\end{aligned}$$

We depict the points $\boldsymbol{p}_1 = \text{Ex}^{\mathcal{M}}_{\sigma_1,s_I}(\Phi) \approx \langle 88.89, -0.11\rangle$, $\boldsymbol{p}_2 = \text{Ex}^{\mathcal{M}}_{\sigma_2,s_I}(\Phi) \approx \langle 107.1, -0.29\rangle$, and $\boldsymbol{p}_3 = \text{Ex}^{\mathcal{M}}_{\sigma_3,s_I}(\Phi) \approx \langle 68.18, -0.09\rangle$ in Figure 3.2. For $i \in \{1,2,3\}$, σ_i achieves $\boldsymbol{p}_i$ which implies $\boldsymbol{p}_i \in Ach^{\mathcal{M}}(\Phi)$. Among the three considered strategies, σ_2 yields the highest value for the first objective $lra(\mathcal{R}_1)$ (depicted on the x-axis) but does not perform very good in terms of the second objective $lra(\mathcal{R}_2)$ (depicted on the y-axis). Vice versa, σ_3 is good for $lra(\mathcal{R}_2)$ but bad for $lra(\mathcal{R}_1)$. Strategy σ_1 is a compromise between the two objectives. However, without further information it is impossible to tell which of the three strategies yields the "best" results.
We can show that the set of achievable points is given by

$$Ach^{\mathcal{M}}(\Phi) = down(conv(\{\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3\}))$$

as sketched by the green area in Figure 3.2. Intuitively, points on the line segments $line(\boldsymbol{p}_1, \boldsymbol{p}_2)$ and $line(\boldsymbol{p}_1, \boldsymbol{p}_3)$ can be achieved by strategies that randomly select to mimic either σ_1 or σ_2 and either σ_1 or σ_3, respectively. We provide more details in Section 3.2.

In the single-objective case ($\ell = 1$), $Ach(\Phi) = Ach(\langle\varphi_1\rangle)$ corresponds to either $[-\infty, p)$ or $[-\infty, p]$ where $p = \text{Ex}^{\mathcal{M}}_{\max}(\varphi_1) = \sup_{\sigma\in\Sigma} \text{Ex}^{\mathcal{M}}_{\sigma}(\varphi_1)$ is included iff the supremum can be attained by some strategy. The value $p \in \overline{\mathbb{R}}$ corresponds to the frontier of $Ach(\langle\varphi_1\rangle)$ meaning that any value $p' > p$ is *not* achievable. We lift this notion to multiple objectives.

Definition 3.2 (Pareto Optimality) A point $\boldsymbol{p} \in cl(Ach^{\mathcal{M}}_{s_I}(\Phi))$ is called *Pareto optimal* for objectives Φ iff $\forall\, \boldsymbol{p}' \gtrsim \boldsymbol{p}\colon\ \boldsymbol{p}' \notin cl(Ach^{\mathcal{M}}_{s_I}(\Phi))$. The set of Pareto optimal points for Φ is called the *Pareto front* for Φ and denoted by $Pareto^{\mathcal{M}}_{s_I}(\Phi)$.

Pareto optimal

Pareto front

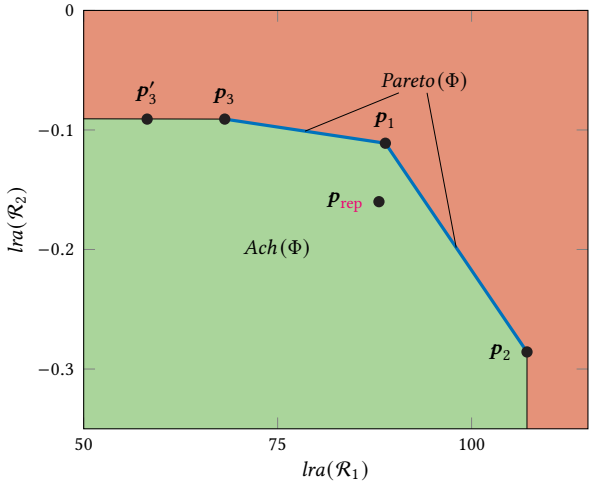


Figure 3.2: Achievable points for two LRA objectives (cf. Examples 3.1 and 3.2)

dominating point

For two points $\boldsymbol{p}, \boldsymbol{p}' \in \overline{\mathbb{R}}^{\ell}$ we say that $\boldsymbol{p}'$ *dominates* $\boldsymbol{p}$ iff $\boldsymbol{p}' \geq \boldsymbol{p}$ (i.e., $\boldsymbol{p} \geq \boldsymbol{p}'$ and $\boldsymbol{p} \neq \boldsymbol{p}'$). We further lift this to strategies $\sigma, \sigma' \in \Sigma^{\mathcal{M}}$ and say that σ' dominates σ iff $\mathrm{Ex}^{\mathcal{M}}_{\sigma',s_I}(\Phi) \geq \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\Phi)$. Again, $\mathcal{M}$ or s_I can be dropped from the notation $\mathit{Pareto}^{\mathcal{M}}_{s_I}(\Phi)$ if they are clear from context.

Example 3.2 Continuing Example 3.1, we now also consider the strategy $\sigma_{\text{rep}} \in \Sigma^{\mathcal{M}}_{\text{PM}}$ with $\sigma_{\text{rep}}(f_i) = \text{rep}$ for all $i \in \{1, 2, 3\}$. We have

$$\mathrm{Ex}^{\mathcal{M}}_{\sigma_{\text{rep}},s_I}(\mathit{lra}(\mathcal{R}_1)) \approx 88.06 \qquad \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\text{rep}},s_I}(\mathit{lra}(\mathcal{R}_{\text{rep}})) \approx -0.16$$

σ_{rep} thus achieves the point $\boldsymbol{p}_{\text{rep}} = \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\text{rep}},s_I}(\Phi) \approx \langle 88.06, 0.16 \rangle$ also indicated in Figure 3.2. However, $\boldsymbol{p}_{\text{rep}}$ is dominated by $\boldsymbol{p}_1$. In other words, σ_1 should be preferred over σ_{rep} as it yields higher values for *both* objectives.
On the other hand, there is no achievable point that yields "better" values than $\boldsymbol{p}_1$, i.e., $\boldsymbol{p}_1$ is Pareto optimal. A similar argument can be made for each of the points on the blue line segments in Figure 3.2—including $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$. It follows that

$$\mathit{Pareto}^{\mathcal{M}}\Phi = \mathit{line}(\boldsymbol{p}_1, \boldsymbol{p}_2) \cup \mathit{line}(\boldsymbol{p}_1, \boldsymbol{p}_3).$$

If a point is Pareto optimal, it is not dominated by any achievable point. On the other

hand, achievable points that are not Pareto optimal are always dominated by at least one Pareto optimal point. The Pareto front lies on the boundary between achievable and unachievable points as formalized in the following lemma.

Lemma 3.1 $Pareto(\Phi) \subseteq bd(Ach(\Phi)) \cup \{\infty\}^\ell$.

Proof. If $\langle \infty, \dots, \infty \rangle \in Pareto(\Phi)$, then $Pareto(\Phi) = \{\langle \infty, \dots, \infty \rangle\} = \{\infty\}^\ell$ since the point $\langle \infty, \dots, \infty \rangle$ dominates any other achievable point. In this case the claim holds.
Now assume $\langle \infty, \dots, \infty \rangle \notin Pareto(\Phi)$. Let $\boldsymbol{p} \in Pareto(\Phi)$. By definition of Pareto optimal points we get $\boldsymbol{p} \in cl(Ach(\Phi))$. It remains to show that $\boldsymbol{p} \in cl(\overline{\mathbb{R}}^\ell \setminus Ach(\Phi))$, i.e., any neighborhood N of $\boldsymbol{p}$ contains a point that is not achievable. Recall from Definition 2.2 on page 17 that we can write N as Cartesian product $N = \bigtimes_{i=1}^{\ell} N_i$, where $N_i \subseteq \overline{\mathbb{R}}$ is a neighborhood of $\boldsymbol{p}(i)$ for each $i \in \{1..\ell\}$. Let $j \in \{1..\ell\}$ such that $\boldsymbol{p}(j) \neq \infty$ and consider the point $\boldsymbol{p}' \in \overline{\mathbb{R}}^\ell$ with

$$\boldsymbol{p}'(i) = \begin{cases} \boldsymbol{p}(i) & \text{if } i \neq j \\ \boldsymbol{p}(j) + \varepsilon/2 & \text{if } i = j, \boldsymbol{p}(j) \in \mathbb{R} \text{ and } N_i \supseteq (\boldsymbol{p}(j) - \varepsilon, \boldsymbol{p}(j) + \varepsilon) \\ a - 1 & \text{if } i = j, \boldsymbol{p}(j) = -\infty \text{ and } N_i \supseteq [-\infty, a) \end{cases}$$

for all $i \in \{1..\ell\}$. We have $\boldsymbol{p}' \in N$ and $\boldsymbol{p}' \notin Ach(\Phi)$. The latter follows because $\boldsymbol{p}' \gtrsim \boldsymbol{p}$ i.e., $\boldsymbol{p}'$ dominates the Pareto optimal point $\boldsymbol{p}$. In summary, we get $\boldsymbol{p} \in cl(Ach(\Phi)) \cap cl(\overline{\mathbb{R}}^\ell \setminus Ach(\Phi)) = bd(Ach(\Phi))$. ∎

The other direction of Lemma 3.1 does not hold: for example, in Figure 3.2 the point $\boldsymbol{p}_3' = \boldsymbol{p}_3 - \langle 10, 0 \rangle$ lies on the boundary of $Ach(\Phi)$ but is not Pareto optimal as it is dominated by the achievable point $\boldsymbol{p}_3$.

We further characterize the relation between $Pareto(\Phi)$ and $Ach(\Phi)$.

Lemma 3.2 $Pareto(\Phi)$ is the smallest set $P \subseteq \overline{\mathbb{R}}^\ell$ with

$$down(P) = cl(Ach(\Phi)).$$

Proof. We first show $down(Pareto(\Phi)) = cl(Ach(\Phi))$ and then show that $Pareto(\Phi)$ is the smallest such set.

Direction "⊆" | Let $\boldsymbol{p} \in down(Pareto(\Phi))$, i.e., there is $\boldsymbol{p}' \in Pareto(\Phi)$ with $\boldsymbol{p} \leq \boldsymbol{p}'$. By definition, $Pareto(\Phi) \subseteq cl(Ach(\Phi))$, implying that also $\boldsymbol{p}' \in cl(Ach(\Phi))$ and therefore $\boldsymbol{p} \in down(cl(Ach(\Phi)))$. Since $cl(Ach(\Phi))$ is downward closed—which we show in Lemma 3.4 on page 70—we conclude $\boldsymbol{p} \in cl(Ach(\Phi))$.

Direction "⊇" | Let $\boldsymbol{p} \in cl(Ach(\Phi))$. We construct a point $\boldsymbol{p}' \in Pareto(\Phi)$ with $\boldsymbol{p} \leq \boldsymbol{p}'$ as follows. A sketch is shown in Figure 3.3. For $i \in \{1..\ell\}$ we set

$$\begin{aligned} \boldsymbol{p}'(i) &:= \max\left\{a \in \overline{\mathbb{R}} \;\middle|\; \boldsymbol{p}_{i\mapsto a} \in cl(Ach(\Phi))\right\} \quad \text{with} \\ \boldsymbol{p}_{i\mapsto a} &:= \langle \boldsymbol{p}'(1), \ldots, \boldsymbol{p}'(i{-}1), a, \boldsymbol{p}(i{+}1), \ldots, \boldsymbol{p}(\ell)\rangle. \end{aligned}$$

$\boldsymbol{p}'$ is well-defined as the value $\boldsymbol{p}'(i)$ for $i \in \{1..\ell\}$ only depends on $\boldsymbol{p}'(j)$ for $j < i$. Furthermore, the "max" always exists as we consider the closed set $cl(Ach(\Phi))$, yielding $\boldsymbol{p}' \in cl(Ach(\Phi))$. As $\boldsymbol{p} \in cl(Ach(\Phi))$, we have $\boldsymbol{p}' \geq \boldsymbol{p}$ by definition of $\boldsymbol{p}'$. We now show that $\boldsymbol{p}' \in Pareto(\Phi)$ which directly yields $\boldsymbol{p} \in down(Pareto(\Phi))$. For that, consider a point $\boldsymbol{q} \in \overline{\mathbb{R}}^{\ell}$ with $\boldsymbol{q} \gtrsim \boldsymbol{p}'$, i.e., there is $j \in \{1..\ell\}$ such that

$$\boldsymbol{q}(j) \;>\; \boldsymbol{p}'(j) \;=\; \max\left\{a \in \overline{\mathbb{R}} \;\middle|\; \boldsymbol{p}_{j\mapsto a} \in cl(Ach(\Phi))\right\}.$$

In the sketch from Figure 3.3 we have $j = 1$. Let $a' := \boldsymbol{q}(j)$. For the point $\boldsymbol{p}_{j\mapsto a'}$ we thus have $\boldsymbol{p}_{j\mapsto a'} \notin cl(Ach(\Phi))$. Furthermore, $\boldsymbol{p}_{j\mapsto a'} \in down(\{\boldsymbol{q}\})$ which also yields—since $cl(Ach(\Phi))$ is downward closed according to Lemma 3.4—that $\boldsymbol{q} \notin cl(Ach(\Phi))$.

Smallest Set | Next, consider an arbitrary set $P \subseteq \overline{\mathbb{R}}^{\ell}$ with $down(P) = cl(Ach(\Phi))$. It remains to show that $Pareto(\Phi) \subseteq P$. Assume towards a contradiction that there is a point $\boldsymbol{p} \in Pareto(\Phi) \setminus P$. We have

$$\boldsymbol{p} \;\in\; Pareto(\Phi) \;\subseteq\; down(Pareto(\Phi)) \;=\; cl(Ach(\Phi)) \;=\; down(P)$$

and therefore $\boldsymbol{p} \in down(P) \setminus P$. Now take $\boldsymbol{p}' \in P$ with $\boldsymbol{p}' \gtrsim \boldsymbol{p}$. We have

$$\boldsymbol{p}' \;\in\; P \;\subseteq\; down(P) \;=\; cl(Ach(\Phi)).$$

$\boldsymbol{p}$ is thus dominated by a point $\boldsymbol{p}' \in cl(Ach(\Phi))$, contradicting $\boldsymbol{p} \in Pareto(\Phi)$. ■

A Pareto optimal point is not necessarily achievable.

Lemma 3.3 In general, $Pareto(\Phi) \not\subseteq Ach(\Phi)$.

Proof. Consider the MA $\mathcal{M}'$ in Figure 3.4 with objectives $\Phi' = \langle \varphi, [\Diamond\{s'\}]\rangle$, where

$$\forall\, \pi \in Paths_{\text{inf}}^{\mathcal{M}'} \colon\; \varphi(\pi) = \sum_{\hat{\pi} \in pref(\pi)} [last(\hat{\pi}) = s] \cdot \frac{1}{2^{|\hat{\pi}|}}.$$

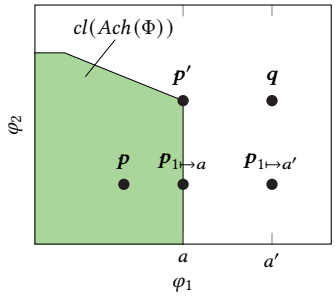


Figure 3.3: Illustration of points considered in the proof of Lemma 3.2

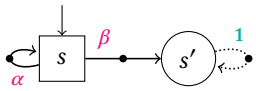


Figure 3.4: MA $\mathcal{M}'$ from the proof of Lemma 3.3

The strategy that always chooses α at state s achieves the point

$$\langle 1 + 1/2 + 1/4 + \ldots,\ 0\rangle \ = \ \langle 2, 0\rangle.$$

The strategy that chooses α for the first $n \in \mathbb{N}$ times and then selects β achieves $\langle 2 - \frac{1}{2^n},\ 1\rangle$. With increasing n, we get arbitrarily close to achieving $\langle 2, 1\rangle$ but the point $\langle 2, 1\rangle$ itself is not achievable. We get

$$\begin{aligned} Ach^{\mathcal{M}'}(\Phi') &= down(\{\langle 2,1\rangle\}) \setminus \{\langle 2,1\rangle\}, \\ cl(Ach^{\mathcal{M}'}(\Phi')) &= down(\{\langle 2,1\rangle\}), \text{ and} \\ Pareto^{\mathcal{M}'}(\Phi') &= \{\langle 2,1\rangle\}. \end{aligned}$$

Two more cases where Pareto optimal points are not achievable are given by $\mathcal{M}_1$ with Φ_1 and $\mathcal{M}_2$ with Φ_2 discussed in Examples 3.6 and 3.7 on page 94 and on page 97. ■

3.2 Closeness Properties of Achievable Points

We present various properties of the sets $Ach(\Phi)$ and $Pareto(\Phi)$. These results allow for a concise representation of (approximations of) the achievable points and provide the formal basis for the algorithm presented in Section 3.4.

A direct consequence of Lemma 3.3 is that $Ach(\Phi)$ might not be a closed set, i.e., we might have $Ach(\Phi) \neq cl(Ach(\Phi))$. However, both sets $Ach(\Phi)$ and its closure $cl(Ach(\Phi))$ are downward closed and convex as we show in the following.

Lemma 3.4 *$Ach(\Phi)$ and $cl(Ach(\Phi))$ are downward closed, i.e.,*

$$Ach(\Phi) = down(Ach(\Phi)) \quad \text{and} \quad cl(Ach(\Phi)) = down(cl(Ach(\Phi))).$$

Proof. It remains to show that $Ach(\Phi)$ is downward closed. Downward closedness of $cl(Ach(\Phi))$ follows from Lemma 2.2 on page 21 Obviously, $Ach(\Phi) \subseteq down(Ach(\Phi))$ holds. To show $Ach(\Phi) \supseteq down(Ach(\Phi))$, consider $\boldsymbol{p} \in down(Ach(\Phi))$. There is $\boldsymbol{p}' \in Ach(\Phi)$ with $\boldsymbol{p} \leq \boldsymbol{p}'$. Let $\boldsymbol{p}'$ be achieved by $\sigma \in \Sigma^{\mathcal{M}}$. The strategy σ also achieves $\boldsymbol{p}$ as $\boldsymbol{p} \leq \boldsymbol{p}' \leq \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\Phi)$. Thus $\boldsymbol{p} \in Ach(\Phi)$. ■

Next, we show that the set of achievable points is not only downward-closed but also convex. However, as the notion of convexity does not straightforwardly generalize to $\overline{\mathbb{R}}^{\ell}$, we limit our attention to the real-valued fragment of $Ach(\Phi)$.

Theorem 3.5 *$Ach(\Phi) \cap \mathbb{R}^{\ell}$ is convex, i.e.,*

$$Ach(\Phi) \cap \mathbb{R}^{\ell} = conv(Ach(\Phi) \cap \mathbb{R}^{\ell}).$$

Before we provide a formal proof for Theorem 3.5, we discuss its correctness on a more intuitive level and establish some auxiliary lemmas. We mention that similar results have been shown for MDP with multiple ω-regular objectives [EKVY08], total reward objectives [FKP12], and long-run average objectives [BBCF+14]. However, lifting these results to the more general setting—in particular to MA with continuous time—is not obvious.

To argue the convexity of $Ach(\Phi) \cap \mathbb{R}^{\ell}$, the crux is to show for any pair of achievable points $\boldsymbol{p}_1, \boldsymbol{p}_2 \in Ach(\Phi) \cap \mathbb{R}^{\ell}$ that $line(\boldsymbol{p}_1, \boldsymbol{p}_2) \subseteq Ach(\Phi) \cap \mathbb{R}^{\ell}$, i.e., any point $\boldsymbol{p}$ on the line connecting $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ is also achievable. For $j \in \{1, 2\}$ let $\sigma_j \in \Sigma^{\mathcal{M}}$ be a strategy achieving $\boldsymbol{p}_j$ and let $\lambda \in [0, 1]$ such that $\boldsymbol{p} = \lambda \cdot \boldsymbol{p}_1 + (1 - \lambda) \cdot \boldsymbol{p}_2 \in line(\boldsymbol{p}_1, \boldsymbol{p}_2)$. From $\boldsymbol{p}_1, \boldsymbol{p}_2 \in \mathbb{R}^{\ell}$ we immediately get $\boldsymbol{p} \in \mathbb{R}^{\ell}$. It thus remains to show that $\boldsymbol{p} \in Ach(\Phi)$. In the following, we construct a strategy that achieves $\boldsymbol{p}$. Intuitively, our strategy makes an initial one-off random choice:

- with probability λ mimic σ_1 and
- with probability $1 - \lambda$ mimic σ_2.

Specifying such a strategy as a function from finite paths to distributions over (enabled) actions as in Definition 2.16 on page 34 is technically involved because such a function can not memorize the outcome of the initial one-off random choice. For a finite path

$\hat{\pi} = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \ldots \xrightarrow{\kappa_{n-1}} s_n$ with $s_n \in PS$ and $\alpha \in \Delta(s_n)$, the value $\sigma(\hat{\pi})(\alpha)$ needs to depend on the previously chosen actions $act(\kappa_0), \ldots, act(\kappa_{n-1})$ in $\hat{\pi}$, and the probability that these choices adhere to either σ_1 or σ_2. Formally, for strategy $\sigma \in \Sigma^{\mathcal{M}}$ and $\hat{\pi}$ as above let

$$p_{\langle \sigma | \hat{\pi} \rangle} \coloneqq \prod_{i=0}^{n-1} \sigma\big(pref(\hat{\pi}, i)\big)\big(act(\kappa_i)\big)$$

which intuitively reflects the probability that we follow strategy σ given that the path $\hat{\pi}$ is observed. Furthermore, for $\sigma_1, \sigma_2 \in \Sigma^{\mathcal{M}}$ and $\lambda \in [0,1]$ with $\lambda_1 \coloneqq \lambda$ and $\lambda_2 \coloneqq 1 - \lambda_1$, we define the strategy $[\sigma_1 \oplus_\lambda \sigma_2] \in \Sigma^{\mathcal{M}}$ by setting

$$[\sigma_1 \oplus_\lambda \sigma_2](\hat{\pi})(\alpha) \coloneqq \frac{\sum_{j=1}^{2} \sigma_j(\hat{\pi})(\alpha) \cdot \lambda_j \cdot p_{\langle \sigma_j | \hat{\pi} \rangle}}{\sum_{j=1}^{2} \lambda_j \cdot p_{\langle \sigma_j | \hat{\pi} \rangle}}$$

for all $\hat{\pi} \in Paths_{\text{fin}}$ with $last(\hat{\pi}) \in PS$ and $\alpha \in \Delta(last(\hat{\pi}))$.

We now establish auxiliary lemmas that allow us to show that the above-mentioned point $\boldsymbol{p} = \lambda \cdot \boldsymbol{p}_1 + (1-\lambda) \cdot \boldsymbol{p}_2$ is achieved by the strategy $[\sigma_1 \oplus_\lambda \sigma_2]$. For a finite path $\hat{\pi} = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \ldots \xrightarrow{\kappa_{n-1}} s_n \in Paths_{\text{fin}}$ we consider the notation

$$[\hat{\pi}]_{Act} \coloneqq \left\{ s_0 \xrightarrow{\kappa'_0} s_1 \xrightarrow{\kappa'_1} \ldots \xrightarrow{\kappa'_n} s_n \in Paths_{\text{fin}} \;\middle|\; \forall\, i \in \{0..n-1\} \colon\; act(\kappa_i) = act(\kappa'_i) \right\},$$

i.e., $[\hat{\pi}]_{Act}$ consists of the paths that consider the same sequence of states and actions as $\hat{\pi}$ but potentially have different durations.

The following lemma allows us to separate the probabilities introduced by a strategy from the probabilities occurring in the MA.

Lemma 3.6 Consider a finite path $\hat{\pi} = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \ldots \xrightarrow{\kappa_{n-1}} s_n \in Paths_{\text{fin}}$ and the strategy $\sigma_{\hat{\pi}} \in \Sigma^{\mathcal{M}}$ given by

$$\sigma_{\hat{\pi}}(\hat{\pi}') \coloneqq \begin{cases} dirac(act(\kappa_{|\hat{\pi}'|})) & \text{if } \exists\, i \in \{0..n-1\} \colon\; \hat{\pi}' \in [pref(\hat{\pi}, i)]_{Act} \\ unif(\Delta(last(\hat{\pi}'))) & \text{otherwise} \end{cases}$$

for $\hat{\pi}' \in Paths_{\text{fin}}$, i.e., $\sigma_{\hat{\pi}}$ deterministically chooses the actions as given in $\hat{\pi}$—unless this is not possible in which case an action is chosen uniformly at random. For every measurable set $\Pi \subseteq [\hat{\pi}]_{Act}$ and every $\sigma \in \Sigma^{\mathcal{M}}, s \in S$ we have

$$\Pr^{\mathcal{M}}_{\sigma,s}(\, Cyl(\Pi)\,) = \int_{\hat{\pi}' \in \Pi} p_{\langle \sigma | \hat{\pi}' \rangle} \, \mathrm{d}\Pr^{|\hat{\pi}|}_{\sigma_{\hat{\pi}},s}(\hat{\pi}'). \tag{3.1}$$

Proof. The proof is by induction over the length n of $\hat{\pi} = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \dots \xrightarrow{\kappa_{n-1}} s_n$. Observe that both sides of Equation (3.1) evaluate to 0 if the first state s_0 visited by $\hat{\pi}$ does not match s. We thus assume $s = s_0$ for the rest of the proof.

Induction Base | If $n = 0$ we have either $\Pi = \{s\}$ or $\Pi = \emptyset$ and thus either $Cyl(\Pi) = Paths_{\text{inf}}(s)$ or $Cyl(\Pi) = \emptyset$. Furthermore, $p_{\langle \sigma | \hat{\pi} \rangle} = 1$. Equation (3.1) follows immediately in both cases.

Induction Step | Now assume that Equation (3.1) holds for paths of length $n-1$, i.e.,

$$\Pr^{\mathcal{M}}_{\sigma,s}(\, Cyl(\Pi^\bullet)\,) \;=\; \int_{\hat{\pi}^\bullet \in \Pi^\bullet} p_{\langle \sigma | \hat{\pi}^\bullet \rangle} \, \mathrm{dPr}^{n-1}_{\sigma_{\hat{\pi}},s}(\hat{\pi}^\bullet).$$

holds for $\Pi^\bullet := \{pref(\hat{\pi}', n-1) \mid \hat{\pi}' \in \Pi\}$. Notice that for $\hat{\pi}^\bullet \in \Pi^\bullet$ we have $last(\hat{\pi}^\bullet) = s_{n-1}$. We distinguish two cases.

- If $s_{n-1} \in PS$ we have $\Pi = \{\hat{\pi}^\bullet \xrightarrow{\alpha} s_n \mid \hat{\pi}^\bullet \in \Pi^\bullet\}$, where $\alpha := act(\kappa_{n-1})$. Equation (3.1) follows with

$$\begin{aligned}
\Pr^{\mathcal{M}}_{\sigma,s}(\, Cyl(\Pi)\,) &= \Pr^{n}_{\sigma,s}(\Pi) \\
&= \int_{\hat{\pi}^\bullet \in \Pi^\bullet} \sigma(\hat{\pi}^\bullet)(\alpha) \cdot \mathbf{P}(s_{n-1}, \alpha, s_n) \, \mathrm{dPr}^{n-1}_{\sigma,s}(\hat{\pi}^\bullet) \\
&= \int_{\hat{\pi}^\bullet \in \Pi^\bullet} \sigma(\hat{\pi}^\bullet)(\alpha) \cdot \mathbf{P}(s_{n-1}, \alpha, s_n) \cdot p_{\langle \sigma | \hat{\pi}^\bullet \rangle} \, \mathrm{dPr}^{n-1}_{\sigma_{\hat{\pi}},s}(\hat{\pi}^\bullet) \\
&= \int_{\hat{\pi}^\bullet \in \Pi^\bullet} p_{\langle \sigma | \hat{\pi}^\bullet \xrightarrow{\alpha} s_n \rangle} \cdot \mathbf{P}(s_{n-1}, \alpha, s_n) \, \mathrm{dPr}^{n-1}_{\sigma_{\hat{\pi}},s}(\hat{\pi}^\bullet) \\
&= \int_{\hat{\pi}^\bullet \in \Pi^\bullet} p_{\langle \sigma | \hat{\pi}^\bullet \xrightarrow{\alpha} s_n \rangle} \cdot \underbrace{\sigma_{\hat{\pi}}(\hat{\pi}^\bullet)(\alpha)}_{=1} \cdot \mathbf{P}(s_{n-1}, \alpha, s_n) \, \mathrm{dPr}^{n-1}_{\sigma_{\hat{\pi}},s}(\hat{\pi}^\bullet) \\
&= \int_{\hat{\pi}' \in \Pi} p_{\langle \sigma | \hat{\pi}' \rangle} \cdot \mathrm{dPr}^{n}_{\sigma_{\hat{\pi}},s}(\hat{\pi}').
\end{aligned}$$

- If $s_{n-1} \in MS$ we consider $T_{\hat{\pi}^\bullet} := \{\langle t, s_n \rangle \in \mathbb{R}_{\geq 0} \mid \hat{\pi}^\bullet \xrightarrow{t} s_n \in \Pi\}$ for $\hat{\pi}^\bullet \in \Pi^\bullet$ and note that the probability of the transition step $\Pr^{Step}_{\sigma,\hat{\pi}^\bullet}(T_{\hat{\pi}^\bullet})$ does not depend on σ,

i.e., $\Pr^{Step}_{\sigma,\hat{\pi}^\bullet}(T_{\hat{\pi}^\bullet}) = \Pr^{Step}_{\sigma_{\hat{\pi}},\hat{\pi}^\bullet}(T_{\hat{\pi}^\bullet})$. Equation (3.1) follows with

$$\begin{aligned}
\Pr^{\mathcal{M}}_{\sigma,s}(\mathit{Cyl}(\Pi)) &= \Pr^{n}_{\sigma,s}(\Pi) \\
&= \int_{\hat{\pi}^\bullet \in \Pi^\bullet} \Pr^{Step}_{\sigma,\hat{\pi}^\bullet}(T_{\hat{\pi}^\bullet}) \, \mathrm{dPr}^{n-1}_{\sigma,s}(\hat{\pi}^\bullet) \\
&= \int_{\hat{\pi}^\bullet \in \Pi^\bullet} \Pr^{Step}_{\sigma,\hat{\pi}^\bullet}(T_{\hat{\pi}^\bullet}) \cdot p_{\langle \sigma|\hat{\pi}^\bullet\rangle} \, \mathrm{dPr}^{n-1}_{\sigma_{\hat{\pi}},s}(\hat{\pi}^\bullet) \\
&= \int_{\hat{\pi}^\bullet \in \Pi^\bullet} p_{\langle \sigma|\hat{\pi}^\bullet\rangle} \cdot \underbrace{\sigma(\hat{\pi}^\bullet)(\blacktriangle)}_{=1} \cdot \Pr^{Step}_{\sigma_{\hat{\pi}},\hat{\pi}^\bullet}(T_{\hat{\pi}^\bullet}) \, \mathrm{dPr}^{n-1}_{\sigma_{\hat{\pi}},s}(\hat{\pi}^\bullet) \\
&= \int_{\hat{\pi}' \in \Pi} p_{\langle \sigma|\hat{\pi}'\rangle} \cdot \mathrm{dPr}^{n}_{\sigma_{\hat{\pi}},s}(\hat{\pi}'). \qquad \blacksquare
\end{aligned}$$

Next, we establish the connection between the strategy $[\sigma_1 \oplus_\lambda \sigma_2]$ and the individual strategies σ_1 and σ_2.

Lemma 3.7 For $\hat{\pi} \in \mathit{Paths}_{\mathrm{fin}}$, $\sigma_1, \sigma_2 \in \Sigma^{\mathcal{M}}$, and $\lambda \in [0,1]$ with $\lambda_1 := \lambda$ and $\lambda_2 := 1 - \lambda_1$ it holds that

$$p_{\langle \sigma|\hat{\pi}^\bullet \xrightarrow{\alpha} s_n\rangle} = \sum_{j=1}^{2} \lambda_j \cdot p_{\langle \sigma_j|\hat{\pi}\rangle}.$$

Proof. For $\hat{\pi} = s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} \ldots \xrightarrow{\kappa_{n-1}} s_n$ we have

$$\begin{aligned}
p_{\langle \sigma|\hat{\pi}^\bullet \xrightarrow{\alpha} s_n\rangle} &= \prod_{i=0}^{n-1} [\sigma_1 \oplus_\lambda \sigma_2](\mathit{pref}(\hat{\pi}, i))(\mathit{act}(\kappa_i)) \\
&= \prod_{i=0}^{n-1} \frac{\sum_{j=1}^{2} \left(\lambda_j \cdot \prod_{k=0}^{i} \sigma_j(\mathit{pref}(\hat{\pi}, k))(\mathit{act}(\kappa_k))\right)}{\sum_{j=1}^{2} \left(\lambda_j \cdot \prod_{k=0}^{i-1} \sigma_j(\mathit{pref}(\hat{\pi}, k))(\mathit{act}(\kappa_k))\right)} \\
&= \frac{\sum_{j=1}^{2} \left(\lambda_j \cdot \prod_{k=0}^{n-1} \sigma_j(\mathit{pref}(\hat{\pi}, k))(\mathit{act}(\kappa_k))\right)}{\sum_{j=1}^{2} \left(\lambda_j \cdot \prod_{k=0}^{0-1} \sigma_j(\mathit{pref}(\hat{\pi}, k))(\mathit{act}(\kappa_k))\right)} \\
&= \sum_{j=1}^{2} \left(\lambda_j \cdot \prod_{k=0}^{n-1} \sigma_j(\mathit{pref}(\hat{\pi}, k))(\mathit{act}(\kappa_k))\right) = \sum_{j=1}^{2} \lambda_j \cdot p_{\langle \sigma_j|\hat{\pi}\rangle}. \qquad \blacksquare
\end{aligned}$$

We now use Lemmas 3.6 and 3.7 to show for arbitrary objectives that the expected value induced by $[\sigma_1 \oplus_\lambda \sigma_2]$ is indeed the weighted sum of the expected values induced by σ_1 and σ_2, respectively.

Lemma 3.8 For every pair of strategies $\sigma_1, \sigma_2 \in \Sigma^{\mathcal{M}}$, weight $\lambda \in [0,1]$, and objective φ we have

$$\mathrm{Ex}^{\mathcal{M}}_{[\sigma_1 \oplus_\lambda \sigma_2]}(\varphi) \;=\; \lambda \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_1}(\varphi) + (1-\lambda) \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_2}(\varphi).$$

Proof. Let $\lambda_1 = \lambda$ and $\lambda_2 = (1-\lambda)$. We show that for any measurable set $\Lambda \subseteq \mathit{Paths}_{\mathrm{inf}}$ we have

$$\Pr^{\mathcal{M}}_{[\sigma_1 \oplus_\lambda \sigma_2]}(\Lambda) \;=\; \sum_{j=1}^{2} \lambda_j \cdot \Pr^{\mathcal{M}}_{\sigma_j}(\Lambda). \tag{3.2}$$

From there, Lemma 3.8 follows via

$$\begin{aligned}\mathrm{Ex}^{\mathcal{M}}_{[\sigma_1 \oplus_\lambda \sigma_2]}(\varphi) \;&=\; \int_{\pi \in \mathit{Paths}_{\mathrm{inf}}} \varphi(\pi)\, \mathrm{dPr}^{\mathcal{M}}_{[\sigma_1 \oplus_\lambda \sigma_2]}(\pi) \\ &=\; \sum_{j=1}^{2} \lambda_j \int_{\pi \in \mathit{Paths}_{\mathrm{inf}}} \varphi(\pi)\, \mathrm{dPr}^{\mathcal{M}}_{\sigma_j}(\Pi) \;=\; \sum_{j=1}^{2} \lambda_j \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_j}(\varphi).\end{aligned}$$

Any measurable $\Lambda \subseteq \mathit{Paths}_{\mathrm{inf}}$ can be expressed via countable unions and complements of cylinder sets of the form $\mathit{Cyl}([\hat{\pi}]_{\mathit{Act}})$ for $\hat{\pi} \in \mathit{Paths}_{\mathrm{fin}}$. It therefore suffices to show Equation (3.2) only for the case where $\Lambda = \mathit{Cyl}(\Pi)$ with $\Pi \subseteq [\hat{\pi}]_{\mathit{Act}}$ for some $\hat{\pi} \in \mathit{Paths}_{\mathrm{fin}}$. Using Lemmas 3.6 and 3.7 we get

$$\begin{aligned}\Pr^{\mathcal{M}}_{[\sigma_1 \oplus_\lambda \sigma_2]}(\Lambda) \;&=\; \Pr^{\mathcal{M}}_{[\sigma_1 \oplus_\lambda \sigma_2]}(\mathit{Cyl}(\Pi)) \\ &=\; \int_{\hat{\pi}' \in \Pi} p_{\langle [\sigma_1 \oplus_\lambda \sigma_2] | \hat{\pi}' \rangle}\, \mathrm{dPr}^{|\hat{\pi}|}_{\sigma_{\hat{\pi}}, s}(\hat{\pi}') \\ &=\; \int_{\hat{\pi}' \in \Pi} \sum_{j=1}^{2} \lambda_j \cdot p_{\langle \sigma_j | \hat{\pi}' \rangle}\, \mathrm{dPr}^{|\hat{\pi}|}_{\sigma_{\hat{\pi}}, s}(\hat{\pi}') \\ &=\; \sum_{j=1}^{2} \lambda_j \cdot \int_{\hat{\pi}' \in \Pi} p_{\langle \sigma_j | \hat{\pi}' \rangle}\, \mathrm{dPr}^{|\hat{\pi}|}_{\sigma_{\hat{\pi}}, s}(\hat{\pi}') \\ &=\; \sum_{j=1}^{2} \lambda_j \cdot \Pr^{\mathcal{M}}_{\sigma_j}(\mathit{Cyl}(\Pi)) \;=\; \sum_{j=1}^{2} \lambda_j \cdot \Pr^{\mathcal{M}}_{\sigma_j}(\Lambda).\end{aligned}$$

■

We are finally ready to prove the convexity of $\mathit{Ach}(\Phi) \cap \mathbb{R}^\ell$.

Proof (of Theorem 3.5). Obviously, $\mathit{Ach}(\Phi) \cap \mathbb{R}^\ell \subseteq \mathit{conv}(\mathit{Ach}(\Phi) \cap \mathbb{R}^\ell)$. For a point $\boldsymbol{p} \in \mathit{conv}(\mathit{Ach}(\Phi) \cap \mathbb{R}^\ell)$ there are two points $\boldsymbol{p}_1, \boldsymbol{p}_2 \in \mathit{Ach}(\Phi) \cap \mathbb{R}^\ell$ and $\lambda \in [0,1]$

such that $\boldsymbol{p} = \lambda \cdot \boldsymbol{p}_1 + (1-\lambda) \cdot \boldsymbol{p}_2$. We have $\boldsymbol{p} \in \mathbb{R}^\ell$ as $\boldsymbol{p}_1, \boldsymbol{p}_2 \in \mathbb{R}^\ell$. Let $\lambda_1 = \lambda$ and $\lambda_2 = 1 - \lambda$. We pick strategies $\sigma_1, \sigma_2 \in \Sigma^{\mathcal{M}}$ that achieve $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, i.e.,

$$\forall\, j \in \{1,2\}\colon\ \boldsymbol{p}_j \leq \mathrm{Ex}^{\mathcal{M}}_{\sigma_j}(\Phi).$$

With Lemma 3.8 it follows that

$$\boldsymbol{p} \;=\; \sum_{j=1}^{2} \lambda_j \cdot \boldsymbol{p}_j \;\leq\; \sum_{j=1}^{2} \lambda_j \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_j}(\Phi) \;=\; \mathrm{Ex}^{\mathcal{M}}_{[\sigma_1 \oplus_\lambda \sigma_2]}(\Phi).$$

Thus, strategy $[\sigma_1 \oplus_\lambda \sigma_2]$ achieves $\boldsymbol{p}$, implying $\boldsymbol{p} \in \mathit{Ach}(\Phi) \cap \mathbb{R}^\ell$. It follows that $\mathit{conv}(\mathit{Ach}(\Phi) \cap \mathbb{R}^\ell) \subseteq \mathit{Ach}(\Phi) \cap \mathbb{R}^\ell$. ■

Corollary 3.9 $\mathit{cl}(\mathit{Ach}(\Phi) \cap \mathbb{R}^\ell)$ is convex.

Proof. As shown in, e.g., [Sol15, Theorem 2.35], the closure of a convex set is always convex. The corollary thus follows from Theorem 3.5. ■

The downward closeness and convexity of the set of achievable points allows for a concise representation of subsets of $\mathit{Ach}(\Phi)$ by a (potentially finite) set of points $P \subseteq \mathbb{R}^\ell$.

Lemma 3.10 If $P \subseteq \mathit{Ach}(\Phi) \cap \mathbb{R}^\ell$, then $\mathit{down}(\mathit{conv}(P)) \subseteq \mathit{Ach}(\Phi)$.

Proof. By the definition of convex hull and downward hull we get that

$$\begin{aligned} P \subseteq \mathit{Ach}(\Phi) \;&\text{ implies }\; \mathit{conv}(P) \subseteq \mathit{conv}(\mathit{Ach}(\Phi)) \\ &\text{ implies }\; \mathit{down}(\mathit{conv}(P)) \subseteq \mathit{down}(\mathit{conv}(\mathit{Ach}(\Phi))). \end{aligned}$$

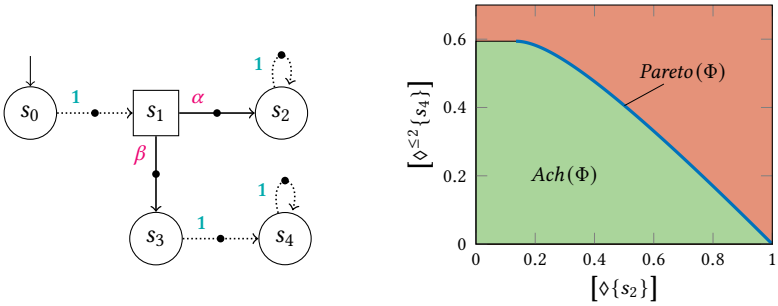
From Theorem 3.5 and Lemma 3.4 we have

$$\mathit{down}(\mathit{conv}(\mathit{Ach}(\Phi))) \;=\; \mathit{down}(\mathit{Ach}(\Phi)) \;=\; \mathit{Ach}(\Phi).$$ ■

Example 3.3 Recall that in Example 3.1 we have

$$\mathit{Ach}^{\mathcal{M}}(\Phi) \;=\; \mathit{down}(\mathit{conv}(\{\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3\})).$$

Lemma 3.11 For some MA $\mathcal{M}$ and objectives Φ with $\mathit{Pareto}(\Phi) \subseteq \mathbb{R}^\ell$ there is no *finite* set of points $P \subseteq \mathbb{R}^\ell$ with $\mathit{down}(\mathit{conv}(P)) = \mathit{cl}(\mathit{Ach}(\Phi))$.



(a) MA $\mathcal{M}$ (b) $Ach(\Phi)$

Figure 3.5: MA and achievable points as in the proof of Lemma 3.11

Proof. We consider the MA $\mathcal{M}$ from Figure 3.5a with the two probability objectives

$$\Phi = \langle \varphi_1, \varphi_2 \rangle = \left\langle \left[\Diamond\{s_2\}\right], \left[\Diamond^{\leq 2}\{s_4\}\right] \right\rangle.$$

Here, we denote the set of paths that reach a state in $G \subseteq S$ within $b \in \mathbb{R}_{\geq 0}$ time units by

$$\Diamond^{\leq b} G \coloneqq \left\{ \pi \in Paths^{\mathcal{M}}_{\text{inf}} \;\middle|\; \exists\, \hat{\pi} \in pref(\pi) \colon\ last(\hat{\pi}) \in G \text{ and } dur(\hat{\pi}) \leq b \right\}.$$

The resulting set of achievable points is illustrated in Figure 3.5b. Intuitively, to achieve Pareto optimality, the decision in state s_1 needs to depend on the exact time we spent in s_0.
Formally, for $a \in [0,2]$ let $\sigma^a \in \Sigma^{\mathcal{M}}$ be the (pure) strategy that chooses action β whenever we stayed at most a time units in s_0, i.e., $\sigma^a(s_0 \xrightarrow{t} s_1)(\beta) = [t \leq a]$. Moreover, let $\Xi \coloneqq \left\{\sigma^a \mid a \in [0,2]\right\}$ be the set of such strategies. We claim that for any strategy $\sigma \in \Sigma^{\mathcal{M}}$ there is some strategy in Ξ that either dominates σ or is equivalent on almost all paths, i.e.,

$$\begin{aligned} \forall\, \sigma \in \Sigma^{\mathcal{M}} \colon\ \exists\, \sigma^a \in \Xi \colon\ \Big(& \mathrm{Ex}^{\mathcal{M}}_{\sigma^a, s_I}(\Phi) \gtrsim \mathrm{Ex}^{\mathcal{M}}_{\sigma, s_I}(\Phi) \\ & \text{or } \forall\, \Pi \subseteq Paths^{\mathcal{M}}_{\text{inf}} \colon\ \Pr^{\mathcal{M}}_{\sigma^a, s_I}(\Pi) = \Pr^{\mathcal{M}}_{\sigma, s_I}(\Pi) \Big). \end{aligned} \tag{3.3}$$

In particular, for two distinct strategies $\sigma_1, \sigma_2 \in \Sigma^{\mathcal{M}}$ the randomized strategy $[\sigma_1 \oplus_\lambda \sigma_2]$ for $\lambda \in (0,1)$ as constructed in the proof of Theorem 3.5 is not equivalent to a strategy in Ξ because Ξ only contains pure strategies. $[\sigma_1 \oplus_\lambda \sigma_2]$ thus must be dominated by some strategy σ^a. Then, the lemma follows as for any finite P with

$down(conv(P)) \subseteq cl(Ach(\Phi))$ there is $a \in [0,2]$ with $\mathrm{Ex}^{\mathcal{M}}_{\sigma_a,s_I}(\Phi) \notin down(conv(P))$. To show Equation (3.3) consider a strategy $\sigma \in \Sigma^{\mathcal{M}}$ and let $a := -\ln(\Pr^{\mathcal{M}}_{\sigma,s_I}(\Diamond\{s_2\}))$, where ln is the natural logarithm. Under strategy σ^a we reach state s_2 whenever the time we spent in s_0 exceeds a. Thus,

$$\mathrm{Ex}^{\mathcal{M}}_{\sigma^a,s_I}(\varphi_1) \;=\; \Pr^{\mathcal{M}}_{\sigma^a,s_I}(\Diamond\{s_2\}) \;=\; e^{-a} \;=\; \Pr^{\mathcal{M}}_{\sigma,s_I}(\Diamond\{s_2\}) \;=\; \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\varphi_1).$$

By a similar argument we get $\Pr^{\mathcal{M}}_{\sigma^a,s_I}(\Diamond\{s_3\}) = \Pr^{\mathcal{M}}_{\sigma,s_I}(\Diamond\{s_3\})$. Moreover, we observe that a strategy can only choose to move to either s_2 or s_3. Thus,

$$\Pr^{\mathcal{M}}_{\sigma,s_I}(\Diamond^{\leq a}\{s_2\}) + \Pr^{\mathcal{M}}_{\sigma,s_I}(\Diamond^{\leq a}\{s_3\}) \;=\; \underbrace{\Pr^{\mathcal{M}}_{\sigma^a,s_I}(\Diamond^{\leq a}\{s_2\})}_{=0} + \Pr^{\mathcal{M}}_{\sigma^a,s_I}(\Diamond^{\leq a}\{s_3\}).$$

We distinguish two cases.

- If $\Pr^{\mathcal{M}}_{\sigma,s_I}(\Diamond^{\leq a}\{s_2\}) > 0$ then the above yields

  $$\Pr^{\mathcal{M}}_{\sigma,s_I}(\Diamond^{\leq a}\{s_3\}) \;<\; \Pr^{\mathcal{M}}_{\sigma^a,s_I}(\Diamond^{\leq a}\{s_3\}) \;=\; \Pr^{\mathcal{M}}_{\sigma^a,s_I}(\Diamond\{s_3\}) \;=\; \Pr^{\mathcal{M}}_{\sigma,s_I}(\Diamond\{s_3\}).$$

  While the probability to reach s_3 is equal under both strategies, s_3 is reached earlier when σ^a is considered. This increases the probability to reach s_4 in time, i.e., $\mathrm{Ex}^{\mathcal{M}}_{\sigma^a,s_I}(\varphi_2) > \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\varphi_2)$. Strategy σ^a thus dominates strategy σ.

- If $\Pr^{\mathcal{M}}_{\sigma,s_I}(\Diamond^{\leq a}\{s_2\}) = 0$ then σ satisfies $\sigma(s_0 \xrightarrow{t} s_1)(\alpha) = 0$ for almost all $t \in [0,a]$. This also holds for strategy σ^a. Since in all other cases σ^a selects action α which leads to s_2, we get

  $$\begin{aligned} &\Pr^{\mathcal{M}}_{\sigma,s_I}(\Diamond\{s_2\}) \\ \leq\; &\sup\left\{\Pr^{\mathcal{M}}_{\sigma',s_I}(\Diamond\{s_2\}) \mid \sigma' \in \Sigma^{\mathcal{M}}, \forall\, t \in [0,a] \colon\; \sigma(s_0 \xrightarrow{t} s_1)(\alpha) = 0\right\} \\ =\; &\Pr^{\mathcal{M}}_{\sigma^a,s_I}(\Diamond\{s_2\}). \end{aligned}$$

  In fact, we already know from above that equality has to hold. Therefore, σ must behave as σ^a on almost all sets of paths.

In both cases we have shown that the claim in Equation (3.3) holds which concludes the proof. ■

3.3 Multi-Objective Model Checking Queries

Multi-objective model checking is concerned with the analysis and verification of trade-offs between various objectives. To make this more concrete, we formalize three different kinds of multi-objective model checking queries that are commonly studied in the literature [EKVY08; FKNP11; FKP12; BBCF+14]. All queries consider an MA $\mathcal{M}$, an initial state s_I, and ℓ objectives $\Phi = \langle \varphi_1, \ldots, \varphi_\ell \rangle$ as input. We impose the following assumption for such inputs.

Assumption 3.1 $\forall\, j \in \{1..\ell\}\colon\ \mathrm{Ex}^{\mathcal{M}}_{\max}(\varphi_j) < \infty$.

The assumption is motivated by the observation that dealing with achievable points and expected values that concern $+\infty$ leads to technical and algorithmical issues. While for some inputs these issues can be resolved with minor additions to our algorithms, other inputs remain a challenge. Incorporating those inputs into our framework is left for future work. More details and examples of such inputs are given in Section 3.4.4. We mention that a similar assumption is also imposed in related works such as [FKNP11; FKP12].

MOA, multi-objective achievability

Problem 3.1: Multi-objective Achievability (MOA)

Input: $\mathcal{M}$, s_I, Φ satisfying Assumption 3.1, point $\boldsymbol{p} \in \mathbb{R}^\ell$

Output: $\begin{cases} \text{'YES'} & \text{if } \boldsymbol{p} \in Ach(\Phi) \\ \text{'NO'} & \text{if } \boldsymbol{p} \notin Ach(\Phi) \end{cases}$

MOQA, multi-objective quantitative achievability

Problem 3.2: Multi-objective Quantitative Achievability (MOQA)

Input: $\mathcal{M}$, s_I, Φ satisfying Assumption 3.1, values $p_2, \ldots, p_\ell \in \mathbb{R}$, precision $\eta \in \mathbb{R}_{\geq 0}$

Output: $\begin{cases} \langle l, u \rangle \in \mathbb{R} \times \mathbb{R} & \text{if } A \neq \emptyset,\ l \leq \sup A \leq u, \text{ and } u - l \leq \eta \\ \text{'NO'} & \text{if } A = \emptyset \end{cases}$

using $A \coloneqq \left\{ p \in \mathbb{R} \;\middle|\; \langle p, p_2, \ldots, p_\ell \rangle \in Ach(\Phi) \right\}$

Problem 3.3: Multi-objective Pareto (MOP)

Input: $\mathcal{M}$, s_I, Φ satisfying Assumption 3.1, precision $\eta \in \mathbb{R}_{\geq 0}$
Output: $\langle L, U \rangle \in 2^{(\mathbb{R}^\ell)} \times 2^{(\mathbb{R}^\ell)}$ with $L \subseteq Ach(\Phi) \cap \mathbb{R}^\ell \subseteq U$ and $\forall\, \boldsymbol{p} \in U\colon\ dist(\boldsymbol{p}, L) \leq \eta$

MOP, multi-objective Pareto

MOA queries yield the most basic variant of multi-objective model checking. They require to specify a threshold $p_j \in \overline{\mathbb{R}}$ for each objective φ_j, $j \in \{1..\ell\}$ with the interpretation that the expected value for φ_j shall be at least p_j. The answer to such a query is a binary 'YES' or 'NO', depending on whether the point $\boldsymbol{p} = \langle p_1, \ldots, p_\ell \rangle$ is achievable or not.

MOQA queries, on the other hand, may yield a numerical answer. Such queries are useful in cases where there are "hard" constraints—given by objectives $\varphi_2, \ldots, \varphi_\ell$ for which the expected values certainly have to satisfy the associated thresholds $p_2, \ldots, p_\ell$—and a "soft" constraint—given by objective φ_1 whose expected value shall be as large as possible without violating the hard constraints. The answer to a MOQA query is either (an approximation of) the largest possible value for the soft constraint or 'NO' in case the hard constraint can not be satisfied. The precision parameter $\eta \in \mathbb{R}_{\geq 0}$ allows us to trade accuracy of the resulting approximation against computational efficiency.

Finally, MOP queries provide a complete overview over all possible trade-offs between the objectives by yielding (an approximation of) the achievable points in $\mathbb{R}^\ell$. An approximation of $Pareto(\Phi) \cap \mathbb{R}^\ell$ can be derived by applying the definition of Pareto optimality (cf. Definition 3.2) or by noting that $Pareto(\Phi)$ is the smallest set P with $down(P) = cl(Ach(\Phi))$ (cf. Lemma 3.2). As for MOQA queries, a precision parameter η can be used to set the desired accuracy of the approximation.

The problem statements above are restricted to inputs and outputs over real numbers—not concerning $-\infty$ or $+\infty$. A consequence of these restrictions together with Assumption 3.1 is that it suffices to consider the real-valued achievable points, i.e., the points in $Ach(\Phi) \cap \mathbb{R}^\ell$ in order to answer all types of queries. We thus may restrict our analysis to strategies $\sigma \in \Sigma^{\mathcal{M}}$ with $\forall\, j \in \{1..\ell\}\colon\ \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\varphi_j) \in \mathbb{R}$.

Extended Problem Variants | Given algorithms for MOA, MOQA, and MOP queries, algorithms for extended variants of these problems with inputs and outputs over extended reals—including $-\infty$ and $+\infty$—can be derived. To show this, we employ the following simple lemma.

Lemma 3.12 Let $\boldsymbol{p} \in \overline{\mathbb{R}}^\ell$ with $\boldsymbol{p}(j) = -\infty$ for some $j \in \{1..\ell\}$. We have that

$\boldsymbol{p} \in Ach(\Phi)$ iff either $\ell = 1$ or $\ell > 1$ and $\boldsymbol{p}' \in Ach(\Phi')$ with

$$\begin{aligned} \boldsymbol{p}' &= \langle \boldsymbol{p}(1), \dots, \boldsymbol{p}(j-1), \boldsymbol{p}(j+1), \dots, \boldsymbol{p}(\ell) \rangle \text{ and} \\ \Phi' &= \langle \varphi(1), \dots, \varphi(j-1), \varphi(j+1), \dots, \varphi(\ell) \rangle. \end{aligned}$$

Proof. Let $\sigma \in \Sigma^{\mathcal{M}}$ be an arbitrary strategy. It holds that $-\infty \leq \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\varphi_j)$. If $\ell = 1$, then σ achieves $\boldsymbol{p} = \langle -\infty \rangle$. Otherwise, we have $\boldsymbol{p} \leq \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\Phi)$ iff $\boldsymbol{p}' \leq \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\Phi')$, meaning that σ achieves $\boldsymbol{p}$ iff σ achieves $\boldsymbol{p}'$. ∎

For an extended MOA query, there are two cases for an input point $\boldsymbol{p} \in \overline{\mathbb{R}}^{\ell} \setminus \mathbb{R}^{\ell}$.

- If $\boldsymbol{p}(j) = +\infty$ for some $j \in \{1..\ell\}$, $\boldsymbol{p}$ can not be achievable due to Assumption 3.1.
- If $\boldsymbol{p}(j) = -\infty$ for some $j \in \{1..\ell\}$ we can repeatedly apply Lemma 3.12 to clear all "$-\infty$"-entries of the given point $\boldsymbol{p}$ as well as the corresponding objectives. If this leaves us with a real-valued point, we can solve the remaining MOA query.

Similar arguments can be made for the input values $p_2, \dots, p_\ell$ of an extended MOQA query. Let $\overline{A} := \left\{ p \in \overline{\mathbb{R}} \,\middle|\, \langle p, p_2, \dots, p_\ell \rangle \in Ach(\Phi) \right\}$. From Assumption 3.1 we get $\sup \overline{A} < +\infty$. Thus, there are three cases left: $\sup \overline{A} \in \mathbb{R}$, $\overline{A} = \{-\infty\}$, and $\overline{A} = \emptyset$. In the two latter cases, the corresponding MOQA instance would yield the answer 'NO'. To further distinguish these two cases, one could solve a separate MOA query for the point $\langle -\infty, p_2, \dots, p_\ell \rangle$.

To also obtain the achievable points $Ach(\Phi) \setminus \mathbb{R}^{\ell}$ for an extended MOP query, we first note that such points do not have a "$+\infty$"-entry (due to Assumption 3.1) but they do have one or more "$-\infty$" entries. One way to obtain these achievable points is to run separate MOP queries for each possible subset

$$\Phi' \in \left\{ \langle \varphi_{j_1}, \dots, \varphi_{j_m} \rangle \mid \emptyset \neq \{j_1, \dots, j_m\} \subsetneq \{1..\ell\} \right\}$$

of objectives and fuse the results together by adding appropriate "$-\infty$"-entries to the resulting points—applying Lemma 3.12. Unfortunately, this requires to solve up to $2^{\ell} - 1$ additional sub-queries of lower dimension. That number can be reduced by only removing objectives φ_j where a strategy $\sigma \in \Sigma^{\mathcal{M}}$ with $\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\varphi_j) = -\infty$ exists.

3.4 A Sandwich Algorithm for Multi-Objective Model Checking

We now present an algorithm that allows to approximate the set $Ach(\Phi)$ in an iterative way, yielding a generic framework that can be steered towards efficiently solving the above-mentioned MOA, MOQA, and MOP queries. The algorithm is based on so-called

sandwich algorithms that are commonly used in the field of convex multi-objective optimization [SAC93; RDH11]. A first application to multi-objective probabilistic systems has been given in [FKP12] which deals with reachability- and total reward objectives on MDPs. In the following, we extend the algorithm towards *all* kinds of objectives for MA (and thus also for MDP).

3.4.1 The Weighted Sum Optimization Problem

The crux for our generic framework lies in forwarding objective-specific computations to a simpler sub-problem.

Problem 3.4: Weighted Sum Optimization (WSO)

Input: $\mathcal{M}$, s_I, Φ satisfying Assumption 3.1,
weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$, precision $\varepsilon_{\texttt{WSO}} \in \mathbb{R}_{\geq 0}$
Output: $\langle v_{\boldsymbol{w}}, \boldsymbol{p}_{\boldsymbol{w}} \rangle \in \mathbb{R} \times Ach(\Phi) \cap \mathbb{R}^\ell$ with
$\sup\left\{\boldsymbol{w} \cdot \mathrm{Ex}_\sigma^{\mathcal{M}}(\Phi) \;\middle|\; \sigma \in \Xi\right\} \leq v_{\boldsymbol{w}} \leq \boldsymbol{w} \cdot \boldsymbol{p}_{\boldsymbol{w}} + \varepsilon_{\texttt{WSO}} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}$
using $\Xi \coloneqq \{\sigma \in \Sigma^{\mathcal{M}} \mid \mathrm{Ex}_\sigma^{\mathcal{M}}(\Phi) \in \mathbb{R}^\ell\}$

WSO, weighted sum optimization

An instance for WSO includes a weight vector $\boldsymbol{w}$ that assigns a non-negative weight $\boldsymbol{w}(j)$ to each objective φ_j, $j \in \{1..\ell\}$. We then consider the weighted sum $\boldsymbol{w} \cdot \mathrm{Ex}_\sigma^{\mathcal{M}}(\Phi) = \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \mathrm{Ex}_\sigma^{\mathcal{M}}(\varphi_j)$ of expected values under different strategies σ. The goal is to find

- a value $v_{\boldsymbol{w}}$ that is an upper bound to the weighted sum for all considered strategies and
- an achievable point $\boldsymbol{p}_{\boldsymbol{w}}$ such that the weighted sum of the entries of $\boldsymbol{p}_{\boldsymbol{w}}$ is sufficiently close to $v_{\boldsymbol{w}}$ i.e., within a distance of $\varepsilon_{\texttt{WSO}} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}$.

As discussed before, we are only interested in the real-valued achievable points. We thus deliberately exclude strategies σ with $\mathrm{Ex}_\sigma^{\mathcal{M}}(\Phi) \in \overline{\mathbb{R}}^\ell \setminus \mathbb{R}^\ell$.

For now, we assume that a "black box" can solve instances for WSO. The details of this computation for arbitrary mixtures of total reachability reward objectives, reward-bounded objectives, and long-run average reward objectives are subject to the subsequent Chapters 4 to 6. To extend our framework towards new types of objectives it suffices to provide a solution method for the corresponding WSO queries as well as means to check the conditions from Lemma 3.13 which ensure that a valid solution $\langle v_{\boldsymbol{w}}, \boldsymbol{p}_{\boldsymbol{w}} \rangle$ exists.

Lemma 3.13 An instance to WSO has a solution if

1. $Ach(\Phi) \cap \mathbb{R}^\ell \neq \emptyset$, and
2. $\varepsilon_{\texttt{WSO}} > 0$ or $down(Ach(\Phi) \cap \mathbb{R}^\ell)$ is closed.

Proof. Due to Item 1 there is $\boldsymbol{p} \in Ach(\Phi) \cap \mathbb{R}^\ell$ and an achieving strategy $\sigma_{\boldsymbol{p}} \in \Sigma^{\mathcal{M}}$ with $\boldsymbol{p} \leq \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\boldsymbol{p}}}(\Phi)$. Since Assumption 3.1 holds we can also follow that $\mathrm{Ex}^{\mathcal{M}}_{\sigma_{\boldsymbol{p}}}(\Phi) \in \mathbb{R}^\ell$, i.e., $\Xi \neq \emptyset$.
As a solution to the WSO instance we may thus take $v_{\boldsymbol{w}} := \sup\left\{\boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) \;\middle|\; \sigma \in \Xi\right\} \in \mathbb{R}$ and derive a point $\boldsymbol{p}_{\boldsymbol{w}}$ as follows.

- If $\varepsilon_{\texttt{WSO}} > 0$, there is a strategy $\sigma_{\boldsymbol{w}} \in \Xi$ with

$$\sup\left\{\boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) \;\middle|\; \sigma \in \Xi\right\} - \boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\boldsymbol{w}}}(\Phi) \;\leq\; \underbrace{\varepsilon_{\texttt{WSO}} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}}_{>0}$$

  and we take $\boldsymbol{p}_{\boldsymbol{w}} := \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\boldsymbol{w}}}(\Phi) \in Ach(\Phi) \cap \mathbb{R}^\ell$

- If $\varepsilon_{\texttt{WSO}} = 0$, Item 2 yields that $down(Ach(\Phi) \cap \mathbb{R}^\ell)$ is closed. We get

$$\begin{aligned}
&\sup\left\{\boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) \;\middle|\; \sigma \in \Xi\right\} \\
=\;& \sup\left\{\boldsymbol{w} \cdot \boldsymbol{p} \;\middle|\; \boldsymbol{p} \in Ach(\Phi) \cap \mathbb{R}^\ell\right\} \\
=\;& \sup\left\{\boldsymbol{w} \cdot \boldsymbol{p} \;\middle|\; \boldsymbol{p} \in down(Ach(\Phi) \cap \mathbb{R}^\ell)\right\} \\
=\;& \max\left\{\boldsymbol{w} \cdot \boldsymbol{p} \;\middle|\; \boldsymbol{p} \in down(Ach(\Phi) \cap \mathbb{R}^\ell)\right\} \\
=\;& \max\left\{\boldsymbol{w} \cdot \boldsymbol{p} \;\middle|\; \boldsymbol{p} \in Ach(\Phi) \cap \mathbb{R}^\ell\right\} \\
=\;& \max\left\{\boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) \;\middle|\; \sigma \in \Xi\right\} \;=\; \boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\boldsymbol{w}}}(\Phi)
\end{aligned}$$

  for some strategy $\sigma_{\boldsymbol{w}} \in \Xi$ that induces the maximum. In this case we take $\boldsymbol{p}_{\boldsymbol{w}} := \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\boldsymbol{w}}}(\Phi) \in Ach(\Phi) \cap \mathbb{R}^\ell$.

In both cases we found a valid solution $\langle v_{\boldsymbol{w}}, \boldsymbol{p}_{\boldsymbol{w}} \rangle$ to the WSO instance. ∎

As sketched in Figure 3.6, a solution $\langle v_{\boldsymbol{w}}, \boldsymbol{p}_{\boldsymbol{w}} \rangle$ to a WSO instance allows us to divide $\mathbb{R}^\ell$ into

1. $down_{\mathbb{R}}(\{\boldsymbol{p}_{\boldsymbol{w}}\})$ (green area)—a set of points that are known to be achievable,
2. $\mathbb{R}^\ell \setminus halfsp(\boldsymbol{w}, v_{\boldsymbol{w}})$ (red area)—a set of points that are known to be not achievable
3. $halfsp(\boldsymbol{w}, v_{\boldsymbol{w}}) \setminus down_{\mathbb{R}}(\{\boldsymbol{p}_{\boldsymbol{w}}\})$ (white area including the blue line)—a set of points for which achievability can neither be inferred nor refuted.

Item 1 follows from Lemma 3.10 and $\boldsymbol{p}_{\boldsymbol{w}} \in Ach(\Phi)$. Item 2 can be inferred by noting that $\boldsymbol{p} \in \mathbb{R}^\ell \setminus halfsp(\boldsymbol{w}, v_{\boldsymbol{w}})$ yields $\boldsymbol{w} \cdot \boldsymbol{p} > v_{\boldsymbol{w}} \geq \sup\left\{\boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) \;\middle|\; \sigma \in \Xi\right\}$ with Ξ as in Problem 3.4. Therefore, Lemma 3.14 implies $\boldsymbol{p} \notin Ach(\Phi) \cap \mathbb{R}^\ell$. Since $\boldsymbol{p} \in \mathbb{R}^\ell$ we must have $\boldsymbol{p} \notin Ach(\Phi)$.

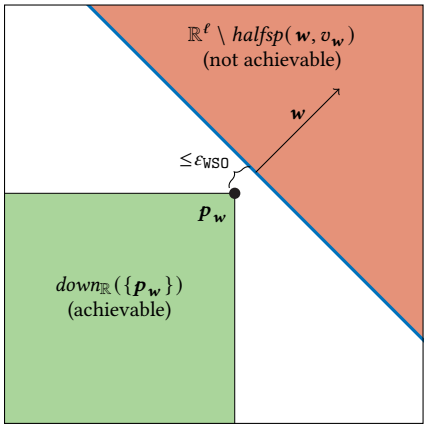


Figure 3.6: Illustration of WSO solutions.

Lemma 3.14 If Assumption 3.1 holds, then $\forall \boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell, \boldsymbol{p} \in Ach(\Phi) \cap \mathbb{R}^\ell$:

$$\boldsymbol{w} \cdot \boldsymbol{p} \;\leq\; \sup\left\{\boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) \;\middle|\; \sigma \in \Sigma^{\mathcal{M}} \text{ and } \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) \in \mathbb{R}^\ell\right\}.$$

Proof. Let $\sigma_{\boldsymbol{p}} \in \Sigma^{\mathcal{M}}$ be a strategy that achieves $\boldsymbol{p} \in Ach(\Phi) \cap \mathbb{R}^\ell$, i.e.,

$$\forall\, j \in \{1..\ell\}\colon\; -\infty \;<\; \boldsymbol{p}(j) \;\leq\; \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\boldsymbol{p}}}(\varphi_j) \;\leq\; \mathrm{Ex}^{\mathcal{M}}_{\max}(\varphi_j) \;<\; +\infty.$$

With the premise that $\boldsymbol{w}$ does not contain negative values, we obtain that

$$\boldsymbol{w} \cdot \boldsymbol{p} \;\leq\; \boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\boldsymbol{p}}}(\varphi_j) \;\leq\; \sup\left\{\boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) \;\middle|\; \sigma \in \Sigma^{\mathcal{M}} \text{ and } \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) \in \mathbb{R}^\ell\right\}. \qquad \blacksquare$$

The next lemma concerns the accuracy of WSO solutions. It states that the shortest distance between the boundary of the halfspace $halfsp(\boldsymbol{w}, v_{\boldsymbol{w}})$ (sketched by the blue line in Figure 3.6) and the point $\boldsymbol{p}_{\boldsymbol{w}}$ is bounded by the provided precision parameter $\varepsilon_{\texttt{WSO}}$.

Lemma 3.15 A solution $\langle v_{\boldsymbol{w}}, \boldsymbol{p}_{\boldsymbol{w}}\rangle$ to a WSO instance satisfies $dist(\boldsymbol{p}_{\boldsymbol{w}}, Q) \leq \varepsilon_{\texttt{WSO}}$ for the plane $Q := \{\boldsymbol{q}' \in \mathbb{R}^\ell \mid \boldsymbol{w} \cdot \boldsymbol{q}' = v_{\boldsymbol{w}}\}$.

Proof. If $\boldsymbol{w} = \boldsymbol{0}$, the claim holds trivially as $v_{\boldsymbol{w}} = 0$ is the only valid solution. We thus assume $\boldsymbol{w} \neq \boldsymbol{0}$. Due to $\boldsymbol{p}_{\boldsymbol{w}} \in Ach(\Phi) \cap \mathbb{R}^\ell$ and Lemma 3.14 we have $\boldsymbol{w} \cdot \boldsymbol{p}_{\boldsymbol{w}} \leq v_{\boldsymbol{w}}$. If $\boldsymbol{w} \cdot \boldsymbol{p}_{\boldsymbol{w}} = v_{\boldsymbol{w}}$ (i.e., $\boldsymbol{p}_{\boldsymbol{w}} \in Q$) we have $dist(Q, \boldsymbol{p}_{\boldsymbol{w}}) = 0$ and the claim again holds trivially. It thus remains to consider the case where $\boldsymbol{w} \cdot \boldsymbol{p}_{\boldsymbol{w}} < v_{\boldsymbol{w}}$

or—equivalently—$\boldsymbol{p}_w \notin halfsp(-\boldsymbol{w}, -v_w)$. Lemma 2.1 on page 20 yields

$$dist(\boldsymbol{p}_w, Q) \;=\; dist(\boldsymbol{p}_w, halfsp(-\boldsymbol{w}, -v_w)) \;=\; \frac{v_w - \boldsymbol{w} \cdot \boldsymbol{p}_w}{\sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}}.$$

The lemma follows since a solution $\langle v_w, \boldsymbol{p}_w \rangle$ for a WSO instance satisfies

$$v_w - \boldsymbol{w} \cdot \boldsymbol{p}_w \;\leq\; \varepsilon_{\texttt{WSO}} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}} \quad \text{iff} \quad \frac{v_w - \boldsymbol{w} \cdot \boldsymbol{p}_w}{\sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}} \;\leq\; \varepsilon_{\texttt{WSO}}.$$

■

3.4.2 Exploration of Achievable Points

sandwich algorithm

The idea of the *sandwich algorithm* is to explore the set of achievable points $Ach(\Phi)$ by solving multiple instances of WSO using different weight vectors.

Lemma 3.16 Let $\langle v_w^{(1)}, \boldsymbol{p}_w^{(1)} \rangle, \ldots, \langle v_w^{(n)}, \boldsymbol{p}_w^{(n)} \rangle$ be solutions to n WSO instances over the same MA $\mathcal{M}$, initial state s_I, and $\ell > 0$ objectives Φ (satisfying Assumption 3.1) and over (potentially different) weight vectors $\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(n)} \in (\mathbb{R}_{\geq 0})^\ell$. Then

$$down_{\mathbb{R}}\Big(conv\Big(\big\{\boldsymbol{p}_w^{(i)} \;\big|\; i \in \{1..n\}\big\}\Big)\Big) \;\subseteq\; Ach(\Phi) \cap \mathbb{R}^\ell \;\subseteq\; \bigcap_{i=1}^{n} halfsp(\boldsymbol{w}^{(i)}, v_w^{(i)}).$$

Proof. The lemma follows from Lemmas 3.10 and 3.14 and similar arguments as for Items 1 and 2 on page 82. ■

Example 3.4 Figure 3.7 illustrates a possible approximation of $Ach(\Phi)$ after solving three WSO instances with weight vectors $\boldsymbol{w}^{(1)} = \langle 2, 2 \rangle$, $\boldsymbol{w}^{(2)} = \langle 2, 0 \rangle$, and $\boldsymbol{w}^{(3)} = \langle 0, 1 \rangle$, respectively.
We have that all points in $down_{\mathbb{R}}(conv(P))$ with $P = \{\boldsymbol{p}_w^{(1)}, \boldsymbol{p}_w^{(2)}, \boldsymbol{p}_w^{(3)}\}$ (indicated by the green area) are known to be achievable. On the other hand, the points in $\mathbb{R}^\ell \setminus \bigcap_{\mathcal{H}}$ with $\mathcal{H} = \{halfsp(\boldsymbol{w}^{(i)}, v_w^{(i)}) \mid i \in \{1, 2, 3\}\}$ (indicated by the red area) are certainly not achievable. Achievability of the remaining points (white area) is still unknown.

The approach solves an increasing number of WSO instances to find more and more achievable points and—at the same time—prunes away unachievable points from the search space. Assuming a sensible selection of weight vectors—which we address later—this allows us to gather enough information to answer MOA, MOQA, and MOP queries as introduced in Section 3.3.

The approach is formalized in Algorithm 3.1. To unify the notations, the different

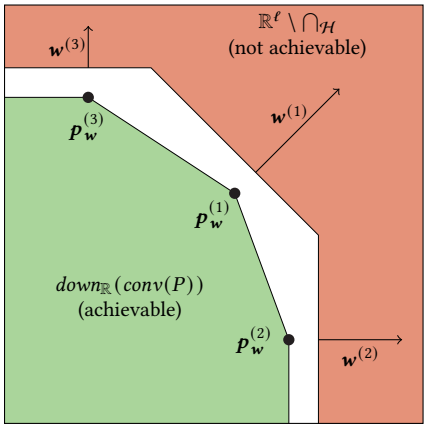


Figure 3.7: Illustration of approximation of $Ach(\Phi)$ with three WSO instances

types of inputs of MOA, MOQA, and MOP queries are encoded using a single information object.

Definition 3.3 (Query Information) $\mathfrak{Q}$ is called a *query information* if either

query information

- $\mathfrak{Q} \in \mathbb{R}^\ell$ (encoding a MOA query),
- $\mathfrak{Q} \in \{?\} \times \mathbb{R}^{\ell-1}$ (encoding a MOQA query), or
- $\mathfrak{Q} = ?$ (encoding a MOP query).

Moreover, Algorithm 3.1 takes as input a precision parameter η which for MOQA and MOP queries corresponds to the input precision. For MOA queries we can set η to an arbitrary[2] value. Algorithm 3.1 makes use of four functions:

- `solveWso`, which solves WSO as in Problem 3.4 for the given input,
- `ensureValidWsoInstances` (cf. Algorithm 3.2), which returns either
  - 'VALID' if subsequent calls to `solveWso` are known to be valid (i.e., satisfy Assumption 3.1 and have a solution, cf. Lemma 3.13),
  - a (potentially trivial) solution to the given query, or
  - an appropriate error message in case the query cannot be handled by our approach,

[2]The choice of η affects our heuristic that chooses precision parameters ε_{WSO} for inputs of WSO as discussed on page 88.

Input: MA $\mathcal{M}$, initial state s_I, objectives $\Phi = \langle \varphi_1, \ldots, \varphi_\ell \rangle$, precision $\eta \in \mathbb{R}_{\geq 0}$, query information $\mathfrak{Q}$
Output: An answer to the associated `MOA`, `MOQA`, or `MOP` query

```
1  answer ← ensureValidWsoInstances(M, s_I, Φ, η, 𝔔)
2  if answer ≠ 'VALID' then return answer
3  P ← ∅                                  // Collects achievable points found so far
4  H ← ∅                                  // Collects halfspaces that over-approximate Ach(Φ)
5  for i = 1, 2, 3, ... do
6      ⟨w, ε_WSO⟩ ← getNextWsoInstance(P, H, i, ℓ, η, 𝔔)
7      ⟨v_w, p_w⟩ ← solveWso(M, s_I, Φ, w, ε_WSO)
8      P ← P ∪ {p_w};   H ← H ∪ {halfsp(w, v_w)}
9      answer ← tryAnswer(P, H, η, 𝔔)
10     if answer ≠ 'UNKNOWN' then return answer
```

Algorithm 3.1: Approximating the set of achievable points

- `getNextWsoInstance` (cf. Algorithm 3.3), which heuristically decides which weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$ and precision parameter $\varepsilon_{\texttt{WSO}} \in \mathbb{R}_{\geq 0}$ should be considered for the next call to `solveWso`, and

- `tryAnswer` (cf. Algorithm 3.4), which checks whether the considered query can be answered based on the currently known approximation of $Ach(\Phi)$ and either returns that answer or 'UNKNOWN' in case the current approximation requires further refinement.

We discuss `ensureValidWsoInstances`, `getNextWsoInstance`, and `tryAnswer` below. The details for `solveWso` assuming various types of objectives are subject to Chapters 4 to 6.

Algorithm 3.1 proceeds as follows. It first checks necessary preconditions for being able to solve instances of `WSO` on the provided input (Lines 1 to 2). If the check is successful, the algorithm maintains two sets, namely a set P of achievable points that have been found so far and a set $\mathcal{H}$ of halfspaces that are known to be a superset of $Ach(\Phi) \cap \mathbb{R}^\ell$. In essence, the elements of P are used to enlarge the currently known set of achievable points whereas the elements of $\mathcal{H}$ are used to prune away unachievable points. Both sets P and $\mathcal{H}$ are populated by solving multiple `WSO` instances (Lines 6 to 8). After each step, we check in Line 9 whether the currently gathered information suffices to answer the given multi-objective query. If a conclusive answer can be given, it is returned in Line 10. Otherwise, the algorithm proceeds with the next iteration.

1 **function** `ensureValidWsoInstances`$(\mathcal{M}, s_I, \Phi, \eta, \mathfrak{Q})$
2 **if** Assumption 3.1 does not hold **then** *// Precondition for* `WSO`
3 **return** 'ERROR: *there is an objective* φ_j *with* $\mathrm{Ex}_{\max}(\varphi_j) = +\infty$'
4 **if** $Ach(\Phi) \cap \mathbb{R}^\ell = \emptyset$ **then** *// Item 1 of Lemma 3.13*
5 **if** $\mathfrak{Q} \in \mathbb{R}^\ell \cup (\{?\} \times \mathbb{R}^{\ell-1})$ **then return** 'NO' *// Answer* `MO(Q)A` *query*
6 **else if** $\mathfrak{Q} = ?$ **then return** $\langle \emptyset, \emptyset \rangle$ *// Answer* `MOP` *query*
 // Item 2 of Lemma 3.13
7 **if** $\eta = 0$ and $down(Ach(\Phi) \cap \mathbb{R}^\ell)$ is not closed **then**
8 **return** 'ERROR: *exact answer for non-closed* $down(Ach(\Phi) \cap \mathbb{R}^\ell)$ *not supported*'
9 **return** 'VALID'

Algorithm 3.2: Ensure that `WSO` instances as in Problem 3.4 can be solved

Theorem 3.17 Algorithm 3.1 maintains the invariant

$$down_{\mathbb{R}}(conv(P)) \subseteq Ach(\Phi) \cap \mathbb{R}^\ell \subseteq \bigcap_{\mathcal{H}}.$$

Proof. The theorem is a direct consequence of Lemma 3.16. ∎

Checking the Input | An implementation for `ensureValidWsoInstances` is given in Algorithm 3.2. It first checks Assumption 3.1 which is a requirement for any `WSO` input (cf. Problem 3.4). Moreover, the algorithm asserts Items 1 and 2 of Lemma 3.13 to ensure that a solution to `WSO` is guaranteed to exist. Observe that if Item 1 of Lemma 3.13 is violated, we can immediately derive an answer to all types of queries (Lines 5 to 6). If the check for Assumption 3.1 or Item 2 of Lemma 3.13 fails, the algorithm returns with an appropriate error message. If all checks pass, 'VALID' is returned in Line 9.

Selecting `WSO` Instances | In Line 6 of Algorithm 3.1 we could—in principle—choose any weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$ and precision parameter $\varepsilon_{\texttt{WSO}} \in \mathbb{R}_{\geq 0}$ for the next instance of `WSO` to be solved without violating the correctness of the sandwich algorithm. In fact, `MOA` queries could be answered with a single iteration of Algorithm 3.1 assuming a "perfect" choice of $\boldsymbol{w}$ and $\varepsilon_{\texttt{WSO}}$ in Line 6. However, such a perfect choice is not possible in practice as we do not have enough information on the trade-offs between objectives—yet. We therefore follow a heuristic approach to select weight vectors $\boldsymbol{w}$ and precision parameters $\varepsilon_{\texttt{WSO}}$ for the `WSO` instances to be solved in Algorithm 3.1. Our heuristic is similar to the heuristic considered in [FKP12]: the selected weight vector

is either $\mathbf{1}_j$ for some $j \in \{1..\ell\}$ or coincides with the normal vector of a halfspace of $down_{\mathbb{R}}(conv(P))$. In addition to [FKP12] we also deal with

- three or more objectives for MOP queries and
- approximative solution methods for WSO instances with precision $\varepsilon_{\texttt{WSO}} > 0$.

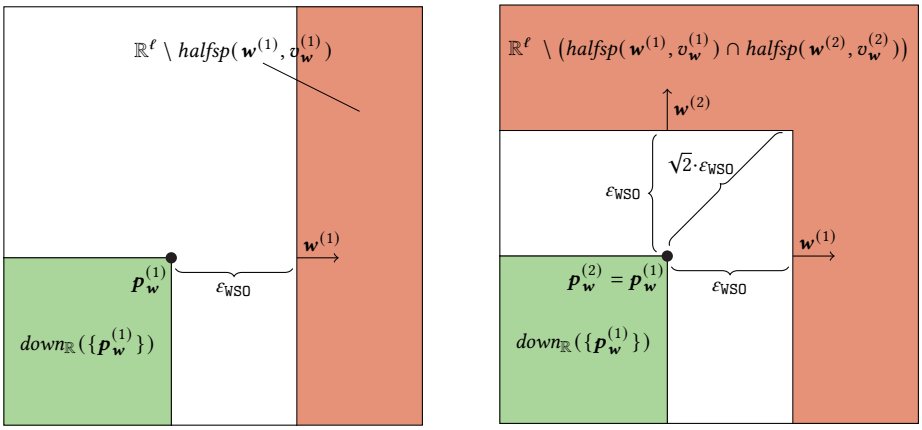
The first item is treated by lifting the ideas of [FKP12] for 2-dimensional queries to the more general case. Towards the latter item, we recall that choosing smaller precision parameters $\varepsilon_{\texttt{WSO}}$ for WSO means that the resulting point $\boldsymbol{p}_{\boldsymbol{w}}$ is closer to the boundary of the resulting halfspace $halfsp(\boldsymbol{w}, v_{\boldsymbol{w}})$ (cf. Lemma 3.15). This yields a tighter approximation of the achievable points with fewer iterations in Algorithm 3.1. On the other hand, solving WSO usually becomes harder with smaller $\varepsilon_{\texttt{WSO}}$. We thus observe a trade-off between

(i) solving fewer but harder WSO instances or

(ii) solving more but easier WSO instances.

We consider a hyper-parameter $\gamma \in (0, 1)$ within our heuristic: if γ is close to 0, we favor (i) and if γ is close to 1 the heuristic leans towards (ii). To accomplish this, we consider the precision $\varepsilon_{\texttt{WSO}} = \gamma \cdot \varepsilon_{\mathfrak{Q}}$ for WSO, where $\varepsilon_{\mathfrak{Q}}$ depends on the type of query to be solved:

- In case of MOA queries, we initially consider $\varepsilon_{\mathfrak{Q}} = \eta$, where η is the input precision of Algorithm 3.1. However, the point of interest might be very close to the boundary of $Ach(\Phi)$. In that case we also take the current gap between $down_{\mathbb{R}}(conv(P))$ and $\bigcap_{\mathcal{H}}$ into account. Intuitively, this gap coincides with the length of the largest line segment that is parallel to the selected weight vector $\boldsymbol{w}$ and whose endpoints lie on the boundaries of $down_{\mathbb{R}}(conv(P))$ and $\bigcap_{\mathcal{H}}$, respectively. We set $\varepsilon_{\mathfrak{Q}}$ to the minimum of η and the computed approximation gap. This way, we make sure that further progress is made in cases where the heuristic chooses the same weight vector $\boldsymbol{w}$ multiple times.
- For MOQA queries, a reference point $\boldsymbol{q} \in \bigcap_{\mathcal{H}} \setminus down_{\mathbb{R}}(conv(P))$ is computed first (details below). We then proceeding as for MOA queries.
- For MOP queries, it is not necessary to consider the current approximation gap since the points close to the boundary of $Ach(\Phi)$ are not of interest for the desired approximation. Instead, we always consider $\varepsilon_{\mathfrak{Q}} = \eta/\sqrt{\ell}$. It is necessary to divide by $\sqrt{\ell}$ to ensure that our approximation is sufficiently precise in case all candidate weight vectors (i.e., normal vectors of halfspaces of $down_{\mathbb{R}}(conv(P))$) have been processed already. We illustrate this in Example 3.5.



(a) 1st iteration (b) 2nd iteration

Figure 3.8: Illustration of obtained precision in the worst case

Example 3.5 Suppose that solving WSO for weight vector $\boldsymbol{w}^{(1)} = \langle 1, 0\rangle$ and precision $\varepsilon_{\mathtt{WSO}}$ yields the point $\boldsymbol{p}_{\boldsymbol{w}}^{(1)}$ and halfspace $halfsp(\boldsymbol{w}^{(1)}, v_{\boldsymbol{w}}^{(1)})$ as shown in Figure 3.8a. In the next step, we consider weight vector $\boldsymbol{w}^{(2)} = \langle 0, 1\rangle$, again with precision $\varepsilon_{\mathtt{WSO}}$. It might be the case that solving the associated instance for WSO yields the same point $\boldsymbol{p}_{\boldsymbol{w}}^{(2)} = \boldsymbol{p}_{\boldsymbol{w}}^{(1)}$ as before[a] and halfspace $halfsp(\boldsymbol{w}^{(2)}, v_{\boldsymbol{w}}^{(2)})$ as indicated in Figure 3.8b. Observe that the largest distance between the green area and some "unknown" point from the white area is $\sqrt{(\varepsilon_{\mathtt{WSO}})^2 + (\varepsilon_{\mathtt{WSO}})^2} = \sqrt{2} \cdot \varepsilon_{\mathtt{WSO}}$. In other words, the resulting approximation can answer a MOP query with input precision η iff $\varepsilon_{\mathtt{WSO}} \leq \eta/\sqrt{2}$.

[a] This can happen when the objectives are not conflicting with each other.

Our heuristic is given in Algorithm 3.3 which implements `getNextWsoInstance`. By default, we set the parameter γ to 0.5 in Line 2. In the first ℓ iterations of Algorithm 3.1 we only put a positive weight to one single objective φ_i for each $i \in \{1..\ell\}$ (Line 3). For the subsequent iterations, we compute a set of candidate weight vectors W with associated halfspaces $H_{\boldsymbol{v}}$ for $\boldsymbol{v} \in W$ in Lines 4 to 7. These halfspaces uniquely represent the polyhedron $down_{\mathbb{R}}(conv(P))$ and can be obtained with a combination of the Quickhull algorithm [BDH96] and Lemma 2.3 on page 22. Each $\boldsymbol{v} \in W$ is orthogonal to a facet of $down_{\mathbb{R}}(conv(P))$. In Lines 8 to 18 the algorithm chooses a weight vector $\boldsymbol{w}$ from the candidate set W depending on the query information $\mathfrak{Q}$.

function `getNextWsoInstance`$(P, \mathcal{H}, i, \ell, \eta, \mathfrak{Q})$

- $\gamma \leftarrow 0.5$ *// Heuristic parameter for approximation trade-offs*
- *// Maximize the i^{th} objective if we are in iteration $i \leq \ell$*
- **if** $i \leq \ell$ **then return** $\langle \mathbf{1}_i\,,\ \gamma \cdot \eta \rangle$
- *// Get representation of $down_{\mathbb{R}}(conv(P))$ (cf. Lemma 2.3 on page 22)*
- $\mathcal{H}_P \leftarrow$ a set of halfspaces such that $\bigcap_{\mathcal{H}_P} = down_{\mathbb{R}}(conv(P))$
- $W \leftarrow \{\boldsymbol{v} \in \mathbb{R}^\ell \mid \exists\, b \in \mathbb{R}\colon\ halfsp(\boldsymbol{v}, b) \in \mathcal{H}_P\}$ *// $W \subseteq (\mathbb{R}_{\geq 0})^\ell$ by Lemma 2.3*
- **for** $\boldsymbol{v} \in W$ **do**
  - $H_{\boldsymbol{v}} \leftarrow halfsp(\boldsymbol{v}, b_{\min})$ with $b_{\min} = \min\{b \in \mathbb{R} \mid halfsp(\boldsymbol{v}, b) \in \mathcal{H}_P\}$
- *// Branch on type of query*
- **if** $\mathfrak{Q} \in \mathbb{R}^\ell \cup (\{?\} \times \mathbb{R}^{\ell-1})$ **then** *// `MO(Q)A` query*
  - **if** $\mathfrak{Q} \in \mathbb{R}^\ell$ **then** $\boldsymbol{q} \leftarrow \mathfrak{Q}$ *// `MOA` query*
  - **else if** $\mathfrak{Q} \in \{?\} \times \mathbb{R}^{\ell-1}$ **then** *// `MOQA` query*
    - *// Get the "best" point we can currently hope to achieve*
    - $\mathcal{H}_{\mathfrak{Q}} \leftarrow \bigcup_{j=2}^{\ell} \{halfsp(-\mathbf{1}_j, -\mathfrak{Q}(j))\}$
    - $\boldsymbol{q} \leftarrow$ `solveLP`$(\mathcal{H} \cup \mathcal{H}_{\mathfrak{Q}}, \mathbf{1}_1)$
  - $\boldsymbol{w} \leftarrow \arg\max_{\boldsymbol{v} \in W}(dist(\boldsymbol{q}, H_{\boldsymbol{v}}))$ *// Halfspace with max. dist. to $\boldsymbol{q}$*
  - $\delta \leftarrow dist($`solveLP`$(\mathcal{H}, \boldsymbol{w}), H_{\boldsymbol{w}})$ *// Current approximation gap*
  - **return** $\langle \boldsymbol{w}\,,\ \gamma \cdot \min(\eta, \delta) \rangle$
- **else if** $\mathfrak{Q} \in$ **?** **then** *// `MOP` query*
  - *// Find the halfspace with maximal distance to $bd(\bigcap_{\mathcal{H}})$*
  - $\boldsymbol{w} \leftarrow \arg\max_{\boldsymbol{v} \in W}\left(dist(\texttt{solveLP}(\mathcal{H}, \boldsymbol{v}), H_{\boldsymbol{v}})\right)$
  - **return** $\langle \boldsymbol{w}\,,\ \gamma \cdot \eta/\sqrt{\ell} \rangle$

Algorithm 3.3: Obtain the next instance of `WSO` to be solved in Algorithm 3.1

maximize y such that:

$$\forall\, i \in \{1..\ell\}: \quad x_i \geq 0$$

$$\sum_{i=1}^{\ell} x_i = 1$$

$$\forall\, \boldsymbol{p} \in P: \quad y \leq \sum_{i=1}^{\ell} x_i \cdot (\boldsymbol{q}(i) - \boldsymbol{p}(i))$$

Figure 3.9: LP from [FKP12] to compute halfspaces of $down_{\mathbb{R}}(conv(P))$

For MOA and MOQA queries we first determine a reference point $\boldsymbol{q}$. For MOA, this is simply the point $\boldsymbol{q} = \mathfrak{Q}$ for which achievability is to be decided (Line 9). For MOQA we compute a point $\boldsymbol{q}$ that

- satisfies all thresholds given for the objectives $\varphi_2, \ldots, \varphi_\ell$ as encoded in the query information $\mathfrak{Q}$ and
- has the largest possible value for φ_1 given the current over-approximation $\bigcap_{\mathcal{H}}$.

Such a point is computed by solving the appropriate LP (cf. Problem 2.1 on page 27) in Line 12. We then choose the normal vector $\boldsymbol{w} \in W$ of a halfspace $H_{\boldsymbol{w}}$ with maximal distance[3] to the reference point $\boldsymbol{q}$ in Line 13. Next, we compute the largest distance d between a point in $\bigcap_{\mathcal{H}}$ and $H_{\boldsymbol{w}}$ in Line 14. This distance intuitively reflects the current approximation gap in direction $\boldsymbol{w}$. To ensure that this gap is in fact getting smaller when solving the next instance of WSO, we take $\gamma \cdot \min(\eta, d)$ as a precision parameter to WSO. Both, the resulting weight vector $\boldsymbol{w}$ and the precision are returned in Line 15.

For MOP queries, we take the weight vector $\boldsymbol{w} \in W$ such that the largest distance between some point in $\bigcap_{\mathcal{H}}$ and the halfspace $H_{\boldsymbol{w}}$ is maximal (Line 17). The resulting weight vector $\boldsymbol{w}$ and precision $\gamma \cdot \eta$ are returned in Line 18.

For MOA and MOQA queries computing the complete set of halfspaces $\mathcal{H}_P$ as indicated in Line 4 can actually be avoided. To this end, we can follow the approach of [FKP12] and replace the computation in Line 13 by solving the LP shown in Figure 3.9 instead. The LP considers P and $\boldsymbol{q}$ as in Algorithm 3.3 as well as $\ell + 1$ variables $y, x_1, \ldots, x_\ell$. A solution $val\colon \{y, x_1, \ldots, x_\ell\} \to \mathbb{R}$ yields a halfspace $halfsp(val(\boldsymbol{x}), val(\boldsymbol{x}) \cdot \boldsymbol{q} - val(y))$ with $val(\boldsymbol{x}) \coloneqq \langle val(x_1), \ldots, val(x_\ell)\rangle$. It can be shown that this is a halfspace of the polyhedron $down_{\mathbb{R}}(conv(P))$ with maximal distance to the reference point $\boldsymbol{q}$.

Answering Multi-objective Queries | The last step of our approach is to decide whether for given sets P and $\mathcal{H}$ we can conclude an answer to the considered multi-

[3] The distance between a point and a halfspace can be computed efficiently using Lemma 2.1 on page 20.

1 **function** `tryAnswer`$(P, \mathcal{H}, \eta, \mathfrak{Q})$
  // Get representation of $down_{\mathbb{R}}(conv(P))$ (cf. Lemma 2.3 on page 22)
2 $\mathcal{H}_P \leftarrow$ a set of halfspaces such that $\bigcap_{\mathcal{H}_P} = down_{\mathbb{R}}(conv(P))$
  // Branch on type of query
3 **if** $\mathfrak{Q} \in \mathbb{R}^\ell$ **then** *// `MOA` query*
4 **if** $\mathfrak{Q} \in \bigcap_{\mathcal{H}_P}$ **then return** 'YES'
5 **else if** $\mathfrak{Q} \notin \bigcap_{\mathcal{H}}$ **then return** 'NO'
6 **else if** $\mathfrak{Q} \in \{?\} \times \mathbb{R}^{\ell-1}$ **then** *// `MOQA` query*
7 $\mathcal{H}_{\mathfrak{Q}} \leftarrow \bigcup_{j=2}^{\ell} \{halfsp(-\mathbf{1}_j, -\mathfrak{Q}(j))\}$
8 $\boldsymbol{p}_{\min} \leftarrow$ `solveLP`$(\mathcal{H}_P \cup \mathcal{H}_{\mathfrak{Q}}, \mathbf{1}_1)$
9 $\boldsymbol{p}_{\max} \leftarrow$ `solveLP`$(\mathcal{H} \cup \mathcal{H}_{\mathfrak{Q}}, \mathbf{1}_1)$
10 **if** $\boldsymbol{p}_{\max}$ = 'Infeasible' **then return** 'NO'
11 **else if** $\boldsymbol{p}_{\min}, \boldsymbol{p}_{\max} \in \mathbb{R}^\ell$ **then** *// Both LPs feasible and bounded*
12 **if** $\boldsymbol{p}_{\max}(1) - \boldsymbol{p}_{\min}(1) \leq \eta$ **then return** $\langle \boldsymbol{p}_{\min}(1),\ \boldsymbol{p}_{\max}(1) \rangle$
13 **else if** $\mathfrak{Q} \in ?$ **then** *// `MOP` query*
14 **if** $\max \left\{ dist(\texttt{solveLP}(\mathcal{H}, \boldsymbol{v}), H) \;\middle|\; H = halfsp(\boldsymbol{v}, b) \in \mathcal{H}_P \right\} \leq \eta$ **then**
15 **return** $\left\langle \bigcap_{\mathcal{H}_P},\ \bigcap_{\mathcal{H}} \right\rangle$
16 **return** 'UNKNOWN'

Algorithm 3.4: Answer query based on current approximation from Algorithm 3.1

objective query. We make use of the invariant from Theorem 3.17:

$$down_{\mathbb{R}}(conv(P)) \;\subseteq\; Ach(\Phi) \cap \mathbb{R}^\ell \;\subseteq\; \bigcap_{\mathcal{H}}.$$

The details of this step are outlined in Algorithm 3.4 which implements `tryAnswer`. Similar to Algorithm 3.3, we consider a set of halfspaces $\mathcal{H}_P$ representing $down_{\mathbb{R}}(conv(P))$.

`MOA` queries can be answered if the considered point is either contained in $down_{\mathbb{R}}(conv(P)) = \bigcap_{\mathcal{H}_P}$ and thus known to be achievable (Line 4) or not contained in $\bigcap_{\mathcal{H}}$ and thus known to be not achievable (Line 5).

For `MOQA` queries we consider two points $\boldsymbol{p}_{\min} \in down_{\mathbb{R}}(conv(P))$ and $\boldsymbol{p}_{\max} \in \bigcap_{\mathcal{H}}$ that satisfy the thresholds imposed on objectives $\varphi_2, \ldots, \varphi_\ell$ and—among all such points—have the largest possible value for φ_1. These points are computed by solving appropriate `LP` instances in Lines 8 and 9. In case no such point $\boldsymbol{p}_{\max}$ exists, we can conclude that the thresholds can not be achieved and thus the answer to the query is 'NO' (Line 10). On the other hand, if both points $\boldsymbol{p}_{\min}$ and $\boldsymbol{p}_{\max}$ exist we can consider their 1^{st} entry to obtain bounds on the requested optimal value for φ_1. If the gap between these bounds is below η we return them in Line 12.

Finally, for MOP queries we consider for each halfspace in $\mathcal{H}_P$ the largest possible distance to a point in $\bigcap_{\mathcal{H}}$. If all these distances are below η, the sets $\bigcap_{\mathcal{H}_P}$ and $\bigcap_{\mathcal{H}}$ will be returned in line Line 15.

If no answer can be given based on the current sets P and $\mathcal{H}$ we return 'UNKNOWN' in Line 16.

Theorem 3.18 If Algorithm 3.1 terminates (without an error) it provides a correct answer to the encoded multi-objective query.

Proof. The theorem is a consequence of Theorem 3.17 and our explanations above.■

Remark 3.1 (Strategy Generation) We can tweak Algorithm 3.1 to also generate an achieving strategy for a given MOA or MOQA query. For that, we assume that `solveWso` in Line 7 also returns a strategy $\sigma_{\boldsymbol{w}}$ that achieves the point $\boldsymbol{p}_{\boldsymbol{w}}$. We store the generated strategies in a map $\mathfrak{S}\colon \mathbb{R}^\ell \to \Sigma^{\mathcal{M}}$. In case of a MOA query, as soon as we showed achievability of the queried point $\mathfrak{Q}$, i.e., $\mathfrak{Q} \in down_{\mathbb{R}}(conv(P))$ we can find a distribution $\mu \in Dist(P)$ such that $\mathfrak{Q} \leq \sum_{\boldsymbol{p}\in P} \mu(\boldsymbol{p}) \cdot \boldsymbol{p}$. A strategy σ that achieves $\mathfrak{Q}$ then intuitively looks as follows. Initially, σ chooses a point $\boldsymbol{p} \in P$ probabilistically according to distribution μ. Once a point $\boldsymbol{p}$ is chosen, σ mimics the strategy $\mathfrak{S}(\boldsymbol{p})$ that achieves $\boldsymbol{p}$. To define such a strategy σ more formally, we may generalize the strategy $[\sigma_1 \oplus_\lambda \sigma_2]$ from the proof of Theorem 3.5 to multiple input strategies $\sigma_1, \ldots, \sigma_n$. The resulting strategy σ satisfies $\mathfrak{Q} \leq \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\Phi)$.
We stress that even if $\mathfrak{S}(\boldsymbol{p}) \in \Sigma^{\mathcal{M}}_{\mathrm{PM}}$ for all $\boldsymbol{p} \in P$, the constructed strategy σ could be neither pure nor memoryless as it needs to memorize the outcome of the initial probabilistic choice.
For a MOQA query, a strategy that achieves $\boldsymbol{p}_{\max}$ as in Algorithm 3.4 can be constructed similarly.

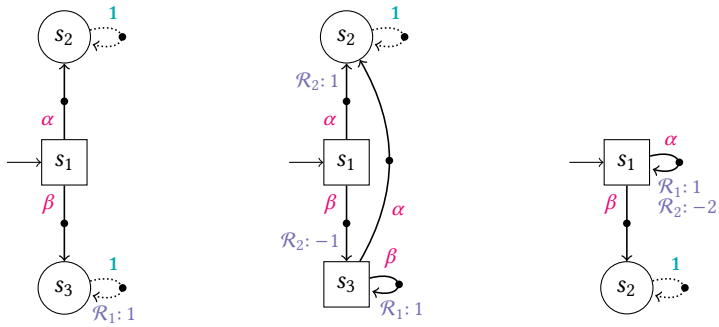
3.4.3 On Completeness of the Approach

The sandwich algorithm with the heuristics as explained above might not terminate in general. Intuitively, the cases where we need to determine achievability of the points that lie exactly on the boundary $bd(Ach(\Phi))$ are problematic.

In light of Lemma 3.11, the problem becomes obvious for MOP queries with precision $\eta = 0$: there might not be a finite set of points P with $down_{\mathbb{R}}(conv(P)) = Ach(\Phi) \cap \mathbb{R}^\ell$.

A similar issue arises for MOA and MOQA queries whenever we need to check achievability for a point $\boldsymbol{p} \in bd(Ach(\Phi))$. In that case, with an increasing number of iterations in Algorithm 3.1, the boundaries of $down_{\mathbb{R}}(conv(P))$ and $\bigcap_{\mathcal{H}}$ can get arbitrarily close to $\boldsymbol{p}$ but might never actually contain $\boldsymbol{p}$.

However, for many types of objectives—in particular the objectives considered



(a) $\mathcal{M}_1$ (b) $\mathcal{M}_2$ (c) $\mathcal{M}_3$

Figure 3.10: Three example MAs with reward assignments demonstrating objectives with infinite value (cf. Example 3.6).

in Chapters 4 to 6—termination can be ensured since a finite set of points P with $down_{\mathbb{R}}(conv(P)) = Ach(\Phi) \cap \mathbb{R}^{\ell}$ exists. For example, to solve WSO instances with precision parameter $\varepsilon_{\texttt{WSO}} = 0$ considering only total reward objectives, it suffices to consider only pure memoryless strategies $\sigma \in \Sigma^{\mathcal{M}}_{\text{PM}}$. There thus are at most $|\Sigma^{\mathcal{M}}_{\text{PM}}| = \prod_{s \in S} |\Delta(s)| \leq |Act|^{|S|}$ different solutions to WSO which means that at some point no new candidate halfspaces will be considered in Algorithm 3.3, eventually triggering termination of Algorithm 3.1.

On practical inputs, the sandwich approach often requires only a few iterations to answer the provided multi-objective query—as we will see in Chapter 9.

3.4.4 Beyond our Assumptions: Dealing with Infinity

We now discuss MAs $\mathcal{M}$ and objectives $\Phi = \langle \varphi_1, \ldots, \varphi_\ell \rangle$ that do not fulfill Assumption 3.1, i.e., $\exists\, j \in \{1..\ell\} \colon \text{Ex}^{\mathcal{M}}_{\max}(\varphi_j) = \infty$. To simplify the notations, we assume that—by potentially re-ordering the objectives—we have that $j = 1$, i.e., $\text{Ex}^{\mathcal{M}}_{\max}(\varphi_1) = \infty$.

Example 3.6 Consider the three MAs $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$ with associated reward assignments $\mathcal{R}_1$ and $\mathcal{R}_2$ as shown in Figure 3.10.
For $\mathcal{M}_1$ and the objectives $\Phi_1 := \langle tot(\mathcal{R}_1), [\Diamond s_2] \rangle$ observe that a strategy that selects α at s_1 with probability $a \in [0, 1)$ achieves point $\langle \infty, a \rangle$. However, if a strategy chooses α with probability 1, no reward for the first objective is collected

and thus $\langle 0, 1\rangle$ is achieved. Therefore,

$$Ach^{\mathcal{M}_1}(\Phi_1) \;=\; \big\{\langle b, a\rangle \in \overline{\mathbb{R}}^2 \;\big|\; b \in \mathbb{R}, a < 1\big\} \cup down\big(\big\{\langle 0, 1\rangle\big\}\big).$$

Next, consider $\mathcal{M}_2$ and the objectives $\Phi_2 \coloneqq \langle tot(\mathcal{R}_1) \cdot [\Diamond s_2], tot(\mathcal{R}_2)\rangle$. Here, we may not achieve value ∞ for the first objective (paths that stay in s_3 forever are not contained in $\Diamond s_2$ and thus get assigned value 0) but we can still achieve any arbitrary value $b \in \mathbb{R}$ by taking action β at s_3 sufficiently often before moving to s_2 via α. If we choose action α at s_1, we will not collect any reward for the first objective but will achieve value 1 for the second objective. If we randomize the decision at s_1 by choosing α with a very high probability, the value for the second objective gets arbitrarily close to 1 and—as long as there is a positive probability to choose β at s_1—we still have a chance to move to s_3 and collect arbitrary many reward for the first objective. We thus get

$$Ach^{\mathcal{M}_2}(\Phi_2) \;=\; \big\{\langle b, a\rangle \in \mathbb{R}^2 \;\big|\; b < \infty, a < 1\big\} \cup down\big(\big\{\langle 0, 1\rangle\big\}\big).$$

Finally, for $\mathcal{M}_3$ and the objectives $\Phi_3 \coloneqq \langle tot(\mathcal{R}_1), tot(\mathcal{R}_2)\rangle$ it is only possible to achieve value b for the first objective if we also collect value $-2 \cdot b$ reward for the second objective. We get

$$Ach^{\mathcal{M}_3}(\Phi_3) \;=\; down\big(\big\{\langle b, -2 \cdot b\rangle \;\big|\; b \in \mathbb{R}_{\geq 0}\big\}\big).$$

In the first two examples we observe that "infinitely good" strategies for the first objective exist that—at the same time—do not yield "infinitely bad" results for the other objective. Using the construction from the proof of Theorem 3.5, we may randomly combine such a strategy with another strategy that only focuses on optimizing the other objectives. In fact, it suffices to assign a very small probability to the "infinitely good for φ_1"-strategy to still achieve arbitrarily high values for φ_1. This allows us to *almost* achieve the same values for the objectives $\varphi_2, \ldots, \varphi_\ell$ that we would obtain when disregarding φ_1 entirely. The following lemma formalizes this insight—allowing us to construct $cl(Ach(\Phi))$ from $cl(Ach(\langle\varphi_2, \ldots, \varphi_\ell\rangle))$.

Lemma 3.19 If $\forall\, \sigma \in \Sigma^{\mathcal{M}}\colon\ \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\varphi_1) > -\infty$ and there is $a_{\min} \in \mathbb{R}$ such that

$$\sup\big\{\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\varphi_1) \;\big|\; \sigma \in \Sigma^{\mathcal{M}}, \forall\, j \in \{1..\ell\}\colon\ \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\varphi_j) \geq a_{\min}\big\} \;=\; \infty,$$

then $cl(Ach(\Phi)) = \overline{\mathbb{R}} \times cl(Ach(\langle\varphi_2, \ldots, \varphi_\ell\rangle))$.

Proof. Let $\Phi' \coloneqq \langle\varphi_2, \ldots, \varphi_\ell\rangle$.

Direction "$\subseteq$" | For $\boldsymbol{p} = \langle p_1, \ldots, p_\ell \rangle \in cl(Ach(\Phi))$ we clearly have $p_1 \in \overline{\mathbb{R}}$ and $\langle p_2, \ldots, p_\ell \rangle \in cl(Ach(\Phi'))$, yielding $\boldsymbol{p} \in \overline{\mathbb{R}} \times cl(Ach(\Phi'))$.

Direction "$\supseteq$" | Let $\boldsymbol{p} = \langle p_1, \ldots, p_\ell \rangle \in \overline{\mathbb{R}} \times cl(Ach(\Phi'))$ and let $N = \bigtimes_{i=1}^{\ell} N_i$ be an arbitrary neighborhood of $\boldsymbol{p}$. We need to show that $N \cap Ach(\Phi) \neq \emptyset$. Without loss of generality we assume that N does not contain any points on its boundary, i.e., $N \cap bd(N) = \emptyset$[a].
Since $\boldsymbol{p}' \coloneqq \langle p_2, \ldots, p_\ell \rangle \in cl(Ach(\Phi'))$ and $N' \coloneqq \bigtimes_{i=2}^{\ell} N_i$ is a neighborhood of $\boldsymbol{p}'$, N' must contain an achievable point $\boldsymbol{q}' \coloneqq \langle q_2, \ldots, q_\ell \rangle \in N' \cap Ach(\Phi')$. Let $\sigma_{\{2..\ell\}} \in \Sigma^{\mathcal{M}}$ be a strategy achieving $\boldsymbol{q}'$, i.e., $\boldsymbol{q}' \leq \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\{2..\ell\}}}(\Phi')$. Observe $\mathrm{Ex}^{\mathcal{M}}_{\sigma_{\{2..\ell\}}}(\varphi_1) > -\infty$ due to the precondition of the lemma.
Furthermore, there is some $\delta \in \mathbb{R}_{>0}$ such that $\boldsymbol{q}' - \mathbf{1} \cdot \delta \in N'$ because $\boldsymbol{q}' \notin bd(N)$ and there is also some $q_1 \in N_1 \cap \mathbb{R}$. For the point $\boldsymbol{q} \coloneqq \langle q_1, q_2 - \delta, \ldots, q_2 - \delta \rangle$ we have that $\boldsymbol{q} \in N$. We now construct a strategy achieving $\boldsymbol{q}$. If $\mathrm{Ex}^{\mathcal{M}}_{\sigma_{\{2..\ell\}}}(\varphi_1) = \infty$ then $\boldsymbol{q}$ is achieved by strategy $\sigma_{\{2..\ell\}}$. Otherwise, consider

$$\lambda \coloneqq \min\left\{ \frac{\delta}{q_j - a_{\min}} \;\middle|\; j \in \{2..\ell\} \text{ and } a_{\min} < q_j < \infty \right\} \cup \left\{ \frac{1}{2} \right\} \quad \text{and}$$

$$b \coloneqq \frac{q_1 - (1-\lambda) \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\{2..\ell\}}}(\varphi_1)}{\lambda}$$

Here, $a_{\min}$ is as in the precondition of the lemma. The $\frac{1}{2}$ in the above expression for λ is an arbitrary value from the interval $(0, 1)$ which is added to ensure that λ is set to a value in $(0, 1)$. By the assumption from the lemma, there must be a strategy σ_1 that achieves the point $\langle b, a_{\min}, \ldots, a_{\min} \rangle$, i.e., $\langle b, a_{\min}, \ldots, a_{\min} \rangle \leq \mathrm{Ex}^{\mathcal{M}}_{\sigma_1}(\Phi)$. We now consider the strategy $[\sigma_1 \oplus_\lambda \sigma_{\{2..\ell\}}]$ using the construction from the proof of Theorem 3.5. By Lemma 3.8 we get

$$\begin{aligned} \mathrm{Ex}^{\mathcal{M}}_{[\sigma_1 \oplus_\lambda \sigma_{\{2..\ell\}}]}(\varphi_1) &= \lambda \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_1}(\varphi_1) + (1-\lambda) \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\{2..\ell\}}}(\varphi_1) \\ &\geq \lambda \cdot b + (1-\lambda) \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\{2..\ell\}}}(\varphi_1) = q_1 \end{aligned}$$

and for all $j \in \{2..\ell\}$ we have

$$\begin{aligned}
\mathrm{Ex}^{\mathcal{M}}_{[\sigma_1 \oplus_\lambda \sigma_{\{2..\ell\}}]}(\varphi_j) &= \lambda \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_1}(\varphi_j) + (1-\lambda) \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma_{\{2..\ell\}}}(\varphi_j) \\
&\geq \lambda \cdot a_{\min} + (1-\lambda) \cdot q_j \\
&= \begin{cases} q_j & \text{if } q_j \in \{-\infty, +\infty\} \\ q_j + \lambda \cdot (a_{\min} - q_j) & \text{if } q_j \in \mathbb{R} \end{cases} \\
&\geq \begin{cases} q_j & \text{if } q_j \in \{-\infty, +\infty\} \text{ or } a_{\min} \geq q_j \\ q_j + \frac{\delta}{q_j - a_{\min}} \cdot (a_{\min} - q_j) & \text{if } a_{\min} < q_j < \infty \end{cases} \\
&\geq q_j - \delta.
\end{aligned}$$

Thus, strategy $[\sigma_1 \oplus_\lambda \sigma_{\{2..\ell\}}]$ achieves $\boldsymbol{q} \in N$ which concludes the proof. ■

[a] If this assumption is not valid, we may show $\tilde{N} \cap Ach(\Phi) \neq \emptyset$ for the neighborhood $\tilde{N} := N \setminus bd(N) \subseteq N$ of $\boldsymbol{p}$ instead which then also implies $N \cap Ach(\Phi) \neq \emptyset$.

Example 3.7 Recall $\mathcal{M}_i$ and Φ_i for $i \in \{1,2,3\}$ as in Example 3.6. For $\mathcal{M}_1$ and Φ_1 the preconditions in Lemma 3.19 hold with $a_{\min} = 0$. With $cl(Ach^{\mathcal{M}_1}([\Diamond s_2])) = down(\{\langle 1 \rangle\})$ we get

$$cl(Ach^{\mathcal{M}_1}(\Phi_1)) = down(\{\langle \infty, 1 \rangle\}).$$

Similar observations can be made for $\mathcal{M}_2$ and Φ_2. For $\mathcal{M}_3$ and Φ_3 the preconditions in Lemma 3.19 do not hold, in particular:

$$\forall\, a \leq 0\colon \ \sup\left\{\mathrm{Ex}^{\mathcal{M}_3}_{\sigma}(tot(\mathcal{R}_1)) \;\middle|\; \sigma \in \Sigma^{\mathcal{M}}, \mathrm{Ex}^{\mathcal{M}_3}_{\sigma}(tot(\mathcal{R}_2)) \geq a\right\} = \frac{a}{2} \neq \infty.$$

Indeed, we have $cl(Ach^{\mathcal{M}_3}(tot(\mathcal{R}_2))) = down(\{\langle 0 \rangle\})$ but $cl(Ach^{\mathcal{M}_3}(\Phi_3)) \neq down(\{\langle \infty, 0 \rangle\})$.

Lemma 3.19 yields an approach to compute (or approximate) $cl(Ach(\Phi))$ in cases where Assumption 3.1 is violated. This is done by computing $\mathcal{I} = \{j_1, \ldots, j_{\ell'}\} := \left\{j \in \{1..\ell\} \;\middle|\; \mathrm{Ex}^{\mathcal{M}}_{\max}(\varphi_j) < \infty\right\}$ and obtaining $cl(Ach(\Phi'))$ for $\Phi' := \langle \varphi_{j_1}, \ldots, \varphi_{j_{\ell'}} \rangle$, using Algorithm 3.1. Then, $cl(Ach(\Phi))$ is constructed by lifting $cl(Ach(\Phi'))$ to ℓ dimensions, i.e.,

$$cl(Ach(\Phi)) = \left\{\langle p_1, \ldots, p_\ell \rangle \in \mathbb{R}^\ell \;\middle|\; \langle p_{j_1}, \ldots, p_{j_{\ell'}} \rangle \in cl(Ach(\Phi'))\right\}.$$

We mention that with this approach it is still not possible to decide achievability for points that lie exactly on the boundary of $Ach(\Phi)$. Moreover, it is necessary to check the preconditions of Lemma 3.19 first. Handling inputs where these preconditions do not hold—such as $\mathcal{M}_3$ and Φ_3 from Examples 3.6 and 3.7—is left for future work.

Chapter Summary

- ➜ The *achievable points* for an MA with multiple objectives indicate the expected values of the objectives that can be achieved simultaneously under the same strategy—revealing potential trade-offs between the objectives.
- ➜ The set of achievable points is always *downward closed* and *convex*, but not necessarily closed.
- ➜ The *sandwich algorithm* yields sound and arbitrarily precise approximations of the set of achievable points. It requires to solve multiple instances of the weighted sum optimization problem (`WSO`).
- ➜ The results from this chapter—including the sandwich algorithm—are applicable to *all* types of objectives. It remains to provide efficient solution methods for `WSO` which is outlined in the subsequent Chapters 4 to 6.

—Chapter 4—

Sound Approximations for Expected Total Rewards

Outlook | Our goal in this chapter is to solve instances of WSO (Problem 3.4 on page 81) concerning only total reachability reward objectives (Definition 2.27 on page 49). Section 4.1 shows that in this case WSO can be reduced to the analysis of (single-objective) expected total rewards on MDPs. In Section 4.2 we discuss established results on the expected total reachability reward problem for MDPs. Section 4.3 presents the classical value iteration algorithm and its shortcomings concerning the accuracy of solutions. Afterwards, we present *sound value iteration*—an extension of the value iteration algorithm that provides sound precision guarantees. We explain the algorithm first for DTMCs in Section 4.4 and then for MDPs in Section 4.5. A survey of related approaches is given in Section 4.6.

Origins | Section 4.1 summarizes results from [FKP12; 13]. While our presentation is new, we emphasize that the underlying insights of Section 4.1 can be associated to the author's master's thesis [1] and are presented in this dissertation for the sake of completeness. The contents of Sections 4.2 and 4.3 can be considered folklore (e.g., [Put94]). Sections 4.4 and 4.5 are revised from [5]. The overview of related work in Section 4.6 is new.

4.1 From WSO to finite single total rewards on MDP

Set-up | We consider a Markov automaton $\mathcal{M} = \langle S, Act, \Delta, \mathbf{P} \rangle$, with dedicated initial state $s_I \in S$, a list of $\ell > 0$ *convergent* total reachability reward objectives $\Phi = \langle tot(\mathcal{R}_1, G_1), \ldots, tot(\mathcal{R}_\ell, G_\ell) \rangle$, a weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$, and a precision parameter $\varepsilon_{\texttt{WSO}} \geq 0$ which together form an instance for WSO.

The goal is to reduce the given instance for WSO to the computation of (single-objective) expected total rewards on MDPs. The reduction is done in various simplification steps as outlined below.

4.1.1 From Total Reachability Reward to Total Reward

The first step is to convert total reachability reward objectives $tot(\mathcal{R}_j, G_j)$ to simpler total reward objectives $tot(\mathcal{R}'_j)$. The idea is to encode visits of goal state sets G_j into the state space of the MA.

goal unfolding

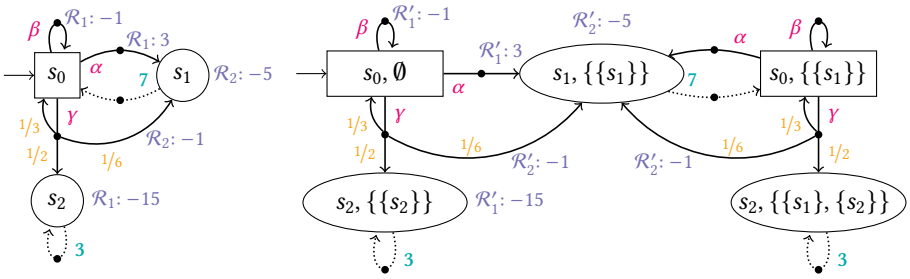
Definition 4.1 (Goal Unfolding) Let $\mathfrak{G} = \{G_1, \ldots, G_\ell\} \subseteq 2^S$ be a set of ℓ sets of goal states for MA $\mathcal{M} = \langle S, Act, \Delta, \mathbf{P}\rangle$. We define the auxiliary function $succ\colon \mathfrak{G} \times S \to \mathfrak{G}$ by $succ(\mathfrak{g}, s') \coloneqq \mathfrak{g} \cup \{G \in \mathfrak{G} \mid s' \in G\}$.
The *goal unfolding* of $\mathcal{M}$ with respect to $\mathfrak{G}$ is given by the MA $unf_{\mathfrak{G}}(\mathcal{M}) = \big\langle S \times 2^{\mathfrak{G}}, Act, \Delta_{\mathfrak{G}}, \mathbf{P}_{\mathfrak{G}}\big\rangle$, where for all $s, s' \in S$ and $\mathfrak{g}, \mathfrak{g}' \subseteq \mathfrak{G}$, we set

- $\Delta_{\mathfrak{G}}(\langle s, \mathfrak{g}\rangle) \coloneqq \Delta(s)$,
- if $s \in MS^{\mathcal{M}}$ then $\mathbf{P}_{\mathfrak{G}}(\langle s, \mathfrak{g}\rangle, \langle s', \mathfrak{g}'\rangle) \coloneqq \mathbf{P}(s, s') \cdot [\mathfrak{g}' = succ(\mathfrak{g}, s')]$, and
- if $s \in PS^{\mathcal{M}}$ then $\mathbf{P}_{\mathfrak{G}}(\langle s, \mathfrak{g}\rangle, \alpha, \langle s', \mathfrak{g}'\rangle) \coloneqq \mathbf{P}(s, \alpha, s') \cdot [\mathfrak{g}' = succ(\mathfrak{g}, s')]$ for all $\alpha \in \Delta(s)$.

Let $\pi = \langle s_0, \mathfrak{g}_0\rangle \xrightarrow{\kappa_0} \langle s_1, \mathfrak{g}_1\rangle \xrightarrow{\kappa_1} \langle s_2, \mathfrak{g}_2\rangle \xrightarrow{\kappa_2} \ldots$ be a path in $unf_{\mathfrak{G}}(\mathcal{M})$. It holds that $\forall\, n \in \mathbb{N}\colon\ \mathfrak{g}_n \subseteq \mathfrak{g}_{n+1}$ and for $G_j \in \mathfrak{g}_n$ it is either $G_j \in \mathfrak{g}_0$ or $\exists\, i \in \{1..n\}\colon\ s_i \in G_j$. In particular, if $\mathfrak{g}_0 = succ(\emptyset, s_0) = \{G \in \mathfrak{G} \mid s_0 \in G\}$, then $G_j \in \mathfrak{g}_n$ holds iff $pref(\pi, n)$ visits some state in G_j. We convert the initial state s_I considered for $\mathcal{M}$ to the initial state $s'_I \coloneqq \langle s_I, succ(\emptyset, s_I)\rangle$ for $unf_{\mathfrak{G}}(\mathcal{M})$. Furthermore, the reward assignments $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$ for $\mathcal{M}$ are converted to reward assignments $\mathcal{R}'_1, \ldots, \mathcal{R}'_\ell$ for $unf_{\mathfrak{G}}(\mathcal{M})$: for $j \in \{1..\ell\}$, the reward assignment $\mathcal{R}'_j$ sets all rewards at states $\langle s, \mathfrak{g}\rangle$ with $G_j \in \mathfrak{g}$ to zero. The remaining states get the same rewards as given by $\mathcal{R}_j$, i.e., for all $s, s' \in S$, $\mathfrak{g}, \mathfrak{g}' \subseteq \mathfrak{G}$, and $\kappa \in Act \cup \mathbb{R}_{\geq 0}$:

$$\mathcal{R}'_j[\langle s, \mathfrak{g}\rangle, \kappa, \langle s', \mathfrak{g}'\rangle] \;\coloneqq\; \mathcal{R}_j[s, \kappa, s'] \cdot [G_j \notin \mathfrak{g}].$$

Example 4.1 Given the MA $\mathcal{M}$ from Figure 4.1a with total reachability reward objectives $tot(\mathcal{R}_1, \{s_1\})$ and $tot(\mathcal{R}_2, \{s_2\})$, Figure 4.1b shows the goal unfolding with respect to the goal state sets $\{s_1\}$ and $\{s_2\}$.



(a) MA $\mathcal{M}$ (b) MA $unf_{\{\{s_1\},\{s_2\}\}}(\mathcal{M})$

Figure 4.1: Example MA and goal unfolding (cf. Example 4.1)

Theorem 4.1

$$Ach_{s_I}^{\mathcal{M}}(\Phi) \;=\; Ach_{s'_I}^{unf_{\mathfrak{G}}(\mathcal{M})}(\langle tot(\mathcal{R}'_1), \ldots, tot(\mathcal{R}'_\ell)\rangle)$$

Proof (sketch). Let $f\colon Paths_{\text{inf}}^{\mathcal{M}}(s_I) \to Paths_{\text{inf}}^{unf_{\mathfrak{G}}(\mathcal{M})}(s'_I)$ be the *bijective* mapping defined by

$$f(s_0 \xrightarrow{\kappa_0} s_1 \xrightarrow{\kappa_1} s_2 \xrightarrow{\kappa_2} \ldots) \;:=\; \langle s_0, \mathfrak{g}_0\rangle \xrightarrow{\kappa_0} \langle s_1, \mathfrak{g}_1\rangle \xrightarrow{\kappa_1} \langle s_2, \mathfrak{g}_2\rangle \xrightarrow{\kappa_2} \ldots$$

where $\mathfrak{g}_0 := succ(\emptyset, s_0)$ and $\mathfrak{g}_{i+1} := succ(\mathfrak{g}_i, s_{i+1})$ for all $i \in \mathbb{N}$. For a path $\pi \in Paths_{\text{inf}}^{\mathcal{M}}(s_I)$ we can show

$$\forall\, j \in \{1..\ell\}\colon\; tot(\mathcal{R}_j, G_j)(\pi) \;=\; tot(\mathcal{R}'_j)(f(\pi)).$$

It remains to show that also the induced path probabilities coincide. To this end we lift f to sets of infinite paths, to finite paths, and to strategies by setting

- $f(\Pi) := \{f(\pi) \mid \pi \in \Pi\}$ for $\Pi \subseteq Paths_{\text{inf}}^{\mathcal{M}}(s_I)$,
- $f(pref(\pi, n)) := pref(f(\pi), n)$ for $\pi \in Paths_{\text{inf}}^{\mathcal{M}}(s_I)$, $n \in \mathbb{N}$, and
- $f(\sigma) \in \Sigma^{unf_{\mathfrak{G}}(\mathcal{M})}$ with $f(\sigma)(\hat{\pi}) := \sigma(f^{-1}(\hat{\pi}))$ for $\sigma \in \Sigma^{\mathcal{M}}$, $\hat{\pi} \in Paths_{\text{fin}}^{unf_{\mathfrak{G}}(\mathcal{M})}(s'_I)$.

We can show for every strategy $\sigma \in \Sigma^{\mathcal{M}}$ and measurable set $\Pi \subseteq Paths_{\text{inf}}^{\mathcal{M}}(s_I)$ that

$$\Pr_{\sigma,s_I}^{\mathcal{M}}(\Pi) \;=\; \Pr_{f(\sigma),s'_I}^{unf_{\mathfrak{G}}(\mathcal{M})}(f(\Pi)).$$

It follows that for every $j \in \{1..\ell\}$:

$$\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(tot(\mathcal{R}_j, G_j)) \;=\; \mathrm{Ex}^{unf_{\mathfrak{G}}(\mathcal{M})}_{f(\sigma),s'_I}(tot(\mathcal{R}'_j)).$$

The theorem follows since the mapping of strategies via f is bijective. ■

Let $\langle v, \boldsymbol{p}\rangle$ be a solution for the WSO instance with input $\mathcal{M}$, s_I, and Φ. From Theorem 4.1 we get that $\langle v, \boldsymbol{p}\rangle$ must also be a solution to the WSO instance obtained when considering $unf_{\mathfrak{G}}(\mathcal{M})$, s'_I, and $\langle tot(\mathcal{R}'_1), \ldots, tot(\mathcal{R}'_\ell)\rangle$ instead.

To avoid cluttering the notations below, we from now on assume a set-up where the given objectives for MA $\mathcal{M}$ are already total reward objectives, i.e.,

$$\Phi \;=\; \langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_\ell)\rangle.$$

We can enforce that this assumption is valid by applying the goal unfolding. However, the size of the considered model potentially increases exponentially in the number of objectives ℓ.

Remark 4.1 (Strategy Complexity) Using f as in the proof of Theorem 4.1, we remark that $\sigma \in \Sigma^{\mathcal{M}}$ is pure iff $f(\sigma) \in \Sigma^{unf_{\mathfrak{G}}(\mathcal{M})}$ is pure. However, we might have that $f(\sigma)$ is memoryless whereas σ is not. Intuitively, σ needs to remember which sets of goal states have been visited already. The required memory is exponential in the number of objectives.

4.1.2 From MA to MDP

underlying MDP

The next simplification step is to solve the WSO instance on the *underlying MDP* $MDP(\mathcal{M})$ (cf. Definition 2.24) instead of the MA $\mathcal{M}$. This abstracts away the continuous-time aspects of the system and therefore simplifies the model analysis and the correctness arguments. The following theorem establishes correctness of this reduction step. Recall the reward assignment $MDP(\mathcal{R})$ for $MDP(\mathcal{M})$ that is constructed from a given reward assignment $\mathcal{R}$ for $\mathcal{M}$ as defined on page 46.

Theorem 4.2 ([13, Theorem 4][a])

$$\begin{aligned} &Ach^{\mathcal{M}}_{s_I}\big(\langle tot(\mathcal{R}_1),\ \ldots,\ tot(\mathcal{R}_\ell)\rangle\big) \\ &= Ach^{MDP(\mathcal{M})}_{s_I}\Big(\big\langle tot(MDP(\mathcal{R}_1)),\ \ldots,\ tot(MDP(\mathcal{R}_\ell))\big\rangle\Big). \end{aligned}$$

[a] This result originates from the author's master's thesis [1, Theorem 4.9] and therefore provides no contribution to this dissertation.

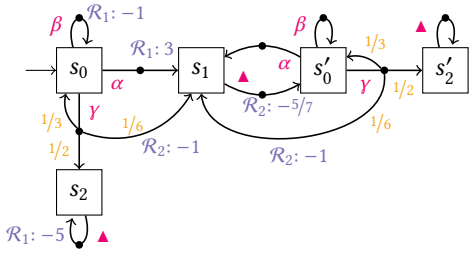


Figure 4.2: Underlying MDP of the MA from Figure 4.1b (cf. Example 4.2)

Proof (sketch). We convert between a strategy σ' for $MDP(\mathcal{M})$ and a strategy σ for $\mathcal{M}$ in a way that both strategies induce the same expected total rewards. Let $\Phi := \langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_\ell)\rangle$ and $\Phi' := \big\langle tot(MDP(\mathcal{R}_1)), \ldots, tot(MDP(\mathcal{R}_\ell))\big\rangle$.
Any strategy $\sigma' \in \Sigma^{MDP(\mathcal{M})}$ can directly be converted to a strategy $\sigma \in \Sigma^{\mathcal{M}}$ with $\mathrm{Ex}^{MDP(\mathcal{M})}_{\sigma',s_I}(\Phi') = \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\Phi)$, i.e., σ' and σ achieve the same points. Intuitively, σ can simply mimic σ' by ignoring the time durations encoded in the input paths. Hence, $Ach^{\mathcal{M}}_{s_I}(\Phi) \supseteq Ach^{MDP(\mathcal{M})}_{s_I}(\Phi')$.
The other direction is more involved: given a strategy $\sigma \in \Sigma^{\mathcal{M}}$, its decisions may depend on the timing information which is not present in paths of $MDP(\mathcal{M})$. For example, σ can select action α at some state s with probability e^{-t}, where t is the time we spend in another state $\hat{s}$ before. However, it has been shown in [1; 13] that we can still mimic such a time-dependent strategy with a time-independent strategy σ' for $MDP(\mathcal{M})$. Intuitively, this is possible because the time we stay in the Markovian states of $\mathcal{M}$ is subject to an exponential distribution which the strategy σ' can emulate with a randomized choice. It follows $Ach^{\mathcal{M}}_{s_I}(\Phi) \subseteq Ach^{MDP(\mathcal{M})}_{s_I}(\Phi')$.■

As in the previous step we avoid notational clutter by assuming a set-up where the input model is already an MDP $M = \langle S, Act, \Delta, \mathbf{P}\rangle$. If that assumption is not valid we can—justified by Theorem 4.2—simply consider the underlying MDP instead.

Example 4.2 Figure 4.2 shows the underlying MDP $M = MDP(unf_{\{\{s_1\},\{s_2\}\}}(\mathcal{M}))$ of the goal unfolding from Figure 4.1b with associated reward assignments $MDP(\mathcal{R}'_j)$. States and reward assignments have been renamed to simplify the examples below.

Input: MDP M, initial state s_I, objectives $\Phi = \langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_\ell)\rangle$
Output: 'VALID' iff $Ach(\Phi) \cap \mathbb{R}^\ell \neq \emptyset$ and $\forall\, j \in \{1..\ell\}\colon\ tot(\mathcal{R}_j)$ is convergent with $\mathrm{Ex}_{\max}(tot(\mathcal{R}_j)) < \infty$

```
// Check if it is possible to collect infinite positive reward
1  for C ∈ MEC(M) do
2      if Pr_max(◇states(C)) = 0 then continue                    // C is not reachable
3      for j ∈ {1..ℓ} do
4          if not R_j⟦C⟧ ≤ 0 then                                 // C contains positive reward
5              if R_j⟦C⟧ ≥ 0 then                                 // C contains no negative reward
6                  return 'ERROR: Ex_max(tot(R_j)) = +∞'
7              else                                               // C contains positive and negative reward
8                  return 'ERROR: tot(R_j) is not convergent'

// Check if no further reward can be collected from some point on
9  S_∃0 ← {s ∈ states(C_0) | C_0 ∈ MEC_0(M, {R_1, ..., R_ℓ})}
10 if Pr_max(◇S_∃0) = 1 then return 'VALID'
11 else return 'ERROR: Ach(Φ) ∩ ℝ^ℓ = ∅'
```

Algorithm 4.1: Check validity and finiteness for expected total rewards

4.1.3 Checking Finiteness

Given an MDP M with total reward objectives $\Phi = \langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_\ell)\rangle$, we now provide means to check Assumption 3.1 on page 78—i.e., $\mathrm{Ex}^M_{\max}(tot(\mathcal{R}_j)) < \infty$ for all $j \in \{1..\ell\}$—and to check whether $Ach(\Phi) \cap \mathbb{R}^\ell \neq \emptyset$ holds. Both checks are part of Algorithm 3.2 on page 87 and required to ensure that the corresponding `WSO` instance actually has a solution (cf. Lemma 3.13 on page 81). In addition, we also ensure that the given total reward objectives are indeed convergent (cf. Definition 2.27 on page 49) and raise an error otherwise. Those checks rely on the well-known fact that almost surely some *maximal end component* (MEC) of M will be reached and never left. Hence, infinite expected total rewards (positive or negative) can only be incurred in ECs. If all the above mentioned checks pass, we can infer that the set of achievable points is always closed. We show this in Lemma 4.6 below. Therefore, there is no need to explicitly check Item 2 of Lemma 3.13.

MEC, maximal end component

The procedure for performing the above-mentioned checks is outlined in Algorithm 4.1. For each MEC C that is reachable from the initial state s_I of M, the algorithm checks if positive reward with respect to some $\mathcal{R}_j$ can be collected while staying inside C (Line 4). If that is the case, we know that either Assumption 3.1 is violated or—if we can also collect negative reward within C—the objective $tot(\mathcal{R}_j)$ is not convergent

(Lines 5 to 8). To see this, consider a strategy for M that reaches the component C with positive probability and—once C has been reached—plays each action inside C uniformly. Under such a strategy, there is a positive probability to take every transition within C infinitely often and thus collecting infinite positive (and potentially negative) rewards.

On the other hand, if there is no reachable MEC with positive reward, we compute in Line 9 the set $S_{\exists 0}$ of states that lie in some 0-EC of $\mathcal{M}$. Observe that once a state in a 0-EC is reached, it is possible to collect no further reward, i.e.,

$$\forall\, s \in S_{\exists 0} \colon\ \exists\, \sigma \in \Sigma^M \colon\ \forall\, j \in \{1..\ell\} \colon\ \mathrm{Ex}_{\sigma,s}(tot(\mathcal{R}_j)) = 0.$$

We check in Line 10 whether the states in $S_{\exists 0}$ can be reached from the initial state s_I almost surely[1]. If that is the case, we have a strategy σ under which we reach and never leave a 0-EC almost surely. This yields $\mathrm{Ex}^M_{\sigma,s_I}(\Phi) \in \mathbb{R}^\ell$, i.e., $Ach(\Phi) \cap \mathbb{R}^\ell \neq \emptyset$. Otherwise—if there always is a positive probability to never reach a 0-EC—there is a positive probability to never stop collecting further (negative) reward, yielding $\mathrm{Ex}^M_{\sigma,s_I}(\Phi) \notin \mathbb{R}^\ell$ for every strategy σ.

From our explanations above we obtain the following lemma.

Lemma 4.3 If Algorithm 4.1 returns 'VALID', then $\boldsymbol{w} \cdot \mathrm{Ex}^M_{\sigma,s_I}(\Phi)$ is defined for every $\sigma \in \Sigma^M$ and $\sup\left\{\boldsymbol{w} \cdot \mathrm{Ex}^M_{\sigma,s_I}(\Phi) \mid \sigma \in \Sigma^M\right\} > -\infty$.

In what follows, we assume that Algorithm 4.1 indeed returns 'VALID' on the given input MDP M with initial state s_I and total reward objectives Φ. Moreover, we assume that every state $s \in S$ can reach the set $S_{\exists 0}$ as given in Line 9 almost surely, i.e., $\forall\, s \in S \colon\ \Pr_{\max,s}(\Diamond S_{\exists 0}) = 1$. This assumption is without loss of generality as we can always consider the submodel

$$M[\![\, \{\langle s, \alpha\rangle \in SA^M \mid s \in S' \text{ and } post(\langle s, \alpha\rangle) \subseteq S'\}]\!]$$

with $S' = \{s \in S \mid \Pr_{\max,s}(\Diamond S_{\exists 0}) = 1\}$ instead of M.

Example 4.3 We consider the MDP M from Figure 4.2 with initial state s_0 and total reward objectives $tot(\mathcal{R}_1)$ and $tot(\mathcal{R}_2)$. The maximal end components of M are

$$MEC(M) \;=\; \big\{\{\langle s_0, \beta\rangle\},\ \{\langle s_1, \blacktriangle\rangle, \langle s_0', \alpha\rangle, \langle s_0', \beta\rangle\},\ \{\langle s_2, \blacktriangle\rangle\},\ \{\langle s_2', \blacktriangle\rangle\}\big\}.$$

Since none of these ECs assign a positive reward, we conclude that both objectives

[1] Maximal almost sure reachability in MDPs can be computed efficiently using graph algorithms, e.g., [BK08, Algorithm 45].

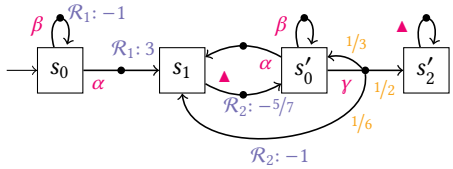


Figure 4.3: Modified MDP without states that can not reach $S_{\exists 0}$ almost surely (cf. Example 4.3)

are convergent and $\mathrm{Ex}_{\max}(tot(\mathcal{R}_j)) < +\infty$ for $j \in \{1, 2\}$.
The maximal 0-ECs for $\mathcal{R}_1$ and $\mathcal{R}_2$ are

$$MEC_0(M, \{\mathcal{R}_1, \mathcal{R}_2\}) \;=\; \big\{\{\langle s'_0, \beta\rangle\},\ \{\langle s'_2, ▲\rangle\}\big\}.$$

We thus have $S_{\exists 0} = \{s'_0, s'_2\}$. This set can be reached almost surely from all states of M except for s_2. To enforce the above mentioned assumption, we therefore remove the state s_2 and the action $\gamma \in \Delta(s_0)$, yielding the MDP shown in Figure 4.3.

4.1.4 From Weighted Sum to Single-Objective

Finally, we outline how the `WSO` instance can be answered with a series of $\ell + 1$ single-objective expected total reward computations. The crux is that the weighted sum over total reward objectives can be transformed to a single total reward objective with reward assignment

$$\mathcal{R} \;:=\; \boldsymbol{w} \cdot \langle \mathcal{R}_1, \ldots, \mathcal{R}_\ell \rangle \;=\; \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \mathcal{R}_j.$$

Lemma 4.4 For all $\sigma \in \Sigma^M$ we have

$$\boldsymbol{w} \cdot \mathrm{Ex}^M_{\sigma,s_I}\big(\langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_\ell)\rangle\big) \;=\; \mathrm{Ex}^M_{\sigma,s_I}\big(tot(\mathcal{R})\big).$$

Proof. From Lemma 2.6 on page 48 we have

$$\boldsymbol{w} \cdot \mathrm{Ex}^M_{\sigma,s_I}\big(\langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_\ell)\rangle\big) \;=\; \mathrm{Ex}^M_{\sigma,s_I}\big(\boldsymbol{w} \cdot \langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_\ell)\rangle\big).$$

It remains to show that $\boldsymbol{w} \cdot \langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_\ell)\rangle = tot(\mathcal{R})$. Consider the path

$\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots \in Paths^M_{\text{inf}}$. We have

$$\begin{aligned}
\big(\boldsymbol{w} \cdot \langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_\ell)\rangle\big)(\pi) &= \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot tot(\mathcal{R}_j)(\pi) \\
&= \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \lim_{n\to\infty} \mathcal{R}_j[pref(\pi, n)] \\
&= \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \lim_{n\to\infty} \sum_{i=0}^{n-1} \mathcal{R}_j[s_i, \alpha_i, s_{i+1}] \\
&= \lim_{n\to\infty} \sum_{i=0}^{n-1} \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \mathcal{R}_j[s_i, \alpha_i, s_{i+1}] \\
&= \lim_{n\to\infty} \sum_{i=0}^{n-1} \mathcal{R}[s_i, \alpha_i, s_{i+1}] \\
&= \lim_{n\to\infty} \mathcal{R}[pref(\pi, n)] \;=\; tot(\mathcal{R}). \qquad \blacksquare
\end{aligned}$$

Assuming that all objectives are total reward objectives, Algorithm 4.2 outlines an implementation for the procedure `solveWso` as considered in Section 3.4. The algorithm extends ideas of [FKP12] towards

- mixtures of positive and negative rewards[2] and
- a proper treatment of approximate solutions.

The former is accomplished by considering the quotient (cf. Definition 2.21) of M with respect to 0-ECs following ideas of [Alf97]. The latter is achieved by computing sound ε-approximations of expected total rewards. More precisely, Algorithm 4.2 considers two kinds of (single-objective) expected total reward computations in Lines 8 and 21 which are both instances of Problem 4.1 introduced in the next section.

After initializing a precision parameter ε and the set $\mathcal{I}_{\boldsymbol{w}}$ of objective indices that intuitively are relevant with respect to the weight vector $\boldsymbol{w}$ in Line 2, the algorithm builds an auxiliary MDP M' in Lines 3 to 6. M' is the quotient model that arises from M after collapsing all 0-ECs with respect to those reward assignments $\mathcal{R}_j$ that are considered with positive weight $\boldsymbol{w}(j) > 0$. Recall from Definition 2.21 on page 42 that the quotient model $M_{\backslash \mathfrak{C}_0}$ considers an action $\perp$ via which we may transition from any collapsed EC $C \in \mathfrak{C}_0$ to a dedicated sink state $s_\perp$. Intuitively, such a transition reflects the possibility to never leave the corresponding EC C in M. If, however, C does not

[2] In [FKP12], all expected total reward objectives consider only positive reward assignments and either only maximizing or only minimizing objectives can be analyzed.

solve WSO

function solveWso($M, s_I, \Phi = \langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_\ell)\rangle, \boldsymbol{w}, \varepsilon_{\texttt{WSO}}$)

// Assuming Alg. 4.1 yields 'VALID' and $\forall\, s \in S\colon \Pr_{\max,s}(\Diamond S_{\exists 0}) = 1$

$\varepsilon \leftarrow \frac{1}{2} \cdot \varepsilon_{\texttt{WSO}} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}$; $\mathcal{I}_{\boldsymbol{w}} \leftarrow \{j \in \{1..\ell\} \mid \boldsymbol{w}(j) > 0\}$

// Get the 0-MECs w.r.t. the reward assignments relevant for $\boldsymbol{w}$

$\mathfrak{C}_0 \leftarrow MEC_0(M, \{\mathcal{R}_j \mid j \in \mathcal{I}_{\boldsymbol{w}}\})$

*// Get those 0-MECs that do not contain a 0-EC w.r.t. **all** rewards*

$\mathfrak{C}_{exit} \leftarrow \{C \in \mathfrak{C}_0 \mid MEC_0(M[\![C]\!], \{\mathcal{R}_1, \ldots, \mathcal{R}_\ell\}) = \emptyset\}$

// Get the 0-MEC quotient and clear trans. from $C \in \mathfrak{C}_{exit}$ to $s_\perp$

$M' \leftarrow (M_{\setminus \mathfrak{C}_0})[\![SA^{(M_{\setminus \mathfrak{C}_0})} \setminus (\mathfrak{C}_{exit} \times \{\perp\})]\!]$

if $\exists\, C \in \mathfrak{C}_0\colon\ s_I \in states(C)$ **then** $s_I' \leftarrow C$ **else** $s_I' \leftarrow s_I$

// Analyze weighted sum total rewards on M'

$\mathcal{R} \leftarrow \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \mathcal{R}_j$

Compute $v \in \mathbb{R}$ and $\sigma' \in \Sigma_{\text{PM}}^{M'}$ with

$$v - \varepsilon \ \leq\ \text{Ex}_{\sigma', s_I'}^{M'}(tot(\mathcal{R})) \ \leq\ \text{Ex}_{\max, s_I'}^{M'}(tot(\mathcal{R})) \ \leq\ v$$

// Lift strategy σ' to a strategy $\sigma \in \Sigma_{\text{PM}}^{M}$

for $C \in \mathfrak{C}_0$ **do**

  if $\sigma'(C) = \langle s, \alpha\rangle \in exits(C)$ **then** $G_C \leftarrow \{s\}$; $\sigma(s) \leftarrow \alpha$

  else *// $\sigma'(C) = \perp$*

    $C' = \{c \in C'' \mid C'' \in MEC_0(M[\![C]\!], \{\mathcal{R}_1, \ldots, \mathcal{R}_\ell\})\}$

    $G_C \leftarrow states(C')$

    for $s \in G_C$ **do** $\sigma(s) \leftarrow \alpha$ for arbitrary α with $\langle s, \alpha\rangle \in C'$

  Compute $\sigma_C \in \Sigma^{M[\![C]\!]}$ with $\forall\, s' \in states(C)\colon \Pr_{\sigma_C, s'}^{M[\![C]\!]}(\Diamond G_C) = 1$

  for $s \in states(C) \setminus G_C$ **do** $\sigma(s) \leftarrow \sigma_C(s)$

for $s \in S \setminus \big(\bigcup_{C \in \mathfrak{C}_0} states(C)\big)$ **do** $\sigma(s) \leftarrow \sigma'(s)$

// Evaluate individual objectives w.r.t. strategy σ

$D \leftarrow M[\![\sigma]\!]$ *// Induced submodel $M[\![\sigma]\!]$ is a DTMC D*

for $j \in \{1..\ell\}$ **do**

  if $j \in \mathcal{I}_{\boldsymbol{w}}$ **then** $\varepsilon_j \leftarrow \frac{\varepsilon}{\boldsymbol{w}(j) \cdot |\mathcal{I}_{\boldsymbol{w}}|}$ **else** $\varepsilon_j \leftarrow \infty$

  Compute p_j with $p_j \leq \text{Ex}^{D}(tot(\mathcal{R}_j)) \ \leq\ p_j + \varepsilon_j$

return $\langle v, \langle p_1, \ldots, p_\ell\rangle\rangle$

Algorithm 4.2: Solving total reward WSO instances

contain a 0-EC with respect to *all* reward assignments (i.e., $C \in \mathfrak{C}_{exit}$), staying at C implies that we will always collect infinite negative reward for at least one reward assignment. For the auxiliary MDP M' we therefore remove the $\bot$-transition from the collapsed ECs $C \in \mathfrak{C}_{exit}$ which is done by considering the appropriate submodel in Line 5. Recall from Section 2.2.6 that we may also apply the reward assignments $\mathcal{R}_1, \dots, \mathcal{R}_\ell$ for M to M' without ambiguity.

In Line 8, we compute the maximal expected total reward as well as an inducing strategy[3] $\sigma' \in \Sigma_{\text{PM}}^{M'}$ for M' and the weighted reward assignment $\mathcal{R}$. However, if $\varepsilon_{\texttt{WSO}} > 0$ then also $\varepsilon > 0$ and we allow a certain approximation error in this computation.

The strategy σ' for M' is converted to a pure and memoryless strategy σ for M in Lines 9 to 17. The intuition is as follows. If σ' selects an exiting state-action pair $\langle s, \alpha \rangle \in \mathit{exits}(C)$ at some collapsed EC C, σ can enforce to leave the EC via the action α at state s, as well. If, on the other hand, σ' selects $\bot$ at collapsed EC C, C must contain at least one 0-EC with respect to *all* reward assignments $\mathcal{R}_1, \dots, \mathcal{R}_\ell$ (otherwise, action $\bot$ would not be available in M'). Strategy σ mimics the choice to stay inside EC C forever by enforcing that one of the contained 0-ECs is reached eventually and never left.

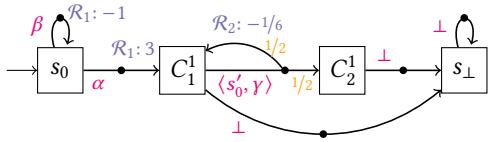
Finally, we evaluate each individual objective $\mathit{tot}(\mathcal{R}_j)$, $j \in \{1..\ell\}$ on the induced DTMC $D = M[\![\sigma]\!] = M[\![\{\langle s, \sigma(s)\rangle \mid s \in S\}]\!]$ in Line 21 to create a point $\langle p_1, \dots, p_\ell \rangle \leq \mathrm{Ex}^D(\Phi)$. Again, we potentially allow some approximation error, where objectives $\mathit{tot}(\mathcal{R}_j)$ with high weights $\boldsymbol{w}(j)$ need to be analyzed with more accuracy.

Example 4.4 Let M be the MDP shown in Figure 4.3. We have

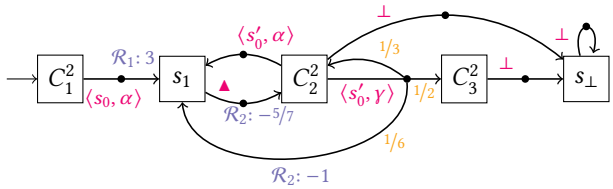
$$\begin{aligned}
MEC_0(M, \{\mathcal{R}_1\}) &= \big\{ \underbrace{\{\langle s_1, \blacktriangle\rangle, \langle s_0', \alpha\rangle, \langle s_0', \beta\rangle\}}_{=:C_1^1}, \underbrace{\{\langle s_2', \blacktriangle\rangle\}}_{=:C_2^1} \big\}, \\
MEC_0(M, \{\mathcal{R}_2\}) &= \big\{ \underbrace{\{\langle s_0, \beta\rangle\}}_{=:C_1^2}, \underbrace{\{\langle s_0', \beta\rangle\}}_{=:C_2^2}, \underbrace{\{\langle s_2', \blacktriangle\rangle\}}_{=:C_3^2} \big\}, \text{ and} \\
MEC_0(M, \{\mathcal{R}_1, \mathcal{R}_2\}) &= \big\{\{\langle s_0', \beta\rangle\}, \{\langle s_2', \blacktriangle\rangle\}\big\}.
\end{aligned}$$

If $\boldsymbol{w} = \langle 1, 0\rangle$, we have to collapse the 0-ECs C_1^1 and C_2^1 for $\mathcal{R}_1$. The resulting MDP M' as constructed in Algorithm 4.2 is shown in Figure 4.4a. Since both collapsed ECs contain a 0-EC for both $\mathcal{R}_1$ and $\mathcal{R}_2$—in particular $\{\langle s_0', \beta\rangle\} \subseteq C_1^1$—we do not remove any $\bot$-transitions from the quotient.
A possible strategy maximizing $\mathit{tot}(\mathcal{R})$ with $\mathcal{R} = 1 \cdot \mathcal{R}_1 + 0 \cdot \mathcal{R}_2 = \mathcal{R}_1$ for M' would be $\sigma' \in \Sigma_{\text{PM}}^{M'}$ with $\sigma'(s_0) = \alpha$ and $\sigma'(C_1^1) = \bot$. We can mimic this strategy also

[3] Maximal expected total rewards can always be attained by a pure and memoryless strategy [BT91].



(a) MDP M' assuming $\boldsymbol{w} = \langle 1, 0 \rangle$



(b) MDP M' assuming $\boldsymbol{w} = \langle 0, 1 \rangle$

Figure 4.4: The MDP M' for two different weight vectors as constructed in Algorithm 4.2 (cf. Example 4.4)

in M by considering $\sigma \in \Sigma_{\mathrm{PM}}^M$ with $\sigma(s_0) = \alpha$, $\sigma(s_1) = \blacktriangle$, and $\sigma(s_0') = \beta$. Observe that setting $\sigma(s_0') = \alpha$ would induce a non-finite value for the objective $tot(\mathcal{R}_2)$. It is therefore crucial that σ enforces to reach the 0-EC $\{\langle s_0', \beta \rangle\}$ contained in C_1^1 almost surely.
Figure 4.4b shows the resulting MDP for the weight vector $\boldsymbol{w} = \langle 0, 1 \rangle$ instead. Observe that component C_1^2 does not contain any 0-EC for all reward assignments and thus does not have an outgoing $\perp$-action. In fact, we should not consider any strategy $\sigma \in \Sigma_{\mathrm{PM}}^M$ that stays inside C_1^2—i.e., $\sigma(s_0) = \beta$—since such a strategy induces a non-finite value for $tot(\mathcal{R}_1)$.

Theorem 4.5 The value v and point $\langle p_1, \ldots, p_\ell \rangle$ returned by Algorithm 4.2 yield a solution to WSO.

Proof. Let $\Xi \coloneqq \{\sigma \in \Sigma^M \mid \mathrm{Ex}_\sigma^M(\Phi) \in \mathbb{R}^\ell\}$ be as in Problem 3.4 on page 81. Using $\boldsymbol{p} \coloneqq \langle p_1, \ldots, p_\ell \rangle$, we have to show that

1. $\boldsymbol{p} \in Ach^M(\Phi) \cap \mathbb{R}^\ell$,
2. $\sup \left\{ \boldsymbol{w} \cdot \mathrm{Ex}_\sigma(\Phi) \;\middle|\; \sigma \in \Xi \right\} \leq v$, and
3. $v \leq \boldsymbol{w} \cdot \boldsymbol{p} + \varepsilon_{\mathtt{WSO}} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}$.

Item 1 | The probability measures $\Pr^D$ and $\Pr^M_\sigma$ coincide. Since

$$\forall\, j \in \{1..\ell\}\colon\ p_j \ \leq\ \mathrm{Ex}^D(tot(\mathcal{R}_j)) \ =\ \mathrm{Ex}^M_\sigma(tot(\mathcal{R}_j))\,,$$

we have $\boldsymbol{p} \leq \mathrm{Ex}^M_\sigma(\Phi)$, i.e., $\boldsymbol{p} \in Ach(\Phi)$. $\boldsymbol{p} \in \mathbb{R}^\ell$ follows from the construction of σ since the strategy enforces to eventually reach and stay inside a 0-EC for $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$.

Item 2 | From Lemma 4.4 we get

$$\sup\left\{\boldsymbol{w}\cdot\mathrm{Ex}^M_\sigma(\Phi)\;\middle|\;\sigma\in\Xi\right\} \ =\ \sup\left\{\mathrm{Ex}^M_\sigma(tot(\mathcal{R}))\;\middle|\;\sigma\in\Xi\right\}.$$

Moreover, we can show that

$$\sup\left\{\mathrm{Ex}^M_\sigma(tot(\mathcal{R}))\;\middle|\;\sigma\in\Xi\right\} \ =\ \sup\left\{\mathrm{Ex}^{M'}_{\sigma'}(tot(\mathcal{R}))\;\middle|\;\sigma'\in\Sigma^{M'}\right\}.$$

To see this, observe that strategy σ' for M' can be converted to a strategy σ for M yielding the same expected total reward by using the construction as in Lines 9 to 17 of Algorithm 4.2. For the other direction, observe that once a 0-EC (for $\mathcal{R}$) is reached in M and a strategy σ is fixed, the decision whether (and via which) transition the EC is exited is purely probabilistic and can thus always be simulated by a potentially randomized choice of a strategy σ' for M'. A more formal argument can be found in [Alf97, Chapter 7]. The claim for Item 2 follows as

$$\sup\left\{\mathrm{Ex}^{M'}_{\sigma'}(tot(\mathcal{R}))\;\middle|\;\sigma'\in\Sigma^{M'}\right\} \ =\ \mathrm{Ex}^{M'}_{\max}(tot(\mathcal{R})) \leq v.$$

Item 3 | Since $2\cdot\varepsilon = \varepsilon_{\texttt{WSO}}\cdot\sqrt{\boldsymbol{w}\cdot\boldsymbol{w}}$, it only remains to show that $v \leq \boldsymbol{w}\cdot\boldsymbol{p} + 2\cdot\varepsilon$ which follows from

$$\begin{aligned}
v \ &\leq\ \mathrm{Ex}^{M'}_{\sigma'}(tot(\mathcal{R})) + \varepsilon \ =\ \mathrm{Ex}^M_\sigma(tot(\mathcal{R})) + \varepsilon \ =\ \boldsymbol{w}\cdot\mathrm{Ex}^M_\sigma(\Phi) + \varepsilon \\
&=\ \Big(\sum_{j=1}^{\ell}\boldsymbol{w}(j)\cdot\mathrm{Ex}^M_\sigma(tot(\mathcal{R}_j))\Big) + \varepsilon \ =\ \Big(\sum_{j=1}^{\ell}\boldsymbol{w}(j)\cdot\mathrm{Ex}^D(tot(\mathcal{R}_j))\Big) + \varepsilon \\
&\leq\ \Big(\sum_{j=1}^{\ell}\boldsymbol{w}(j)\cdot(p_j+\varepsilon_j)\Big) + \varepsilon \ =\ \boldsymbol{w}\cdot\boldsymbol{p} + \varepsilon + \sum_{j=1}^{\ell}\boldsymbol{w}(j)\cdot\varepsilon_j \\
&=\ \boldsymbol{w}\cdot\boldsymbol{p} + \varepsilon + \sum_{j=1}^{\ell}[\boldsymbol{w}(j) > 0]\cdot\frac{\varepsilon}{|\mathcal{I}_{\boldsymbol{w}}|} \ =\ \boldsymbol{w}\cdot\boldsymbol{p} + \varepsilon + \varepsilon \ =\ \boldsymbol{w}\cdot\boldsymbol{p} + 2\cdot\varepsilon. \qquad \blacksquare
\end{aligned}$$

The correctness of our algorithms yields a simple way to prove the following.

Lemma 4.6 If Algorithm 4.1 returns 'VALID', then $down(Ach(\Phi) \cap \mathbb{R}^\ell)$ is closed for total reward objectives Φ and is the downward convex hull of at most $|\Sigma^M_{\text{PM}}| = \prod_{s \in S} |\Delta(s)|$ points.

Proof. If $\varepsilon_{\texttt{WSO}} = 0$, the point returned by Algorithm 4.2 is of the form

$$\langle p_1, \ldots, p_\ell \rangle \;=\; \left\langle \mathrm{Ex}^M_{\sigma,s_I}(tot(\mathcal{R}_1)),\; \ldots,\; \mathrm{Ex}^M_{\sigma,s_I}(tot(\mathcal{R}_1)) \right\rangle$$

for some $\sigma \in \Sigma^M_{\text{PM}}$. The number of considered strategies can be bounded by $|\Sigma^M_{\text{PM}}| = \prod_{s \in S} |\Delta(s)| \leq |Act|^{|S|} < \infty$. Thus, when Algorithm 4.2 is invoked multiple times with different weight vectors $\boldsymbol{w}$, only finitely many different points can be returned. Now suppose that we run Algorithm 3.1 on page 86 with the corresponding MOP query using precision $\eta = 0$. Following our argumentation from Section 3.4.3, the set of points P in Algorithm 3.1 will at some point stabilize which eventually yields termination of the algorithm. The correctness of the algorithm then implies $Ach(\Phi) \cap \mathbb{R}^\ell = down_{\mathbb{R}}(P)$ which is a polyhedron and therefore closed. ■

4.1.5 Attracting Goal States

With the simplifications discussed before, it only remains to analyze single-objective expected total rewards on MDPs (and DTMCs). Furthermore, our constructions—in particular collapsing 0-ECs—ensure that we can always consider an attracting set of goal states.

attracting state

Definition 4.2 (Attracting States) A set of states $G \subseteq S$ is *attracting* for MDP M and reward assignment $\mathcal{R}$ if for every state $s \in S$

- $\exists\, \sigma \in \Sigma^M\colon\ \Pr_{\sigma,s}(\Diamond G) = 1$ and
- $\forall\, \sigma \in \Sigma^M\colon\ \Pr_{\sigma,s}(\Diamond G) < 1$ implies $\mathrm{Ex}_{\sigma,s}(tot(\mathcal{R}, G)) = -\infty$.

An attracting set of states G for an MDP M can be reached almost surely from any state s under some strategy σ. Clearly, $\mathrm{Ex}_{\sigma,s}(tot(\mathcal{R}, G)) \in \mathbb{R}$ must hold for that strategy since the probability of never reaching G (and thus potentially collecting infinite reward) is zero. Furthermore, any other strategy that does not reach G almost surely yields a value of $-\infty$. It follows that $\mathrm{Ex}_{\max}(tot(\mathcal{R}, G)) \in \mathbb{R}$ and that the maximum is only attained by a strategy that reaches G almost surely.

The total reward analysis queries considered in Algorithm 4.2 can also be seen as total *reachability* reward queries for an attracting set of goal states: for M' and $\mathcal{R}$ constructed in Lines 5 and 7, the set $\{s_\perp\}$ is attracting and the objectives $tot(\mathcal{R})$ and $tot(\mathcal{R}, \{s_\perp\})$ coincide as no further reward can be collected once $s_\perp$ is reached. Furthermore, for the induced DTMC $D = M[\![\sigma]\!]$ from Line 18, the set $S_\perp$ of states

that lie on a BSCC of D is attracting[4] and—by construction of strategy σ—consist only of 0-ECs of M for reward assignments $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$. Hence, $tot(\mathcal{R}_j)$ and $tot(\mathcal{R}_j, S_\perp)$ coincide for D.

4.2 The Expected Total Reachability Reward Problem

Our aim for the rest of this chapter is to provide sound and arbitrarily tight bounds for maximal expected total reachability rewards—assuming an attracting set of goal states. As shown in, e.g., [BT91], the maximal expected reward can always be attained by a pure and memoryless strategy. In Line 8 of Algorithm 4.2, we are also interested in obtaining such an optimal (or at least close to optimal) strategy. We address how such a strategy can be obtained in Section 4.5.4. For now, we focus on the computation (or approximation) of the expected reward values, i.e., we want to solve the following problem.

Problem 4.1: Expected Total Reachability Reward Problem

Input $M = \langle S, Act, \Delta, \mathbf{P}\rangle$, initial state $s_I \in S$, reward assignment $\mathcal{R}$, attracting set of goal states G, precision $\varepsilon \in \mathbb{R}_{\geq 0}$
Output $\langle l, u\rangle \in \mathbb{R} \times \mathbb{R}$ with $u - \varepsilon \leq l \leq \mathrm{Ex}_{\max}(tot(\mathcal{R}, G)) \leq u$

Problem 4.1 can be solved in polynomial time—even in the exact case where $\varepsilon = 0$—by solving an LP that we outline in Section 4.6.3. In fact, the problem is P-complete.

Theorem 4.7 The decision problem variant of Problem 4.1—asking whether a given pair $\langle l, u\rangle$ is a solution to the instance M, s_I, $\mathcal{R}$, G, ε—is P-complete, even if restricted to instances with $\varepsilon > 0$.

Proof. We refer to [PT87] for a brief introduction to the necessary concepts from complexity theory used in this proof.
The above-mentioned LP formulation yields containment in P. For P-hardness, we follow the proof of [PT87, Theorem 1] that shows P-hardness for discounted reward objectives using a log-space reduction from the *circuit value problem* (CVP).
An instance for CVP is a sequence $T = t_1, \ldots t_k$ of $k > 0$ triples $t_i := \langle \mathrm{op}_i, \mathrm{left}_i, \mathrm{right}_i\rangle$ for $i \in \{1..k\}$, where

- $\mathrm{op}_i \in \{true,\ false,\ \wedge,\ \vee\}$, and

[4] This is true for *every* DTMC, cf. [BK08, Theorem 10.28].

- $\text{left}_i,\ \text{right}_i \in \begin{cases} \{0\} & \text{if } \text{op}_i \in \{\mathit{true},\ \mathit{false}\} \\ \{1..i-1\} & \text{if } \text{op}_i \in \{\wedge,\ \vee\}. \end{cases}$

The value $v(t_i) \in \{\mathit{true},\ \mathit{false}\}$ of a triple $t_i = \langle \text{op}_i, \text{left}_i, \text{right}_i \rangle$ is inductively defined by

$$v(\langle \text{op}_i, \text{left}_i, \text{right}_i \rangle) := \begin{cases} \text{op}_i & \text{if } \text{op}_i \in \{\mathit{true},\ \mathit{false}\} \\ v(t_{\text{left}_i})\ \text{op}_\text{i}\ v(t_{\text{right}_i}) & \text{if } \text{op}_i \in \{\wedge,\ \vee\}. \end{cases}$$

CVP asks whether we have $v(t_k) = \mathit{true}$.
For a given CVP instance $T = t_1, \ldots t_k$ we construct the instance M_T, s_I, $\mathcal{R}_T$, G, ε, $\langle l, u \rangle$, for the decision problem variant of the expected total reachability reward problem, where

- $M_T := \langle \{t_1, \ldots, t_k\}, \{\alpha, \beta\}, \Delta_T, \mathbf{P}_T \rangle$ is the MDP such that for $i \in \{1..k\}$ we have

$$\Delta_T(t_i) := \begin{cases} \{\alpha, \beta\} & \text{if } t_i = \langle \vee, \text{left}_i, \text{right}_i \rangle \\ \{\alpha\} & \text{otherwise,} \end{cases}$$

  and the non-zero transition probabilities are given as follows:
  - if $t_i = \langle \vee, \text{left}_i, \text{right}_\text{i} \rangle$ then $\mathbf{P}_T(t_i, \alpha, t_{\text{left}_i}) = \mathbf{P}_T(t_i, \beta, t_{\text{right}_i}) = 1$,
  - if $t_i = \langle \wedge, \text{left}_i, \text{right}_\text{i} \rangle$ then $\mathbf{P}_T(t_i, \alpha, t_{\text{left}_i}) = \mathbf{P}_T(t_i, \alpha, t_{\text{right}_i}) = 1/2$, and
  - if $t_i = \langle \text{op}_i, 0, 0 \rangle$ with $\text{op}_i \in \{\mathit{true},\ \mathit{false}\}$ then $\mathbf{P}_T(t_i, \alpha, t_i) = 1$,
- $s_I := t_k$ is the initial state,
- $\mathcal{R}_T$ is the reward assignment such that for all $i \in \{1..k\}$, $t_i \neq \langle \mathit{false}, 0, 0 \rangle$ and all $\gamma \in \Delta(t_i)$:

$$\mathcal{R}_T[t_i, \gamma, \langle \mathit{false}, 0, 0 \rangle] := -1$$

  and all other rewards are zero,
- $G := \{\langle \mathit{true}, 0, 0 \rangle,\ \langle \mathit{false}, 0, 0 \rangle\}$ is the set of goal states,
- $\varepsilon := (1/2)^k$ is the precision parameter, and
- $\langle l, u \rangle := \langle -\varepsilon, 0 \rangle$ is the given solution candidate.

The above instance—including a binary encoding of the value $\varepsilon = (1/2)^k$—can be computed with $O(\log(k))$ space complexity.
Intuitively, the (maximal) expected reward at a state $t_i = \langle \wedge, \text{left}_i, \text{right}_i \rangle$ of M_T is zero iff the expected reward at both successor states t_{left_i} and t_{right_i} is zero. On the

other hand, the expected reward at state $t_i = \langle \vee, \text{left}_i, \text{right}_i \rangle$ is zero iff the expected reward at t_{left_i} or at t_{right_i} is zero. Using the same reasoning as in the proof of [PT87, Theorem 1] we can show for all $i \in \{1..k\}$ that

$$v(t_i) \;=\; \mathit{true} \quad \text{iff} \quad \mathrm{Ex}^{M_T}_{\max,t_i}(\mathit{tot}(\mathcal{R}_T, G)) \;=\; 0.$$

Moreover, if the maximal expected reward at t_i is *not* zero, then for every strategy $\sigma \in \Sigma^M_{\text{PM}}$ there must be a path $\hat{\pi} \in \mathit{Paths}^{M_T}_{\text{fin}}(t_i)$ of length $|\hat{\pi}| \leq i$ from t_i to $\mathit{last}(\hat{\pi}) = \langle \mathit{false}, 0, 0 \rangle$ that—under strategy σ—has probability at least $(1/2)^{i-1}$ and on which at least reward -1 is incurred. Thus, or all $i \in \{1..k\}$ we have

$$\mathrm{Ex}^{M_T}_{\max,t_i}(\mathit{tot}(\mathcal{R}_T, G)) \;\neq\; 0 \quad \text{implies} \quad \mathrm{Ex}^{M_T}_{\max,t_i}(\mathit{tot}(\mathcal{R}_T, G)) \;\leq\; -(1/2)^{i-1}.$$

By combining the two observations above and by noting that $-(1/2)^{k-1} < -(1/2)^k = -\varepsilon = l$ we get

$$v(t_k) \;=\; \mathit{true} \quad \text{iff} \quad l \;\leq\; \mathrm{Ex}^{M_T}_{\max,t_i}(\mathit{tot}(\mathcal{R}_T, G)) \;\leq\; u$$

which justifies the correctness of the construction and thus concludes the proof. ■

We emphasize that the hardness proof above also applies to the approximative variant of Problem 4.1, where we require $\varepsilon > 0$ instead of $\varepsilon \geq 0$. From a complexity theory perspective, computing (sound) approximations of expected total reachability rewards is thus as hard as computing exact expected values.

From a more practical perspective, however, it can often be observed that approaches for exact computations do not scale to large MDPs. In what follows, we consider instances for Problem 4.1 with $\varepsilon > 0$, i.e., we allow a certain approximation error. Such a setting enables approaches like *sound value iteration* (Sections 4.4 and 4.5) that can often deal with practical MDP models considering hundreds of millions of states.

4.3 Value Iteration

In this section we describe the classical value iteration algorithm for computing expected total reachability rewards (e.g., [Put94; BK08]). In particular we outline issues regarding the accuracy of obtained solutions.

Set-up | We fix an instance $M, s_I, \mathcal{R}, G, \varepsilon$ for the expected total reachability reward problem with $\varepsilon > 0$.

Notations | We assume an arbitrary but fixed order $s_1 \prec \ldots \prec s_n$ on the states $S = \{s_1, \ldots, s_n\}$ of M which allows us to consider vectors $\boldsymbol{x} = \langle x_{s_1}, \ldots, x_{s_n} \rangle \in \mathbb{R}^{|S|}$ to

uniquely associate some value $\boldsymbol{x}(s) := x_s \in \mathbb{R}$ to each state $s \in S$. The notation $\langle x_s \rangle_{s \in S}$ is used for the $|S|$-dimensional vector that assigns value $x_s \in \mathbb{R}$ to state s.

4.3.1 The Bellman Operator

Bellman operator

Definition 4.3 (Bellman Operator [Bel57]) For a reward assignment $\mathcal{R}$ and a set of states G, the *Bellman operator* $\mathcal{V}_{\mathcal{R},G} \colon \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ is given for all $\boldsymbol{x} \in \mathbb{R}^{|S|}$ by

$$\mathcal{V}_{\mathcal{R},G}(\boldsymbol{x}) := \left\langle [s \notin G] \cdot \max \left\{ \sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot \left(\boldsymbol{x}(s') + \mathcal{R}[s,\alpha,s']\right) \;\middle|\; \alpha \in \Delta(s) \right\} \right\rangle_{s \in S}.$$

For $k \in \mathbb{N}$ we further introduce the operator $\mathcal{V}^k_{\mathcal{R},G}$ which results from applying the Bellman operator k times, i.e., $\mathcal{V}^0_{\mathcal{R},G}(\boldsymbol{x}) := \boldsymbol{x}$ and $\mathcal{V}^{k+1}_{\mathcal{R},G}(\boldsymbol{x}) := \mathcal{V}_{\mathcal{R},G}(\mathcal{V}^k_{\mathcal{R},G}(\boldsymbol{x}))$. The following well-known result (e.g., [BT91, Proposition 2]) relates the Bellman operator to expected total reachability rewards.

Theorem 4.8 If G is attracting, the fixpoint $\boldsymbol{x}^*$ of the Bellman operator $\mathcal{V}_{\mathcal{R},G}$ is unique and satisfies

$$\boldsymbol{x}^* = \mathcal{V}_{\mathcal{R},G}(\boldsymbol{x}^*) = \left\langle \mathrm{Ex}_{\max,s}(tot(\mathcal{R},G)) \right\rangle_{s \in S}.$$

Furthermore, we have

$$\forall\, \boldsymbol{x} \in \mathbb{R}^{|S|} \colon \lim_{k \to \infty} \mathcal{V}^k_{\mathcal{R},G}(\boldsymbol{x}) = \boldsymbol{x}^*.$$

Due to Theorem 4.8, we can compute expected total reachability rewards by computing the (unique) fixpoint of the Bellman operator $\mathcal{V}_{\mathcal{R},G}$. Furthermore, applying the Bellman operator repeatedly on some arbitrary starting vector $\boldsymbol{x} \in \mathbb{R}^{|S|}$ converges to its fixpoint $\boldsymbol{x}^*$.

4.3.2 The Value Iteration Algorithm

value iteration

Theorem 4.8 gives rise to the *value iteration* [Bel57] algorithm outlined in Algorithm 4.3. Starting with vector $\boldsymbol{x}^{(0)} = \mathbf{0} \in \mathbb{R}^{|S|}$, the algorithm computes $\boldsymbol{x}^{(k)} = \mathcal{V}^k_{\mathcal{R},G}(\mathbf{0})$ for increasing k (Line 3). When for all $s \in S$ the absolute difference between $\boldsymbol{x}^{(k)}(s)$ and $\boldsymbol{x}^{(k-1)}(s)$ falls below the provided precision threshold ε, the algorithm stops and returns the current value for the initial state s_I (Line 4).

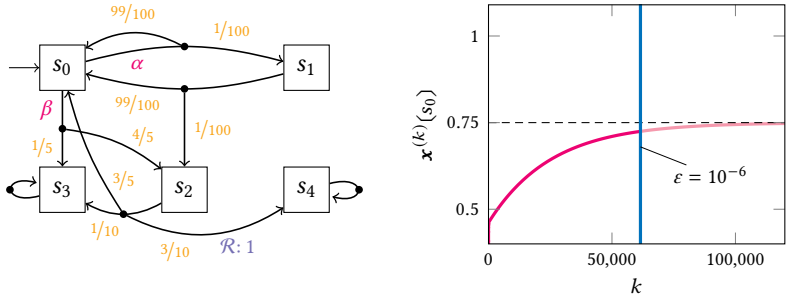
Value iteration is commonly implemented by probabilistic model checkers[5] and can be considered the de-facto standard for computing expected total rewards and related objectives such as reachability probabilities. A major asset of the algorithm is

[5] see [8] for an overview of probabilistic model checking tools and implemented algorithms.

Input: MDP M, initial state s_I, reward assignment $\mathcal{R}$, attracting set of goal states $G \subseteq S$, precision $\varepsilon > 0$
Output: Approximation of $\mathrm{Ex}_{\sigma,s_I}(tot(\mathcal{R}, G))$

1 $\boldsymbol{x}^{(0)} \leftarrow \mathbf{0}$
2 **for** $k = 1, 2, 3, \ldots$ **do**
3 $\quad \boldsymbol{x}^{(k)} \leftarrow \mathcal{V}_{\mathcal{R},G}(\boldsymbol{x}^{(k-1)})$ *// i.e.,* $\boldsymbol{x}^{(k)} = \mathcal{V}^k_{\mathcal{R},G}(\mathbf{0})$
4 $\quad$ **if** $\max_{s \in S} |\boldsymbol{x}^{(k)}(s) - \boldsymbol{x}^{(k-1)}(s)| < \varepsilon$ **then return** $\boldsymbol{x}^{(k)}(s_I)$

Algorithm 4.3: The value iteration algorithm



(a) MDP M (b) value iteration progress

Figure 4.5: MDP and illustration of value iteration (cf. Example 4.5)

scalability on practical models as value iteration can deal with large models considering millions or even billions of states.

However, there are no sound guarantees on the accuracy of the obtained result as pointed out in, e.g., [WKHB08; HM18]. The rational behind the termination criterion in Line 4 is that—as soon as the values between two successive iterations do not change much—the current value $\boldsymbol{x}^{(k)}$ is likely to be close to the fixpoint $\boldsymbol{x}^*$ of $\mathcal{V}_{\mathcal{R},G}$. While this might often the case in practice, there are no formal precision guarantees and the results obtained by value iteration can be significantly different from the actual expected rewards.

Example 4.5 Consider the MDP M from Figure 4.5a with reward assignment $\mathcal{R}$ and attracting set $G = \{s_3, s_4\}$. Observe that the expected value of the objective $tot(\mathcal{R}, G)$ coincides with the probability to eventually reach s_4.
If a strategy always selects action α at s_0, we will cycle through the strongly con-

nected component consisting of states s_0, s_1, and s_2 until we either move from s_2 to s_3 or from s_2 to s_4. The former case happens with probability $1/1+3 = 1/4$ and no reward is accumulated. The latter case occurs with probability $3/1+3 = 3/4$ and we collect a reward of 1 before moving to s_4. We thus obtain an expected total reachability reward of $3/4$. Selecting β at s_0 has a higher probability of reaching s_3—and thus not s_4—yielding a smaller expected value. Therefore, $\mathrm{Ex}^{M}_{\max,s_0}(tot(\mathcal{R}, G)) = 3/4 = 0.75$. We illustrate the progress of value iteration on this example in Figure 4.5b where we plot the number of iterations k against the computed value at the initial state s_0 (pink line). In this case, value iteration converges very slowly—requiring thousands of iterations to get close to the actual solution 0.75. As a consequence, the improvement per iteration becomes very small. Assuming that we set $\varepsilon = 10^{-6}$, the algorithm prematurely terminates in iteration $k = 61\,532$ (blue vertical line) and returns the value $\boldsymbol{x}^{(k)}(s_0) \approx 0.7248$.

Remark 4.2 (Relative termination criterion) Model checking tools often implement an alternative termination criterion for value iteration where the relative difference between two successive iterations is considered instead of the absolute difference, i.e., the algorithm terminates as soon as

$$\max_{s \in S} \left| \frac{\boldsymbol{x}^{(k)}(s) - \boldsymbol{x}^{(k-1)}(s)}{\boldsymbol{x}^{(k)}(s)} \right| < \varepsilon$$

holds. While this relative criterion does improve the accuracy in cases where the values are close to zero, it still does not provide any formal guarantees.

As demonstrated in Example 4.5, the value iteration algorithm does not provide a reliable solution to Problem 4.1. In the following we present *sound value iteration*—an algorithm for computing sound approximations for expected total reachability rewards as in Problem 4.1. For the sake of simplicity we first restrict to the case where the given MDP is deterministic, i.e., a DTMC.

4.4 Sound Value Iteration for DTMCs

SVI, sound value iteration

Set-up | In the following, let $D = \langle S, \mathbf{P} \rangle$ be a DTMC with initial state $s_I \in S$, reward assignment $\mathcal{R}$, and attracting goal state set $G \subseteq S$. Furthermore, we assume a precision parameter $\varepsilon > 0$. To solve the corresponding instance of Problem 4.1, we have to find $l, u \in \mathbb{R}$ with

$$u - \varepsilon \leq l \leq \mathrm{Ex}^{D}_{s_I}(tot(\mathcal{R}, G)) \leq u.$$

Notations | Let $S_? := S \setminus G$. For $k \in \mathbb{N}$ we define the objective $tot^{\leq k}(\mathcal{R}, G)$ and the set of paths $\Box^{\leq k} S_?$ as follows. For $\pi = s_0 \to s_1 \to \cdots \in Paths^D_{\text{inf}}$ let $n^G_\pi := \min\{i \in \mathbb{N} \mid s_i \in G\}$, i.e., the state $s_{n^G_\pi}$ is the first goal state visited by π. If π does not visit a goal state, we assume $n^G_\pi = \infty$. We set

$$\forall\, \pi \in Paths^D_{\text{inf}}\colon\ tot^{\leq k}(\mathcal{R}, G)(\pi) := \mathcal{R}[pref(\pi, \min(n^G_\pi, k))] \text{ and}$$
$$\Box^{\leq k} S_? := \{\pi \in Paths^D_{\text{inf}} \mid n^G_\pi > k\}.$$

In other words, $\text{Ex}(tot^{\leq k}(\mathcal{R}, G))$ is the expected total reachability reward for $\mathcal{R}$ and G accumulated only within the first k steps. Moreover, $\Pr(\Box^{\leq k} S_?)$ is the probability that we do not visit a state from G within the first k steps (and thus only see $S_?$-states within the first k steps).

4.4.1 Obtaining Sound Approximations

The central idea of our approach is to split the expected total reachability reward into the sum of

(i) the expected reward accumulated *within* k steps and

(ii) the expected reward accumulated *only after* k steps.

We obtain Item (i) via k iterations of (standard) value iteration. A second instance of value iteration computes the probability that the goal state set G has *not* been reached within k iterations. We show that from this information safe lower- and upper bounds for Item (ii) can be derived. The following theorem captures the key insight of sound value iteration.

Theorem 4.9 For $k \in \mathbb{N}$ with $\forall\, s \in S_?\colon \Pr_s(\Box^{\leq k} S_?) < 1$ we have

$$\begin{aligned}
&\text{Ex}_{s_I}(tot^{\leq k}(\mathcal{R}, G)) + \Pr_{s_I}(\Box^{\leq k} S_?) \cdot \min_{s \in S_?} \frac{\text{Ex}_s(tot^{\leq k}(\mathcal{R}, G))}{1 - \Pr_s(\Box^{\leq k} S_?)} \\
&\leq \text{Ex}_{s_I}(tot(\mathcal{R}, G)) \\
&\leq \text{Ex}_{s_I}(tot^{\leq k}(\mathcal{R}, G)) + \Pr_{s_I}(\Box^{\leq k} S_?) \cdot \max_{s \in S_?} \frac{\text{Ex}_s(tot^{\leq k}(\mathcal{R}, G))}{1 - \Pr_s(\Box^{\leq k} S_?)}\,.
\end{aligned}$$

Proof. The theorem is a direct consequence of Lemmas 4.10 and 4.11 that we show below. ∎

Theorem 4.9 allows to approximate $\text{Ex}_{s_I}(tot(\mathcal{R}, G))$ by computing $\text{Ex}_s(tot^{\leq k}(\mathcal{R}, G))$ and $\Pr_s(\Box^{\leq k} S_?)$ for all $s \in S_?$ and increasing $k \in \mathbb{N}$. This can be realized with a procedure based on value iteration that we outline in Section 4.4.2. The obtained

bounds on $\mathrm{Ex}_{s_I}(tot(\mathcal{R}, G))$ can be tightened arbitrarily since $\Pr_s(\Box^{\leq k} S_?)$ approaches 0 for increasing k. We prove Theorem 4.9 using the following two lemmas which essentially show that the second summand of the lower (upper) bound in Theorem 4.9 is a lower (upper) bound for the expected reward accumulated only after k steps (cf. Item (ii) above).

Lemma 4.10 For $k \in \mathbb{N}$ and $a, b \in \mathbb{R}$ with $\forall\, s \in S_?\colon\ a \leq \mathrm{Ex}_s(tot(\mathcal{R}, G)) \leq b$ we have:

$$\begin{aligned} &\mathrm{Ex}_{s_I}(tot^{\leq k}(\mathcal{R}, G)) + \Pr_{s_I}(\Box^{\leq k} S_?) \cdot a \\ \leq\ &\mathrm{Ex}_{s_I}(tot(\mathcal{R}, G)) \\ \leq\ &\mathrm{Ex}_{s_I}(tot^{\leq k}(\mathcal{R}, G)) + \Pr_{s_I}(\Box^{\leq k} S_?) \cdot b. \end{aligned}$$

Proof. We consider the objective $tot^{>k}(\mathcal{R}, G)$ which reflects the total reachability reward accumulated *after* the first k steps. Formally, we set for all $\pi = s_0 \to s_1 \to \cdots \in Paths^D_{\mathrm{inf}}$:

$$tot^{>k}(\mathcal{R}, G)(\pi) \ :=\ \begin{cases} 0 & \text{if } \exists\, i \in \{0..k\}\colon\ s_i \in G \\ tot(\mathcal{R}, G)(s_k \to s_{k+1} \to \dots) & \text{otherwise.} \end{cases}$$

It holds that $tot(\mathcal{R}, G)(\pi) = tot^{\leq k}(\mathcal{R}, G) + tot^{>k}(\mathcal{R}, G)$. As a consequence, we get

$$\mathrm{Ex}_{s_I}(tot(\mathcal{R}, G)) \ =\ \mathrm{Ex}_{s_I}(tot^{\leq k}(\mathcal{R}, G)) + \mathrm{Ex}_{s_I}(tot^{>k}(\mathcal{R}, G)). \tag{4.1}$$

Next, we consider the set of finite paths of length k visiting only states in $S_?$, formally given by

$$\Pi_k \ :=\ \left\{ s_0 \to \dots \to s_k \in Paths^D_{\mathrm{fin}} \;\middle|\; \forall\, i \in \{0..k\}\colon\ s_i \in S_? \right\}$$

we have $\Box^{\leq k} S_? = \bigcup_{\hat{\pi} \in \Pi_k} \{\pi \in Paths^D_{\mathrm{inf}} \mid pref(\pi, k) = \hat{\pi}\}$. Using that Π_k is finite (i.e., countable) and that for $\pi \in Paths^D_{\mathrm{inf}} \setminus \Box^{\leq k} S_?$ we have $tot^{>k}(\mathcal{R}, G)(\pi) = 0$, it holds

that

$$\begin{aligned}
\mathrm{Ex}_{s_I}(tot^{>k}(\mathcal{R},G)) &= \int_{\pi\in\Box^{\le k}S_?} tot^{>k}(\mathcal{R},G)(\pi)\,\mathrm{dPr}_{s_I}(\pi)\\
&= \sum_{\hat\pi\in\Pi_k}\int_{\substack{\pi\in Paths^D_{\inf}\\ pref(\pi,k)=\hat\pi}} tot^{>k}(\mathcal{R},G)(\pi)\,\mathrm{dPr}_{s_I}(\pi)\\
&= \sum_{\hat\pi\in\Pi_k}\mathrm{Pr}_{s_I}(\hat\pi)\cdot\int_{\pi\in Paths^D_{\inf}} tot(\mathcal{R},G)(\pi)\,\mathrm{dPr}_{last(\hat\pi)}(\pi)\\
&= \sum_{\hat\pi\in\Pi_k}\mathrm{Pr}_{s_I}(\hat\pi)\cdot\mathrm{Ex}_{last(\hat\pi)}(tot(\mathcal{R},G)). \qquad (4.2)
\end{aligned}$$

Here,

$$\mathrm{Pr}_{s_I}(\hat\pi) = [s_0 = s_I]\cdot\prod_{i=0}^{k-1}\mathbf{P}(s_i,s_{i+1})$$

denotes the probability for the finite path $\hat\pi = s_0\to\ldots\to s_k\in\Pi_k$ which we could pull out of the integral above as it is a common factor for all paths π with prefix $\hat\pi$. With $a,b\in\mathbb{R}$ as in the lemma we get from Equation (4.2) that

$$\begin{aligned}
\mathrm{Ex}_{s_I}(tot^{>k}(\mathcal{R},G)) &\ge \sum_{\hat\pi\in\Pi_k}\mathrm{Pr}_{s_I}(\hat\pi)\cdot a = \mathrm{Pr}_{s_I}(\Box^{\le k}S_?)\cdot a \text{ and}\\
\mathrm{Ex}_{s_I}(tot^{>k}(\mathcal{R},G)) &\le \sum_{\hat\pi\in\Pi_k}\mathrm{Pr}_{s_I}(\hat\pi)\cdot b = \mathrm{Pr}_{s_I}(\Box^{\le k}S_?)\cdot b.
\end{aligned}$$

Together with Equation (4.1) the lemma follows immediately. ∎

Lemma 4.10 provides lower- and upper bounds on the expected total reachability reward at a given state s_I which are tightened arbitrarily when increasing k. Still, it is required to have *some* a priori bounds $a,b\in\mathbb{R}$ on the occurring expected values. Such bounds might be derived from context, e.g., $a=0$ and $b=1$ for probability objectives. Moreover, [BKLP+17] provides efficient methods to compute some valid (but often not very tight) a priori bounds.

When applying sound value iteration, however, it is not required to compute such bounds as we can use the results of our computations to obtain (and refine) the bounds a and b from Lemma 4.10 on-the-fly. For this, we may use the following lemma.

Lemma 4.11 For $k\in\mathbb{N}$ with $\forall\, s\in S_?\colon\ \mathrm{Pr}_s(\Box^{\le k}S_?)<1$ we have for all $\hat s\in S_?$

$$\min_{s\in S_?}\frac{\mathrm{Ex}_s(tot^{\le k}(\mathcal{R},G))}{1-\mathrm{Pr}_s(\Box^{\le k}S_?)} \le \mathrm{Ex}_{\hat s}(tot(\mathcal{R},G)) \le \max_{s\in S_?}\frac{\mathrm{Ex}_s(tot^{\le k}(\mathcal{R},G))}{1-\mathrm{Pr}_s(\Box^{\le k}S_?)}.$$

Proof. We only show the upper bound on $\mathrm{Ex}_{\hat{s}}(tot(\mathcal{R}, G))$. The reasoning for the lower bound is similar. Let $s_{\max} \in \arg\max_{s \in S_?} \mathrm{Ex}_s(tot(\mathcal{R}, G))$. Applying Lemma 4.10 using $s_I = s_{\max}$ and $b = \mathrm{Ex}_{s_{\max}}(tot(\mathcal{R}, G))$ yields

$$\mathrm{Ex}_{s_{\max}}(tot(\mathcal{R}, G)) \;\leq\; \mathrm{Ex}_{s_{\max}}(tot^{\leq k}(\mathcal{R}, G)) + \mathrm{Pr}_{s_{\max}}(\Box^{\leq k} S_?) \cdot \mathrm{Ex}_{s_{\max}}(tot(\mathcal{R}, G)).$$

We solve the inequality for $\mathrm{Ex}_{s_{\max}}(tot(\mathcal{R}, G))$ (assuming $\mathrm{Pr}_s(\Box^{\leq k} S_?) < 1$ for all $s \in S_?$):

$$\mathrm{Ex}_{s_{\max}}(tot(\mathcal{R}, G)) \;\leq\; \frac{\mathrm{Ex}_{s_{\max}}(tot^{\leq k}(\mathcal{R}, G))}{1 - \mathrm{Pr}_{s_{\max}}(\Box^{\leq k} S_?)} \;\leq\; \max_{s \in S_?} \frac{\mathrm{Ex}_s(tot^{\leq k}(\mathcal{R}, G))}{1 - \mathrm{Pr}_s(\Box^{\leq k} S_?)}.$$

The lemma follows by noting $\mathrm{Ex}_{\hat{s}}(tot(\mathcal{R}, G)) \leq \mathrm{Ex}_{s_{\max}}(tot(\mathcal{R}, G))$. ∎

4.4.2 Extending Value Iteration

Our approach is to compute $\mathrm{Ex}_s(tot^{\leq k}(\mathcal{R}, G))$ and $\mathrm{Pr}_s(\Box^{\leq k} S_?)$ for all states $s \in S$ and increasing $k \in \mathbb{N}$. Using Theorem 4.9, this produces sound and arbitrarily tight bounds for $\mathrm{Ex}_{s_I}(tot(\mathcal{R}, G))$. Our algorithm makes use of two separate instances $\mathcal{V}_{\mathcal{R},G}$ and $\mathcal{V}_{\circ,G}$ of the Bellman operator (Definition 4.3), where $\mathcal{V}_{\circ,G}$ assumes that all assigned rewards are 0, i.e., for $\boldsymbol{x} \in \mathbb{R}^{|S|}$ we have

$$\mathcal{V}_{\circ,G}(\boldsymbol{x}) \;=\; \Big\langle [s \notin G] \cdot \textstyle\sum_{s' \in S} \mathbf{P}(s, s') \cdot \boldsymbol{x}(s') \Big\rangle_{s \in S}.$$

Lemma 4.12 For $k \in \mathbb{N}$ and we have

$$\begin{aligned} \big\langle \mathrm{Ex}_s(tot^{\leq k}(\mathcal{R}, G)) \big\rangle_{s \in S} &= \mathcal{V}^k_{\mathcal{R},G}(\mathbf{0}) \;\text{ and} \\ \big\langle \mathrm{Pr}_s(\Box^{\leq k} S_?) \big\rangle_{s \in S} &= \mathcal{V}^k_{\circ,G}(\langle [s \in S_?] \rangle_{s \in S}). \end{aligned}$$

Proof (sketch). Using the terminology of [FKNP11], $\mathrm{Ex}_s(tot^{\leq k}(\mathcal{R}, G))$ is a step-bounded cumulative reward property and $\mathrm{Pr}_s(\Box^{\leq k} S_?)$ is a special instance of an instantaneous reward property. In both cases, [FKNP11, Section 5] provides equations that coincide with the lemma. ∎

Corollary 4.13 For $k > 0$ and $s \in S_?$ we have

$$\begin{aligned} \mathrm{Ex}_s(tot^{\leq k}(\mathcal{R}, G)) &= \sum_{s' \in S} \mathbf{P}(s, s') \cdot \big(\mathrm{Ex}_{s'}(tot^{\leq k-1}(\mathcal{R}, G)) + \mathcal{R}[s, s']\big) \;\text{ and} \\ \mathrm{Pr}_s(\Box^{\leq k} S_?) &= \sum_{s' \in S} \mathbf{P}(s, s') \cdot \mathrm{Pr}_{s'}(\Box^{\leq k-1} S_?) \end{aligned}$$

Input: DTMC D, initial state s_I, reward assignment $\mathcal{R}$, attracting set of goal states $G \subseteq S$, precision $\varepsilon > 0$
Output: $\langle l, u \rangle$ with $l \leq \mathrm{Ex}_{s_I}(tot(\mathcal{R}, G)) \leq u$

1 $\boldsymbol{x}^{(0)} \leftarrow \mathbf{0}; \boldsymbol{y}^{(0)} \leftarrow \left\langle [s \in S_?] \right\rangle_{s \in S}$
2 $a \leftarrow -\infty; b \leftarrow +\infty$ *// or a, b with $\forall\, s \in S_? \colon\ a \leq \mathrm{Ex}_s(tot(\mathcal{R}, G)) \leq b$*
3 **for** $k = 1, 2, 3, \ldots$ **do**
4 $\quad \boldsymbol{x}^{(k)} \leftarrow \mathcal{V}_{\mathcal{R},G}(\boldsymbol{x}^{(k-1)})$ *// i.e., $\boldsymbol{x}^{(k)} = \langle \mathrm{Ex}_s(tot^{\leq k}(\mathcal{R}, G)) \rangle_{s \in S}$*
5 $\quad \boldsymbol{y}^{(k)} \leftarrow \mathcal{V}_{\circ,G}(\boldsymbol{y}^{(k-1)})$ *// i.e., $\boldsymbol{y}^{(k)} = \langle \mathrm{Pr}_s(\Box^{\leq k} S_?) \rangle_{s \in S}$*
6 $\quad$ **if** $\boldsymbol{y}^{(k)} < \mathbf{1}$ **then**
7 $\quad\quad a \leftarrow \max\left(a, \min_{s \in S_?} \frac{\boldsymbol{x}^{(k)}(s)}{1 - \boldsymbol{y}^{(k)}(s)}\right)$
8 $\quad\quad b \leftarrow \min\left(b, \max_{s \in S_?} \frac{\boldsymbol{x}^{(k)}(s)}{1 - \boldsymbol{y}^{(k)}(s)}\right)$
9 $\quad l \leftarrow \boldsymbol{x}^{(k)}(s_I) + a \cdot \boldsymbol{y}^{(k)}(s_I)$
10 $\quad u \leftarrow \boldsymbol{x}^{(k)}(s_I) + b \cdot \boldsymbol{y}^{(k)}(s_I)$
11 $\quad$ **if** $u - l < \varepsilon$ **then return** $\langle l, u \rangle$

Algorithm 4.4: The sound value iteration algorithm for DTMCs

We compute $\left\langle \mathrm{Ex}_s(tot^{\leq k}(\mathcal{R}, G)) \right\rangle_{s \in S}$ for $k \in \mathbb{N}$ by performing k iterations of value iteration as in Algorithm 4.3. $\left\langle \mathrm{Pr}_s(\Box^{\leq k} S_?) \right\rangle_{s \in S}$ is computed similarly—except that we consider different starting values and the operator $\mathcal{V}_{\circ,G}$ instead of $\mathcal{V}_{\mathcal{R},G}$. Algorithm 4.4 combines the two value iteration instances using vectors $\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(k)}$ in iteration $k \in \mathbb{N}$. In addition, we maintain values a and b that—with Lemma 4.11—satisfy the invariant

$$\forall\, s \in S_? \colon\ a \leq \mathrm{Ex}_s(tot(\mathcal{R}, G)) \leq b.$$

In Line 2 it is also possible to initialize a and b with any values that satisfy this invariant which—in practice—may improve the runtime of our algorithm. As mentioned before, such bounds on the occurring expected values can sometimes be inferred from context or by applying ideas of [BKLP+17]. In Lines 9 and 10 we apply Lemma 4.10 to obtain sound lower and upper bounds l and u for the expected total reachability reward at the initial state. Finally, Line 11 checks if those bounds are sufficiently tight and thus providing a solution to the DTMC variant of Problem 4.1. Since $\mathrm{Pr}_s(\Box^{\leq k} S_?)$ approaches 0 for increasing k, the procedure terminates for any $\varepsilon > 0$.

Theorem 4.14 Algorithm 4.4 terminates for any given input with a solution $\langle l, u \rangle$ for Problem 4.1.

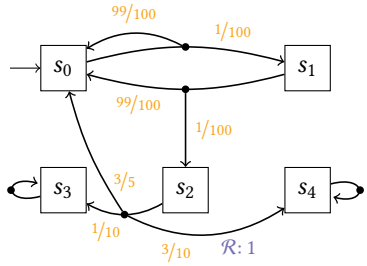


Figure 4.6: Example DTMC (cf. Example 4.6)

Example 4.6 We apply Algorithm 4.4 to the DTMC D in Figure 4.6 which is the induced submodel for the MDP from Figure 4.5a and the strategy that always selects α at s_0. As in Example 4.5 we consider the depicted reward assignment $\mathcal{R}$ and attracting set $G = \{s_3, s_4\}$. After $k = 3$ iterations we get (assuming order $s_0 \prec s_1 \prec s_2$ and omitting entries for $s_3, s_4 \in G$):

$$\boldsymbol{x}^{(3)} = \langle 0.00003,\ 0.003,\ 0.3 \rangle \quad \text{and} \quad \boldsymbol{y}^{(3)} = \langle 0.99996,\ 0.996,\ 0.6 \rangle$$

Moreover, it is $a = b = 0.75$ since

$$\forall\, s \in S_? \colon \frac{\boldsymbol{x}^{(3)}(s)}{1 - \boldsymbol{y}^{(3)}(s)} = \frac{3}{4} = 0.75.$$

The algorithm terminates for any $\varepsilon > 0$ since $l = u = 0.00003 + 0.99996 \cdot 0.75 = 0.75$ which indeed coincides with the value $\mathrm{Ex}^D_{s_I}(tot(\mathcal{R}, G))$ that we computed in Example 4.5.
In contrast to sound value iteration, classical value iteration converges very slowly in this example (cf. Figure 4.5b). Thus, even if some oracle could give us the minimum number of required iterations to obtain an ε-approximation with value iteration, *sound* value iteration would still converge faster (assuming ε is significantly smaller than $\mathrm{Ex}^D_{s_I}(tot(\mathcal{R}, G)) = 0.75$).

We remark that the values $l = \boldsymbol{x}^{(k)}(s_I) + a \cdot \boldsymbol{y}^{(k)}(s_I)$ and $u = \boldsymbol{x}^{(k)}(s_I) + b \cdot \boldsymbol{y}^{(k)}(s_I)$ returned by Algorithm 4.4 correspond to the initial state value obtained when applying the Bellman operator k times with starting vector $\langle a \cdot [s \in S_?] \rangle_{s \in S}$ and $\langle b \cdot [s \in S_?] \rangle_{s \in S}$, respectively. More generally, after running Algorithm 4.4 for k iterations, we can easily obtain $\mathcal{V}^k_{\mathcal{R},G}(\langle c \cdot [s \in S_?] \rangle_{s \in S})$ for arbitrary $c \in \mathbb{R}$ without restarting the iterations from scratch. This is an advantage of sound value iteration over the related *interval iteration* which has to find suitable starting vectors a priori (cf. Section 4.6).

Lemma 4.15 After k iterations of Algorithm 4.4 we have

$$\forall c \in \mathbb{R}\colon\ \boldsymbol{x}^{(k)} + c \cdot \boldsymbol{y}^{(k)} \ = \ \mathcal{V}^k_{\mathcal{R},G}(\langle c \cdot [s \in S_?]\rangle_{s\in S}).$$

Proof. The proof is by induction over k. For $k = 0$ we get

$$\boldsymbol{x}^{(0)} + c \cdot \boldsymbol{y}^{(0)} \ = \ \langle c \cdot [s \in S_?]\rangle_{s\in S} \ = \ \mathcal{V}^0_{\mathcal{R},G}(\langle c \cdot [s \in S_?]\rangle_{s\in S}).$$

Now assume $k > 0$. For $s \in G$ we have

$$\boldsymbol{x}^{(k)}(s) + c \cdot \boldsymbol{y}^{(k)}(s) \ = \ 0 \ = \ \mathcal{V}^k_{\mathcal{R},G}(\langle c \cdot [\hat{s} \in S_?]\rangle_{\hat{s}\in S})(s)$$

and for $s \in S_?$ we get

$$\begin{aligned}
&\boldsymbol{x}^{(k)}(s) + c \cdot \boldsymbol{y}^{(k)}(s)\\
&= \mathcal{V}_{\mathcal{R},G}(\boldsymbol{x}^{(k-1)})(s) + c \cdot \mathcal{V}_{\circ,G}(\boldsymbol{y}^{(k-1)})(s)\\
&= \sum_{s'\in S} \mathbf{P}(s,s') \cdot \left(\boldsymbol{x}^{(k-1)}(s') + \mathcal{R}[s,s']\right) + c \cdot \sum_{s'\in S} \mathbf{P}(s,s') \cdot \boldsymbol{y}^{(k-1)}(s')\\
&= \sum_{s'\in S} \mathbf{P}(s,s') \cdot \left(\boldsymbol{x}^{(k-1)}(s') + c \cdot \boldsymbol{y}^{(k-1)}(s') + \mathcal{R}[s,s']\right)\\
&= \sum_{s'\in S} \mathbf{P}(s,s') \cdot \left(\mathcal{V}^{k-1}_{\mathcal{R},G}(\langle c \cdot [\hat{s} \in S_?]\rangle_{\hat{s}\in S})(s') + \mathcal{R}[s,s']\right)\\
&= \mathcal{V}_{\mathcal{R},G}(\mathcal{V}^{k-1}_{\mathcal{R},G}(\langle c \cdot [\hat{s} \in S_?]\rangle_{\hat{s}\in S}))(s) \ = \ \mathcal{V}^k_{\mathcal{R},G}(\langle c \cdot [\hat{s} \in S_?]\rangle_{\hat{s}\in S})(s).
\end{aligned}$$

■

4.4.3 Sound Gauss-Seidel Value Iteration

We lift the idea of *Gauss-Seidel value iteration* [Put94]—a well-known optimization of the classical value iteration algorithm—to sound value iteration.

Gauss-Seidel value iteration

Recall that we assume some arbitrary but fixed order $s_1 \prec \ldots \prec s_n$ on the states $S = \{s_1, \ldots, s_n\}$ of the DTMC D. We write $s \preceq s'$ iff either $s = s'$ or $s \prec s'$. For Gauss-Seidel value iteration we consider the operator $\mathcal{U}_{\mathcal{R},G}$ instead of $\mathcal{V}_{\mathcal{R},G}$.

Definition 4.4 (Gauss-Seidel Operator) For a reward assignment $\mathcal{R}$ and a set of states G of a DTMC, the *Gauss-Seidel operator* $\mathcal{U}_{\mathcal{R},G}\colon \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ is given by

Gauss-Seidel operator

$$\mathcal{U}_{\mathcal{R},G}(\boldsymbol{x}) \ \coloneqq \ \Big\langle [s \notin G] \cdot \sum_{s'\in S} \mathbf{P}(s,\alpha,s') \cdot \big([s' \prec s] \cdot \mathcal{U}_{\mathcal{R},G}(\boldsymbol{x}(s')) + [s \preceq s'] \cdot \boldsymbol{x}(s') \ + \ \mathcal{R}[s,\alpha,s']\big)\Big\rangle_{s\in S}$$

for all $\boldsymbol{x} \in \mathbb{R}^{|S|}$.

Whereas the Bellman operator from Definition 4.3 considers only the values from the given input vector (i.e., the result from the previous iteration), the Gauss-Seidel operator always considers the most recently computed value for each state. This is well-defined when computing the state values in the order given by "$\prec$". The benefit of using the Gauss-Seidel operator is twofold. Firstly, Gauss-Seidel value iteration intuitively converges faster since it always uses the most recent values. Secondly, it is possible to implement Gauss-Seidel value iteration using a single value vector $\boldsymbol{x} \in \mathbb{R}^{|S|}$ while an implementation of classical value iteration requires at least two vectors $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{|S|}$ for storing the values of both the current and the previous iteration.

The Gauss-Seidel variant of sound value iteration arises from Algorithm 4.4 by replacing the Bellman operators $\mathcal{V}_{\mathcal{R},G}$ and $\mathcal{V}_{\circ,G}$ by the Gauss-Seidel operators $\mathcal{U}_{\mathcal{R},G}$ and $\mathcal{U}_{\circ,G}$. We now argue that this modified algorithm still yields a solution to Problem 4.1, i.e., sound ε-approximations of the expected reachability reward $\mathrm{Ex}_{s_I}(tot(\mathcal{R},G))$.

For $k \in \mathbb{N}$ we define the objective $tot^{\preceq k}_{\prec}(\mathcal{R},G)$ and the set of paths $\Box^{\preceq k}_{\prec} S_?$ which are similar to $tot^{\leq k}(\mathcal{R},G)$ and $\Box^{\leq k} S_?$ but we only "count" steps from s to s' iff $s \preceq s'$ holds. Formally, for $\pi = s_0 \to s_1 \to \cdots \in Paths^D_{\inf}$ let $n^G_\pi := \min\{i \in \mathbb{N} \mid s_i \in G\}$, i.e., the state $s_{n^G_\pi}$ is the first goal state visited by π or $n^G_\pi = \infty$ if no goal state is visited. Furthermore, let

$$n^k_{\pi,\preceq} \;:=\; \min\left\{n \in \mathbb{N} \mid \textstyle\sum_{i=0}^{n-1} [\![s_i \preceq s_{i+1}]\!] = k\right\},$$

i.e., $pref(\pi, n^k_{\pi,\preceq})$ is the smallest prefix of π in which we have k steps from a state s_i to a state s_{i+1} with $s_i \preceq s_{i+1}$. Since $|S| < \infty$, we can follow that $n^k_{\pi,\preceq} < \infty$ for all paths π and $k \in \mathbb{N}$. We set

$$\begin{aligned} \forall\, \pi \in Paths^D_{\inf}\colon\ tot^{\preceq k}_{\prec}(\mathcal{R},G)(\pi) \;&:=\; \mathcal{R}[pref(\pi, \min(n^G_\pi, n^k_{\pi,\preceq}))] \text{ and} \\ \Box^{\preceq k}_{\prec} S_? \;&:=\; \{\pi \in Paths^D_{\inf} \mid n^G_\pi > n^k_{\pi,\preceq}\}. \end{aligned}$$

The correctness of the Gauss-Seidel variant of sound value iteration follows by restating our results from Sections 4.4.1 and 4.4.2 using

- $tot^{\preceq k}_{\prec}(\mathcal{R},G)$ instead of $tot^{\leq k}(\mathcal{R},G)$,
- $\Box^{\preceq k}_{\prec} S_?$ instead of $\Box^{\leq k} S_?$, and
- $\mathcal{U}_{\mathcal{R},G}$ instead of $\mathcal{V}_{\mathcal{R},G}$.

In particular, we obtain the following analogons to Theorem 4.9 and Lemma 4.12 whose proofs can be adapted straightforwardly.

Theorem 4.16 For $k \in \mathbb{N}$ with $\forall s \in S_?\colon \Pr_s(\Box^{\leq k}_{<} S_?) < 1$ we have

$$\begin{aligned} &\mathrm{Ex}_{s_I}(tot^{\leq k}_{<}(\mathcal{R}, G)) + \Pr_{s_I}(\Box^{\leq k}_{<} S_?) \cdot \min_{s \in S_?} \frac{\mathrm{Ex}_s(tot^{\leq k}_{<}(\mathcal{R}, G))}{1 - \Pr_s(\Box^{\leq k}_{<} S_?)} \\ \leq\ &\mathrm{Ex}_{s_I}(tot(\mathcal{R}, G)) \\ \leq\ &\mathrm{Ex}_{s_I}(tot^{\leq k}_{<}(\mathcal{R}, G)) + \Pr_{s_I}(\Box^{\leq k}_{<} S_?) \cdot \max_{s \in S_?} \frac{\mathrm{Ex}_s(tot^{\leq k}_{<}(\mathcal{R}, G))}{1 - \Pr_s(\Box^{\leq k}_{<} S_?)}\,. \end{aligned}$$

Lemma 4.17 For $k \in \mathbb{N}$ we have

$$\begin{aligned} \left\langle \mathrm{Ex}_s(tot^{\leq k}_{<}(\mathcal{R}, G)) \right\rangle_{s \in S} &= \mathcal{U}^k_{\mathcal{R},G}(\mathbf{0}) \text{ and} \\ \left\langle \Pr_s(\Box^{\leq k}_{<} S_?) \right\rangle_{s \in S} &= \mathcal{U}^k_{\circ,G}(\langle [s \in S_?] \rangle_{s \in S}). \end{aligned}$$

4.4.4 Sound Topological Value Iteration

Recall the decomposition of DTMC D into maximal strongly connected components $SCC(D)$ from Section 2.2.6. For $C, C' \in SCC(D)$ we write $C \hookrightarrow C'$ iff there is a path from some $s \in states(C)$ to some $s' \in states(C')$. We can show that $\hookrightarrow$ yields a partial order. In particular, $C \hookrightarrow C'$ and $C' \hookrightarrow C$ implies $C = C$ because otherwise, $C \cup C'$ would be a (larger) SCC, violating the assumption that C and C' are maximal. Furthermore, the expected total reward at some $s \in states(C)$ only depends on the SCC C itself and the SCCs C' with $C \hookrightarrow C'$. Other SCCs C'' with $C \not\hookrightarrow C''$ are not reachable from s.

These observations give rise to *topological value iteration* [CBGK08; DMWG11]—an improvement of the classical value iteration algorithm that we now lift to sound value iteration. Instead of analyzing the complete model at once, the idea is to analyze the maximal SCCs one-after-the-other. Let $\dot{\hookrightarrow}$ be some linearization of $\hookrightarrow$, i.e.,

topological value iteration

- $C \hookrightarrow C'$ implies $C \mathrel{\dot{\hookrightarrow}} C'$ and
- $C_1 \mathrel{\dot{\hookrightarrow}} \cdots \mathrel{\dot{\hookrightarrow}} C_n$ for $SCC(D) = \{C_1, \ldots, C_n\}$.

We compute the expected rewards within each SCC in reverse order from C_n to C_1.

Definition 4.5 (SCC Model) For SCC C of DTMC D, the *SCC model* is the DTMC $D[\![C]\!]^\perp = \langle states(C) \uplus \{s_\perp\}, \mathbf{P}[\![C]\!]^\perp \rangle$, where for $s, s' \in states(C)$

SCC model

- $\mathbf{P}[\![C]\!]^\perp(s, s') = \mathbf{P}(s, s')$,
- $\mathbf{P}[\![C]\!]^\perp(s, s_\perp) = \sum_{\hat{s} \in S \setminus states(C)} \mathbf{P}(s, \hat{s})$, and

- $\mathbf{P}[\![C]\!]^{\perp}(s_{\perp}, s_{\perp}) = 1$.

The *SCC Rewards* of reward assignment $\mathcal{R}$ for D and a vector $\boldsymbol{x} \in \mathbb{R}^{|S|}$ is the reward assignment $\langle \mathcal{R}, \boldsymbol{x} \rangle[\![C]\!]^{\perp}$ for $D[\![C]\!]^{\perp}$, where for $s, s' \in \mathit{states}(C)$

- $\langle \mathcal{R}, \boldsymbol{x} \rangle[\![C]\!]^{\perp}[s, s'] = \mathcal{R}[s, s']$,
- $\langle \mathcal{R}, \boldsymbol{x} \rangle[\![C]\!]^{\perp}[s, s_{\perp}] = \sum_{\hat{s} \in S \setminus \mathit{states}(C)} \left(\frac{\mathbf{P}(s, \hat{s})}{\mathbf{P}[\![C]\!]^{\perp}(s, s_{\perp})} \cdot \left(\mathcal{R}[s, \hat{s}] + \boldsymbol{x}(\hat{s}) \right) \right)$, if $s_{\perp} \in \mathit{post}(s)$, and
- $\langle \mathcal{R}, \boldsymbol{x} \rangle[\![C]\!]^{\perp}[s_{\perp}, s_{\perp}] = 0$.

SCC models are similar to submodels (cf. Definition 2.20 on page 41) except that the considered component C must not necessarily be closed. Transitions leading to states $\hat{s} \in S \setminus \mathit{states}(C)$ are redirected to a dedicated sink state $s_{\perp}$. SCC rewards are a straightforward restriction of a reward assignment $\mathcal{R}$ to an SCC model $D[\![C]\!]^{\perp}$. However, when exiting the SCC C by moving from $s \in \mathit{states}(C)$ to $\hat{s} \notin \mathit{states}(C)$, we consider the reward $\mathcal{R}[s, \hat{s}]$ plus a one-off reward given by $\boldsymbol{x}(\hat{s})$ for some vector $\boldsymbol{x}$. Only the values $\boldsymbol{x}(\hat{s})$ for states $\hat{s}$ reachable from C—i.e., $\hat{s} \in \mathit{states}(C')$ for some $C' \in \mathit{SCC}(D)$ with $C \hookrightarrow C'$—are relevant.

The following theorem relates total rewards in D to total rewards in $D[\![C]\!]^{\perp}$ by slightly generalizing results from [DMWG11] towards *sound* approximations. We say that two values $a, b \in \mathbb{R}$ are ε-close for some $\varepsilon \in \mathbb{R}_{\geq 0}$ iff $|a - b| \leq \varepsilon$.

Theorem 4.18 For $C \in \mathit{SCC}(D)$ and some $\varepsilon \in \mathbb{R}_{\geq 0}$ consider a vector $\boldsymbol{x} \in \mathbb{R}^{|S|}$ such that $\boldsymbol{x}(\hat{s})$ and $\mathrm{Ex}^{D}_{\hat{s}}(\mathit{tot}(\mathcal{R}, G))$ are ε-close for all states $\hat{s} \in S \setminus \mathit{states}(C)$ that are reachable from C. Then, $\mathrm{Ex}^{D}_{s}(\mathit{tot}(\mathcal{R}, G))$ and $\mathrm{Ex}^{D[\![C]\!]^{\perp}}_{s}(\mathit{tot}(\langle \mathcal{R}, \boldsymbol{x} \rangle[\![C]\!]^{\perp}, G))$ are ε-close for all $s \in \mathit{states}(C)$.

Proof. Let $\boldsymbol{x}^{*}, \boldsymbol{x}^{\mathrm{diff}} \in \mathbb{R}^{|S|}$ be given by

$$\boldsymbol{x}^{*} \coloneqq \left\langle \mathrm{Ex}^{D}_{s}(\mathit{tot}(\mathcal{R}, G)) \right\rangle_{s \in S} \quad \text{and} \quad \boldsymbol{x}^{\mathrm{diff}} \coloneqq \boldsymbol{x} - \boldsymbol{x}^{*}.$$

Observe that $\boldsymbol{x} = \boldsymbol{x}^{*} + \boldsymbol{x}^{\mathrm{diff}}$. Using a similar calculation as in the proof of Lemma 4.4 we get for $s \in \mathit{states}(C)$:

$$\begin{aligned} \mathrm{Ex}^{D[\![C]\!]^{\perp}}_{s}(\mathit{tot}(\langle \mathcal{R}, \boldsymbol{x} \rangle[\![C]\!]^{\perp}, G)) &= \mathrm{Ex}^{D[\![C]\!]^{\perp}}_{s}(\mathit{tot}(\langle \mathcal{R}, \boldsymbol{x}^{*} \rangle[\![C]\!]^{\perp}, G)) + \\ &\quad \mathrm{Ex}^{D[\![C]\!]^{\perp}}_{s}(\mathit{tot}(\langle \boldsymbol{x}^{\mathrm{diff}} \rangle[\![C]\!]^{\perp}, G)). \end{aligned} \tag{4.3}$$

Here, $\langle \boldsymbol{x}^{\text{diff}} \rangle [\![C]\!]^{\perp}$ is the reward assignment for the SCC model $D[\![C]\!]^{\perp}$ that only assigns the rewards given by $\boldsymbol{x}^{\text{diff}}$ when moving to $s_{\perp}$ and all other rewards are assumed to be 0.
With Theorem 4.8 we get that $\boldsymbol{x}^*$ is the fixpoint of the Bellman operator $\mathcal{V}_{\mathcal{R},G}$ with respect to the DTMC D. Let $\boldsymbol{x}^C$ be the fixpoint of the Bellman operator $\mathcal{V}_{\langle \mathcal{R}, \boldsymbol{x} \rangle [\![C]\!]^{\perp}, G}$ with respect to the SCC model $D[\![C]\!]^{\perp}$. We get

$$
\begin{aligned}
\boldsymbol{x}^C(s) &= \mathcal{V}_{\langle \mathcal{R}, \boldsymbol{x} \rangle [\![C]\!]^{\perp}, G}(\boldsymbol{x}^C)(s) \\
&= [s \notin G] \cdot \sum_{s' \in states(C) \uplus \{s_{\perp}\}} \mathbf{P}[\![C]\!]^{\perp}(s, s') \cdot \left(\boldsymbol{x}^C(s') + \langle \mathcal{R}, \boldsymbol{x}^* \rangle [\![C]\!]^{\perp}[s, s'] \right) \\
&= [s \notin G] \cdot \Bigg(\sum_{s' \in states(C)} \Big(\mathbf{P}(s, s') \cdot \left(\boldsymbol{x}^C(s') + \mathcal{R}[s, s'] \right) \Big) + \\
&\qquad \mathbf{P}[\![C]\!]^{\perp}(s, s_{\perp}) \cdot \sum_{\hat{s} \in S \setminus states(C)} \Big(\frac{\mathbf{P}(s, \hat{s})}{\mathbf{P}[\![C]\!]^{\perp}(s, s_{\perp})} \cdot \left(\mathcal{R}[s, \hat{s}] + \boldsymbol{x}^*(\hat{s}) \right) \Big) \Bigg) \\
&= [s \notin G] \cdot \sum_{s' \in S} \Big(\mathbf{P}(s, s') \cdot \left(\hat{\boldsymbol{x}}(s') + \mathcal{R}[s, s'] \right) \Big) \;=\; \mathcal{V}_{\mathcal{R},G}(\hat{\boldsymbol{x}})(s)
\end{aligned}
$$

with $\hat{\boldsymbol{x}} \in \mathbb{R}^{|S|}$ given for $s \in S$ by

$$
\hat{\boldsymbol{x}}(s) = \begin{cases} \boldsymbol{x}^C(s) & \text{if } s \in states(C) \\ \boldsymbol{x}^*(s) & \text{if } s \notin states(C). \end{cases}
$$

Thus, $\hat{\boldsymbol{x}}$ is a fixpoint of the Bellman operator $\mathcal{V}_{\mathcal{R},G}$ for DTMC D. Since the fixpoint is unique, we get $\hat{\boldsymbol{x}} = \boldsymbol{x}^*$. Therefore,

$$
\mathrm{Ex}_s^{D[\![C]\!]^{\perp}}(tot(\langle \mathcal{R}, \boldsymbol{x}^* \rangle [\![C]\!]^{\perp}, G)) \;=\; \boldsymbol{x}^C(s) \;=\; \boldsymbol{x}^*(s) \;=\; \mathrm{Ex}_s^D(tot(\mathcal{R}, G)).
$$

It remains to show that the absolute value of the second summand in Equation (4.3) is at most ε. Since any path of $D[\![C]\!]^{\perp}$ can only take a transition from $s \in states(C)$ to $s_{\perp}$ at most once, a non-zero reward for $\langle \boldsymbol{x}^{\text{diff}} \rangle [\![C]\!]^{\perp}$ can also only be collected at most once per path. Since $|\boldsymbol{x}^{\text{diff}}(\hat{s})| \leq \varepsilon$ for all $\hat{s}$, we get for $s \in states(C)$:

$$
\left| \mathrm{Ex}_s^{D[\![C]\!]^{\perp}}(tot(\langle \boldsymbol{x}^{\text{diff}} \rangle [\![C]\!]^{\perp}, G)) \right| \;\leq\; \varepsilon. \qquad \blacksquare
$$

Algorithm 4.5 makes use of Theorem 4.18 to provide a solution to Problem 4.1. In Line 1 it decomposes the input DTMC into its maximal SCCs and computes some

linearization $\hookrightarrow$ of the order $\hookrightarrow$ discussed above[6]. Line 2 computes the size m of the largest chain of the order $\hookrightarrow$, i.e., the maximal number of different SCCs that a path of D can visit. This number is used to account for approximation errors that can accumulate as discussed below. Lines 4 to 6 process the SCCs in reverse order, starting with C_m (which necessarily is a BSCC). For each processed SCC C_i, a sound approximation of the values $\mathrm{Ex}_s^{D[\![C_i]\!]^\perp}(tot(\langle \mathcal{R}, \boldsymbol{x}\rangle[\![C_i]\!]^\perp, G))$ for $s \in states(C_i)$ is obtained and the corresponding entries of $\boldsymbol{x}$ are updated. In this step, we can make use of Algorithm 4.4—or any other approach that provides sound approximations (cf. Section 4.6). After all SCCs are processed, the algorithm returns an interval of width ε around the value $\boldsymbol{x}(s_I)$ as a solution to (the DTMC variant of) Problem 4.1.

In Line 6, it is necessary to divide the considered precisions by m because the obtained values $\boldsymbol{x}(s)$ are only *approximations* of $\mathrm{Ex}_s^{D[\![C_i]\!]^\perp}(tot(\langle \mathcal{R}, \boldsymbol{x}\rangle[\![C_i]\!]^\perp, G))$. To elaborate on this argument, suppose that for all states $\hat{s}$ reachable from the currently processed SCC C_i we have that $\boldsymbol{x}(\hat{s})$ and $\mathrm{Ex}_{\hat{s}}^{D}(tot(\mathcal{R}, G))$ are ε'-close for some $\varepsilon' \in \mathbb{R}_{\geq 0}$. Then, for $s \in states(C_i)$ we know by Theorem 4.18 that $\mathrm{Ex}_s^{D}(tot(\mathcal{R}, G))$ and $\mathrm{Ex}_s^{D[\![C_i]\!]^\perp}(tot(\langle \mathcal{R}, \boldsymbol{x}\rangle[\![C_i]\!]^\perp, G))$ are ε'-close. However, the value v_s computed in Line 6 is only $\frac{\varepsilon}{2\cdot m}$-close to $\mathrm{Ex}_s^{D[\![C_i]\!]^\perp}(tot(\langle \mathcal{R}, \boldsymbol{x}\rangle[\![C_i]\!]^\perp, G))$. Therefore, we can only infer that v_s is $(\frac{\varepsilon}{2\cdot m} + \varepsilon')$-close to $\mathrm{Ex}_s^{D}(tot(\mathcal{R}, G))$. Such an error accumulation can happen at most m times, yielding a worst-case error of $m \cdot \frac{\varepsilon}{2\cdot m} = \frac{\varepsilon}{2}$, i.e., the value $\boldsymbol{x}(s_I)$ is at least $\frac{\varepsilon}{2}$-close to $\mathrm{Ex}_{s_I}^{D}(tot(\mathcal{R}, G))$. A similar accumulation of approximation errors when analyzing components of an MDP individually has been observed in [HH16].

The values v_s from Line 6 are only relevant for the initial state s_I and for states s that have an *incoming* transition from another SCC $C_j \hookrightarrow C_i$, $j \leq i$. Hence, in the case of Algorithm 4.4 we can stop its iterations once the values at those relevant states are sufficiently precise.

4.5 Sound Value Iteration for MDPs

SVI, sound value iteration

Set-up | In this section we consider an MDP $M = \langle S, Act, \Delta, \mathbf{P}\rangle$ with initial state $s_I \in S$, reward assignment $\mathcal{R}$, attracting goal state set $G \subseteq S$, and a precision parameter $\varepsilon > 0$. Our goal is to provide an answer to the corresponding instance of Problem 4.1.

Notations | As in Section 4.4, we consider the set of states $S_? := S \setminus G$ as well as the objective $tot^{\leq k}(\mathcal{R}, G)$ and the set of paths $\square^{\leq k} S_?$ with $k \in \mathbb{N}$, whose definitions are straightforwardly lifted to MDPs.

[6] $\hookrightarrow$ can be obtained as a byproduct of Tarjan's SCC decomposition algorithm [Tar72].

Input: DTMC D, initial state s_I, reward assignment $\mathcal{R}$, attracting set of goal states $G \subseteq S$, precision $\varepsilon > 0$
Output: $\langle l, u \rangle$ with $l \leq \mathrm{Ex}_{s_I}(tot(\mathcal{R}, G)) \leq u$

$\{C_1, \dots C_n\} \leftarrow SCC(D)$ with $C_1 \hookrightarrow \cdots \hookrightarrow C_n$
$m \leftarrow$ size of largest chain $C_{i_1} \hookrightarrow \dots \hookrightarrow C_{i_m}$
$\boldsymbol{x} \leftarrow \mathbf{0}$
for $i = n, n-1, \dots, 1$ **do**
  for $s \in states(C_i)$ **do** *// Analyze C_i e.g., with Algorithm 4.4*
    $\boldsymbol{x}(s) \leftarrow v_s$ with $\left|\mathrm{Ex}_s^{D[\![C_i]\!]^\perp}(tot(\langle \mathcal{R}, \boldsymbol{x} \rangle [\![C_i]\!]^\perp, G)) - v_s\right| \leq \frac{\varepsilon}{2 \cdot m}$
return $\langle \boldsymbol{x}(s_I) - \frac{\varepsilon}{2},\ \boldsymbol{x}(s_I) + \frac{\varepsilon}{2} \rangle$

Algorithm 4.5: Sound topological value iteration

4.5.1 From DTMCs to MDPs

Our results from Section 4.4.1 also hold for the DTMC $M[\![\sigma]\!]$ induced by MDP M and pure memoryless strategy $\sigma \in \Sigma^M_{\mathrm{PM}}$. Hence, if an optimal strategy—i.e., a strategy σ inducing the maximal expected total reachability reward—would be known a priori, we could build the induced DTMC and apply sound value iteration for DTMCs to obtain sound lower and upper bounds for $\mathrm{Ex}^{M[\![\sigma]\!]}_{s_I}(tot(\mathcal{R}, G)) = \mathrm{Ex}^M_{\max, s_I}(tot(\mathcal{R}, G))$. Even if the strategy σ is not optimal, a lower bound for the expected reward for $M[\![\sigma]\!]$ is also a lower bound for the maximal expected reward for M since

$$\forall\, \sigma \in \Sigma^M_{\mathrm{PM}} \colon\ \mathrm{Ex}^{M[\![\sigma]\!]}_{s_I}(tot(\mathcal{R}, G)) \ \leq\ \mathrm{Ex}^M_{\max, s_I}(tot(\mathcal{R}, G)).$$

However, such a statement can not be made for upper bounds. Thus, *the key challenge in lifting sound value iteration to MDPs is finding strategy choices that yield a valid upper bound.* Instead of finding such a strategy a priori, our approach is to build a strategy on-the-fly, during the iteration steps. This requires to lift our observations towards strategies that repeat their choices every k steps.

Definition 4.6 For $k > 0$, a strategy $\sigma \in \Sigma^M$ is called *k-repeating* iff

k-repeating strategy

$$\forall\, \hat{\pi} \in Paths^M_{\mathrm{fin}} \colon\ \sigma(\hat{\pi}) \ =\ \sigma(pref(\hat{\pi}, |\hat{\pi}| \bmod k)).$$

The set of k-repeating strategies for MDP M is denoted by Σ^M_k.

A strategy is memoryless iff it is 1-repeating. Furthermore, any memoryless strategy is also k-repeating for all $k > 0$. Hence, maximal expected total reachability rewards can always be attained by some pure, k-repeating strategy.

We lift Lemmas 4.10 and 4.11 to MDPs under a fixed k-repeating strategy.

Lemma 4.19 For $k > 0$, $\sigma \in \Sigma_k^M$, and $a, b \in \mathbb{R}$ with $\forall\, s \in S_?$:

$$a \leq \mathrm{Ex}_{\sigma,s}(tot(\mathcal{R}, G)) \leq b$$

we have:

$$\begin{aligned} &\mathrm{Ex}_{\sigma,s_I}(tot^{\leq k}(\mathcal{R}, G)) + \Pr_{\sigma,s_I}(\Box^{\leq k} S_?) \cdot a \\ \leq\ &\mathrm{Ex}_{\sigma,s_I}(tot(\mathcal{R}, G)) \\ \leq\ &\mathrm{Ex}_{\sigma,s_I}(tot^{\leq k}(\mathcal{R}, G)) + \Pr_{\sigma,s_I}(\Box^{\leq k} S_?) \cdot b. \end{aligned}$$

Proof. Let $\Pi \subseteq Paths_{\mathrm{inf}}^M$ be an arbitrary set of paths with some common prefix $\hat{\pi} = pref(\pi, k)$ for all $\pi \in \Pi$. Furthermore, let

$$\Pi' \;=\; \left\{ last(\hat{\pi}) \xrightarrow{\alpha_k} s_{k+1} \xrightarrow{\alpha_{k+1}} \cdots \in Paths_{\mathrm{inf}}^M \;\middle|\; \hat{\pi} \xrightarrow{\alpha_k} s_{k+1} \xrightarrow{\alpha_{k+1}} \cdots \in \Pi \right\}$$

be the set of postfixes of paths from Π starting from position k. Since $\sigma \in \Sigma_k^M$ repeats its decisions after k steps, i.e., after observing $\hat{\pi}$, we can show that

$$\Pr_{\sigma,s_I}(\Pi) \;=\; \Pr_{\sigma,s_I}(\hat{\pi}) \cdot \Pr_{\sigma,last(\hat{\pi})}(\Pi').$$

Using this observation, we can straightforwardly adapt the proof of Lemma 4.10 to also show Lemma 4.19. ∎

Lemma 4.20 For $k > 0$ and $\sigma \in \Sigma_k^M$ with $\forall\, s \in S_?\colon \Pr_{\sigma,s}(\Box^{\leq k} S_?) < 1$ we have for all $\hat{s} \in S_?$:

$$\min_{s \in S_?} \frac{\mathrm{Ex}_{\sigma,s}(tot^{\leq k}(\mathcal{R}, G))}{1 - \Pr_{\sigma,s}(\Box^{\leq k} S_?)} \;\leq\; \mathrm{Ex}_{\sigma,\hat{s}}(tot(\mathcal{R}, G)) \;\leq\; \max_{s \in S_?} \frac{\mathrm{Ex}_{\sigma,s}(tot^{\leq k}(\mathcal{R}, G))}{1 - \Pr_{\sigma,s}(\Box^{\leq k} S_?)}.$$

Proof. Using Lemma 4.19, the proof is analog to the proof of Lemma 4.11. ∎

Any k-repeating strategy $\sigma \in \Sigma_k^M$ can be used to provide a valid lower bound for $\mathrm{Ex}_{\max,s_I}(tot(\mathcal{R}, G))$ since

$$\begin{aligned} &\mathrm{Ex}_{\sigma,s_I}(tot^{\leq k}(\mathcal{R}, G)) + \Pr_{\sigma,s_I}(\Box^{\leq k} S_?) \cdot \min_{s \in S_?} \frac{\mathrm{Ex}_{\sigma,s}(tot^{\leq k}(\mathcal{R}, G))}{1 - \Pr_s(\Box^{\leq k} S_?)} \\ \leq\ &\mathrm{Ex}_{\sigma,s_I}(tot(\mathcal{R}, G)) \;\leq\; \mathrm{Ex}_{\max,s_I}(tot(\mathcal{R}, G)). \end{aligned}$$

However, using Lemma 4.19, we only get an upper bound for the expected value

induced by σ:

$$\mathrm{Ex}_{\sigma,s_I}(tot(\mathcal{R},G)) \;\leq\; \mathrm{Ex}_{\sigma,s_I}(tot^{\leq k}(\mathcal{R},G)) + \mathrm{Pr}_{\sigma,s_I}(\Box^{\leq k} S_?) \cdot b,$$

where some fixed $b \in \mathbb{R}$ as in Lemma 4.19 is assumed. In the case where σ does induce the maximal expected value, this upper bound is also valid for $\mathrm{Ex}_{\max,s_I}(tot(\mathcal{R},G))$. Our approach is to find a (k-repeating) strategy σ that maximizes the resulting upper bound. Here, it suffices to restrict to k-repeating strategies since the maximal expected value is always attained by some memoryless strategy. The following theorem is the basis of sound value iteration for MDPs.

Theorem 4.21 For $k \in \mathbb{N}$, $b \in \mathbb{R}$, and $\sigma \in \Sigma_k^M$ such that

- $\forall\, s \in S_?\colon\ \mathrm{Ex}_{\max,s}(tot(\mathcal{R},G)) \leq b$,
- $\sigma \in \arg\max_{\sigma' \in \Sigma^M} \left(\mathrm{Ex}_{\sigma',s_I}(tot^{\leq k}(\mathcal{R},G)) + \mathrm{Pr}_{\sigma',s_I}(\Box^{\leq k} S_?) \cdot b\right)$, and
- $\forall\, s \in S_?\colon\ \mathrm{Pr}_{\sigma,s}(\Box^{\leq k} S_?) < 1$

it holds that

$$\begin{aligned}
&\mathrm{Ex}_{\sigma,s_I}(tot^{\leq k}(\mathcal{R},G)) + \mathrm{Pr}_{\sigma,s_I}(\Box^{\leq k} S_?) \cdot \min_{s \in S_?} \frac{\mathrm{Ex}_{\sigma,s}(tot^{\leq k}(\mathcal{R},G))}{1 - \mathrm{Pr}_{\sigma,s}(\Box^{\leq k} S_?)} \\
&\leq \mathrm{Ex}_{\max,s_I}(tot(\mathcal{R},G)) \\
&\leq \mathrm{Ex}_{\sigma,s_I}(tot^{\leq k}(\mathcal{R},G)) + \mathrm{Pr}_{\sigma,s_I}(\Box^{\leq k} S_?) \cdot b.
\end{aligned}$$

Proof. The theorem is a direct consequence of Lemmas 4.19 and 4.20. ∎

Our goal is to efficiently obtain bounds on $\mathrm{Ex}_{\max,s_I}(tot(\mathcal{R},G))$ using Theorem 4.21 for increasing $k \in \mathbb{N}$. In the theorem we assume that some upper bound $b \in \mathbb{R}$ on the largest occurring expected reward is known. We can use Lemma 4.20 to obtain the upper bound

$$b \;=\; \max_{s \in S_?} \frac{\mathrm{Ex}_{\sigma,s}(tot^{\leq k}(\mathcal{R},G))}{1 - \mathrm{Pr}_{\sigma,s}(\Box^{\leq k} S_?)} \;\geq\; \max_{s \in S_?} \mathrm{Ex}_{\max,s}(tot(\mathcal{R},G)), \tag{4.4}$$

where $\sigma \in \Sigma_k^M$ satisfies

$$\sigma \in \arg\max_{\sigma' \in \Sigma^M} \left(\mathrm{Ex}_{\sigma',s_I}(tot^{\leq k}(\mathcal{R},G)) + \mathrm{Pr}_{\sigma',s_I}(\Box^{\leq k} S_?) \cdot \max_{s \in S_?} \frac{\mathrm{Ex}_{\sigma',s}(tot^{\leq k}(\mathcal{R},G))}{1 - \mathrm{Pr}_{\sigma',s}(\Box^{\leq k} S_?)}\right).$$

There is always such a k-repeating strategy since only the first k steps are relevant for

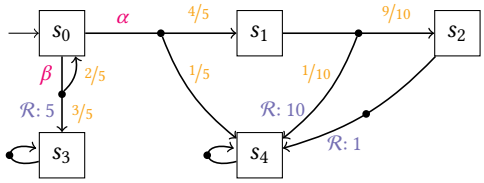


Figure 4.7: Example MDP (cf. Examples 4.7 and 4.8).

the considered objectives. However, the computation of such a strategy as well as the induced values $\mathrm{Ex}_{\sigma,s}(tot^{\leq k}(\mathcal{R}, G))$ and $\Pr_{\sigma,s}(\Box^{\leq k} S_?)$ for $s \in S_?$ and increasing $k \in \mathbb{N}$ can not trivially be implemented with a value iteration based procedure. Intuitively, this is because in step k we can not necessarily build upon the results from iteration $k-1$ as the following example illustrates.

Example 4.7 Consider the MDP M given in Figure 4.7 with $G = \{s_3, s_4\}$, $S_? = \{s_0, s_1, s_2\}$, and $\mathcal{R}$ as indicated. In the following, we show that in order to apply Theorem 4.21 with b as in Equation (4.4), action α at s_0 has to be considered for $k = 1$. However, for $k = 2$ we have to consider the strategy that takes β in *both* steps, i.e., we cannot use the results for $k = 1$ to compute the results for $k = 2$. For $k = 1$, it suffices to distinguish between the two strategies σ_α and σ_β which always select α and β at s_0, respectively. Assuming order $s_0 \prec s_1 \prec s_2$, we obtain

$$\begin{aligned}
\left\langle \mathrm{Ex}_{\sigma_\alpha,s}(tot^{\leq 1}(\mathcal{R}, G)) \right\rangle_{s \in S_?} &= \langle 0, 1, 1 \rangle \\
\left\langle \Pr_{\sigma_\alpha,s}(\Box^{\leq 1} S_?) \right\rangle_{s \in S_?} &= \langle 4/5, 9/10, 0 \rangle \\
\left\langle \mathrm{Ex}_{\sigma_\beta,s}(tot^{\leq 1}(\mathcal{R}, G)) \right\rangle_{s \in S_?} &= \langle 3, 1, 1 \rangle \\
\left\langle \Pr_{\sigma_\beta,s}(\Box^{\leq 1} S_?) \right\rangle_{s \in S_?} &= \langle 2/5, 9/10, 0 \rangle.
\end{aligned}$$

Using Equation (4.4), let

$$b_\alpha = \max_{s \in S_?} \frac{\mathrm{Ex}_{\sigma_\alpha,s}(tot^{\leq k}(\mathcal{R}, G))}{1 - \Pr_{\sigma_\alpha,s}(\Box^{\leq 1} S_?)} = \max(0, 10, 1) = 10$$

and similarly $b_\beta = \max(5, 10, 1) = 10$. Only σ_α satisfies the preconditions of Theorem 4.21 since

$$\begin{aligned}
\mathrm{Ex}_{\sigma_\alpha,s_I}(tot^{\leq 1}(\mathcal{R}, G)) + \Pr_{\sigma_\alpha,s_I}(\Box^{\leq 1} S_?) \cdot b_\alpha &= 0 + 4/5 \cdot 10 = 8 \\
> \mathrm{Ex}_{\sigma_\beta,s_I}(tot^{\leq 1}(\mathcal{R}, G)) + \Pr_{\sigma_\beta,s_I}(\Box^{\leq 1} S_?) \cdot b_\beta &= 3 + 2/5 \cdot 10 = 7.
\end{aligned}$$

Hence, we consider action α at s_0.
Now, let $k = 2$. We have to distinguish between the strategies σ_α and σ_β as above as well as $\sigma_{\beta\alpha}$ which selects β only at the first visit of s_0 and α at any potential later visit. Other strategies are not relevant for our MDP. We have

$$\begin{aligned}
\left\langle \mathrm{Ex}_{\sigma_\alpha,s}(tot^{\leq 2}(\mathcal{R}, G)) \right\rangle_{s\in S_?} &= \left\langle {}^{4}/_{5}, {}^{19}/_{10}, 1 \right\rangle \\
\left\langle \mathrm{Pr}_{\sigma_\alpha,s}(\Box^{\leq 2} S_?) \right\rangle_{s\in S_?} &= \left\langle {}^{18}/_{25}, 0, 0 \right\rangle \\
\left\langle \mathrm{Ex}_{\sigma_\beta,s}(tot^{\leq 2}(\mathcal{R}, G)) \right\rangle_{s\in S_?} &= \left\langle {}^{21}/_{5}, {}^{19}/_{10}, 1 \right\rangle \\
\left\langle \mathrm{Pr}_{\sigma_\beta,s}(\Box^{\leq 2} S_?) \right\rangle_{s\in S_?} &= \left\langle {}^{4}/_{25}, 0, 0 \right\rangle \\
\left\langle \mathrm{Ex}_{\sigma_{\beta\alpha},s}(tot^{\leq 2}(\mathcal{R}, G)) \right\rangle_{s\in S_?} &= \left\langle 3, {}^{19}/_{10}, 1 \right\rangle \\
\left\langle \mathrm{Pr}_{\sigma_{\beta\alpha},s}(\Box^{\leq 2} S_?) \right\rangle_{s\in S_?} &= \left\langle {}^{8}/_{25}, 0, 0 \right\rangle.
\end{aligned}$$

The corresponding bounds are $b_\alpha = \max({}^{20}/_{7}, {}^{19}/_{10}, 1) = {}^{20}/_{7} \approx 2.9$, $b_\beta = \max(5, {}^{19}/_{10}, 1) = 5$, and $b_{\beta\alpha} = \max({}^{75}/_{17}, {}^{19}/_{10}, 1) = {}^{75}/_{17} \approx 4.4$. Only σ_β satisfies the preconditions in Theorem 4.21. That means a value iteration-based procedure would have to consider action β at s_0 in *both* iterations. Since for $k = 1$ we considered α, this essentially means that for $k = 2$ we would have to restart the iterations from scratch.

4.5.2 Constraining the Upper Bound

As illustrated in Example 4.7, considering a different upper bound b for every $k \in \mathbb{N}$ requires to reassess all performed strategy choices which hinders an efficient, value iteration-like implementation. In the following, we first show that this problem is avoided when considering the same fixed upper bound b throughout all iterations. Then, we argue that the approach can be a bit more liberal by maintaining a range of possible values for b under which the choices of earlier iterations remain valid.

Fixed Upper Bound | First, assume that some fixed value $b \in \mathbb{R}$ as in Theorem 4.21 is given, i.e.,

$$\forall\, s \in S_?\colon\ \mathrm{Ex}_{\max,s}(tot(\mathcal{R}, G)) \leq b.$$

Similar to the DTMC case, such a value could for instance be obtained using domain knowledge or methods from [BKLP+17]. To apply Theorem 4.21 for increasing $k = 0, 1, 2, \ldots$, we need to consider a k-repeating strategy $\sigma_k \in \Sigma_k^M$ that satisfies the preconditions of the theorem. The strategies constructed in the following lemma are *pure* i.e., we assume $\sigma_k\colon Paths_{\mathrm{fin}}^M \to Act$.

Lemma 4.22 Consider a family of pure strategies $\sigma_0, \sigma_1, \sigma_2, \ldots \Sigma^M_{\mathrm{P}}$ such that for every $k > 0$ the following conditions hold:

(I) for all $s \in S_?$:

$$\sigma_k(s) \in \arg\max_{\alpha \in A(s)} \sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot \Big(\mathrm{Ex}_{\sigma_{k-1},s'}(tot^{\leq k-1}(\mathcal{R},G)) + \mathcal{R}[s,\alpha,s'] + \mathrm{Pr}_{\sigma_{k-1},s'}(\Box^{\leq k-1} S_?) \cdot b\Big),$$

(II) and for paths $s_0 \xrightarrow{\alpha_0} \ldots \xrightarrow{\alpha_{n-1}} s_n \in Paths^M_{\mathrm{fin}}$ with $0 < n < k$ we have

$$\sigma_k(s_0 \xrightarrow{\alpha_0} \ldots \xrightarrow{\alpha_{n-1}} s_n) = \sigma_{k-1}(s_1 \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{n-1}} s_n).$$

Then, the following holds for all $k \in \mathbb{N}$ and $s \in S$:

$$\sigma_k \in \arg\max_{\sigma \in \Sigma^M} \big(\mathrm{Ex}_{\sigma,s}(tot^{\leq k}(\mathcal{R},G)) + \mathrm{Pr}_{\sigma,s}(\Box^{\leq k} S_?) \cdot b\big).$$

Proof. The proof is by induction over $k \in \mathbb{N}$. For $k = 0$ the choices of σ_0 have no effect and thus the claim holds trivially. Similarly, the claim holds immediately for $k > 0$ and $s \in G$. For $k > 0$, the induction hypothesis states that the claim holds for σ_{k-1}, i.e.,

$$\sigma_{k-1} \in \arg\max_{\sigma' \in \Sigma^M} \big(\mathrm{Ex}_{\sigma',s}(tot^{\leq k-1}(\mathcal{R},G)) + \mathrm{Pr}_{\sigma',s}(\Box^{\leq k-1} S_?) \cdot b\big). \tag{4.5}$$

Condition (II) intuitively states that after the first transition step, σ_k simulates σ_{k-1} for the next $k-1$ steps. Therefore, for $s \in S_?$ we have (cf. Corollary 4.13):

$$\mathrm{Ex}_{\sigma_k,s}(tot^{\leq k}(\mathcal{R},G)) = \sum_{s' \in S} \mathbf{P}(s,\sigma_k(s),s') \cdot \big(\mathrm{Ex}_{\sigma_{k-1},s'}(tot^{\leq k-1}(\mathcal{R},G)) + \mathcal{R}[s,\sigma_k(s),s']\big)$$

$$\mathrm{Pr}_{\sigma_k,s}(\Box^{\leq k} S_?) \cdot b = \sum_{s' \in S} \mathbf{P}(s,\sigma_k(s),s') \cdot \big(\mathrm{Pr}_{\sigma_{k-1},s'}(\Box^{\leq k-1} S_?) \cdot b\big)$$

Combining the two equations yields

$$\begin{aligned} &\mathrm{Ex}_{\sigma_k,s}(tot^{\leq k}(\mathcal{R},G)) + \mathrm{Pr}_{\sigma_k,s}(\Box^{\leq k} S_?) \cdot b \\ &= \sum_{s' \in S} \mathbf{P}(s,\sigma_k(s),s') \cdot \\ &\qquad \big(\mathrm{Ex}_{\sigma_{k-1},s'}(tot^{\leq k-1}(\mathcal{R},G)) + \mathcal{R}[s,\sigma_k(s),s'] + \mathrm{Pr}_{\sigma_{k-1},s'}(\Box^{\leq k-1} S_?) \cdot b\big) \end{aligned}$$

(Condition (I))

$$= \max_{\alpha \in \Delta(s)} \sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot \left(\mathrm{Ex}_{\sigma_{k-1},s'}(tot^{\leq k-1}(\mathcal{R},G)) + \mathcal{R}[s,\alpha,s'] + \Pr_{\sigma_{k-1},s'}(\Box^{\leq k-1} S_?) \cdot b\right)$$

(Equation (4.5))

$$= \max_{\alpha \in \Delta(s)} \sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot \left(\max_{\sigma' \in \Sigma^M} \left(\mathrm{Ex}_{\sigma',s'}(tot^{\leq k-1}(\mathcal{R},G)) + \Pr_{\sigma',s'}(\Box^{\leq k-1} S_?) \cdot b\right) + \mathcal{R}[s,\alpha,s']\right)$$

$$= \max_{\sigma \in \Sigma^M} \left(\mathrm{Ex}_{\sigma,s}(tot^{\leq k}(\mathcal{R},G)) + \Pr_{\sigma,s}(\Box^{\leq k} S_?) \cdot b\right). \qquad \blacksquare$$

Lemma 4.22 enables a value-iteration like procedure, where in iteration step $k > 0$ we compute a strategy σ_k that satisfies Conditions (I) and (II) of Lemma 4.22 as well as the induced values $\mathrm{Ex}_{\sigma_k,s}(tot^{\leq k}(\mathcal{R},G))$ and $\Pr_{\sigma_k,s}(\Box^{\leq k} S_?)$ for all $s \in S$ using the corresponding results from the previous step $k-1$. Since the conditions from Lemma 4.22 only concerns the first k steps of a path, we can assume that the strategy σ_k is k-repeating. This allows us to obtain bounds on $\mathrm{Ex}_{\max,s_I}(tot(\mathcal{R},G))$ using Theorem 4.21. Section 4.5.3 provides further details of the procedure.

Variable Upper Bound | Next, we investigate the effect of the upper bound b on the constructed strategy σ_k from Lemma 4.22. The key observation is that the same strategy can also be valid for other (smaller) upper bounds $b' < b$ that are potentially established during the procedure. To see this, observe that for a fixed k there are only finitely many different (pure) k-repeating strategies σ_k but there are uncountably many upper bounds $b \geq \mathrm{Ex}^M_{\max,s_I}(tot(\mathcal{R},G))$. Thus, two different upper bounds b and b' might yield the very same strategy σ_k. As we argue below, the *set* of upper bounds for which our construction yields a certain strategy σ_k is convex, i.e., a (potentially empty) interval $I \subseteq \mathbb{R}$.

Our approach is as follows: given some upper bound b, we compute a strategy σ_k for increasing $k \in \mathbb{N}$ as mentioned above. In addition, we keep track of the smallest value $d \leq b$ such that if we restarted the algorithm with new upper bound d (instead of b), the same strategy σ_k would be obtained. We refer to d as the *decision value*. Since the upper bound has no further effect on former iterations, it is safe to change it to any value b' with

$$\max\left(d, \mathrm{Ex}^M_{\max,s_I}(tot(\mathcal{R},G))\right) \;\leq\; b' \;\leq\; b.$$

We further illustrate the approach. Assuming some $k > 0$ and $s \in S_?$, the construction

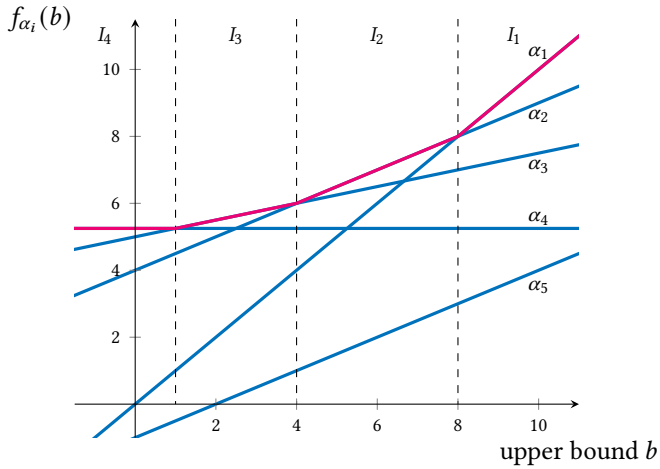


Figure 4.8: Illustration of decision points.

of σ_k as in Lemma 4.22 requires to find an action $\alpha \in \Delta(s)$ that maximizes

$$\begin{aligned} &\sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot \Big(\mathrm{Ex}_{\sigma_{k-1},s'}(tot^{\leq k-1}(\mathcal{R},G)) + \mathcal{R}[s,\alpha,s'] + \Pr_{\sigma_{k-1},s'}(\Box^{\leq k-1} S_?) \cdot b\Big) \\ = &\sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot \Big(\mathrm{Ex}_{\sigma_{k-1},s'}(tot^{\leq k-1}(\mathcal{R},G)) + \mathcal{R}[s,\alpha,s']\Big) + \\ &\quad b \cdot \sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot \Pr_{\sigma_{k-1},s'}(\Box^{\leq k-1} S_?) =: f_\alpha(b). \end{aligned} \tag{4.6}$$

For a fixed $\alpha \in \Delta(s)$, the above expression can be seen as a function $f_\alpha(b)$ that is linear in b and has a slope in $[0,1]$. We illustrate an example case assuming $\Delta(s) = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$ in Figure 4.8, where each blue line labeled with α_i plots the value $f_{\alpha_i}(b)$ from Equation (4.6) when choosing α_i (y-axis) against the value of the upper bound b (x-axis).

We also indicate the maximum over all actions in pink. Since all functions $f_{\alpha_i}(b)$ are linear in b, we may assign a (potentially empty) interval $I_i \subseteq \mathbb{R}$ to each action α_i such that if $b \in I_i$, we know that α_i induces the maximum, i.e., α_i is a valid choice for σ_k at state s. In the case of Figure 4.8 we get

$$I_1 = [8, +\infty) \quad I_2 = [4,8] \quad I_3 = [1,4] \quad I_4 = (-\infty, 1] \quad I_5 = \emptyset$$

For example, if the upper bound is $b = 7$ we have $b \in I_2 = [4,8]$, meaning that

the action α_2 is a valid choice for $\sigma_k(s)$. In fact, the same choice would be made for any other upper bound $b' \in [4, 8]$. If at a later point in the computation (e.g., via Lemma 4.20) we find out that 5 is an upper bound as well, we could set b to 5 without invalidating the decision at state s in iteration k. We take the intersection of all such intervals for any state $s \in S_?$ and any iteration step k which results in an interval I such that the considered strategies $\sigma_0, \sigma_1, \sigma_2, \ldots$ are valid (in the sense of Lemma 4.22) for all $b' \in I$. As we are only interested in *decreasing* the upper bound b, it suffices to only keep track of the lower boundary of I which we refer to as the decision value d. In the scenario sketched in Figure 4.8 with $b = 7$, the decision value d is (at least) 4.

If initially no upper bound b is known, we consider actions that induce the maximum when b approaches ∞. Put differently, we take an action α at state s that induces the maximum for all upper bounds b in an interval of the form $(-\infty, +\infty)$ or $[d_s, +\infty)$ for some $d_s \in \mathbb{R}$. This is always the case for some action

$$\alpha_{\text{opt}} \in \operatorname*{arg\,max}_{\alpha \in A} \sum_{s' \in S} \mathbf{P}(s, \alpha, s') \cdot \Big(\mathrm{Ex}_{\sigma_{k-1}, s'}(tot^{\leq k-1}(\mathcal{R}, G)) + \mathcal{R}[s, \alpha, s'] \Big), \quad \text{where}$$

$$A = \operatorname*{arg\,max}_{\alpha \in \Delta(s)} \sum_{s' \in S} \mathbf{P}(s, \alpha, s') \cdot \Pr_{\sigma_{k-1}, s'}(\Box^{\leq k-1} S_?).$$

Keeping track of the decision value d allows us to update the upper bound b in every iteration according to Lemma 4.20. If at some point this would imply $b < d$, we consider d as the new upper bound instead in order to not invalidate strategy choices from earlier iterations.

4.5.3 Extending Value Iteration

The sound value iteration algorithm for MDPs is outlined in Algorithm 4.6. It extends the procedure for DTMCs (cf. Algorithm 4.4) towards the selection of optimal actions and the computation of the decision value. In particular, Algorithm 4.6 computes vectors $\boldsymbol{x}^{(k)}$ and $\boldsymbol{y}^{(k)}$ in each iteration k with

$$\boldsymbol{x}^{(k)} = \langle \mathrm{Ex}_{\sigma_k, s}(tot^{\leq k}(\mathcal{R}, G)) \rangle_{s \in S} \quad \text{and} \quad \boldsymbol{y}^{(k)} = \langle \Pr_{\sigma_k, s}(\Box^{\leq k} S_?) \rangle_{s \in S},$$

where σ_k is a strategy as in Lemma 4.22.

Lines 3 to 4 consider the case where $G \subseteq S$ is not reached almost surely from a state $s \in S_?$ under a strategy $\sigma \in \Sigma^M$. Since G is attracting, this case yields $\mathrm{Ex}_{\sigma, s}(tot(\mathcal{R}, G)) = -\infty$ for some $s \in S_?$ which means that there are end components (ECs) within $S_?$ in which negative reward can be accumulated. We require to pre-compute an upper bound $b \in \mathbb{R}$ to guarantee termination of the algorithm. Such an upper bound can be computed either via sound value iteration or the pre-computations of [BKLP+17] when applied to the quotient (cf. Definition 2.21 on page 42) of M with respect to the ECs

Input: MDP M, initial state s_I, reward assignment $\mathcal{R}$, attracting set of goal states $G \subseteq S$, precision $\varepsilon > 0$
Output: $\langle l, u \rangle$ with $l \leq \mathrm{Ex}_{\max,s_I}(tot(\mathcal{R}, G)) \leq u$

$\boldsymbol{x}^{(0)} \leftarrow \mathbf{0}$; $\boldsymbol{y}^{(0)} \leftarrow \big\langle [s \in S_?] \big\rangle_{s \in S}$
$a \leftarrow -\infty$; $b \leftarrow +\infty$; $d \leftarrow -\infty$
if $\exists\, s \in S_?\colon \Pr_{\min,s}(\Diamond G) < 1$ **then** *// Check if a priori upper bound needed*
  Set b such that $\max_{s \in S_?} \big(\mathrm{Ex}_{\max,s}(tot(\mathcal{R}, G))\big) \leq b < +\infty$
for $k = 1, 2, 3, \ldots$ **do**
  for $s \in S$ **do**
    if $s \in G$ **then** $\boldsymbol{x}^{(k)}(s) \leftarrow 0$; $\boldsymbol{y}^{(k)}(s) \leftarrow 0$
    else *// $s \in S_?$*
      for $\alpha \in \Delta(s)$ **do** *// Evaluate all enabled actions*
        $x_\alpha \leftarrow \sum_{s' \in S} \mathbf{P}(s, \alpha, s') \cdot \big(\boldsymbol{x}^{(k-1)}(s') + \mathcal{R}[s, \alpha, s']\big)$
        $y_\alpha \leftarrow \sum_{s' \in S} \mathbf{P}(s, \alpha, s') \cdot \boldsymbol{y}^{(k-1)}(s')$
      // Get set $A \subseteq \Delta(s)$ of candidate actions
      if $b = +\infty$ **then** $A \leftarrow \arg\max_{\alpha \in \Delta(s)} y_\alpha$; $A \leftarrow \arg\max_{\alpha \in A} x_\alpha$
      else $A \leftarrow \arg\max_{\alpha \in \Delta(s)} (x_\alpha + b \cdot y_\alpha)$
      Select some action α from A
      // Update decision value
      $d \leftarrow \max\left(d, \max\left\{ \frac{x_\beta - x_\alpha}{y_\alpha - y_\beta} \;\middle|\; \beta \in \Delta(s) \setminus \{\alpha\} \text{ and } y_\alpha > y_\beta \right\}\right)$
      // Set new values for s
      $\boldsymbol{x}^{(k)}(s) \leftarrow x_\alpha$; $\boldsymbol{y}^{(k)}(s) \leftarrow y_\alpha$
  if $\boldsymbol{y}^{(k)} < \mathbf{1}$ **then**
    $a \leftarrow \max\left(a, \min_{s \in S_?} \frac{\boldsymbol{x}^{(k)}(s)}{1 - \boldsymbol{y}^{(k)}(s)}\right)$
    $b \leftarrow \min\left(b, \max\left(d, \max_{s \in S_?} \frac{\boldsymbol{x}^{(k)}(s)}{1 - \boldsymbol{y}^{(k)}(s)}\right)\right)$
  $l \leftarrow \boldsymbol{x}^{(k)}(s_I) + a \cdot \boldsymbol{y}^{(k)}(s_I)$
  $u \leftarrow \boldsymbol{x}^{(k)}(s_I) + b \cdot \boldsymbol{y}^{(k)}(s_I)$
  if $u - l < \varepsilon$ **then return** $\langle l, u \rangle$

Algorithm 4.6: The sound value iteration algorithm for MDPs

that contain negative rewards. If $\mathcal{R}$ only considers non-positive values, we can simply set b to 0.

In Line 7, we set the correct entries of $\boldsymbol{x}$ and $\boldsymbol{y}$ for goal states $s \in G$. The remaining states $s \in S_?$ are considered in Lines 8 to 16. Given some $s \in S_?$, we first evaluate each enabled action $\alpha \in \Delta(s)$ by computing associated values x_α and y_α. Intuitively, x_α is the entry associated to s when applying the Bellman operator $\mathcal{V}_{\mathcal{R},G}(\boldsymbol{x}^{(k-1)})$ under the assumption that α is the only enabled action at s. Similarly, y_α is the value obtained with Bellman operator $\mathcal{V}_{\circ,G}$. The next step is to compute a set of actions $A \subseteq \Delta(s)$ (Lines 12 to 13) from which we pick some arbitrary action $\alpha \in A$ in Line 14. If some real-valued upper bound $b < +\infty$ is known, the selection of the action α is as in Lemma 4.22. Otherwise, we pick an action that is optimal for arbitrarily high upper bounds as discussed above. In Line 15, we potentially update the decision value d to make sure that it satisfies $d > b'$ for all $b' < b$ and $\beta \in \Delta(s) \setminus \{\alpha\}$ with

$$x_\alpha + b' \cdot y_\alpha \;<\; x_\beta + b' \cdot y_\beta \quad\Longleftrightarrow\quad b' \;<\; \frac{x_\beta - x_\alpha}{y_\alpha - y_\beta},$$

i.e., β would become the "better" choice if we considered upper bound b' instead of b. The equivalence above is valid due to $y_\alpha > y_\beta$ which—using $x_\alpha + b \cdot y_\alpha \geq x_\beta + b \cdot y_\beta$—follows from

$$\begin{aligned} & x_\alpha + b \cdot y_\alpha - (x_\alpha + b' \cdot y_\alpha) \;>\; x_\beta + b \cdot y_\beta - (x_\beta + b' \cdot y_\beta) \\ \Longleftrightarrow\quad & (b - b') \cdot y_\alpha \;>\; (b - b') \cdot y_\beta \\ \Longleftrightarrow\quad & y_\alpha \;>\; y_\beta. \end{aligned}$$

We set the corresponding entry of the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in Line 16.

The remaining steps of the algorithm are similar to the DTMC variant of sound value iteration shown in Algorithm 4.4. When updating the upper bound b in Line 19, we make sure that the new value does not fall below the decision value d. The algorithm returns the obtained bounds l and u with $l \leq \mathrm{Ex}_{\max,s_I}(tot(\mathcal{R},G)) \leq u$ once those bounds are sufficiently precise.

Similar to the DTMC variant, termination of the algorithm is guaranteed because G is assumed to be attracting, i.e., for all $s \in S_?$ we get that $\Pr_{\sigma_k,s}(\Box^{\leq k} S_?)$ approaches 0 for increasing k and considered strategy σ_k. In particular, if strategies $\sigma \in \Sigma^M$ with $\Pr_{\sigma,s}(\Box^{\leq k} S_?) = 1$ for all $k \in \mathbb{N}$ exist, then Lines 3 to 4 ensure that the candidate action set A determined in Line 13 eventually contains actions that enforce leaving the set $S_?$.

Theorem 4.23 Algorithm 4.6 terminates for any given input with a solution $\langle l, u \rangle$ for Problem 4.1.

Sound Gauss-Seidel value iteration (Section 4.4.3) and sound topological value

iteration (Section 4.4.4) can be lifted to the MDP case in a straightforward way.

Example 4.8 Reconsider the MDP from Figure 4.7 with $G = \{s_3, s_4\}$ as in Example 4.7. The expected value for $tot(\mathcal{R}, G)$ is maximized when a strategy always selects β at s_0, yielding $\mathrm{Ex}_{\max}(tot(\mathcal{R}, G)) = 5$. We now illustrate how Algorithm 4.6 approximates this value by sketching the first two iterations.
We assume $s_0 \prec s_1 \prec s_2$ and omit the entries for $s_3, s_4 \in G$ to improve readability. Initially, we get

$$\boldsymbol{x}^{(0)} = \langle 0, 0, 0 \rangle \quad \boldsymbol{y}^{(0)} = \langle 1, 1, 1 \rangle \quad \langle a, b, d \rangle = \langle -\infty, +\infty, -\infty \rangle.$$

For $k = 1$, we get the following values at s_0:

$$x_\alpha = 0 \quad y_\alpha = 4/5 \quad x_\beta = 3 \quad y_\beta = 2/5.$$

Thus, the set of candidate actions at s_0 is $A = \{\alpha\}$, i.e., we select action α, yielding

$$\boldsymbol{x}^{(1)} = \langle 0, 1, 1 \rangle \quad \boldsymbol{y}^{(1)} = \langle 4/5, 9/10, 0 \rangle \quad \langle a, b, d \rangle = \langle 0, 10, 15/2 \rangle.$$

Switching to $k = 2$ yields

$$x_\alpha = 4/5 \quad y_\alpha = 18/25 \quad x_\beta = 3 \quad y_\beta = 8/25$$

at state s_0. Since $4/5 + 10 \cdot 18/25 = 8 > 6.2 = 3 + 10 \cdot 8/25$, we select action α, which results in

$$\boldsymbol{x}^{(2)} = \langle 4/5, 19/10, 1 \rangle \quad \boldsymbol{y}^{(2)} = \langle 18/25, 0, 0 \rangle \quad \langle a, b, d \rangle = \langle 1, 15/2, 15/2 \rangle.$$

The decision value $d = 15/2$ "prevents" that the upper bound b is set to

$$\frac{4/5}{1 - 18/25} = \frac{20}{7} < \mathrm{Ex}_{\max}(tot(\mathcal{R}, G)) = 5\,.$$

The bounds obtained after this step are

$$l = 4/5 + 1 \cdot 18/25 = 1.52 \quad \text{and} \quad u = 4/5 + 15/2 \cdot 18/25 = 6.2\,.$$

4.5.4 Obtaining Optimal Strategies

We discuss how to obtain a pure and memoryless strategy $\sigma \in \Sigma^M_{\mathrm{PM}}$ with

$$l \leq \mathrm{Ex}_{\sigma, s_I}(tot(\mathcal{R}, G)),$$

where l is the lower bound returned by Algorithm 4.6. Such a strategy is required, e.g., in Line 8 of Algorithm 4.2.

Consider the mapping $\mathfrak{S}\colon \mathbb{R}^{|S|} \to 2^{\Sigma^M_{\text{PM}}}$, where for $\boldsymbol{x} \in \mathbb{R}^{|S|}$ we have $\sigma \in \mathfrak{S}(\boldsymbol{x})$ iff

$$\forall\, s \in S_?\colon \quad \sigma(s) \;\in\; \operatorname*{arg\,max}_{\alpha \in \Delta(s)} \sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot \big(\boldsymbol{x}^*(s') + \mathcal{R}[s,\alpha,s']\big).$$

Intuitively, the strategies in $\mathfrak{S}(\boldsymbol{x})$ select an action that induces the maximum value when applying the Bellman operator $\mathcal{V}_{\mathcal{R},G}$ on $\boldsymbol{x}$. In other words, if $\mathcal{V}^{\sigma}_{\mathcal{R},G}$ denotes the Bellman operator for the induced DTMC $M[\![\sigma]\!]$ with $\sigma \in \mathfrak{S}(\boldsymbol{x})$, then $\mathcal{V}_{\mathcal{R},G}(\boldsymbol{x}) = \mathcal{V}^{\sigma}_{\mathcal{R},G}(\boldsymbol{x})$.

Lemma 4.24 Let $\boldsymbol{x}^* = \big\langle \mathrm{Ex}_{\max,s}(tot(\mathcal{R},G)) \big\rangle_{s\in S}$ be the unique fixpoint of the Bellman operator $\mathcal{V}_{\mathcal{R},G}$. Then,

$$\sigma \;\in\; \mathfrak{S}(\boldsymbol{x}^*) \quad \text{iff} \quad \boldsymbol{x}^* = \big\langle \mathrm{Ex}_{\sigma,s}(tot(\mathcal{R},G)) \big\rangle_{s\in S}.$$

Proof. The lemma is a reformulation of [Put94, Theorem 7.1.7]. ∎

Assuming that the precise fixpoint $\boldsymbol{x}^*$ of $\mathcal{V}_{\mathcal{R},G}$ is known, we can efficiently compute a pure and memoryless strategy $\sigma \in \mathfrak{S}(\boldsymbol{x}^*)$ that—according to Lemma 4.24—induces the maximal expected reachability reward. When applying sound value iteration (Algorithm 4.6), the fixpoint $\boldsymbol{x}^*$ is not known precisely. However, at iteration k we can bound the fixpoint from below and from above by

$$\boldsymbol{x}^{(k)} + a \cdot \boldsymbol{y}^{(k)} \;\leq\; \boldsymbol{x}^* \;\leq\; \boldsymbol{x}^{(k)} + b \cdot \boldsymbol{y}^{(k)}.$$

Those bounds are tightened arbitrarily when the number of iterations k is increased. We now assume that k is sufficiently large so that $-\infty < a \leq b < +\infty$.

From our observations, three approaches for obtaining a strategy can naturally be derived:

- $\sigma \in \mathfrak{S}(\boldsymbol{x}^{(k)} + a \cdot \boldsymbol{y}^{(k)})$,
- $\sigma \in \mathfrak{S}(\boldsymbol{x}^{(k)} + b \cdot \boldsymbol{y}^{(k)})$, or
- $\sigma \in \mathfrak{S}(\boldsymbol{x}^{(k)} + \frac{a+b}{2} \cdot \boldsymbol{y}^{(k)})$.

Observe that the second item yields a strategy with the same actions as considered in Line 14 of Algorithm 4.6. Unfortunately, none of these approaches yield the desired guarantee on the performance of the obtained strategy. More precisely, we may obtain a strategy σ with $l > \mathrm{Ex}_{\sigma,s_I}(tot(\mathcal{R},G))$ where l is the lower bound returned by Algorithm 4.6. We demonstrate this in the following examples.

Example 4.9 Consider the MDP M_1 from Figure 4.9a with indicated reward assignments $\mathcal{R}$ and attracting set $G = \{s_4\}$. We assume order $s_0 \prec s_1 \prec s_2 \prec s_3$ and omit entries for $s_4 \in G$. Running Algorithm 4.6 for two iteration on this example yields

$$\boldsymbol{x}^{(1)} = \langle 2, 4, 0, 16\rangle \quad \boldsymbol{y}^{(1)} = \langle 1/2, 1/2, 1/2, 0\rangle \quad \langle a, b, d\rangle = \langle 0, 16, 6\rangle$$

and

$$\boldsymbol{x}^{(2)} = \langle 4, 4, 8, 16\rangle \quad \boldsymbol{y}^{(2)} = \langle 1/4, 1/4, 0, 0\rangle \quad \langle a, b, d\rangle = \langle 16/3, 16, 6\rangle,$$

where α is considered at s_0 in both iterations. At this point, we get $l = 4 + 16/3 \cdot 1/4 = 16/3$ and

$$\mathfrak{S}(\boldsymbol{x}^{(2)} + a \cdot \boldsymbol{y}^{(2)}) = \mathfrak{S}(\langle 16/3, 16/3, 8, 16, 0\rangle) = \{\sigma_\beta\}$$

with $\sigma_\beta \colon s_0 \mapsto \beta$, since

$$\underbrace{1/2 \cdot 4 + 1/2 \cdot 16/3 = 14/3}_{\text{value for } \alpha} < \underbrace{15/3 = 5}_{\text{value for } \beta}.$$

Note that $\mathrm{Ex}^{M_1}_{\sigma_\beta, s_I}(tot(\mathcal{R}, G)) = 5 < l$.

Example 4.10 Consider the MDP M_2 from Figure 4.9b with indicated reward assignments $\mathcal{R}$ and attracting set $G = \{s_2\}$. We assume order $s_0 \prec s_1$ and omit entries for $s_2 \in G$. In the first iteration of Algorithm 4.6 we consider α at s_0 and get

$$\boldsymbol{x}^{(1)} = \langle 1, 0\rangle \quad \boldsymbol{y}^{(1)} = \langle 3/4, 0\rangle \quad \langle a, b, d\rangle = \langle 0, 4, -4\rangle.$$

At this point, we get $l = 1 + 0 \cdot 3/4 = 1$ and

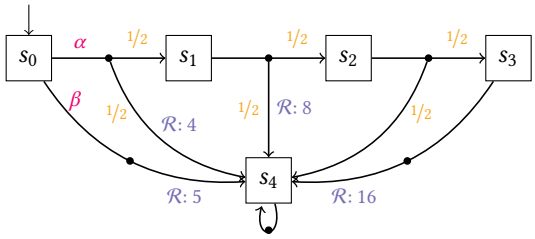
$$\mathfrak{S}(\boldsymbol{x}^{(2)} + b \cdot \boldsymbol{y}^{(2)}) = \mathfrak{S}(\langle 4, 0, 0\rangle) = \{\sigma_\beta\}$$

with $\sigma_\beta \colon s_0 \mapsto \beta$, since

$$\underbrace{1/4 \cdot 4 + 3/4 \cdot 0 = 1}_{\text{value for } \alpha} < \underbrace{2 = 1/2 \cdot 4}_{\text{value for } \beta}.$$

Similarly,

$$\mathfrak{S}(\boldsymbol{x}^{(2)} + (a+b)/2 \cdot \boldsymbol{y}^{(2)}) = \mathfrak{S}(\langle 5/2, 0, 0\rangle) = \{\sigma_\beta\}$$



(a) MDP M_1 (b) MDP M_2

Figure 4.9: Example MDPs (cf. Examples 4.9 and 4.10).

with $\sigma_\beta : s_0 \mapsto \beta$ as before, since

$$\underbrace{1/4 \cdot 4 + 3/4 \cdot 0 \;=\; 1}_{\text{value for } \alpha} \;<\; \underbrace{5/4 \;=\; 1/2 \cdot 5/2}_{\text{value for } \beta}.$$

In both cases we obtain the strategy σ_β with $\mathrm{Ex}^{M_2}_{\sigma_\beta, s_I}(tot(\mathcal{R}, G)) = 0 < l$.

We follow a pragmatic approach to obtain an optimal (or at least close to optimal) pure memoryless strategy. Upon termination of Algorithm 4.6 we consider

$$z^{(k)} := x^{(k)} + \frac{a+b}{2} \cdot y^{(k)}$$

as an approximation of the fixpoint x^* of $\mathcal{V}_{\mathcal{R},G}$ and determine a strategy $\sigma \in \mathfrak{S}(z^{(k)})$ as suggested by Lemma 4.24. We argue that the obtained strategy *usually* satisfies

$$l \leq \mathrm{Ex}_{\sigma, s_I}(tot(\mathcal{R}, G)) \tag{4.7}$$

on most practical instances.

However, a formal guarantee can not be provided as shown in Example 4.10 above. To resolve this problem, we may validate (or refute) Equation (4.7) by running sound value iteration on the induced DTMC $M[\![\sigma]\!]$. If the expected reachability reward for $M[\![\sigma]\!]$ is below l, we can refine the strategy by continuing the execution of Algorithm 4.6 on M with a smaller precision parameter $\varepsilon' < \varepsilon$ which yields a better approximation of the fixpoint x^*.

Such a check can also easily be incorporated into Algorithm 4.2 by asserting that

the returned value v and point $\langle p_1, \ldots, p_\ell \rangle$ satisfy

$$v \leq \boldsymbol{w} \cdot \langle p_1, \ldots, p_\ell \rangle + \varepsilon_{\texttt{WSO}} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}.$$

This way, the performance of the considered strategy can be validated without significant computational overhead. We stress that situations in which the obtained strategy needs to be refined are very rare.

4.6 Related Work

We end the chapter with an overview of related approaches that compute expected total reachability rewards. We also empirically compare a selection of the mentioned approaches in Section 9.2.

4.6.1 Approaches based on Value Iteration

The value iteration algorithm [Bel57] as discussed in Section 4.3 is arguably the most popular algorithm for probabilistic model checking. Probabilistic model checkers, including EPMC [FHLS+22], MCSTA [HH14], PRISM [KNP11], and STORM [12], commonly use (a variant of) value iteration as their default method to verify MDPs against expected total rewards and reachability probabilities. Those implementations usually consider a finite-precision floating point number representation. When using an infinite precision number representation, [BKNP+19] shows that it is EXPTIME-complete to perform k iterations of value iteration (with k encoded in binary).

Several variations and improvements to value iteration have been suggested. These include Gauss-Seidel- and topological value iteration as presented in Sections 4.4.3 and 4.4.4. Asynchronous value iteration [BT89; GB93] updates the values for some states more frequently than for others. The selection of states for which the value is updated can either be based on heuristics [CBGK08; MKI20] or available information of the (only partially known) MDP [MLG05; BCCF+14]. [WB12; WHGZ+16; KHK21] implement value iteration using GPUs—showing drastic speed-ups compared to CPU-based implementations. [HH15] makes use of secondary storage to enable value iteration on very large models.

Issues regarding premature convergence of value iteration in a probabilistic model checking context have first been pointed out in [WKHB08]. Several extensions of value iteration have been proposed as a remedy. Besides the above-mentioned sound value iteration algorithm, this includes *interval iteration* [HM18; BKLP+17] (also known as bounded value iteration [MLG05; BCCF+14]), *optimistic value iteration* [HK20], the approach of Hansen [Han17], *bisection value iteration* [LX22], and *rational search* [MBCS+20]. Below, we discuss these methods in more detail.

The value iteration algorithm is also applicable to stochastic games (SGs) as discussed in, e.g., [Kwi16]. Approaches for SGs that deal with convergence issues have been developed in [KKKW18; PTHH20]. Those approaches have to take special care of 0-ECs which—in contrast to the MDP case—cannot necessarily be collapsed.

II, interval iteration

Interval Iteration | The interval iteration algorithm runs two instances of value iteration to compute $\mathcal{V}^k_{\mathcal{R},G}(\boldsymbol{x})$ and $\mathcal{V}^k_{\mathcal{R},G}(\boldsymbol{y})$ for increasing $k \in \mathbb{N}$ and starting vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{|S|}$ with $\boldsymbol{x} \leq \boldsymbol{x}^* \leq \boldsymbol{y}$, where $\boldsymbol{x}^* = \left\langle \mathrm{Ex}_{\max,s}(tot(\mathcal{R}, G)) \right\rangle_{s \in S}$ is the unique fixpoint of $\mathcal{V}_{\mathcal{R},G}$. It can be shown that $\mathcal{V}^k_{\mathcal{R},G}(\boldsymbol{x}) \leq \boldsymbol{x}^* \leq \mathcal{V}^k_{\mathcal{R},G}(\boldsymbol{y})$ for all $k \in \mathbb{N}$, i.e., interval iteration provides sound and arbitrarily tight bounds for $\boldsymbol{x}^* = \left\langle \mathrm{Ex}_{\max,s}(tot(\mathcal{R}, G)) \right\rangle_{s \in S}$.

When computing reachability *probabilities*, the starting vectors $\boldsymbol{x} = \mathbf{0}$ and $\boldsymbol{y} = \mathbf{1}$ are always valid. For expected rewards, [BKLP+17] proposes a method that for each state s provides an upper bound for the expected number of times s can be visited before reaching a goal state. From this information, starting vectors of the form $\boldsymbol{x} = \langle a \cdot [\![s \in S_?]\!] \rangle_{s \in S}$ and $\boldsymbol{x} = \langle b \cdot [\![s \in S_?]\!] \rangle_{s \in S}$ are derived with $a, b \in \mathbb{R}$ and $\forall s \in S_?\colon\ a \leq \mathrm{Ex}_{\max,s}(tot(\mathcal{R}, G)) \leq b$.

The use of finite-precision floating point numbers in practical implementations makes interval iteration—as well as all other variants of (sound) value iteration—susceptible to round-off errors. [Har22] alters the rounding mode during floating point operations in a way that the lower- and upper bounds obtained from the interval iteration algorithm are always correct.

We mention that—considering Lemma 4.15 and a straightforward generalization towards MDP—the bounds obtained by sound value iteration are similar to those of interval iteration except that the values a and b can be computed and improved during the iterations. Hence, if we use the methods of [BKLP+17] to compute initial bounds a and b for sound value iteration (cf. Line 2 of Algorithm 4.4 and Line 2 of Algorithm 4.6), the approximation obtained after k iterations of sound value iteration is at least as tight (or even tighter) as the approximation obtained after k steps of interval iteration. On the other hand, the costs per iteration are slightly higher in the case of sound value iteration due to additional bookkeeping—especially in the MDP case.

OVI, optimistic value iteration

Optimistic Value Iteration | Optimistic value iteration can be divided into two phases. In the first phase, it calls the classical value iteration algorithm, yielding $\mathcal{V}^k_{\mathcal{R},G}(\mathbf{0})$ for some k. Under the assumption that all rewards occurring in the model are non-negative[7], those values already provide a lower bound for $\boldsymbol{x}^* = \left\langle \mathrm{Ex}_{\max,s}(tot(\mathcal{R}, G)) \right\rangle_{s \in S}$. In the second phase, an upper bound is established by computing some candidate vector $\boldsymbol{y} \geq \mathcal{V}^k_{\mathcal{R},G}(\mathbf{0})$ according to a heuristic. Then, the Bellman operator is applied repeatedly with starting vector $\boldsymbol{y}$. At the same time,

[7] [HK20] does not consider models with both positive and negative rewards.

the lower bound is improved by continuing the iterations from the first phase, i.e., $\mathcal{V}^{k+k'}_{\mathcal{R},G}(\mathbf{0})$ and $\mathcal{V}^{k'}_{\mathcal{R},G}(\boldsymbol{y})$ are computed for increasing $k' \in \mathbb{N}$. If for some state s and some k' we get $\mathcal{V}^{k+k'}_{\mathcal{R},G}(\mathbf{0})(s) > \mathcal{V}^{k'}_{\mathcal{R},G}(\boldsymbol{y})(s)$, it can be concluded that $\boldsymbol{y}$ is *not* an upper bound for $\boldsymbol{x}^*$ and the second phase is restarted with a new candidate vector $\boldsymbol{y}'$. On the other hand, if $\mathcal{V}^{k'}_{\mathcal{R},G}(\boldsymbol{y}) \geq \mathcal{V}^{k'+1}_{\mathcal{R},G}(\boldsymbol{y})$ for some k', then it can be shown that also $\mathcal{V}^{k'+1}_{\mathcal{R},G}(\boldsymbol{y}) \geq \boldsymbol{x}^*$ holds, i.e., an upper bound for $\boldsymbol{x}^*$ has been established. A similar idea is also described in [MS20].

The key advantage of optimistic value iteration over both interval iteration and sound value iteration is that the first phase only considers a single value iteration instance. If the values from the first phase are already close to the fixpoint, there is often only a small computational overhead induced by the second phase.

In the context of multi-objective model checking, queries involving both positive and negative rewards arise naturally when considering trade-offs between maximizing and minimizing objectives. A possible extension of optimistic value iteration towards this case has not been studied, yet. Such an extension will likely add some additional overhead to also establish a valid lower bound. In our experiments in Chapter 9 we follow a naive approach that pre-computes a starting vector $\boldsymbol{x} \leq \boldsymbol{x}^*$—as for interval iteration [BKLP+17]—that is used for the classical value iteration in the first phase. The correctness of interval iteration implies that for every $k > 0$, $\mathcal{V}^{k}_{\mathcal{R},G}(\boldsymbol{x})$ is a valid lower bound for $\boldsymbol{x}^*$.

The Approach of Hansen | The approach of [Han17] is based on an observation from [Ber05, Example 7.13], where error bounds of classical value iteration are provided. Those bounds consider for each state $s \in S_?$ the expected number of transition steps until reaching a state in G from s. Unfortunately, computing these values exactly is impractical. The core idea of [Han17] is to provide sound and efficiently computable lower bounds and/or upper bounds on the expected number of steps until reaching G. Using the original bounds from [Ber05, Example 7.13], this approach then also yield sound approximations of the expected total reachability rewards.

For the two special cases where all assigned rewards are *strictly positive* or all assigned rewards are *strictly negative*, the bounds can be derived using only a single instance of value iteration. In the more general case, two instances of value iteration are required: one as in the classical value iteration algorithm and one to obtain bounds on the expected number of steps. The former instance requires a starting vector $\boldsymbol{y} \geq \boldsymbol{x}^*$—similar to interval iteration.

There are strong similarities between the approach of [Han17] and sound value iteration. While the former considers the expected number of steps until $S_? = S \setminus G$ is left, the latter concerns the probability to stay inside $S_?$ for at least $k \in \mathbb{N}$ steps.

Bisection Value Iteration | Bisection value iteration maintains lower- and upper bounds $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{|S|}$ of the unique fixpoint $\boldsymbol{x}^*$ of the Bellman operator $\mathcal{V}_{\mathcal{R},G}$, i.e., $\boldsymbol{x} \leq \boldsymbol{x}^* \leq \boldsymbol{y}$. The bounds are refined using a bisection approach. For simplicity, the following assumes that rewards are non-negative. [LX22] does not consider mixtures of positive and negative rewards.

The algorithm is divided into three phases. In the first phase, lower value bounds $\boldsymbol{x} \leq \boldsymbol{x}^*$ are computed using classical value iteration—similar to the first phase of optimistic value iteration. Initial upper bounds $\boldsymbol{y} \geq \boldsymbol{x}^*$ are obtained by considering $\boldsymbol{y} = \mathbf{1}$ for reachability probabilities, and ideas from interval iteration [BKLP+17] for expected rewards. In the second phase, candidate vectors are repeatedly guessed using a heuristic that roughly considers the middle values of the currently known lower- and upper bounds. The algorithm attempts to derive and verify valid lower- or upper bounds from the candidates and then updates $\boldsymbol{x}$ or $\boldsymbol{y}$ accordingly. This step again borrows ideas from optimistic value iteration, using the fact that $\mathcal{V}_{\mathcal{R},G}(\boldsymbol{z}) \sim \boldsymbol{z}$ implies $\boldsymbol{x}^* \sim \mathcal{V}_{\mathcal{R},G}(\boldsymbol{z}) \sim \boldsymbol{z}$ for $\sim \in \{\leq, \geq\}$. After a heuristically determined number of refinement steps, the algorithm proceeds with the third phase in which the lower- and upper bounds are further refined by repeatedly applying the Bellman operator, similar to interval iteration. The procedure stops as soon as the bounds are sufficiently precise.

Rational Search | Rational search [MBCS+20] computes expected total rewards *exactly*, i.e., for the case where $\varepsilon = 0$. The idea is to use classical value iteration to compute some approximation $\boldsymbol{x}$ of the fixpoint $\boldsymbol{x}^*$ of the Bellman operator. Then, the vector $\boldsymbol{x}$ is "sharpened" by computing for each state s the rational number with minimal size—in terms of magnitude of its numerator and denominator—in an interval around $\boldsymbol{x}(s)$. This yields a new vector $\boldsymbol{x}'$ for which it is checked whether $\mathcal{V}_{\mathcal{R},G}(\boldsymbol{x}') = \boldsymbol{x}'$. If equality holds, it can be concluded that $\boldsymbol{x}' = \boldsymbol{x}^*$ is the precise fixpoint of $\mathcal{V}_{\mathcal{R},G}$. Otherwise, value iteration is continued and the above steps are repeated using a better approximation of $\boldsymbol{x}^*$.

4.6.2 Approaches based on Strategy Iteration

Strategy iteration [How60] (also known as policy iteration) is sketched in Algorithm 4.7. strategy iteration
Starting with some arbitrary pure and memoryless strategy, the algorithm evaluates the current strategy by computing expected total reachability rewards on the induced DTMC (Line 3). Next—using the mapping $\mathfrak{S}$ from Section 4.5.4—we check in Line 4 if the choices made by the current strategy induce the maximum value at each state when applying the Bellman operator $\mathcal{V}_{\mathcal{R},G}$. If that is the case, it immediately follows that the fixpoint of $\mathcal{V}_{\mathcal{R},G}$ has been found. Otherwise, an improved strategy for the next iteration is obtained in Line 5.

Input: MDP M, initial state s_I, reward assignment $\mathcal{R}$, attracting set of goal states $G \subseteq S$
Output: $\mathrm{Ex}_{\sigma,s_I}(tot(\mathcal{R},G))$

1 Select arbitrary $\sigma_0 \in \Sigma^M_{\mathrm{PM}}$
2 **for** $k = 1, 2, 3, \ldots$ **do**
3 $\quad \boldsymbol{x}^{(k)} \leftarrow \left\langle \mathrm{Ex}_{\sigma_k,s}(tot(\mathcal{R},G)) \right\rangle_{s\in S}$
4 $\quad$ **if** $\sigma_k \in \mathfrak{S}(\boldsymbol{x}^{(k)})$ **then return** $\boldsymbol{x}^{(k)}(s_I)$
5 $\quad$ Select arbitrary $\sigma_{k+1} \in \mathfrak{S}(\boldsymbol{x}^{(k)})$

Algorithm 4.7: The strategy iteration algorithm

minimize x_{s_I} such that:
$$\forall \langle s,\alpha\rangle \in SA\colon \quad x_s \;\geq\; [s \notin G] \cdot \sum_{s'\in S} \mathbf{P}(s,\alpha,s') \cdot (x_{s'} + \mathcal{R}[s,\alpha,s'])$$

Figure 4.10: LP for computing $\mathrm{Ex}_{\max,s_I}(tot(\mathcal{R},G))$

Similar to rational search, strategy iteration can be employed to compute expected total reachability rewards *exactly*, i.e., for $\varepsilon = 0$. This, however, also requires to evaluate the induced DTMC in Line 3 exactly. Storm [12] implements such a method, where the DTMC computations are translated to a linear equation system that is solved exactly using the Eigen Library [GJ+10].

Unfortunately, if the induced DTMC is *not* evaluated exactly (e.g., using sound value iteration) no guarantees for the obtained solution can be made (cf. Section 4.5.4).

4.6.3 Approaches based on Linear Programming

The fixpoint $\boldsymbol{x}^* = \mathcal{V}_{\mathcal{R},G}(\boldsymbol{x}^*)$ of the Bellman operator can also be characterized using the LP from Figure 4.10. The LP considers a variable x_s for each state s. A solution $val\colon \{x_s \mid s \in S\} \to \mathbb{R}$ yields $\boldsymbol{x}^* = \langle val(x_s)\rangle_{s\in S}$. We can thus compute expected total reachability rewards by solving the corresponding LP instance using off-the-shelf tools such as GLPK [GNU20].

Unfortunately, the LP based method does not scale very well compared to the iterative methods mentioned before. To speed up the procedure—especially for the case where the LP is to be solved exactly—[Gir14] provides methods to initialize the simplex algorithm with a "good" strategy that can, e.g., be obtained from the value iteration algorithm.

Table 4.1: Capabilities and limitations of SVI and related approaches

	SVI	II	OVI	HVI	BVI
Gauss-Seidl variant	yes	yes	yes	yes	yes
topological variant	yes	yes*	yes	yes	yes
improves VI bounds	yes	no	no	if $\mathcal{R}\{<,>\}0$	yes
anytime algorithm	yes	no	only lower	yes	yes
requires...					
...$\mathcal{R}\{\leq,\geq\}0$	no	no	yes†	no	yes†
...starting vectors	optional	$\boldsymbol{x}\leq\boldsymbol{x}^*\leq\boldsymbol{y}$	implicitly	$\boldsymbol{y}\geq\boldsymbol{x}^*$	$\boldsymbol{x}\leq\boldsymbol{x}^*\leq\boldsymbol{y}$
...attracting G	yes	yes	if $\mathcal{R}\geq 0$	yes	yes
...two VI instances	yes	yes	2nd phase	if $\neg(\mathcal{R}\{<,>\}0)$	3rd phase

* [BKLP+17] introduces topological interval iteration that propagates both lower and upper bounds from one SCC to the other.

† Extensions of OVI and BVI to mixtures of positive and negative rewards are possible, but not addressed in the originating works.

4.6.4 Comparison with Sound Value Iteration

Table 4.1 compares sound value iteration (SVI) with its most related approaches, namely interval iteration (II), optimistic value iteration (OVI), the approach by Hansen (HVI), and bisection value iteration (BVI). All these approaches are based on classical value iteration (VI) and can be used to obtain sound ε-approximations (for $\varepsilon > 0$) of expected total reachability rewards on MDPs.

In the table, we use the notation $\mathcal{R}\{<,>\}0$ to denote the case where either all assigned rewards are strictly less than 0 or all assigned rewards are strictly greater than 0. The notations $\mathcal{R}\{\leq,\geq\}0$ and $\mathcal{R}\geq 0$ are similar.

All considered methods support a Gauss-Seidel variant and topological optimizations. As mentioned in Example 4.6, SVI can potentially converge faster (i.e., with fewer iterations) than VI—even if (hypothetically) the minimum number of required iterations for VI is known. HVI has a similar property in the case where $\mathcal{R}\{<,>\}0$ holds. On the other hand, II and OVI both use the solution from classical value iteration as their lower and/or upper bound. For BVI, the values obtained from VI in the first phase are potentially improved when new bounds are found during the bisection in the second phase.

SVI, II, and HVI are anytime algorithms in the sense that the currently established lower- and upper bounds are accessible at any given iteration. OVI, on the other hand, can only provide lower bounds in intermediate iterations. BVI can also return both bounds at any time. However, the upper bounds are only improved in the second and

third phase.

[HK20; LX22] have not considered mixtures of positive and negative rewards when introducing OVI and BVI, respectively. Thus, $\mathcal{R}\{\leq, \geq\}0$ needs to be assumed for the original algorithms. II and BVI require two starting vectors $\boldsymbol{x}, \boldsymbol{y}$ with $\boldsymbol{x} \leq \boldsymbol{x}^* \leq \boldsymbol{y}$, where $\boldsymbol{x}^*$ is the precise solution. Those vectors have to be computed a priori. HVI only requires an upper bound $\boldsymbol{y} \geq \boldsymbol{x}^*$. OVI does not explicitly require an a priori starting vector. However, OVI relies on the fact that either $\mathbf{0} \leq \boldsymbol{x}^*$ or $\boldsymbol{x}^* \leq \mathbf{0}$ holds, which is implied by the restriction to non-negative or non-positive rewards. SVI does not require any starting vectors. However, if upper or lower bounds are known—e.g. 0 and 1 when computing reachability probabilities—they can be provided to accelerate the procedure. If all rewards are non-positive, OVI does not require that the set of goal states G is attracting (cf. Definition 4.2). In all other cases, goal states have to be made attracting which potentially requires to consider the quotient (cf. Definition 2.21) with respect to 0-ECs as outlined in Section 4.1.4. Finally, we point out that both SVI and II require two separate instances of VI. During the first phase of OVI, a single instance suffices. If $\mathcal{R}\{<, >\}0$ holds, HVI can produce sound ε-approximations using a single value iteration instance. BVI only needs two VI instances in its third phase.

Chapter Summary

- ➜ The weighted sum optimization problem (`WSO`) concerning only expected total reachability reward objectives can be reduced to a series of *single-objective* expected total reward analyzes.
- ➜ The classical value iteration algorithm for expected total reachability rewards does not provide guarantees on the accuracy of the obtained solution due to premature convergence.
- ➜ *Sound value iteration* extends value iteration so that sound lower- and upper bounds can be provided. This requires a second instance of value iteration to compute the probability to reach the goal states only after k steps.

—Chapter 5—

Mixtures with Long-Run Average Objectives

Outlook | We consider instances of WSO (Problem 3.4 on page 81) over *long-run average (LRA) reward objectives* (Definition 2.28 on page 53). We first discuss some prerequisites concerning single-objective LRA reward analysis in Section 5.1. Section 5.2 presents our approach to solve WSO instances (cf. Problem 3.4) over *multiple* LRA objectives. We also consider instances of WSO where LRA reward objectives occur simultaneously with *total (reachability) reward objectives* (Definition 2.27 on page 49) in Section 5.3. Such mixtures arise naturally, e.g., in economics when comparing one-time investments versus long-term benefits. Section 5.4 surveys related work.

Origins | This chapter resembles [11]. The presentation of results has been revised. Section 5.1 summarizes well-known results for single-objective LRA analysis, e.g., [Put94; Alf97; GHHK+14; BWH17; ACDK+17].

Set-up | Let $\mathcal{M} = \langle S, Act, \Delta, \mathbf{P} \rangle$ be an MA with initial state $s_I \in S$. We consider $\ell_T \geq 0$ total reward objectives and $\ell_L > 0$ LRA reward objectives:

$$\Phi \;=\; \big\langle tot(\mathcal{R}_1)\,,\; \ldots\,,\; tot(\mathcal{R}_{\ell_T})\,,\; lra(\mathcal{R}_{\ell_T+1})\,,\; \ldots\,,\; lra(\mathcal{R}_{\ell_T+\ell_L}) \big\rangle,$$

yielding a total number of $\ell \coloneqq \ell_T + \ell_L > 0$ objectives. Moreover, we consider a weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^{\ell}$ and a precision parameter $\varepsilon_{\texttt{WSO}} \geq 0$. Our goal is to solve the corresponding WSO instance. While we focus on MA, applying our results to step-based LRA rewards on MDPs is straightforward.

Remark 5.1 (Total Reachability Rewards) For simplicity, we only consider total reward objectives. Total *reachability* reward objectives $tot(\mathcal{R}, G)$ for $\emptyset \neq G \subseteq S$ can be transformed to total reward objectives $tot(\mathcal{R}')$ via the goal unfolding of $\mathcal{M}$ (cf. Section 4.1.1). Theorem 4.1 on page 101 can straightforwardly be generalized to show that this transformation preserves the set of achievable points $Ach(\Phi)$—even if Φ contains LRA reward objectives.

Assumptions | Without loss of generality, every state $s \in S$ is assumed to be reachable from the initial state s_I. To ensure well-definedness of the considered LRA reward objectives, we forbid Zeno behavior (cf. Section 2.2.7).

Assumption 5.1 MA $\mathcal{M}$ has no Zeno behavior.

Assumption 5.1 is checked by analyzing the maximal end components of $\mathcal{M}$ (cf. Lemma 2.5 on page 45). Non-Zenoness is a common assumption for the verification of MAs, see e.g., [HH12; GHHK+14; BWH17; 2; BHH21]. Recall from Lemma 2.9 on page 53 that Assumption 5.1 implies $\mathrm{Ex}_\sigma^{\mathcal{M}}\big(lra(\mathcal{R})\big) \in \mathbb{R}$ for all strategies $\sigma \in \Sigma^{\mathcal{M}}$ and reward assignments $\mathcal{R}$.

For the total reward objectives, we impose similar assumptions as in Chapter 4.

Assumption 5.2 For $j \in \{1..\ell_T\}$, the total reward objective $tot(\mathcal{R}_j)$ is convergent and satisfies $\mathrm{Ex}_{\max}^{\mathcal{M}}(tot(\mathcal{R}_j)) < \infty$. Furthermore, there is a strategy $\sigma \in \Sigma^{\mathcal{M}}$ such that $\mathrm{Ex}_\sigma^{\mathcal{M}}\big(\langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_{\ell_T})\rangle\big) \in \mathbb{R}^{\ell_T}$.

Section 4.1.3 discusses how to check Assumption 5.2 algorithmically.

5.1 Single-objective LRA Computation

We first discuss known results and techniques for the optimization of a *single* LRA reward objective $lra(\mathcal{R})$ for reward assignment $\mathcal{R}$. More precisely, we consider the problem of computing—or approximating—the maximal expected value $\mathrm{Ex}_{\max}^{\mathcal{M}}(lra(\mathcal{R}))$. It is well-known (e.g., [Put94, Theorem 9.1.8]) that this value is attained by a pure memoryless strategy. We are also interested in finding such a strategy $\sigma \in \Sigma_{\mathrm{PM}}^{\mathcal{M}}$ with $\mathrm{Ex}_{\max}^{\mathcal{M}}(lra(\mathcal{R})) = \mathrm{Ex}_\sigma^{\mathcal{M}}(lra(\mathcal{R}))$.

Intuitively, rewards collected at states that are only visited *finitely* often become less and less relevant in the long-run. In fact, long-run average reward objectives only depend on rewards assigned to states (or transitions) that can be visited (or taken) infinitely often with positive probability. As discussed in Section 2.2.6—in particular Lemma 2.4 on page 41—these states and transitions are characterized by the end

components (ECs) of $\mathcal{M}$. Let $\mathcal{R}_{EC}$ be the reward assignment for $\mathcal{M}$ that is obtained from $\mathcal{R}$ by setting all rewards that are not collected within an EC to zero, i.e., for $s, s' \in S$ and $\kappa \in Act \cup \mathbb{R}_{\geq 0}$ we have

$$\mathcal{R}_{EC}[s,\kappa,s'] \;=\; \big[\exists\, C \in EC(\mathcal{M}) \colon\; \big(s \in C \cap MS \text{ or } \langle s,\kappa\rangle \in C \cap SA\big)\big] \cdot \mathcal{R}[s,\kappa,s'].$$

The following lemma yields that rewards collected outside of ECs are not relevant for LRA objectives.

Lemma 5.1 For all $\sigma \in \Sigma^{\mathcal{M}}$ we have $\mathrm{Ex}^{\mathcal{M}}_{\sigma}\big(lra(\mathcal{R})\big) \;=\; \mathrm{Ex}^{\mathcal{M}}_{\sigma}\big(lra(\mathcal{R}_{EC})\big)$.

Proof. Using Lemma 2.4 and the non-Zenoness of $\mathcal{M}$, we can show that

$$\Pr^{\mathcal{M}}_{\sigma}\big(\{\pi \in Paths^{\mathcal{M}}_{\mathrm{inf}} \mid lra(\mathcal{R})(\pi) \neq lra(\mathcal{R}_{EC})(\pi)\}\big) \;=\; 0.$$

The lemma follows since the objective values coincide on almost all paths. ∎

The computation of $\mathrm{Ex}^{\mathcal{M}}_{\max}(lra(\mathcal{R}))$ is divided into two phases outlined below.

Phase 1: EC Analysis | We first consider the individual submodels $\mathcal{M}[\![C]\!]$ for each *maximal* EC $C \in MEC(\mathcal{M})$. As shown in, e.g., [Put94, Theorem 8.3.2], the maximal LRA reward value $\mathrm{Ex}^{\mathcal{M}[\![C]\!]}_{\max}(lra(\mathcal{R}))$ is independent of the initial state $s \in states(C)$. Intuitively, this is because any other state s' can be reached from s almost surely.

Lemma 5.2 For all $s, s' \in states(C)$: $\mathrm{Ex}^{\mathcal{M}[\![C]\!]}_{\max,s}\big(lra(\mathcal{R})\big) \;=\; \mathrm{Ex}^{\mathcal{M}[\![C]\!]}_{\max,s'}\big(lra(\mathcal{R})\big)$.

For the computation of $\mathrm{Ex}^{\mathcal{M}}_{\max}(lra(\mathcal{R}))$ we solve the following problem for each $C \in MEC(\mathcal{M})$.

Problem 5.1: Expected LRA Reward Problem for ECs

Input MA $\mathcal{M}$, $C \in EC(\mathcal{M})$, reward assignment $\mathcal{R}$, precision $\varepsilon \in \mathbb{R}_{\geq 0}$
Output $\langle v, \sigma\rangle \in \mathbb{R} \times \Sigma^{\mathcal{M}[\![C]\!]}_{\mathrm{PM}}$ with

$$v - \varepsilon \;\leq\; \mathrm{Ex}^{\mathcal{M}[\![C]\!]}_{\sigma}\big(lra(\mathcal{R})\big) \;\leq\; \mathrm{Ex}^{\mathcal{M}[\![C]\!]}_{\max}\big(lra(\mathcal{R})\big) \;\leq\; v$$

There are various approaches to efficiently solve Problem 5.1—and the corresponding variant for MDPs—based on

- *linear programming* [Alf97; BBCF+14; GTHR+14]
- *strategy iteration* [WBBH+10; KM17], or
- *value iteration* [ACDK+17; BWH17; But20].

The former two support exact computations (with precision $\varepsilon = 0$) whereas value iteration is usually preferred for approximate computations ($\varepsilon > 0$) due to its fast convergence on many practical instances.

Example 5.1 Consider the MA $\mathcal{M}$ from Figure 5.1a with initial state s_3 and reward assignment $\mathcal{R}$. We compute the maximal LRA-reward values for the two MECs

$$MEC(\mathcal{M}) = \Big\{ \underbrace{\{s_2, \langle s_4, \alpha\rangle, \langle s_4, \beta\rangle, s_6\}}_{=:C_1}, \underbrace{\{\langle s_5, \alpha\rangle, \langle s_7, \alpha\rangle, \langle s_7, \beta\rangle, s_8\}}_{=:C_2} \Big\}.$$

When residing in the MEC C_1, the time is distributed over the two Markovian states s_2 and s_6, yielding a reward of $\mathcal{R}[s_2] = 6$ and $\mathcal{R}[s_6] = 1$ per time unit, respectively. To maximize the expected LRA reward in this MEC, an optimal strategy $\sigma_{C_1} \in \Sigma_{\mathrm{PM}}^{\mathcal{M}[\![C_1]\!]}$ selects action α in s_4 as this increases the time we spend in s_2, yielding

$$\mathrm{Ex}_{\max}^{\mathcal{M}[\![C_1]\!]}\big(lra(\mathcal{R})\big) = \mathrm{Ex}_{\sigma_{C_1}}^{\mathcal{M}[\![C_1]\!]}\big(lra(\mathcal{R})\big) = 3/5 \cdot 6 + 2/5 \cdot 1 = 4.$$

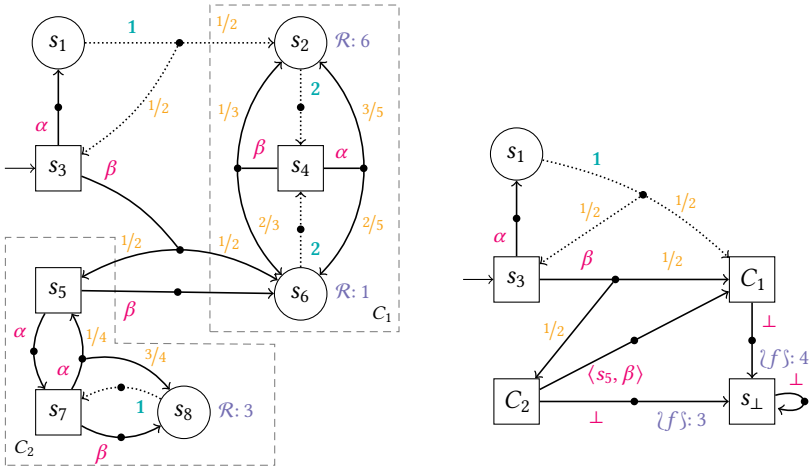
The MEC C_2 contains just the single Markovian state s_8 in which all the time is spent. When we stay inside this MEC, we therefore always collect a reward of $\mathcal{R}[s_8] = 3$ per time unit, i.e., for an arbitrary strategy $\sigma_{C_2} \in \Sigma_{\mathrm{PM}}^{\mathcal{M}[\![C_2]\!]}$ we have

$$\mathrm{Ex}_{\max}^{\mathcal{M}[\![C_2]\!]}\big(lra(\mathcal{R})\big) = \mathrm{Ex}_{\sigma_{C_2}}^{\mathcal{M}[\![C_2]\!]}\big(lra(\mathcal{R})\big) = 3.$$

Phase 2: Reduction to Total Reward | Once each MEC has been analyzed, we fuse their results together. Intuitively, we want to reach MECs with a high LRA value. This is formalized as a total reward objective on the *quotient* $\mathcal{M}_{\backslash MEC}$ of $\mathcal{M}$ with respect to all MECs (cf. Definition 2.21 on page 42). Recall that in $\mathcal{M}_{\backslash MEC}$ each MEC C is collapsed into a single state C. There are two types of enabled actions at those states:

- actions $\langle s, \alpha\rangle \in exits(C)$ which resemble the corresponding successor state distribution $\mathbf{P}(\langle s, \alpha\rangle)$ of $\mathcal{M}$ and
- action $\bot$ which leads to a dedicated sink state $s_\bot$.

Intuitively, selecting an action $\langle s, \alpha\rangle \in exits(C)$ at state C of $\mathcal{M}_{\backslash MEC}$ coincides with a strategy for $\mathcal{M}$ that enforces to leave C via the exiting state-action pair $\langle s, \alpha\rangle$. On the other hand, selecting action $\bot$ at state C reflects a strategy for $\mathcal{M}$ that upon visiting the EC C will stay in this EC forever. For a function $f\colon MEC(\mathcal{M}) \to \mathbb{R}$, let $\wr f \wr$ be the



(a) MA $\mathcal{M}$ with rewards $\mathcal{R}$ (b) Quotient $\mathcal{M}_{\setminus MEC}$

Figure 5.1: Example for MEC-based LRA analysis (cf. Examples 5.1 and 5.2)

reward assignment for $\mathcal{M}_{\setminus MEC}$ that assigns reward $f(C)$ to the transitions that reflect staying inside C, i.e.,

$$\forall C \in MEC(\mathcal{M}) \colon \ \wr f \wr[C, \perp, s_\perp] \ := \ f(C)$$

and all other rewards are set to zero. The following theorem—adapted from [GHHK+14; ACDK+17]—reduces the computation of $\mathrm{Ex}^{\mathcal{M}}_{\max}(lra(\mathcal{R}))$ to a total reward query for the quotient model $\mathcal{M}_{\setminus MEC}$ that can be solved using, e.g., methods from Chapter 4.

Theorem 5.3 Let $f \colon MEC(\mathcal{M}) \to \mathbb{R}$ and $\varepsilon \in \mathbb{R}_{\geq 0}$ such that

$$\forall C \in MEC(\mathcal{M}) \colon \ f(C) - \varepsilon \ \leq \ \mathrm{Ex}^{\mathcal{M}[\![C]\!]}_{\max}\big(lra(\mathcal{R})\big) \ \leq \ f(C).$$

For any $s \in S$ and $\hat{s} \in S \cup MEC(\mathcal{M})$ such that $\hat{s} = C$ if $s \in states(C)$ for some $C \in MEC(\mathcal{M})$ and $\hat{s} = s$ otherwise, we have

$$\mathrm{Ex}^{\mathcal{M}_{\setminus MEC}}_{\max,\hat{s}}\big(tot(\wr f \wr)\big) - \varepsilon \ \leq \ \mathrm{Ex}^{\mathcal{M}}_{\max,s}\big(lra(\mathcal{R})\big) \ \leq \ \mathrm{Ex}^{\mathcal{M}_{\setminus MEC}}_{\max,\hat{s}}\big(tot(\wr f \wr)\big).$$

Intuitively, if $f(C)$ is an ε-approximation of the LRA value for each MEC C (as computed in the Phase 1), then the expected total reward $\mathrm{Ex}^{\mathcal{M}_{\setminus MEC}}_{\max,\hat{s}_I}\big(tot(\wr f \wr)\big)$ yields an ε-approximation of the overall LRA value $\mathrm{Ex}^{\mathcal{M}}_{\max,s_I}\big(lra(\mathcal{R})\big)$. The approximation

error from the MEC analysis (reflected by ε) does not accumulate because any path in $\mathcal{M}_{\setminus MEC}$ can only collect a non-zero reward w.r.t. $\wr f \wr$ at most once—when transitioning to the sink state $s_\perp$.

Algorithm | Algorithm 5.1 summarizes the computation. It performs the two phases outlined above in Lines 1 to 5. Both phases each allow an approximation error of $\varepsilon/2$—which accumulate to an overall error of $2 \cdot \varepsilon/2 = \varepsilon$. In Lines 6 to 13, the algorithm constructs a corresponding pure memoryless strategy σ for $\mathcal{M}$ that

- for states within a MEC C either enforces leaving C via $\hat{\sigma}(C) = \langle s, \alpha \rangle \in \mathit{exits}(C)$ or—if $\hat{\sigma}(C) = \perp$—mimics the optimal LRA value strategy σ_C for $\mathcal{M}[\![C]\!]$ (Lines 6 to 12), and

- for states not contained in a MEC mimics the strategy $\hat{\sigma}$ for $\mathcal{M}_{\setminus MEC}$ (Line 13).

Theorem 5.4 Algorithm 5.1 returns $\langle v, \sigma \rangle$ with $\sigma \in \Sigma^{\mathcal{M}}_{\mathrm{PM}}$ and

$$v - \varepsilon \;\leq\; \mathrm{Ex}^{\mathcal{M}}_{\sigma}\big(\mathit{lra}(\mathcal{R})\big) \;\leq\; \mathrm{Ex}^{\mathcal{M}}_{\max}\big(\mathit{lra}(\mathcal{R})\big) \;\leq\; v.$$

Example 5.2 We apply Algorithm 5.1 to compute $\mathrm{Ex}^{\mathcal{M}}_{\max,s_3}\big(\mathit{lra}(\mathcal{R})\big)$ for the MA $\mathcal{M}$ from Figure 5.1a. We consider the exact case, i.e., $\varepsilon = 0$.
Let the function $f\colon \mathit{MEC}(\mathcal{M}) \to \mathbb{R}$ assign the maximal expected LRA reward values to the two MECs C_1 and C_2 of $\mathcal{M}$ as computed in Example 5.1, i.e., $f(C_1) = 4$ and $f(C_2) = 3$. The quotient $\mathcal{M}_{\setminus MEC}$ with reward assignment $\wr f \wr$ is depicted in Figure 5.1b. To maximize the expected value of the total reward objective $\mathit{tot}(\wr f \wr)$, a strategy $\hat{\sigma} \in \Sigma^{\mathcal{M}_{\setminus MEC}}_{\mathrm{PM}}$ has to ensure that state C_1 of $\mathcal{M}_{\setminus MEC}$ is reached almost surely and that action $\perp$ is taken at C_1. For example, let $\hat{\sigma}(s_3) = \alpha$, $\hat{\sigma}(C_1) = \perp$, and $\hat{\sigma}(C_2) = \langle s_5, \beta \rangle$. This yields

$$\mathrm{Ex}^{\mathcal{M}_{\setminus MEC}}_{\max,s_3}\big(\mathit{tot}(\wr f \wr)\big) \;=\; \mathrm{Ex}^{\mathcal{M}_{\setminus MEC}}_{\hat{\sigma},s_3}\big(\mathit{tot}(\wr f \wr)\big) \;=\; 4.$$

Due to $\hat{\sigma}(C_2) = \langle s_5, \beta \rangle \in \mathit{exits}(C_2)$, we compute a strategy $\sigma^{\mathrm{exit}}_{C_2} \in \Sigma^{\mathcal{M}[\![C_2]\!]}_{\mathrm{PM}}$ that makes sure that state s_5 is reached almost surely from any other state inside this MEC, i.e., $\sigma^{\mathrm{exit}}_{C_2}(s_7) = \alpha$.
It follows that the maximal expected LRA reward for the original MA $\mathcal{M}$ is given by $\mathrm{Ex}^{\mathcal{M}}_{\max,s_3}\big(\mathit{lra}(\mathcal{R})\big) = \mathrm{Ex}^{\mathcal{M}_{\setminus MEC}}_{\max,s_3}\big(\mathit{tot}(\wr f \wr)\big) = 4$ and is attained by the strategy

Input: MA $\mathcal{M}$, initial state s_I, objective $lra(\mathcal{R})$, precision $\varepsilon \in \mathbb{R}_{\geq 0}$ such that Assumption 5.1 holds (i.e., $\mathcal{M}$ is non-Zeno)
Output: $\langle v, \sigma \rangle \in \mathbb{R} \times \Sigma_{\text{PM}}^{\mathcal{M}}$ with $v - \varepsilon \leq \text{Ex}_{\sigma}^{\mathcal{M}}\big(lra(\mathcal{R})\big) \leq \text{Ex}_{\max}^{\mathcal{M}}\big(lra(\mathcal{R})\big) \leq v$

// Phase 1: solve Problem 5.1 for each MEC
for $C \in MEC(\mathcal{M})$ **do**
  Compute $v_C \in \mathbb{R}$ and $\sigma_C \in \Sigma_{\text{PM}}^{\mathcal{M}[\![C]\!]}$ with
    $v_C - \varepsilon/2 \leq \text{Ex}_{\sigma_C}^{\mathcal{M}[\![C]\!]}\big(lra(\mathcal{R})\big) \leq \text{Ex}_{\max}^{\mathcal{M}[\![C]\!]}\big(lra(\mathcal{R})\big) \leq v_C$

// Phase 2: compute total rewards on quotient
Build quotient $\mathcal{M}_{\setminus MEC}$ and rewards $\wr f \wr$ for function $f : C \mapsto v_C$
if $s_I \in states(C)$ for some $C \in MEC(\mathcal{M})$ **then** $\hat{s}_I \leftarrow C$ **else** $\hat{s}_I \leftarrow s_I$
Compute $v \in \mathbb{R}$ and $\hat{\sigma} \in \Sigma_{\text{PM}}^{\mathcal{M}_{\setminus MEC}}$ with
  $v - \varepsilon/2 \leq \text{Ex}_{\hat{\sigma},\hat{s}_I}^{\mathcal{M}_{\setminus MEC}}\big(tot(\wr f \wr)\big) \leq \text{Ex}_{\max,\hat{s}_I}^{\mathcal{M}_{\setminus MEC}}\big(tot(\wr f \wr)\big) \leq v$

// Build resulting strategy $\sigma \in \Sigma_{\text{PM}}^{\mathcal{M}}$
for $C \in MEC(\mathcal{M})$ **do**
  if $\hat{\sigma}(C) = \langle s, \alpha \rangle \in exits(C)$ **then** *// Enforce leaving C via $\langle s, \alpha \rangle$*
    $\sigma(s) \leftarrow \alpha$
    Compute $\sigma_C^{\text{exit}} \in \Sigma_{\text{PM}}^{\mathcal{M}[\![C]\!]}$ with $\forall\, s' \in states(C)\colon \Pr_{\sigma_C^{\text{exit}},s'}^{\mathcal{M}[\![C]\!]}\big(\Diamond\{s\}\big) = 1$
    for $s' \in states(C) \setminus \{s\}$ **do** $\sigma(s') \leftarrow \sigma_C^{\text{exit}}(s')$
  else *// $\hat{\sigma}(C) = \bot$—mimic optimal LRA strategy σ_C for $\mathcal{M}[\![C]\!]$*
    for $s \in states(C)$ **do** $\sigma(s) \leftarrow \sigma_C(s)$
for $s \in S \setminus \big(\bigcup_{C \in MEC(\mathcal{M})} states(C)\big)$ **do** $\sigma(s) \leftarrow \hat{\sigma}(s)$
return $\langle v, \sigma \rangle$

Algorithm 5.1: Computing optimal single-objective LRA rewards

$\sigma \in \Sigma^{\mathcal{M}}_{\text{PM}}$ with

$$\sigma(s_3) = \hat{\sigma}(s_3) = \alpha, \qquad \sigma(s_4) = \sigma_{C_1}(s_4) = \alpha,$$
$$\sigma(s_5) = \beta, \text{ and} \qquad \sigma(s_7) = \sigma^{\text{exit}}_{C_2}(s_7) = \alpha.$$

5.2 Solving LRA Instances for WSO

We now deal with solving multi-dimensional WSO instances that only consider LRA reward objectives, i.e., we are given $\ell \geq 2$ LRA reward objectives $\Phi = \langle lra(\mathcal{R}_1), \ldots, lra(\mathcal{R}_\ell)\rangle$ for $\mathcal{M}$, a weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$, and a precision parameter $\varepsilon_{\texttt{WSO}} \geq 0$.

Similar to our approach for total rewards (cf. Section 4.1.4), the idea is to reduce the corresponding WSO instance to standard single-objective methods for LRA as discussed in the previous section. Consider the reward assignment $\mathcal{R}$ for $\mathcal{M}$ given by the weighted sum of the reward assignments $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$:

$$\mathcal{R} \coloneqq \boldsymbol{w} \cdot \langle \mathcal{R}_1, \ldots, \mathcal{R}_\ell \rangle = \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \mathcal{R}_j.$$

The following lemma is analogous to Lemma 4.4 on page 106 and shows that we can consider the expected value of objective $lra(\mathcal{R})$ instead of the weighted sum of the individual expected LRA objective values. This allows us to combine all objectives into a single reward assignment and then apply single-objective model checking for LRA rewards.

Lemma 5.5 For all $\sigma \in \Sigma^{\mathcal{M}}$ we have

$$\boldsymbol{w} \cdot \text{Ex}^{\mathcal{M}}_{\sigma,s_I}\big(\langle lra(\mathcal{R}_1), \ldots, lra(\mathcal{R}_\ell)\rangle\big) \leq \text{Ex}^{\mathcal{M}}_{\sigma,s_I}\big(lra(\mathcal{R})\big),$$

where equality holds if $\sigma \in \Sigma^{\mathcal{M}}_{\text{PM}}$.

Proof. The proof is similar to the proof of Lemma 4.4 on page 106. From Lemma 2.6 on page 48 we have

$$\boldsymbol{w} \cdot \text{Ex}^{\mathcal{M}}_{\sigma,s_I}\big(\langle lra(\mathcal{R}_1), \ldots, lra(\mathcal{R}_\ell)\rangle\big) = \text{Ex}^{\mathcal{M}}_{\sigma,s_I}\big(\boldsymbol{w} \cdot \langle lra(\mathcal{R}_1), \ldots, lra(\mathcal{R}_\ell)\rangle\big).$$

It remains to consider the objectives $\boldsymbol{w} \cdot \langle lra(\mathcal{R}_1), \ldots, lra(\mathcal{R}_\ell)\rangle$ and $lra(\mathcal{R})$.
Since $\mathcal{M}$ is non-Zeno, Lemma 2.9 on page 53 yields for $\mathcal{R}' \in \{\mathcal{R}, \mathcal{R}_1, \ldots, \mathcal{R}_\ell\}$ and

almost[a] all paths $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots \in Paths^{\mathcal{M}}_{\text{inf}}$:

$$lra(\mathcal{R}')(\pi) \;=\; \liminf_{n\to\infty} \frac{\mathcal{R}'[pref(\pi,n)]}{dur(pref(\pi,n))} \in \mathbb{R}.$$

It follows that

$$\begin{aligned}
\big(\boldsymbol{w}\cdot\langle lra(\mathcal{R}_1),\ldots,lra(\mathcal{R}_\ell)\rangle\big)(\pi) \;&=\; \sum_{j=1}^{\ell} \boldsymbol{w}(j)\cdot lra(\mathcal{R}_j)(\pi)\\
&=\; \sum_{j=1}^{\ell} \boldsymbol{w}(j)\cdot \liminf_{n\to\infty} \frac{\mathcal{R}_j[pref(\pi,n)]}{dur(pref(\pi,n))}\\
&\overset{(*)}{\leq}\; \liminf_{n\to\infty} \sum_{j=1}^{\ell} \boldsymbol{w}(j)\cdot \frac{\sum_{i=0}^{n-1}\mathcal{R}_j[s_i,\alpha_i,s_{i+1}]}{dur(pref(\pi,n))}\\
&=\; \liminf_{n\to\infty} \frac{\sum_{i=0}^{n-1}\sum_{j=1}^{\ell}\boldsymbol{w}(j)\cdot\mathcal{R}_j[s_i,\alpha_i,s_{i+1}]}{dur(pref(\pi,n))}\\
&=\; \liminf_{n\to\infty} \frac{\sum_{i=0}^{n-1}\mathcal{R}[s_i,\alpha_i,s_{i+1}]}{dur(pref(\pi,n))}\\
&=\; \liminf_{n\to\infty} \frac{\mathcal{R}[pref(\pi,n)]}{dur(pref(\pi,n))} \;=\; lra(\mathcal{R})(\pi).
\end{aligned}$$

If $\sigma \in \Sigma^{\mathcal{M}}_{\text{PM}}$, Lemma 2.10 on page 55 yields that

$$\Pr^{\mathcal{M}}_{\sigma,s_I}\big(\{\pi \in Paths^{\mathcal{M}}_{\text{inf}} \mid lra(\mathcal{R}')(\pi) = lra^{\sup}(\mathcal{R}')(\pi)\}\big) \;=\; 1.$$

Therefore, for almost all σ-consistent paths, the limit inferior above can be replaced by a simple limit meaning that the inequality $(*)$ can be replaced by an equality. We thus get

$$\big(\boldsymbol{w}\cdot\langle lra(\mathcal{R}_1),\ldots,lra(\mathcal{R}_\ell)\rangle\big)(\pi) \;=\; lra(\mathcal{R})(\pi)$$

for almost all σ-consistent paths $\pi \in Paths^{\mathcal{M}}_{\text{inf}}$. ■

[a] The set of paths for which the limit diverges to infinity has probability zero.

Lemma 5.5 gives rise to Algorithm 5.2 for solving instances of WSO considering only LRA reward objectives. The approach is similar to Algorithm 4.2 on page 108 for total reward objectives. The first step is to compute the optimal value v with an inducing pure memoryless strategy $\sigma \in \Sigma^{\mathcal{M}}_{\text{PM}}$ for the weighted sum of objectives in Lines 3 to 4. This is done by analyzing the *single* objective $lra(\mathcal{R})$ using Algorithm 5.1. In a second step, the individual objectives $lra(\mathcal{R}_1), \ldots lra(\mathcal{R}_\ell)$ are evaluated with respect to the

solve WSO

function solveWso($\mathcal{M}, s_I, \Phi = \langle lra(\mathcal{R}_1), \ldots, lra(\mathcal{R}_\ell)\rangle, \boldsymbol{w}, \varepsilon_{\texttt{WSO}}$)

// Assuming $\mathcal{M}$ is non-Zeno (Assumption 5.1)

$\varepsilon \leftarrow \frac{1}{2} \cdot \varepsilon_{\texttt{WSO}} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}$; $\mathcal{I}_{\boldsymbol{w}} \leftarrow \{j \in \{1..\ell\} \mid \boldsymbol{w}(j) > 0\}$

// Analyze weighted sum of LRA objectives

$\mathcal{R} \leftarrow \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \mathcal{R}_j$

Compute $\langle v, \sigma\rangle \in \mathbb{R} \times \Sigma^{\mathcal{M}}_{\mathrm{PM}}$ with
$v - \varepsilon \;\leq\; \mathrm{Ex}^{\mathcal{M}}_{\sigma}\big(lra(\mathcal{R})\big) \;\leq\; \mathrm{Ex}^{\mathcal{M}}_{\max}\big(lra(\mathcal{R})\big) \;\leq\; v$ *// Using Algorithm 5.1*

// Evaluate individual objectives w.r.t. strategy σ

$D \leftarrow \mathcal{M}[\![\sigma]\!]$ *// Induced submodel $\mathcal{M}[\![\sigma]\!]$ has no nondeterminism*

for $j \in \{1..\ell\}$ **do**

  if $j \in \mathcal{I}_{\boldsymbol{w}}$ **then** $\varepsilon_j \leftarrow \frac{\varepsilon}{\boldsymbol{w}(j) \cdot |\mathcal{I}_{\boldsymbol{w}}|}$ **else** $\varepsilon_j \leftarrow \infty$

  Compute $p_j \in \mathbb{R}$ with $p_j \leq \mathrm{Ex}^{D}(lra(\mathcal{R}_j)) \;\leq\; p_j + \varepsilon_j$

return $\langle v, \langle p_1, \ldots, p_\ell\rangle\rangle$

Algorithm 5.2: Solving LRA reward WSO instances

strategy σ—now using a slightly simplified variant of Algorithm 5.1 that does not have to deal with nondeterminism. This yields a lower bound $p_j \leq \mathrm{Ex}^{\mathcal{M}}_{\sigma}(lra(\mathcal{R}_j))$ for each $j \in \{1..\ell\}$. In both steps the precision parameter $\varepsilon = \frac{1}{2} \cdot \varepsilon_{\texttt{WSO}} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}$ is used which ensures that Algorithm 5.2 correctly solves WSO (Problem 3.4 on page 81).

Theorem 5.6 The value v and point $\boldsymbol{p} = \langle p_1, \ldots, p_\ell\rangle$ returned by Algorithm 5.2 yield a solution to WSO.

Proof. Lemma 2.9 on page 53 yields $\mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) \in \mathbb{R}^{\ell}$ for all $\sigma \in \Sigma^{\mathcal{M}}$. Using σ and $D = \mathcal{M}[\![\sigma]\!]$ as computed in the algorithm, we get

$$\boldsymbol{p} \leq \mathrm{Ex}^{D}(\Phi) = \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi),$$

implying $\boldsymbol{p} \in Ach^{\mathcal{M}}(\Phi)$. Furthermore, by Lemma 5.5 we have

$$\sup\left\{\boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma'}(\Phi) \;\middle|\; \sigma' \in \Sigma^{\mathcal{M}}\right\} \;\leq\; \mathrm{Ex}^{\mathcal{M}}_{\max}(lra(\mathcal{R})) \;\leq\; v.$$

It remains to show that $v \leq \boldsymbol{w} \cdot \boldsymbol{p} + \varepsilon_{\texttt{WSO}} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}$ which follows as in the proof of Theorem 4.5 on page 110 using $\mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(lra(\mathcal{R})) = \boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\Phi)$, which is a consequence of Lemma 5.5 and the fact that the constructed strategy σ is pure memoryless. ■

5.3 Combining Long-Run Average and Total Rewards

We finally consider WSO instances over arbitrary combinations of $\ell_T \geq 1$ total reward and $\ell_L \geq 1$ LRA reward objectives

$$\Phi \;=\; \big\langle tot(\mathcal{R}_1)\,,\;\ldots\,,\;tot(\mathcal{R}_{\ell_T})\,,\;lra(\mathcal{R}_{\ell_T+1})\,,\;\ldots\,,\;lra(\mathcal{R}_{\ell})\big\rangle,$$

where $\ell := \ell_T + \ell_L$. For the given objectives and a fixed weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^{\ell}$, we define two reward assignments $\mathcal{R}_{tot}$ and $\mathcal{R}_{lra}$ for $\mathcal{M}$ given by

$$\mathcal{R}_{tot} \;:=\; \sum_{j=1}^{\ell_T} \boldsymbol{w}(j) \cdot \mathcal{R}_j \quad \text{and} \quad \mathcal{R}_{lra} \;:=\; \sum_{j=\ell_T+1}^{\ell} \boldsymbol{w}(j) \cdot \mathcal{R}_j.$$

The following lemma is a combination of Lemma 4.4 on page 106 and Lemma 5.5. It reduces the weighted sum of expected objective values considered in the WSO instance to the expected value of the *single* objective $(tot(\mathcal{R}_{tot}) + lra(\mathcal{R}_{lra}))$, where for $\pi \in Paths_{\text{inf}}(\mathcal{M})$ we have

$$\big(tot(\mathcal{R}_{tot}) + lra(\mathcal{R}_{lra})\big)(\pi) \;:=\; tot(\mathcal{R}_{tot})(\pi) + lra(\mathcal{R}_{lra})(\pi).$$

Lemma 5.7 For all $\sigma \in \Sigma^{\mathcal{M}}$ it holds that

$$\boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\Phi) \;\leq\; \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}\big(tot(\mathcal{R}_{tot}) + lra(\mathcal{R}_{lra})\big),$$

where equality holds if $\sigma \in \Sigma^{\mathcal{M}}_{\text{PM}}$

Proof. Using Lemmas 2.6, 4.4 and 5.5, we get

$$\begin{aligned}
\boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}(\Phi) \;&=\; \left(\sum_{j=1}^{\ell_T} \boldsymbol{w}(j) \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}\big(tot(\mathcal{R}_j)\big)\right) + \left(\sum_{j=\ell_T+1}^{\ell} \boldsymbol{w}(j) \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}\big(lra(\mathcal{R}_j)\big)\right) \\
&\overset{*}{\leq}\; \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}\big(tot(\mathcal{R}_{tot})\big) + \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}\big(lra(\mathcal{R}_{lra})\big) \\
&=\; \mathrm{Ex}^{\mathcal{M}}_{\sigma,s_I}\big(tot(\mathcal{R}_{tot}) + lra(\mathcal{R}_{lra})\big).
\end{aligned}$$

As in the proof of Lemma 5.5, we can replace the inequality $(*)$ by an equality if $\sigma \in \Sigma^{\mathcal{M}}_{\text{PM}}$. ∎

We provide analysis methods for objectives of the form $tot(\mathcal{R}_{tot}) + lra(\mathcal{R}_{lra})$—i.e. combinations of total rewards and LRA rewards. The procedure for pure LRA queries discussed in Section 5.1 reduces the analysis to an expected total reward computation on the quotient model $\mathcal{M}_{\setminus MEC}$. The idea for combined queries is to also incorporate

other total reward objectives for $\mathcal{M}$ in the quotient model. However, special care has to be taken concerning total rewards collected within ECs of $\mathcal{M}$ that are no longer present in the quotient $\mathcal{M}_{\setminus MEC}$. We deal with this issue by *considering the quotient only for ECs in which no reward w.r.t. total reward objectives is collected.*

Due to Assumption 5.2, we get that all non-zero total rewards collected in an EC have to be negative, i.e.,

$$\forall\, j \in \{1..\ell_T\} \colon\ \forall\, C \in EC(\mathcal{M}) \colon\ \mathcal{R}_j[\![C]\!] \ \leq\ 0.$$

Strategies that induce a total reward of $-\infty$ for some objective $tot(\mathcal{R}_j)$ will not be taken into account for `WSO` as they do not achieve a point in $\mathbb{R}^\ell$. Therefore, transitions within ECs that accumulate *negative* total reward should only be taken *finitely* often. These transitions have to be discarded when computing the expected LRA rewards. In summary, only the 0-MECs (cf. Definition 2.22) $C \in MEC_0(\mathcal{M}, \{\mathcal{R}_1, \ldots, \mathcal{R}_{\ell_T}\})$ are relevant for the LRA computation. Assumption 5.2 ensures that there must be at least one such 0-MEC as we require $\mathrm{Ex}^{\mathcal{M}}_{\sigma}\big(\langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_{\ell_T})\rangle\big) \in \mathbb{R}^{\ell_T}$ to hold for at least one strategy $\sigma \in \Sigma^{\mathcal{M}}$.

Algorithm 5.3 outlines the procedure for optimizing weighted sums of LRA- and total reward objectives. The algorithm extends Algorithm 5.1 towards mixtures with total rewards. When computing the LRA values in the ECs of $\mathcal{M}$ in Lines 2 to 4, Algorithm 5.3 only considers those ECs $\mathfrak{C} \subseteq EC(\mathcal{M})$ in which no (total) reward is collected—as argued above. Then, the quotient model $\mathcal{M}_{\setminus\mathfrak{C}}$ and a reward assignment $\hat{\mathcal{R}}$ incorporating all total- and LRA rewards is build and analyzed (Lines 5 to 8).

$\mathcal{M}_{\setminus\mathfrak{C}}$ might still contain ECs different from $\{\langle s_\perp, \perp\rangle\}$. Those ECs shall be left eventually to avoid collecting infinite negative reward for a total reward objective $tot(\mathcal{R}_j)$. If the weight $\boldsymbol{w}(j)$ for such an objective is zero, the rewards of $\mathcal{R}_j$ are not present in $\mathcal{R}_{tot}$ (and $\hat{\mathcal{R}}$). It is therefore necessary to explicitly restrict the analysis in Line 8 to strategies that almost surely (i.e., with probability 1) reach the dedicated state $s_\perp$ of the quotient $\mathcal{M}_{\setminus\mathfrak{C}}$. This particular type of expected total reward computation can be realized using ideas from Algorithm 4.2. In a nutshell, we collapse the 0-MECs $\mathfrak{C}' = MEC_0(\mathcal{M}_{\setminus\mathfrak{C}}, \{\hat{\mathcal{R}}\})$, i.e., we consider the quotient $(\mathcal{M}_{\setminus\mathfrak{C}})_{\setminus\mathfrak{C}'}$. However, this quotient should not introduce the transitions $C \xrightarrow{\perp} s_\perp$ for $C \in \mathfrak{C}'$ to reflect that those ECs shall be left eventually.

In Lines 9 to 16, the strategies σ_C for the 0-MECs and $\hat{\sigma}$ for the quotient $\mathcal{M}_{\setminus\mathfrak{C}}$ are combined into one strategy $\sigma \in \Sigma^{\mathcal{M}}_{\mathrm{PM}}$. This step is completely analogous to Lines 6 to 13 of Algorithm 5.1.

Using Lemma 5.7, we obtain the correctness of the algorithm.

Input: MA $\mathcal{M}$, initial state s_I, $\ell = \ell_T + \ell_L$ objectives Φ as above, weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$, precision $\varepsilon \in \mathbb{R}_{>0}$ such that Assumptions 5.1 and 5.2 hold.
Output: $\langle v, \sigma \rangle \in \mathbb{R} \times (\Sigma_{\text{PM}}^{\mathcal{M}} \cap \Xi)$ using $\Xi = \{\sigma' \in \Sigma^{\mathcal{M}} \mid \text{Ex}_{\sigma'}^{\mathcal{M}}(\Phi) \in \mathbb{R}^\ell\}$ with

$$v - \varepsilon \;\leq\; \boldsymbol{w} \cdot \text{Ex}_{\sigma}^{\mathcal{M}}(\Phi) \;\leq\; \sup\left\{\boldsymbol{w} \cdot \text{Ex}_{\hat{\sigma}}^{\mathcal{M}}(\Phi) \;\middle|\; \hat{\sigma} \in \Xi\right\} \;\leq\; v$$

$\mathcal{R}_{tot} \leftarrow \sum_{j=1}^{\ell_T} \boldsymbol{w}(j) \cdot \mathcal{R}_j$; $\mathcal{R}_{lra} \leftarrow \sum_{j=\ell_T+1}^{\ell} \boldsymbol{w}(j) \cdot \mathcal{R}_j$
// Solve Problem 5.1 for each 0-MEC
$\mathfrak{C} \leftarrow MEC_0(\mathcal{M}, \{\mathcal{R}_1, \ldots, \mathcal{R}_{\ell_T}\})$ *// Get 0-MECs w.r.t. total rewards*
for $C \in \mathfrak{C}$ **do**
  Compute $v_C \in \mathbb{R}$ and $\sigma_C \in \Sigma_{\text{PM}}^{\mathcal{M}[\![C]\!]}$ with
  $v_C - \varepsilon/2 \;\leq\; \text{Ex}_{\sigma_C}^{\mathcal{M}[\![C]\!]}\big(lra(\mathcal{R}_{lra})\big) \;\leq\; \text{Ex}_{\max}^{\mathcal{M}[\![C]\!]}\big(lra(\mathcal{R}_{lra})\big) \;\leq\; v_C$
// Compute total rewards on quotient
Build quotient $\mathcal{M}_{\backslash\mathfrak{C}}$ and rewards $\wr f \wr$ for function $f\colon C \mapsto v_C$
if $s_I \in states(C)$ for some $C \in \mathfrak{C}$ **then** $\hat{s}_I \leftarrow C$ **else** $\hat{s}_I \leftarrow s_I$
$\hat{\mathcal{R}} \leftarrow (\mathcal{R}_{tot})_{\backslash\mathfrak{C}} + \wr f \wr$ *// Using quotient rewards $(\mathcal{R}_{tot})_{\backslash\mathfrak{C}}$ from Example 2.13*
Compute $v \in \mathbb{R}$ and $\hat{\sigma} \in \big(\Sigma_{\text{PM}}^{\mathcal{M}_{\backslash\mathfrak{C}}} \cap \Theta\big)$ using $\Theta := \left\{\hat{\sigma}' \;\middle|\; \Pr_{\hat{\sigma}', \hat{s}_I}^{\mathcal{M}_{\backslash\mathfrak{C}}}(\Diamond\{s_\perp\}) = 1\right\}$ with

$$v - \varepsilon/2 \;\leq\; \text{Ex}_{\hat{\sigma}, \hat{s}_I}^{\mathcal{M}_{\backslash\mathfrak{C}}}\big(tot(\hat{\mathcal{R}})\big) \;\leq\; \sup\left\{\text{Ex}_{\hat{\sigma}', \hat{s}_I}^{\mathcal{M}_{\backslash\mathfrak{C}}}\big(tot(\hat{\mathcal{R}})\big) \;\middle|\; \hat{\sigma}' \in \Theta\right\} \;\leq\; v$$

// Build resulting strategy $\sigma \in \Sigma_{\text{PM}}^{\mathcal{M}}$
for $C \in \mathfrak{C}$ **do**
  if $\hat{\sigma}(C) = \langle s, \alpha \rangle \in exits(C)$ **then** *// Enforce leaving C via $\langle s, \alpha \rangle$*
    $\sigma(s) \leftarrow \alpha$
    Compute $\sigma_C^{\text{exit}} \in \Sigma_{\text{PM}}^{\mathcal{M}[\![C]\!]}$ with $\forall\, s' \in states(C)\colon \Pr_{\sigma_C^{\text{exit}}, s'}^{\mathcal{M}[\![C]\!]}\big(\Diamond\{s\}\big) = 1$
    for $s' \in states(C) \setminus \{s\}$ **do** $\sigma(s') \leftarrow \sigma_C^{\text{exit}}(s')$
  else *// $\hat{\sigma}(C) = \perp$—mimic optimal LRA strategy σ_C for $\mathcal{M}[\![C]\!]$*
    for $s \in states(C)$ **do** $\sigma(s) \leftarrow \sigma_C(s)$
for $s \in S \setminus \big(\bigcup_{C \in \mathfrak{C}} states(C)\big)$ **do** $\sigma(s) \leftarrow \hat{\sigma}(s)$
return $\langle v, \sigma \rangle$

Algorithm 5.3: Computing optimal expected values for combined objectives

Theorem 5.8 Algorithm 5.3 returns $\langle v, \sigma \rangle$ with

- $\sigma \in \Xi = \{\sigma' \in \Sigma^{\mathcal{M}} \mid \mathrm{Ex}^{\mathcal{M}}_{\sigma'}(\Phi) \in \mathbb{R}^{\ell}\}$ and
- $v - \varepsilon \leq \boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\sigma}(\Phi) \leq \sup\left\{\boldsymbol{w} \cdot \mathrm{Ex}^{\mathcal{M}}_{\hat{\sigma}}(\Phi) \;\middle|\; \hat{\sigma} \in \Xi\right\} \leq v.$

Algorithm 5.3 allows us to solve `WSO` instances over combinations of LRA- and total reward objectives by using a slight variation of Algorithm 5.2, where in Lines 3 to 4 we compute the pair $\langle v, \sigma \rangle$ by invoking Algorithm 5.3 instead.

Example 5.3 Consider the MA $\mathcal{M}$ from Figure 5.2a with objectives

$$\Phi = \left\langle tot(\mathcal{R}_1),\ tot(\mathcal{R}_2),\ lra(\mathcal{R}_3)\right\rangle$$

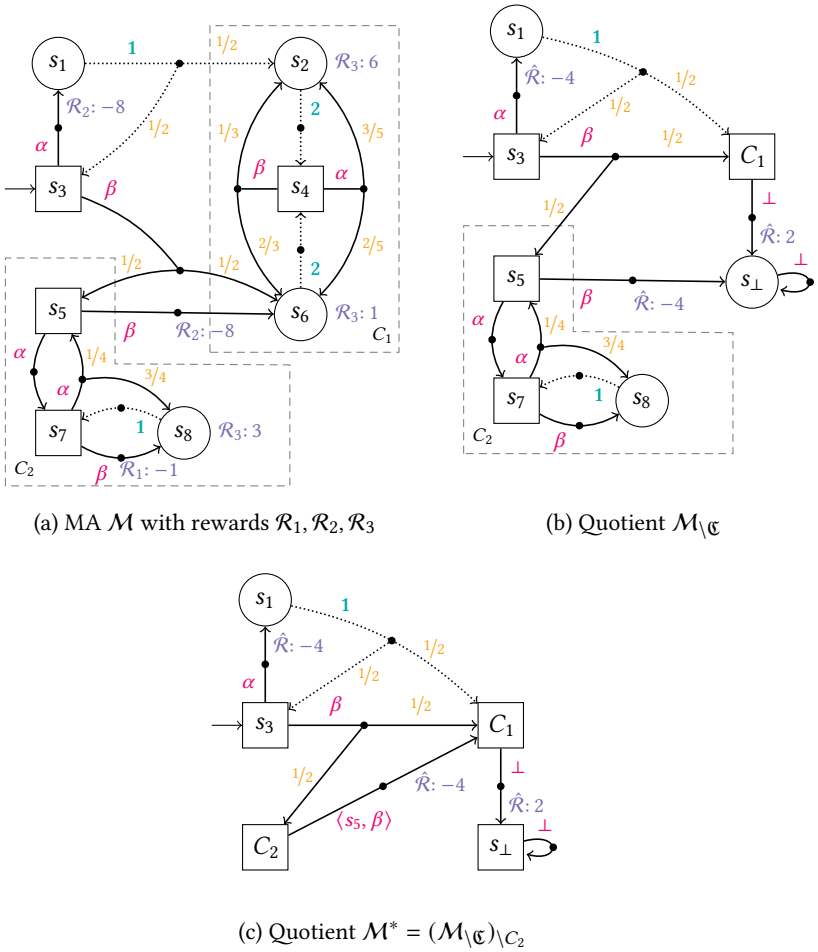
and weight vector $\boldsymbol{w} = \langle 0, 1/2, 1/2\rangle$. MA $\mathcal{M}$ and $lra(\mathcal{R}_3)$ coincide with the query from Examples 5.1 and 5.2.
We apply Algorithm 5.3 with $\varepsilon = 0$. The MEC C_1 of $\mathcal{M}$ is a 0-MEC with respect to $\mathcal{R}_1$ and $\mathcal{R}_2$. However C_2 is not (and does not contain) such a 0-MEC since the transition $\mathcal{R}_1[s_7, \beta, s_8] = -1 \neq 0$. We get $\mathfrak{C} = MEC_0(\mathcal{M}, \{\mathcal{R}_1, \mathcal{R}_2\}) = \{C_1\}$. The LRA reward value for C_1 is (cf. Example 5.1)

$$v_{C_1} = \mathrm{Ex}^{\mathcal{M}[\![C_1]\!]}_{\max}\big(lra(\mathcal{R}_{lra})\big) = \mathrm{Ex}^{\mathcal{M}[\![C_1]\!]}_{\max}\big(lra(1/2 \cdot \mathcal{R}_3)\big) = 1/2 \cdot 4 = 2.$$

The quotient $\mathcal{M}_{\setminus\mathfrak{C}}$ with its reward assignment $\hat{\mathcal{R}} = (\mathcal{R}_{tot})_{\setminus\mathfrak{C}} + \wr f \wr$ as in Algorithm 5.3 is shown in Figure 5.2b. For this quotient, the maximal expected total reward for objective $tot(\hat{\mathcal{R}})$ is induced by a strategy $\hat{\sigma}'$ that selects β at s_3 and α at s_5 which yields

$$\mathrm{Ex}^{\mathcal{M}_{\setminus\mathfrak{C}}}_{\max, s_3}(tot(\hat{\mathcal{R}})) = \mathrm{Ex}^{\mathcal{M}_{\setminus\mathfrak{C}}}_{\hat{\sigma}', s_3}(tot(\hat{\mathcal{R}})) = \underbrace{1/2 \cdot 2}_{\text{go to } C_1} + \underbrace{1/2 \cdot 0}_{\text{go to } C_2} = 1$$

However, the strategy $\hat{\sigma}'$ does not reach state $s_\perp$ almost surely. Indeed, when lifting $\hat{\sigma}'$ to the original MA $\mathcal{M}$, we get an expected value of $-\infty$ for the objective $tot(\mathcal{R}_1)$—which is considered invalid. We enforce strategies that reach $s_\perp$ almost surely by also collapsing the 0-MECs of $\mathcal{M}_{\setminus\mathfrak{C}}$ w.r.t. $\hat{\mathcal{R}}$, i.e., we also collapse C_2 which yields MA $\mathcal{M}^*$ as outlined in Figure 5.2c. Observe that there is no direct transition from C_2 to $s_\perp$ in $\mathcal{M}^*$. Now the maximal expected total reward for



(a) MA $\mathcal{M}$ with rewards $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$

(b) Quotient $\mathcal{M}_{\setminus \mathfrak{C}}$

(c) Quotient $\mathcal{M}^* = (\mathcal{M}_{\setminus \mathfrak{C}})_{\setminus C_2}$

Figure 5.2: Example for mixed LRA- and total reward analysis (cf. Example 5.3)

objective $tot(\hat{\mathcal{R}})$ is obtained when selecting β at s_3 and $\langle s_5, \beta \rangle$ at C_2 yielding

$$v = \mathrm{Ex}_{\max,s_3}(tot(\hat{\mathcal{R}})) = \underbrace{1/2 \cdot 2}_{\text{go to } C_1} + \underbrace{1/2 \cdot (-4+2)}_{\text{go to } C_2} = 0.$$

Our approach for solving the considered WSO instances also correctly deals with the exact case ($\varepsilon_{\mathtt{WSO}} = 0$) by using existing, exact methods for single-objective LRA- and total reward analysis. As the number of memoryless deterministic strategies of $\mathcal{M}$ is bounded, Algorithm 5.3 also provides only finitely many different answers $\langle v, \sigma \rangle$ for fixed $\mathcal{M}$ and Φ and varying weight vectors $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$. Similar to Lemma 4.6 on page 112, we conclude the following—extending results for pure LRA queries [BBCF+14] to mixtures with total rewards.

Lemma 5.9 If Assumptions 5.1 and 5.2 hold, then $down(Ach(\Phi) \cap \mathbb{R}^\ell)$ is closed and is the downward convex hull of at most $|\Sigma^{\mathcal{M}}_{\mathrm{PM}}| = \prod_{s \in PS} |\Delta(s)|$ points.

5.4 Related Work

Computing the expected value of a *single* step-based LRA reward objective is a classical problem in MDP verification [Put94]. Similar to the total reward case, solutions are commonly based on *linear programming* [Alf97], *strategy iteration* [WBBH+10; KM17], or *value iteration* [ACDK+17]. These approaches have been lifted to LRA objectives for MAs [GHHK+14; GTHR+14; BWH17; But20].

[Alf97; EJ11; BDK14] consider the expected long-run *ratio* of two reward assignments for an MDP. As pointed out in [GHHK+14, Theorem 4.2], LRA objectives for MAs can be seen as special instances of such MDP ratio objectives.

Related work for *multiple* LRA reward objectives commonly builds upon linear programming (LP) which yields a polynomial time-complexity for the achievability problem [Cha07; BBCF+14; CKK17; ABAV20]. The LP formulation of [BBCF+14; CKK17] is implemented in MultiGain [BCFK15]—an extension of Prism [KNP11] for multi-objective LRA rewards.

[RRS17; CKK17] consider multi-objective combinations of *percentile queries* on MDP and LRA objectives. [BDKK+17] treats *resilient systems*, ensuring constraints on a repair mechanism while maximizing the expected LRA reward when being operational. The trade-off between expected LRA rewards and their variance is analyzed in [BCFK17]. [GZ18; Kře21] combine LRA objectives with constraints specified in linear temporal logic (LTL).

Multiple LRA reward objectives for *stochastic games* have been treated using LP [CD16] and value iteration over convex sets [BKTW15; BKW18]; the latter is

included in PRISM-GAMES [KNPS20; KPW18]. [BMRL+18] consider a setting where the average *accumulated* reward is optimized while maintaining that the accumulated reward stays within predefined bounds. These approaches can also be applied to MDPs when viewed as one-player stochastic games.

To our knowledge, arbitrary *mixtures* of LRA- and total reward objectives have not been considered so far. There is also no related work on multiple LRA reward objectives for *MAs*.

Chapter Summary

- ➜ Single LRA reward objectives are evaluated in *two phases*:
  Phase 1: compute the LRA value for each maximal end component
  Phase 2: optimize an expected total reward objective on the quotient
- ➜ The weighted sum optimization problem (WSO) for LRA reward objectives is reducible to *single-objective* LRA analysis.
- ➜ Combinations with total reward objectives are handled by dealing with 0-MECs.

—Chapter 6—

Multi-Reward Bounded Objectives

Outlook | We consider instances for the weighted sum optimization problem (WSO, Problem 3.4 on page 81) concerning an MDP and arbitrary combinations of bounded reward objectives (Definition 2.31 on page 57) and bounded reachability objectives (Definition 2.33 on page 58). We outline two approaches for those queries. First, Section 6.1 presents a classical unfolding approach that extends the state space with accumulated rewards and reached goals—resulting in the so-called unfolding MDP. We formulate total reward objectives for the unfolding MDP and argue that the given WSO instance can be solved using techniques from Chapter 4. The second approach—outlined in Section 6.2—avoids an explicit construction of the potentially large unfolding MDP by exploiting certain regularities of the unfolding. This procedure is more memory-efficient, in particular when there are very large reward bound values. Section 6.3 ends the chapter with an overview of related work.

Origins | The contents of this chapter originate from [3; 10]. Their presentation has been revised and generalized towards bounded reward objectives as well as more expressive bounded reachability objectives as given in Definitions 2.31 and 2.33 on page 57 and on page 58. The explicit unfolding approach from Section 6.1 is based on [AHK03; RRS17]. Avoiding an explicit construction of the unfolding MDP has also been described in [BDDK+14; HH16] for objectives that only consider a single reward bound.

Set-up | We consider an MDP $M = \langle S, Act, \Delta, \mathbf{P} \rangle$, with dedicated initial state $s_I \in S$ and a list of $\ell_R \geq 0$ *convergent* bounded reward objectives and $\ell_G \geq 0$ bounded reachability objectives

$$\Phi = \Big\langle bnd^{\mathcal{B}}(\mathcal{R}'_1, \mathcal{I}_1), \ldots, bnd^{\mathcal{B}}(\mathcal{R}'_{\ell_R}, \mathcal{I}_{\ell_R}), bnd^{\mathcal{B}}_{\mathcal{G}}(c_{\ell_R+1}, \mathcal{J}_{\ell_R+1}), \ldots, bnd^{\mathcal{B}}_{\mathcal{G}}(c_{\ell_R+\ell_G}, \mathcal{J}_{\ell_R+\ell_G}) \Big\rangle,$$

where the total number of objectives is given by $\ell := \ell_R + \ell_G > 0$ and all objectives refer to (potentially different entries of) the same lists of

- $d > 0$ reward bounds $\mathcal{B} = \left\langle \llparenthesis \mathcal{R}_1 \sim_1 b_1 \rrparenthesis, \ldots, \llparenthesis \mathcal{R}_d \sim_d b_d \rrparenthesis \right\rangle$ and
- $m \geq 0$ bounded goals $\mathcal{G} = \left\langle \langle \mathcal{I}_1, G_1 \rangle, \ldots, \langle \mathcal{I}_m, G_m \rangle \right\rangle$ over $\mathcal{B}$.

Furthermore, we consider a weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$, and a precision parameter $\varepsilon_{\texttt{WSO}}$ which—together with M, s_I, and Φ—form an instance to WSO (Problem 3.4 on page 81).

As outlined in Section 2.3.3, (unbounded) total reachability reward objectives can also be written as bounded reward objectives, i.e., we implicitly also consider mixtures of bounded and unbounded objectives. Remark 6.3 addresses the incorporation of long-run average reward objectives.

Assumptions | We make the following simplifying assumption regarding the reward bounds in $\mathcal{B}$.

Assumption 6.1 For all $i \in \{1..d\}$ we have $\mathcal{B}(i) = \llparenthesis \mathcal{R}_i \sim_i b_i \rrparenthesis$ with

(i) $\mathcal{R}_i(s, \alpha, s') \in \mathbb{N}$ for all $s, s' \in S$ and $\alpha \in \Delta(s) \subseteq Act$,

(ii) $\sim_i \in \{\leq, >\}$, and

(iii) $b_i \in \mathbb{N}$.

In case a reward bound $\mathcal{B}(i) = \llparenthesis \mathcal{R}_i \sim_i b_i \rrparenthesis$ violates Assumption 6.1, we can transform it to an equivalent reward bound that satisfies the assumption *if* the inverses of all assigned non-zero rewards have a common multiple, i.e., there is $a_i \in \mathbb{R} \setminus \{0\}$ such that

$$\forall\, s, s' \in S,\ \alpha \in \Delta(s) \colon\ \mathcal{R}_i(s, \alpha, s') \neq 0 \implies \exists\, n \in \mathbb{N} \colon\ n \cdot (\mathcal{R}_i(s, \alpha, s'))^{-1} = a_i.$$

The transformation—outlined below—in particular allows us to consider cases where all rewards assigned by $\mathcal{R}_i$ are non-negative (or non-positive) rational numbers. Optimal expected values for objectives with reward bounds considering arbitrary mixtures of positive and negative rewards can not be computed with our techniques—and are not even computable in general (cf. Theorem 2.14 on page 60).

The transformation for reward bound $\llparenthesis \mathcal{R}_i \sim_i b_i \rrparenthesis$ with common multiple a_i as above is as follows. First, we note that the scaled reward assignment $(a_i \cdot \mathcal{R}_i)$ only assigns rewards in $\mathbb{N}$: for $s, s' \in S$ and $\alpha \in \Delta(s)$ with $\mathcal{R}_i(s, \alpha, s') \neq 0$ there is $n \in \mathbb{N}$ with

$$n \cdot (\mathcal{R}_i(s, \alpha, s'))^{-1} = a_i \quad \iff \quad n = a_i \cdot \mathcal{R}_i(s, \alpha, s') = (a_i \cdot \mathcal{R}_i)(s, \alpha, s').$$

We can thus satisfy Item (i) of Assumption 6.1 by replacing the original reward bound

with the equivalent bound $\wr(a_i \cdot \mathcal{R}_i) \sim'_i (a_i \cdot b_i)\wr$ as in Lemma 2.13 on page 59. Since the comparison with the new bound value $a_i \cdot b_i$ according to relation $\sim'_i$ is only made with natural numbers, we can always satisfy Items (ii) and (iii) by adapting the strictness of $\sim'_i$ and by rounding, incrementing, or decrementing $a_i \cdot b_i$ accordingly. If this introduces a negative bound value, the reward bound is known to be either always active (e.g., $\wr\mathcal{R} > -1\wr$) or never active (e.g., $\wr\mathcal{R} \leq -1\wr$). We can remove such bounds from $\mathcal{B}$ and simplify the objectives accordingly.

We also assume that the considered bounded reachability objectives $bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})$ are not immediately satisfied at the initial state s_I.

Assumption 6.2 For every bounded reachability objective $bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})$, there is some index $j \in \mathcal{J}$ with $\mathcal{G}(j) = \langle \mathcal{I}_j, G_j \rangle$ such that either

- $s_I \notin G_j$ or
- some $i \in \mathcal{I}_j$ refers to a bound $\mathcal{B}(i) = \wr\mathcal{R}_i \sim_i b_i\wr$ that is initially not active (i.e., $\sim_i = >$).

Assumption 6.2 allows for a more straightforward transformation of $bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})$ to a total reward objective as outlined in Section 6.1.3. The assumption can be checked in a preprocessing step. If an objective $bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})$ violates the assumption, the objective will always have value c (for all strategies). Hence, there is no further analysis required for those objectives.

Example 6.1 As a running example for this chapter, consider the MDP M from Figure 6.1 with initial state s_0, $d = 2$ reward bounds $\mathcal{B} = \langle \wr\mathcal{R}_1 \leq 1\wr, \wr\mathcal{R}_2 > 3\wr \rangle$, $m = 1$ bounded goal $\mathcal{G} = \langle \langle \{2\}, \{s_2\} \rangle \rangle$, and objectives

$$\Phi = \langle bnd^{\mathcal{B}}(\mathcal{R}', \{1\}),\ bnd^{\mathcal{B}}_{\mathcal{G}}(1, \{1\}) \rangle.$$

The first objective $\Phi(1)$ asks for the expected reward that is accumulated for $\mathcal{R}'$ as long as at most 1 reward is accumulated for $\mathcal{R}_1$. The second objective $\Phi(2)$ concerns the probability for accumulating more than 3 reward for $\mathcal{R}_2$ and then reaching s_2 afterwards.
Let $\sigma \in \Sigma^M$ be the strategy for M that repeatedly selects action α at s_0 until the self-loop transition $s_0 \xrightarrow{\alpha} s_0$ has been taken twice. After that, σ always selects β at s_0.
The strategy σ yields the maximal expected value for both objectives: intuitively, σ attempts to move to s_1 as often as possible to collect as much reward for $\mathcal{R}'$ as possible. This maximizes the value for the first objective $\Phi(1)$. Once moving to s_1 has failed two times, more than 1 reward has been accumulated for $\mathcal{R}_1$ and there

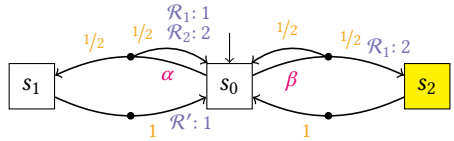


Figure 6.1: Example MDP M with reward assignments $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}'$ and goal state set $\{s_2\}$.

is no further benefit when moving to s_1. At this point, the expected accumulated reward with respect to $\mathcal{R}'$ is 2 and we have collected more than 3 reward for $\mathcal{R}_2$. The strategy now optimizes for the second objective by repeatedly attempting to reach s_2, which eventually succeeds almost surely.
The Pareto front for Φ is thus given by a single point:

$$Pareto^M_{s_0}(\Phi) = \left\{\mathrm{Ex}^M_{\sigma,s_I}(\Phi)\right\} = \left\{\langle 2, 1\rangle\right\}.$$

6.1 Explicit Unfolding

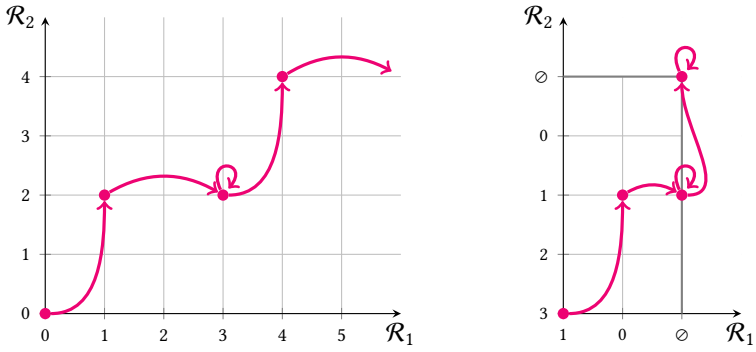
The classical approach to analyze reward-bounded objectives is to unfold the accumulated rewards into the state space. In [RRS17], this has been applied to multi-objective model checking of bounded reachability objectives, where each individual objective considers a single bounded goal with a single reward bound. We now extend this approach towards *general combinations of multi-reward bounded objectives.*

Consider some path $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \dots$ of MDP M. To compute the value of a bounded reward objective $bnd^{\mathcal{B}}(\mathcal{R}, \mathcal{I})(\pi)$ or a bounded reachability objective $bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})(\pi)$ for π, we intuitively have to keep track of the accumulated rewards for each relevant reward bound $\wr\mathcal{R}_i \sim_i b_i\wr$ to decide if the reward bound is active on a given fragment of π.

Example 6.2 For the MDP M from Figure 6.1 and Example 6.1, consider the path $\pi \in Paths^M_{\mathrm{inf}}(s_0)$ with

$$\pi = s_0 \xrightarrow{\alpha} s_0 \xrightarrow{\beta} s_2 \to s_0 \xrightarrow{\alpha} s_0 \xrightarrow{\beta} s_2 \to s_0 \xrightarrow{\alpha} \dots = \left(s_0 \xrightarrow{\alpha} s_0 \xrightarrow{\beta} s_2 \to\right)^\omega.$$

We plot the collected rewards for reward assignments $\mathcal{R}_1$ and $\mathcal{R}_2$ in Figure 6.2a. Starting from $\langle 0, 0\rangle$, the first transition yields rewards 1 and 2 for $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively: we jump to coordinate $\langle 1, 2\rangle$. Moving to s_2 incurs reward 2 for $\mathcal{R}_1$,



(a) Accumulated rewards (b) Reward epochs

Figure 6.2: An illustration of reward epochs (cf. Examples 6.2 and 6.3)

jumping to coordinate $\langle 3, 2\rangle$. The next transition—back to s_0—has no reward, so we stay at $\langle 3, 2\rangle$. This series of updates repeats ad infinitum.
Considering the reward bounds from Example 6.1, we observe that the reward bound $⁅\mathcal{R}_1 \leq 1⁆$ is only active until we reach coordinate $\langle 3, 2\rangle$ while reward bound $⁅\mathcal{R}_2 > 3⁆$ becomes active once we are at coordinate $\langle 4, 4\rangle$.

Besides the accumulated rewards, we also have to remember which of the referred bounded goals are already satisfied when evaluating a bounded reachability objective. Therefore, to obtain the objective value accumulated in the n^{th} step of a path π, the complete prefix $pref(\pi, n)$ is relevant. This is different from, e.g., total reward objectives $tot(\mathcal{R})$, where only the transition $s_{n-1} \xrightarrow{\alpha_{n-1}} s_n$ is relevant for the value accumulated at step n of π.

Our goal in this section is to build a new MDP—the unfolding MDP—from our input MDP M on which we can encode the given reward-bounded objectives as (unbounded) total reward objectives. To allow such a transformation, we "attach" some extra information to the states of the unfolding MDP from which we can infer

(i) which reward bounds are currently active, and

(ii) which bounded goals are already satisfied.

This extra information is formally given in the form of *reward epochs* and *goal satisfactions*—two central concepts of our approach.

6.1.1 Reward Epochs

reward epoch

Definition 6.1 (Reward Epoch) A *d-dimensional reward epoch* (or simply *epoch*) is a tuple

$$\boldsymbol{e} = \langle e_1, \ldots, e_d \rangle \in (\mathbb{N} \cup \{\oslash\})^d.$$

The set of all d-dimensional epochs is denoted by $\mathbb{E}_d := (\mathbb{N} \cup \{\oslash\})^d$.

Given d reward bounds $\mathcal{B} = \langle \wr \mathcal{R}_1 \sim_1 b_1 \wr, \ldots, \wr \mathcal{R}_d \sim_d b_d \wr \rangle$, a reward epoch intuitively keeps track of the rewards *that can still be accumulated for each reward bound without changing the activity status of that bound.* Starting with the initial epoch

$$\boldsymbol{e}^{\mathcal{B}}_{init} := \langle b_1, \ldots, b_d \rangle,$$

the current epoch evolves by *subtracting* element-wise the rewards accumulated for $\mathcal{R}_1, \ldots, \mathcal{R}_d$. Values below zero are collapsed using the dedicated symbol $\oslash$. Formally, define the *monus* operation $\ominus \colon \mathbb{E}_d \times \mathbb{N}^d \to \mathbb{E}_d$, given for $\boldsymbol{e} \in \mathbb{E}_d$ and $\boldsymbol{r} \in \mathbb{N}^d$ by

$$\forall\, i \in \{1..d\} \colon \quad (\boldsymbol{e} \ominus \boldsymbol{r})(i) := \begin{cases} \boldsymbol{e}(i) - \boldsymbol{r}(i) & \text{if } \boldsymbol{e}(i) \neq \oslash \text{ and } \boldsymbol{e}(i) - \boldsymbol{r}(i) \geq 0 \\ \oslash & \text{otherwise.} \end{cases}$$

Taking a transition in the MDP M from $s \in S$ via $\alpha \in \Delta(s)$ to $s' \in \mathit{post}(\langle s, \alpha \rangle)$ changes the current epoch $\boldsymbol{e}$ to $\boldsymbol{e}' = \boldsymbol{e} \ominus \boldsymbol{r}^{\mathcal{B}}_{s,\alpha,s'}$, with

$$\boldsymbol{r}^{\mathcal{B}}_{s,\alpha,s'} := \langle \mathcal{R}_1(s, \alpha, s'), \ldots, \mathcal{R}_d(s, \alpha, s') \rangle \in \mathbb{N}^d.$$

Example 6.3 Reconsider path π and reward bounds $\mathcal{B} = \langle \wr \mathcal{R}_1 \leq 1 \wr, \wr \mathcal{R}_2 > 3 \wr \rangle$ as in Examples 6.1 and 6.2. Starting from $\boldsymbol{e}_0 = \boldsymbol{e}^{\mathcal{B}}_{init} = \langle 1, 3 \rangle$, the resulting development of reward epoch is shown in Figure 6.2b.
As already outlined in Example 6.2, the first transition incurs reward 1 for $\mathcal{R}_1$ and reward 2 for $\mathcal{R}_2$. Subsequently, the remaining rewards that can still be collected without affecting the activity of the two reward bounds are $1 - 1 = 0$ and $3 - 2 = 1$, respectively. We move to the reward epoch $\boldsymbol{e}_1 = \boldsymbol{e}_0 \ominus \langle 1, 2 \rangle = \langle 0, 1 \rangle$. The second transition incurs reward 2 for $\mathcal{R}_1$ and reward 0 for $\mathcal{R}_2$. Hence, the next epoch is $\boldsymbol{e}_2 = \boldsymbol{e}_1 \ominus \langle 2, 0 \rangle = \langle \oslash, 1 \rangle$. From this point on, the first reward bound $\wr \mathcal{R}_1 \leq 1 \wr$ is no longer active as we have collected more than one reward for $\mathcal{R}_1$.
Next, we take a transition that does not incur any rewards and thus the epoch is $\boldsymbol{e}_3 = \boldsymbol{e}_2 \ominus \langle 0, 0 \rangle = \langle \oslash, 1 \rangle$. The fourth transition yields again rewards 1 and 2, respectively, and the epoch changes to $\boldsymbol{e}_4 = \boldsymbol{e}_3 \ominus \langle 1, 2 \rangle = \langle \oslash, \oslash \rangle$. After that, we have collected more than 3 reward for $\mathcal{R}_2$ and thus the second reward bound $\wr \mathcal{R}_2 > 3 \wr$ becomes active. Subsequent transitions do not change the epoch $\langle \oslash, \oslash \rangle$.

Recall from Section 2.3.3 that a bound $\mathcal{B}(i) = \wr \mathcal{R}_i \sim_i b_i \wr$ is called *active* at position n of a path $\pi \in Paths^M_{\text{inf}}$ iff $\mathcal{R}_i[pref(\pi, n)] \sim_i b_i$. We now lift this notion to reward epochs. Consider the function $active^{\mathcal{B}} \colon \mathbb{E}_d \to 2^{\{1..d\}}$ with

$$active^{\mathcal{B}}(\boldsymbol{e}) \;:=\; \big\{ i \in \{1..d\} \;\big|\; (\boldsymbol{e}(i) \in \mathbb{N} \text{ and } \sim_i = \leq) \text{ or } (\boldsymbol{e}(i) = \oslash \text{ and } \sim_i = >) \big\}.$$

The set $active^{\mathcal{B}}(\boldsymbol{e})$ identifies the active bounds for a given epoch $\boldsymbol{e}$. The following lemma shows that this is consistent to the path-based notion of active bounds.

Lemma 6.1 For path $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \dots$ of MDP M and infinite sequence of epochs $\boldsymbol{e}_0, \boldsymbol{e}_1, \dots \in \mathbb{E}_d$ with $\boldsymbol{e}_0 = \boldsymbol{e}^{\mathcal{B}}_{init}$ and $\forall\, n \in \mathbb{N} \colon\; \boldsymbol{e}_{n+1} = \boldsymbol{e}_n \ominus \boldsymbol{r}^{\mathcal{B}}_{s_n, \alpha_n, s_{n+1}}$, we have for all $i \in \{1..d\}$ and $n \in \mathbb{N}$ that

$$\mathcal{R}_i[pref(\pi, n)] \sim_i b_i \quad \text{iff} \quad i \in active^{\mathcal{B}}(\boldsymbol{e}_n).$$

Proof. We first show by induction that for all $n \in \mathbb{N}$:

- $\boldsymbol{e}_n(i) \in \mathbb{N}$ implies $\mathcal{R}_i[pref(\pi, n)] = b_i - \boldsymbol{e}_n(i)$, and
- $\boldsymbol{e}_n(i) = \oslash$ implies $\mathcal{R}_i[pref(\pi, n)] > b_i$.

Induction Base | For $n = 0$ we have $\boldsymbol{e}_0(i) = \boldsymbol{e}^{\mathcal{B}}_{init}(i) = b_i \in \mathbb{N}$ and $\mathcal{R}_i[pref(\pi, 0)] = 0 = b_i - b_i = b_i - \boldsymbol{e}_0(i)$.

Induction Step | Assuming that the claim holds for fixed $n \in \mathbb{N}$, we distinguish the following cases.

- If $\boldsymbol{e}_n(i) \in \mathbb{N}$, the induction hypothesis yields

$$\begin{aligned} \mathcal{R}_i[pref(\pi, n+1)] &= \mathcal{R}_i[pref(\pi, n)] + \mathcal{R}_i(s_n, \alpha_n, s_{n+1}) \\ &= b_i - \boldsymbol{e}_n(i) + \mathcal{R}_i(s_n, \alpha_n, s_{n+1}). \end{aligned} \tag{6.1}$$

  We further distinguish whether the reward $\mathcal{R}_i(s_n, \alpha_n, s_{n+1})$ incurred at the $(n{+}1)^{\text{th}}$ step of π exceeds the epoch value $\boldsymbol{e}_n(i)$.

  - If $\mathcal{R}_i(s_n, \alpha_n, s_{n+1}) > \boldsymbol{e}_n(i)$, then $\boldsymbol{e}_{n+1}(i) = \oslash$. Furthermore, using Equation (6.1) we get

$$\mathcal{R}_i[pref(\pi, n+1)] \;=\; b_i + \underbrace{\mathcal{R}_i(s_n, \alpha_n, s_{n+1}) - \boldsymbol{e}_n(i)}_{>0} \;>\; b_i.$$

  - If $\mathcal{R}_i(s_n, \alpha_n, s_{n+1}) \leq \boldsymbol{e}_n(i)$, then we have $\boldsymbol{e}_{n+1}(i) = \boldsymbol{e}_n(i) - \mathcal{R}_i(s_n, \alpha_n, s_{n+1}) \in \mathbb{N}$ and thus $\boldsymbol{e}_n(i) = \boldsymbol{e}_{n+1}(i) + \mathcal{R}_i(s_n, \alpha_n, s_{n+1})$. Inserting into Equation (6.1)

yields

$$\mathcal{R}_i[\mathit{pref}(\pi, n+1)] \;=\; b_i - \boldsymbol{e}_{n+1}(i).$$

- If $\boldsymbol{e}_n(i) = \infty$, then $\boldsymbol{e}_{n+1}(i) = \infty$ and $\mathcal{R}_i[\mathit{pref}(\pi, n)] > b_i$ holds. Since $\mathcal{R}_i$ only assigns non-negative rewards, we have

$$\mathcal{R}_i[\mathit{pref}(\pi, n+1)] \;=\; \underbrace{\mathcal{R}_i[\mathit{pref}(\pi, n)]}_{>b_i} + \underbrace{\mathcal{R}_i(s_n, \alpha_n, s_{n+1})}_{\geq 0} \;>\; b_i.$$

Inferring the Lemma | Since $b_i - \boldsymbol{e}_n(i) \leq b_i$ for all $n \in \mathbb{N}$, a consequence of the claim shown above is that

$$\boldsymbol{e}_n(i) \;\in\; \mathbb{N} \quad \text{iff} \quad \mathcal{R}_i[\mathit{pref}(\pi, n)] \;\leq\; b_i,$$

or, equivalently,

$$\boldsymbol{e}_n(i) \;=\; \infty \quad \text{iff} \quad \mathcal{R}_i[\mathit{pref}(\pi, n)] \;>\; b_i.$$

The following case distinction concludes the proof of the lemma.

- If $\sim_i = \leq$, then

$$i \;\in\; \mathit{active}^{\mathcal{B}}(\boldsymbol{e}_n) \quad \text{iff} \quad \boldsymbol{e}_n(i) \;\in\; \mathbb{N} \quad \text{iff} \quad \mathcal{R}_i[\mathit{pref}(\pi, n)] \;\leq\; b_i.$$

- If $\sim_i = >$, then

$$i \;\in\; \mathit{active}^{\mathcal{B}}(\boldsymbol{e}_n) \quad \text{iff} \quad \boldsymbol{e}_n(i) \;=\; \infty \quad \text{iff} \quad \mathcal{R}_i[\mathit{pref}(\pi, n)] \;>\; b_i. \qquad \blacksquare$$

6.1.2 Goal Satisfactions

goal satisfaction

Definition 6.2 (Goal Satisfaction) A *goal satisfaction* over m bounded goals is a set $\mathfrak{g} \subseteq \{1..m\}$. $\mathbb{G}_m := 2^{\{1..m\}}$ denotes the set of all goal satisfactions over m bounded goals.

Given bounded goals $\mathcal{G} = \big\langle \langle \mathcal{I}_1, G_1 \rangle, \ldots, \langle \mathcal{I}_m, G_m \rangle \big\rangle$ over reward bounds $\mathcal{B}$, a goal satisfaction $\mathfrak{g} \in \mathbb{G}_m$ encodes which goals are already satisfied. The function $\mathit{sat}^{\mathcal{B}}_{\mathcal{G}} \colon \mathbb{E}_d \times S \to \mathbb{G}_m$ with

$$\mathit{sat}^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}, s) \;:=\; \big\{ j \in \{1..m\} \;\big|\; \mathcal{I}_j \subseteq \mathit{active}^{\mathcal{B}}(\boldsymbol{e}) \text{ and } s \in G_j \big\}$$

intuitively yields the "local" goal satisfaction for a given epoch $\boldsymbol{e} \in \mathbb{E}_d$ and a given state $s \in S$: when the model execution arrives at state s and epoch $\boldsymbol{e}$, the bounded goals

given by $sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}, s)$ are satisfied at that particular point of the execution. However, we are not only interested in the currently satisfied bounded goals, but also in the bounded goals that have been satisfied before. Goal satisfactions are used to keep track of this information. Starting with the initial goal satisfaction $sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}^{\mathcal{B}}_{init}, s_I)$, the idea is to add new indices $j \in \{1..m\}$ to the current goal satisfaction $\mathfrak{g}$, whenever the corresponding bounded goal $\mathcal{G}(j) = \langle \mathcal{I}_j, G_j \rangle$ is satisfied.

Example 6.4 For M, $\mathcal{B}$, and $\mathcal{G} = \big\langle \langle \{2\}, \{s_2\} \rangle \big\rangle$ as in Example 6.1 we have

$$sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}, s) = \begin{cases} \{1\} & \text{if } \boldsymbol{e}(2) = \oslash \text{ and } s = s_2 \\ \emptyset & \text{otherwise.} \end{cases}$$

For the path $\pi = s_0 \xrightarrow{\alpha} s_0 \xrightarrow{\beta} s_2 \to s_0 \xrightarrow{\alpha} s_0 \xrightarrow{\beta} s_2 \to s_0 \xrightarrow{\alpha} \ldots =$ and the sequence of epochs $\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots$ as in Examples 6.2 and 6.3, we initially consider goal satisfaction $\mathfrak{g}_0 = sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}^{\mathcal{B}}_{init}, s_0) = sat^{\mathcal{B}}_{\mathcal{G}}(\langle 1, 3 \rangle, s_0) = \emptyset$. The goal satisfaction does not change for the first four steps, i.e., $\mathfrak{g}_0 = \mathfrak{g}_1 = \mathfrak{g}_2 = \mathfrak{g}_3 = \mathfrak{g}_4 = \emptyset$. In the fifth step of π we visit s_2 and the current reward epoch is $\boldsymbol{e}_5 = \langle \oslash, \oslash \rangle$, i.e., the second reward bound is active. Hence, the goal satisfaction changes to $\mathfrak{g}_5 = \{1\} = \mathfrak{g}_6 = \mathfrak{g}_7 = \ldots$. Thus, $\pi \models \mathcal{G}(1) = \langle \{2\}, \{s_2\} \rangle$.

Lemma 6.2 Let $\mathcal{J} \subseteq \{1..m\}$ be a set of indices referring to bounded goals. For path $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \ldots$ of MDP M, infinite sequence of epochs $\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots$ as in Lemma 6.1, and infinite sequence of goal satisfactions $\mathfrak{g}_0, \mathfrak{g}_1, \ldots$ with $\mathfrak{g}_0 = sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}_0, s_0)$ and $\forall\, n \in \mathbb{N}\colon\ \mathfrak{g}_{n+1} = \mathfrak{g}_n \cup sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}_{n+1}, s_{n+1})$, we have

$$\forall\, j \in \mathcal{J}\colon\ \pi \models \langle \mathcal{I}_j, G_j \rangle \quad \text{iff} \quad \exists\, n \in \mathbb{N}\colon\ \mathcal{J} \subseteq \mathfrak{g}_n$$

Proof. Recall the definition of the "$\models$" relation from page 58:

$$\pi \models \langle \mathcal{I}_j, G_j \rangle \quad \text{iff} \quad \exists\, \hat{\pi} \in pref(\pi)\colon\ last(\hat{\pi}) \in G_j \text{ and } \forall\, i \in \mathcal{I}_j\colon\ \mathcal{R}_i[\hat{\pi}] \sim_i b_i.$$

Using Lemma 6.1, we get for $j \in \{1..m\}$:

$$\begin{aligned} \pi \models \langle \mathcal{I}_j, G_j \rangle \quad &\text{iff} \quad \exists\, n \in \mathbb{N}\colon\ s_n \in G_j \text{ and } \mathcal{I}_j \subseteq active^{\mathcal{B}}(\boldsymbol{e}_n) \\ &\text{iff} \quad \exists\, n \in \mathbb{N}\colon\ j \in sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}_n, s_n) \\ &\text{iff} \quad \exists\, n \in \mathbb{N}\colon\ j \in \mathfrak{g}_n. \end{aligned}$$

Now assume that $\forall\, j \in \mathcal{J}\colon\ \pi \models \langle \mathcal{I}_j, G_j \rangle$. The above equivalence yields that for each $j \in \mathcal{J}$ there is $n_j \in \mathbb{N}$ with $j \in \mathfrak{g}_{n_j}$. By noting that $\mathfrak{g}_0 \subseteq \mathfrak{g}_1 \subseteq \ldots$, we conclude

$\mathcal{J} \subseteq \mathfrak{g}_n$ for $n = \max\{n_j \mid j \in \mathcal{J}\}$.
On the other hand, if $\mathcal{J} \subseteq \mathfrak{g}_n$ for some $n \in \mathbb{N}$, then the above equivalence immediately yields $\pi \models \langle \mathcal{I}_j, G_j \rangle$ for all $j \in \mathcal{J}$. ■

6.1.3 Unfolding MDP

We now define the *unfolding MDP*—an MDP on which the given reward-bounded objectives can be restated using (unbounded) total reward objectives. The idea is to incorporate reward epochs and goal satisfactions into the state space in a way that Lemmas 6.1 and 6.2 can be applied to paths in the unfolding MDP. For a given reward-bounded objective, this transformation then allows us to determine the value accumulated at a certain step $\langle s_n, \boldsymbol{e}_n, \mathfrak{g}_n \rangle \xrightarrow{\alpha_n} \langle s_{n+1}, \boldsymbol{e}_{n+1}, \mathfrak{g}_{n+1} \rangle$ of a path in the unfolding MDP by only considering the information given locally at that step.

unfolding MDP

Definition 6.3 (Unfolding MDP) The *unfolding MDP* of MDP $M = \langle S, Act, \Delta, \mathbf{P} \rangle$, $d > 0$ reward bounds $\mathcal{B} = \big\langle \llbracket \mathcal{R}_1 \sim_1 b_1 \rrbracket, \ldots, \llbracket \mathcal{R}_d \sim_d b_d \rrbracket \big\rangle$, and $m \geq 0$ bounded goals $\mathcal{G} = \big\langle \langle \mathcal{I}_1, G_1 \rangle, \ldots, \langle \mathcal{I}_m, G_m \rangle \big\rangle$ is given by the MDP $un^{\mathcal{B}}_{\mathcal{G}}(M) = \big\langle S_{un}, Act, \Delta_{un}, \mathbf{P}_{un} \big\rangle$, where

- $S_{un} \coloneqq S \times \mathbb{E}_d \times \mathbb{G}_m$,
- for $\langle s, \boldsymbol{e}, \mathfrak{g} \rangle \in S_{un}$: $\Delta_{un}\big(\langle s, \boldsymbol{e}, \mathfrak{g} \rangle\big) \coloneqq \Delta(s)$,
- and for $s_{un} = \langle s, \boldsymbol{e}, \mathfrak{g} \rangle$, $s'_{un} = \langle s', \boldsymbol{e}', \mathfrak{g}' \rangle \in S_{un}$ and $\alpha \in \Delta(s)$:

$$\mathbf{P}_{un}\big(s_{un}, \alpha, s'_{un}\big) \coloneqq \mathbf{P}(s, \alpha, s') \cdot [\boldsymbol{e}' = \boldsymbol{e} \ominus \boldsymbol{r}^{\mathcal{B}}_{s,\alpha,s'}] \cdot [\mathfrak{g}' = \mathfrak{g} \cup sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}', s')].$$

When the reward bounds $\mathcal{B}$ and bounded goals $\mathcal{G}$ are clear from context, we write $un(M)$ instead of $un^{\mathcal{B}}_{\mathcal{G}}(M)$. Given the initial state $s_I \in S$ of M, we consider

$$s^{un}_I \coloneqq \big\langle s_I, \boldsymbol{e}^{\mathcal{B}}_{init}, sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}^{\mathcal{B}}_{init}, s_I) \big\rangle \in S_{un}$$

as the corresponding initial state of $un(M)$. The unfolding MDP is infinite since there are infinitely many epochs $\boldsymbol{e} \in \mathbb{E}_d$. However, we are only interested in the fragment of $un(M)$ that is reachable from s^{un}_I. Since the rewards for $\mathcal{R}_1, \ldots, \mathcal{R}_d$ are required to be non-negative integers (cf. Assumption 6.1), the entries of successor epochs can only become smaller (or $\oslash$). Thus, the states reachable from s^{un}_I are contained in the finite set

$$S^{fin}_{un} \coloneqq \big\{ \langle s, \boldsymbol{e}, \mathfrak{g} \rangle \in S_{un} \;\big|\; \forall\, i \in \{1..d\} \colon \boldsymbol{e}(i) = \oslash \text{ or } 0 \leq \boldsymbol{e}(i) \leq b_i \big\}.$$

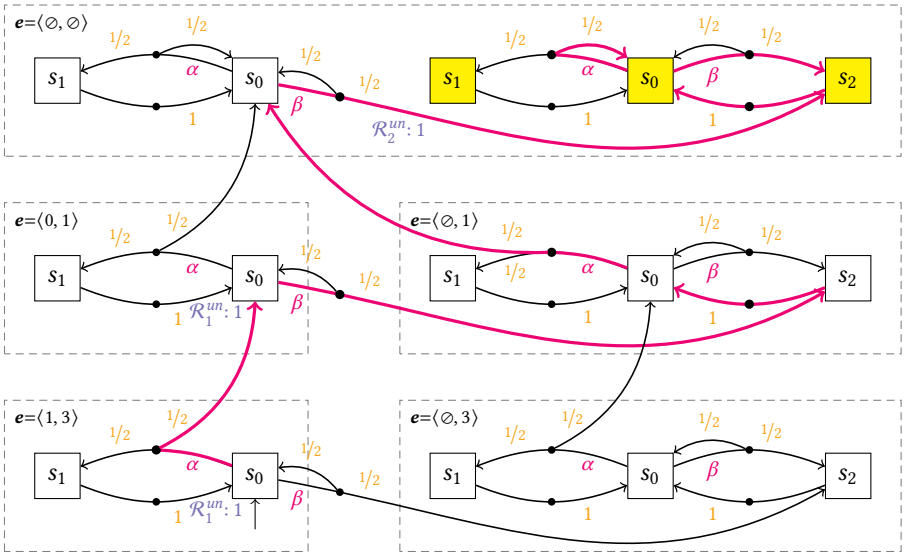


Figure 6.3: Reachable fragment of the unfolding MDP $un_{\mathcal{G}}^{\mathcal{B}}(M)$ of MDP M with reward assignments $\mathcal{R}_1^{un}$ and $\mathcal{R}_2^{un}$, and path $un_{\mathrm{p}}(\pi)$ in pink (cf. Examples 6.5 to 6.7).

Example 6.5 The unfolding MDP $un_{\mathcal{G}}^{\mathcal{B}}(M)$ of MDP M from Figure 6.1 with reward bounds $\mathcal{B}$ and bounded goals $\mathcal{G}$ from Example 6.1 is shown in Figure 6.3. We only depict the fragment of $un_{\mathcal{G}}^{\mathcal{B}}(M)$ that is reachable from $s_I^{un} = \langle s_0, \langle 1, 3\rangle, \emptyset\rangle$. In the figure, the epoch $\boldsymbol{e}$ of a state $\langle s, \boldsymbol{e}, \mathfrak{g}\rangle$ of $un_{\mathcal{G}}^{\mathcal{B}}(M)$ is derived from the dashed rectangle in which the state is drawn. Furthermore, states with $\mathfrak{g} = \{1\}$ are depicted in yellow. For the remaining states we have $\mathfrak{g} = \emptyset$.

Remark 6.1 The unfolding MDP can be seen as a generalization of the *goal unfolding* from Definition 4.1 on page 100 towards reward epochs and *bounded* goals. In fact, the observations below are strongly related to those of Section 4.1.1.

Next, we define total reward objectives for the unfolding MDP $un(M)$ to replace the given $\ell = \ell_R + \ell_G$ reward-bounded objectives

$$\Phi = \left\langle bnd^{\mathcal{B}}(\mathcal{R}'_1, \mathcal{I}_1), \ldots, bnd^{\mathcal{B}}(\mathcal{R}'_{\ell_R}, \mathcal{I}_{\ell_R}), bnd^{\mathcal{B}}_{\mathcal{G}}(c_{\ell_R+1}, \mathcal{J}_{\ell_R+1}), \ldots, bnd^{\mathcal{B}}_{\mathcal{G}}(c_{\ell_R+\ell_G}, \mathcal{J}_{\ell_R+\ell_G})\right\rangle.$$

For each objective $\Phi(j)$ $(j \in \{1..\ell\})$, we introduce a new reward assignment $\mathcal{R}_j^{un}$ for

$un(M)$ which for $\langle s, \boldsymbol{e}, \mathfrak{g}\rangle \in S_{un}$, $\alpha \in \Delta(\langle s, \boldsymbol{e}, \mathfrak{g}\rangle)$, and $\langle s', \boldsymbol{e}', \mathfrak{g}'\rangle \in post(\langle s, \boldsymbol{e}, \mathfrak{g}\rangle, \alpha)$ is defined as follows (cf. Lemmas 6.1 and 6.2).

- If $1 \leq j \leq \ell_R$—i.e., $\Phi(j) = bnd^{\mathcal{B}}(\mathcal{R}'_j, \mathcal{I}_j)$ is a bounded reward objective—then

$$\mathcal{R}^{un}_j[\langle s, \boldsymbol{e}, \mathfrak{g}\rangle, \alpha, \langle s', \boldsymbol{e}', \mathfrak{g}'\rangle] \;\coloneqq\; \mathcal{R}'_j[s, \alpha, s'] \cdot [\mathcal{I}_j \subseteq active^{\mathcal{B}}(\boldsymbol{e}')].$$

- If $\ell_R < j \leq \ell$—i.e., $\Phi(j) = bnd^{\mathcal{B}}_{\mathcal{G}}(c_j, \mathcal{J}_j)$ is a bounded reachability objective—then

$$\mathcal{R}^{un}_j[\langle s, \boldsymbol{e}, \mathfrak{g}\rangle, \alpha, \langle s', \boldsymbol{e}', \mathfrak{g}'\rangle] \;\coloneqq\; c_j \cdot [\mathcal{J}_j \not\subseteq \mathfrak{g}] \cdot [\mathcal{J}_j \subseteq \mathfrak{g}'].$$

Example 6.6 The reward assignments $\mathcal{R}^{un}_1$ and $\mathcal{R}^{un}_2$ for the two objectives $\Phi = \big\langle bnd^{\mathcal{B}}(\mathcal{R}', \{1\}),\ bnd^{\mathcal{B}}_{\mathcal{G}}(1, \{1\})\big\rangle$ from Example 6.1 are indicated in Figure 6.3.

Due to the following theorem, we can replace the reward-bounded objectives Φ for M by the total reward objectives for $un(M)$:

$$\Phi^{un} \;\coloneqq\; \big\langle tot(\mathcal{R}^{un}_1), \ldots, tot(\mathcal{R}^{un}_\ell)\big\rangle.$$

Theorem 6.3

$$Ach^{M}_{s_I}(\Phi) \;=\; Ach^{un(M)}_{s^{un}_I}(\Phi^{un}).$$

Towards a proof of Theorem 6.3, we establish a one-to-one correspondence between (i) paths and (ii) strategies of M and $un(M)$, respectively.

Consider the mapping

$$un_{\mathrm{p}} \colon Paths^{M}_{\mathrm{inf}}(s_I) \to Paths^{un(M)}_{\mathrm{inf}}(s^{un}_I),$$

where for $s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots \in Paths^{M}_{\mathrm{inf}}(s_I)$ we set

$$un_{\mathrm{p}}(s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \ldots) \;\coloneqq\; \langle s_0, \boldsymbol{e}_0, \mathfrak{g}_0\rangle \xrightarrow{\alpha_0} \langle s_1, \boldsymbol{e}_1, \mathfrak{g}_1\rangle \xrightarrow{\alpha_1} \ldots$$

with epochs $\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots$ and goal satisfactions $\mathfrak{g}_0, \mathfrak{g}_1, \ldots$ as in Lemmas 6.1 and 6.2, i.e.,

$$\langle s_0, \boldsymbol{e}_0, \mathfrak{g}_0\rangle \;=\; \big\langle s_I, \boldsymbol{e}^{\mathcal{B}}_{init}, sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}^{\mathcal{B}}_{init}, s_I)\big\rangle \;=\; s^{un}_I$$

and for all $n \in \mathbb{N}$:

$$\boldsymbol{e}_{n+1} \;=\; \boldsymbol{e}_n \ominus \boldsymbol{r}^{\mathcal{B}}_{s_n, \alpha_n, s_{n+1}} \quad \text{and} \quad \mathfrak{g}_{n+1} \;=\; \mathfrak{g}_n \cup sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}_{n+1}, s_{n+1}).$$

Example 6.7 For the path π from Examples 6.2 and 6.3, Figure 6.3 indicates the path $un_{\mathrm{p}}(\pi)$ in pink.

We use un_{p} to relate paths of the original MDP to paths of the unfolding MDP. The mapping is restricted to paths that start at the corresponding initial states, which ensures that the mapping is bijective as argued below.

Lemma 6.4 The mapping un_{p} is well-defined (i.e., paths in M are mapped to valid paths in $un(M)$) and bijective.

Proof. Well-definedness follows immediately from Definition 6.3. Bijectivity holds because

- $\pi \neq \pi'$ implies $un_{\mathrm{p}}(\pi) \neq un_{\mathrm{p}}(\pi')$ and
- for every $\pi^{un} \in Paths_{\mathrm{inf}}^{un(M)}(s_I^{un})$, there is a unique path $\pi \in Paths_{\mathrm{inf}}^{M}(s_I)$ with $\pi^{un} = un_{\mathrm{p}}(\pi)$ which is obtained by removing the epochs and goal satisfactions from π^{un}. ∎

For simplicity, we overload the notation $un_{\mathrm{p}}(\cdot)$ with *finite* paths in a straightforward way, i.e., for $\pi \in Paths_{\mathrm{inf}}^{M}(s_I)$, $n \in \mathbb{N}$:

$$un_{\mathrm{p}}(pref(\pi, n)) \;\coloneqq\; pref(un_{\mathrm{p}}(\pi), n).$$

We also map strategies of M to strategies of $un(M)$. We slightly deviate from strategies as introduced in Definition 2.16 on page 34, by considering strategies that only operate on paths that start in the initial state. To this end, let $\Sigma^{M}(s_I)$ be the set of (partially defined) strategies $\sigma\colon Paths_{\mathrm{fin}}^{M}(s_I) \to Dist(Act)$. The probability measure $\Pr_{\sigma,s_I}^{M}$ remains well-defined for strategies $\sigma \in \Sigma^{M}(s_I)$ since paths not starting in s_I always have probability 0. The set $\Sigma^{un(M)}(s_I^{un})$ is defined similarly. This allows us to relate strategies for M and $un(M)$ using the mapping $un_{\mathrm{p}}(\pi)$—which is only defined on paths starting in the initial state.

Formally, consider the mapping $un_{\Sigma}\colon \Sigma^{M}(s_I) \to \Sigma^{un(M)}(s_I^{un})$, where for $\sigma \in \Sigma^{M}(s_I)$ and $\hat{\pi}^{un} \in Paths_{\mathrm{fin}}^{un(M)}(s_I^{un})$ we set

$$un_{\Sigma}(\sigma)(\hat{\pi}^{un}) \;\coloneqq\; \sigma(\hat{\pi}) \quad \text{for unique } \hat{\pi} \in Paths_{\mathrm{fin}}^{M}(s_I) \text{ with } \pi^{un} = un_{\mathrm{p}}(\pi).$$

Intuitively, the strategy $un_{\Sigma}(\sigma)$ for $un(M)$ "simulates" the strategy σ for M by ignoring the epochs and goal satisfactions from the given path $\hat{\pi}^{un}$.

Example 6.8 Reconsider the strategy $\sigma \in \Sigma^M$ from Example 6.1. The strategy $un_\Sigma(\sigma)$ for the unfolding MDP in Figure 6.3 is the pure memoryless strategy which for $\langle s_0, \boldsymbol{e}, \mathfrak{g}\rangle \in S_{un}$ is given by

$$un_\Sigma(\sigma)(\langle s_0, \boldsymbol{e}, \mathfrak{g}\rangle) \;=\; \begin{cases} \alpha & \text{if } \boldsymbol{e}(1) \neq \oslash \\ \beta & \text{if } \boldsymbol{e}(1) = \oslash. \end{cases}$$

Lemma 6.5 un_Σ is bijective.

Proof. Clearly, $\sigma \neq \sigma'$ implies $un_\Sigma(\sigma) \neq un_\Sigma(\sigma')$. Furthermore, for every $\sigma^{un} \in \Sigma^{un(M)}(s_I^{un})$ we have $\sigma^{un} = un_\Sigma(\sigma)$ for the strategy $\sigma \in \Sigma^M(s_I)$ with

$$\forall\, \hat{\pi} \in Paths^M_{\text{fin}}(s_I)\colon \;\; \sigma(\hat{\pi}) \;\coloneqq\; \sigma^{un}(un_{\text{p}}(\hat{\pi})). \qquad \blacksquare$$

The following lemma states that the unfolding MDP preserves path probabilities.

Lemma 6.6 For all $\sigma \in \Sigma^M$ and all measurable $\Pi \subseteq Paths^M_{\text{inf}}(s_I)$, we have

$$\Pr^M_{\sigma,s_I}(\Pi) \;=\; \Pr^{un(M)}_{un_\Sigma(\sigma),s_I^{un}}\big(\{un_{\text{p}}(\pi) \mid \pi \in \Pi\}\big).$$

Proof (sketch). Since measurable sets of infinite paths can be constructed from cylinder sets of finite paths, it suffices to show

$$\Pr^M_{\sigma,s_I}(\{\hat{\pi}\}) \;=\; \Pr^{un(M)}_{un_\Sigma(\sigma),s_I^{un}}(\{un_{\text{p}}(\hat{\pi})\})$$

for all *finite* $\hat{\pi} \in Paths^M_{\text{fin}}(s_I)$. This equality holds by a simple induction over the length of $\hat{\pi}$. $\blacksquare$

Next, we show that given an individual path in the MDP M, the value of a *reward-bounded objective in* Φ coincides with the value of the corresponding *total reward objective in* Φ^{un} on the associated path in $un(M)$.

Lemma 6.7 For $\pi \in Paths^M_{\text{inf}}(s_I)$ and $j \in \{1..\ell\}$:

$$\Phi(j)(\pi) \;=\; \Phi^{un}(j)(un_{\text{p}}(\pi)).$$

Proof. Let $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \ldots$ and $un_{\text{p}}(\pi) = \langle s_0, \boldsymbol{e}_0, \mathfrak{g}_0\rangle \xrightarrow{\alpha_0} \langle s_1, \boldsymbol{e}_1, \mathfrak{g}_1\rangle \xrightarrow{\alpha_1} \ldots$. We distinguish between bounded reward objectives and bounded reachability objectives.

- If $1 \leq j \leq \ell_R$—i.e., $\Phi(j) = bnd^{\mathcal{B}}(\mathcal{R}'_j, \mathcal{I}_j)$ is a bounded reward objective—then,

using Definition 2.31 on page 57, Lemma 6.1, and the definition of $\mathcal{R}_j^{un}$ from above:

$$\begin{aligned} bnd^{\mathcal{B}}(\mathcal{R}'_j, \mathcal{I}_j)(\pi) &= \liminf_{n\to\infty} (\mathcal{R}'_j)^{\mathcal{B}(\mathcal{I}_j)}[pref(\pi, n)] \\ &= \liminf_{n\to\infty} \sum_{k=1}^{n} \Big(\mathcal{R}'_j[s_{k-1}, \alpha_{k-1}, s_k] \cdot \big[\forall\, i \in \mathcal{I}_j \colon\ \mathcal{R}_i[pref(\pi, k)] \sim_i b_i\big] \Big) \\ &= \liminf_{n\to\infty} \sum_{k=1}^{n} \Big(\mathcal{R}'_j[s_{k-1}, \alpha_{k-1}, s_k] \cdot \big[\mathcal{I}_j \subseteq active^{\mathcal{B}}(\boldsymbol{e}_k)\big] \Big) \\ &= \liminf_{n\to\infty} \sum_{k=1}^{n} \Big(\mathcal{R}_j^{un}[\langle s_{k-1}, \boldsymbol{e}_{k-1}, \mathfrak{g}_{k-1}\rangle, \alpha_{k-1}, \langle s_k, \boldsymbol{e}_k, \mathfrak{g}_k\rangle] \Big) \\ &= tot(\mathcal{R}_j^{un})(un_{\mathrm{p}}(\pi)). \end{aligned}$$

- If $\ell_R < j \leq \ell$—i.e., $\Phi(j) = bnd^{\mathcal{B}}_{\mathcal{G}}(c_j, \mathcal{J}_j)$ is a bounded reachability objective—then

$$\begin{aligned} bnd^{\mathcal{B}}_{\mathcal{G}}(c_j, \mathcal{J}_j)(\pi) &= c_j \cdot \big[\forall\, i \in \mathcal{J}_j \colon\ \pi \models \langle \mathcal{I}_i, G_i\rangle\big] \\ &= c_j \cdot \big[\exists\, n \in \mathbb{N} \colon\ \mathcal{J}_j \subseteq \mathfrak{g}_n\big] \end{aligned}$$

From Assumption 6.2 we can infer that $\mathcal{J}_j \not\subseteq \mathfrak{g}_0 = sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}^{\mathcal{B}}_{init}, s_I)$. Thus,

$$\exists\, n \in \mathbb{N} \colon\ \mathcal{J}_j \subseteq \mathfrak{g}_n \quad \Longleftrightarrow \quad \exists\, n \in \mathbb{N} \colon\ \mathcal{J}_j \not\subseteq \mathfrak{g}_n \text{ and } \mathcal{J}_j \subseteq \mathfrak{g}_{n+1}.$$

Furthermore, due to $\mathfrak{g}_0 \subseteq \mathfrak{g}_1 \subseteq \ldots$ we follow that there can be at most one $n \in \mathbb{N}$ with $\mathcal{J}_j \not\subseteq \mathfrak{g}_n$ and $\mathcal{J}_j \subseteq \mathfrak{g}_{n+1}$. Hence,

$$\begin{aligned} bnd^{\mathcal{B}}_{\mathcal{G}}(c_j, \mathcal{J}_j)(\pi) &= c_j \cdot \big[\exists\, n \in \mathbb{N} \colon\ \mathcal{J}_j \subseteq \mathfrak{g}_n\big] \\ &= \lim_{n\to\infty} \sum_{k=0}^{n} \Big(c_j \cdot \big[\mathcal{J}_j \not\subseteq \mathfrak{g}_k \text{ and } \mathcal{J}_j \subseteq \mathfrak{g}_{k+1}\big] \Big) \\ &= \lim_{n\to\infty} \sum_{k=0}^{n} \Big(\mathcal{R}_j^{un}[\langle s_k, \boldsymbol{e}_k, \mathfrak{g}_k\rangle, \alpha_k, \langle s_{k+1}, \boldsymbol{e}_{k+1}, \mathfrak{g}_{k+1}\rangle] \Big) \\ &= tot(\mathcal{R}_j^{un})(un_{\mathrm{p}}(\pi)). \end{aligned}$$

■

We now apply Lemmas 6.4 to 6.7 to show Theorem 6.3.

Proof (of Theorem 6.3). For $\sigma \in \Sigma^M(s_I)$ and $j \in \{1..\ell\}$ we have

$$\begin{aligned}
&\mathrm{Ex}^M_{\sigma,s_I}(\Phi(j)) \\
&= \int_{\pi \in Paths^M_{\inf}(s_I)} \Phi(j)(\pi)\, \mathrm{dPr}^M_{\sigma,s_I}(\pi) \\
&= \int_{\pi \in Paths^M_{\inf}(s_I)} \Phi^{un}(j)\big(un_{\mathrm{p}}(\pi)\big)\, \mathrm{dPr}^{un(M)}_{un_\Sigma(\sigma),s_I^{un}}\big(un_{\mathrm{p}}(\pi)\big) \\
&= \int_{\pi^{un} \in Paths^{un(M)}_{\inf}(s_I^{un})} \Phi^{un}(j)\big(un_{\mathrm{p}}(\pi)\big)\, \mathrm{dPr}^{un(M)}_{un_\Sigma(\sigma),s_I^{un}}\big(\pi^{un}\big) \\
&= \mathrm{Ex}^{un(M)}_{un_\Sigma(\sigma),s_I^{un}}(\Phi^{un}(j)).
\end{aligned}$$

Lifting to multiple objectives and using bijectivity of un_Σ yields

$$\exists\, \sigma \in \Sigma^M \colon \mathrm{Ex}^M_{\sigma,s_I}(\Phi) \geq \boldsymbol{p} \quad \Longleftrightarrow \quad \exists\, \sigma^{un} \in \Sigma^{un(M)} \colon \mathrm{Ex}^{un(M)}_{\sigma^{un},s_I^{un}}(\Phi^{un}) \geq \boldsymbol{p}$$

for all $\boldsymbol{p} \in \overline{\mathbb{R}}^\ell$. From this we conclude that Theorem 6.3 holds, i.e.,

$$Ach^M_{s_I}(\Phi) \;=\; Ach^{un(M)}_{s_I^{un}}(\Phi^{un}). \qquad \blacksquare$$

Due to Theorem 6.3, the set $Ach^M_{s_I}(\Phi)$ with reward-bounded objectives Φ has similar properties as in the total reward case discussed in Chapter 4. In particular, the following is a consequence of Lemma 4.6 on page 112.

Lemma 6.8 If Algorithm 4.1 on page 104 returns 'VALID' for input $un(M)$, s_I^{un}, and Φ^{un}, then $Ach^M_{s_I}(\Phi)$ is closed.

Another consequence from our observations in Chapter 4 is that for solving the WSO instance given by MDP M, initial state s_I, (reward-bounded) objectives Φ, weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$, and precision parameter $\varepsilon_{\mathtt{WSO}}$, it suffices to consider strategies $\sigma \in \Sigma^M$ such that $un_\Sigma(\sigma)$ is a *pure, memoryless* strategy for $un(M)$.

Finally, Theorem 6.3 yields an algorithm for solving the above instance for WSO:

1. build the fragment of the unfolding MDP $un^{\mathcal{B}}_{\mathcal{G}}(M)$ reachable from s_I^{un} with total reward objectives Φ^{un} as above, and
2. solve the corresponding total reward WSO instance with $un^{\mathcal{B}}_{\mathcal{G}}(M)$, s_I^{un}, and Φ^{un} instead of M, s_I, and Φ (e.g., using techniques from Chapter 4).

However, the reachable fragment of the unfolding MDP has up to $|S| \cdot \prod_{i=1}^{d}(b_i{+}2) \cdot 2^m$ states. While usually in practice the number d of reward bounds and the number m of

bounded goals remain small, the reward bound values $b_1, \ldots, b_d$ often become very large so that *explicitly building the unfolding MDP becomes infeasible.*

6.2 Sequential Epoch Analysis

In the previous section, we have reduced the analysis of reward-bounded objectives on an MDP M to the analysis of (unbounded) total reward objectives on the unfolding MDP $un^{\mathcal{B}}_{\mathcal{G}}(M)$. We now conduct this unfolding-based analysis *without explicitly creating the unfolding MDP*. The key idea is to exploit regularities in the unfolding that allow us to decompose the large unfolding MDP into small components—called *epoch MDPs*—that

- each have a similar transition structure which can be obtained without building the complete unfolding MDP and
- can be analyzed sequentially in a certain order.

The approach is to analyze the epoch MDPs one-by-one—using results from previously analyzed epochs in a *dynamic programming* manner. This idea has also been considered for objectives with just a single reward bound in [BDDK+14; HH16] and takes strong inspiration from *topological value iteration* [CBGK08; DMWG11] which we also discussed in Section 4.4.4.

Set-up | We consider the unfolding MDP $un^{\mathcal{B}}_{\mathcal{G}}(M) = un(M) = \langle S_{un}, Act, \Delta_{un}, \mathbf{P}_{un} \rangle$ of MDP $M = \langle S, Act, \Delta, \mathbf{P} \rangle$, $d > 0$ reward bounds $\mathcal{B} = \langle \wr \mathcal{R}_1 \sim_1 b_1 \wr, \ldots, \wr \mathcal{R}_d \sim_d b_d \wr \rangle$, and $m \geq 0$ bounded goals $\mathcal{G} = \langle \langle \mathcal{I}_1, G_1 \rangle, \ldots, \langle \mathcal{I}_m, G_m \rangle \rangle$ as in Definition 6.3. Furthermore, we consider the initial state $s^{un}_I \in S_{un}$ as well as $\ell > 0$ reward assignments $\mathcal{R}^{un}_1, \ldots, \mathcal{R}^{un}_\ell$ for $un(M)$, which give rise to ℓ total reward objectives $\Phi^{un} := \langle tot(\mathcal{R}^{un}_1), \ldots, tot(\mathcal{R}^{un}_\ell) \rangle$. For our approach, there is no need to explicitly create $un(M)$ and $\mathcal{R}^{un}_1, \ldots, \mathcal{R}^{un}_\ell$. An implicit representation using M, $\mathcal{B}$, $\mathcal{G}$, and the original reward-bounded objectives Φ suffices.

As before, our goal is to answer the resulting WSO instance for given weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$, and precision parameter ε_{WSO}.

6.2.1 Epoch MDPs

Let us fix a reward epoch $\boldsymbol{e} \in \mathbb{E}_d$. Intuitively, the epoch MDP for $\boldsymbol{e}$ arises when restricting the states S_{un} of the unfolding MDP to those with epoch $\boldsymbol{e}$, i.e., triples $\langle s, \boldsymbol{e}, \mathfrak{g} \rangle$ with $s \in S$, $\mathfrak{g} \in \mathbb{G}_m$, and fixed epoch $\boldsymbol{e}$. Transitions leading to different epochs $\boldsymbol{e}' \neq \boldsymbol{e}$ are redirected to a dedicated sink state $s_\perp$.

epoch MDP

Definition 6.4 (Epoch MDP) The *epoch MDP* for MDP M, epoch $\boldsymbol{e} \in \mathbb{E}_d$, reward bounds $\mathcal{B}$ and bounded goals $\mathcal{G}$ is given by the MDP

$$ep^{\mathcal{B}}_{\mathcal{G}}(M, \boldsymbol{e}) = \big\langle S_{\boldsymbol{e}}, Act \uplus \{\bot\}, \Delta_{\boldsymbol{e}}, \mathbf{P}_{\boldsymbol{e}} \big\rangle,$$

where—using the unfolding MDP $un^{\mathcal{B}}_{\mathcal{G}}(M) = \big\langle S_{un}, Act, \Delta_{un}, \mathbf{P}_{un} \big\rangle$ as in Definition 6.3—we have

- $S_{\boldsymbol{e}} \coloneqq \big(S \times \{\boldsymbol{e}\} \times \mathbb{G}_m\big) \uplus \{s_\bot\} \subseteq S_{un} \uplus \{s_\bot\}$,
- for $\hat{s} \in S_{\boldsymbol{e}}$: $\Delta_{\boldsymbol{e}}(\hat{s}) \coloneqq \begin{cases} \Delta_{un}(\hat{s}) & \text{if } \hat{s} \neq s_\bot \\ \{\bot\} & \text{if } \hat{s} = s_\bot, \end{cases}$
- and for $\hat{s}, \hat{s}' \in S_{\boldsymbol{e}} \setminus \{s_\bot\}$ and $\alpha \in \Delta_{\boldsymbol{e}}(\hat{s})$:
  - $\mathbf{P}_{\boldsymbol{e}}\big(\hat{s}, \alpha, \hat{s}'\big) \coloneqq \mathbf{P}_{un}\big(\hat{s}, \alpha, \hat{s}'\big)$,
  - $\mathbf{P}_{\boldsymbol{e}}\big(\hat{s}, \alpha, s_\bot\big) \coloneqq \sum_{s'_{un} \in (S_{un} \setminus S_{\boldsymbol{e}})} \mathbf{P}_{un}\big(\hat{s}, \alpha, s'_{un}\big)$,
  - $\mathbf{P}_{\boldsymbol{e}}\big(s_\bot, \bot, s_\bot\big) \coloneqq 1$, and
  - $\mathbf{P}_{\boldsymbol{e}}\big(s_\bot, \bot, \hat{s}'\big) \coloneqq 0$.

When $\mathcal{B}$ and $\mathcal{G}$ are clear from the context, we usually omit them from the notation and just write $ep(M, \boldsymbol{e})$. We slightly abuse notations by applying memoryless strategies $\sigma \in \Sigma^{un(M)}$ for the unfolding MDP also to the epoch MDPs $ep(M, \boldsymbol{e})$ for any $\boldsymbol{e} \in \mathbb{E}_d$. This is well-defined since

- the choice at $s_\bot \in S_{\boldsymbol{e}}$ is unique as $\Delta(s_\bot) = \{\bot\}$ is a singleton set, and
- for any other state $\langle s, \boldsymbol{e}, \mathfrak{g} \rangle \in S_{\boldsymbol{e}} \setminus \{s_\bot\}$ we also have $\langle s, \boldsymbol{e}, \mathfrak{g} \rangle \in S_{un}$ and thus $\sigma(\langle s, \boldsymbol{e}, \mathfrak{g} \rangle)$ is defined.

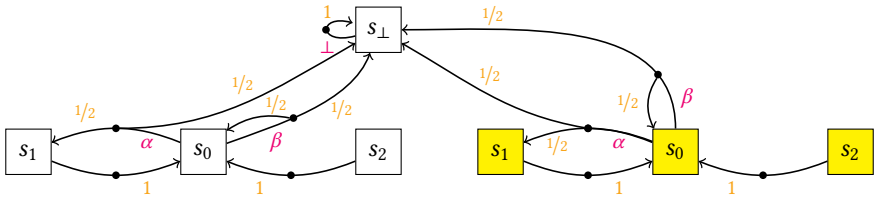
Example 6.9 Figure 6.4 shows the epoch MDPs for M, $\mathcal{B}$, and $\mathcal{G}$ as in Figure 6.1 and Example 6.1 for all possible epochs $\boldsymbol{e} \in \mathbb{E}_d = \mathbb{E}_2$.

Given an epoch $\boldsymbol{e} \in \mathbb{E}_d$ and a reward assignment $\mathcal{R}^{un}$ for $un(M)$, we now construct a corresponding reward assignment for the epoch MDP $ep(M, \boldsymbol{e})$. We also incorporate a function $f \colon S_{un} \to \mathbb{R}$ for assigning rewards upon leaving the current epoch.

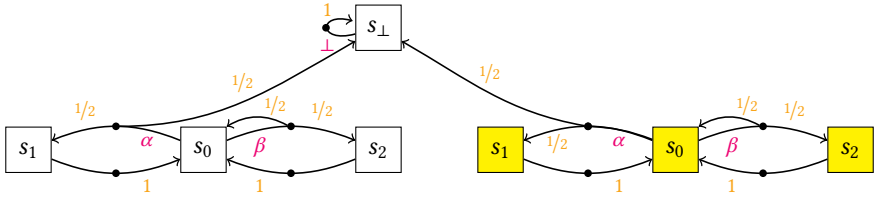
induced reward assignment

Definition 6.5 (Induced Reward Assignment) The reward assignment for $ep(M, \boldsymbol{e})$ induced by $\mathcal{R}^{un}$ and f as above is denoted by $\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}}$, and for $\hat{s} \in S_{\boldsymbol{e}} \setminus \{s_\bot\}$ and $\alpha \in \Delta_{\boldsymbol{e}}(\hat{s})$ given by

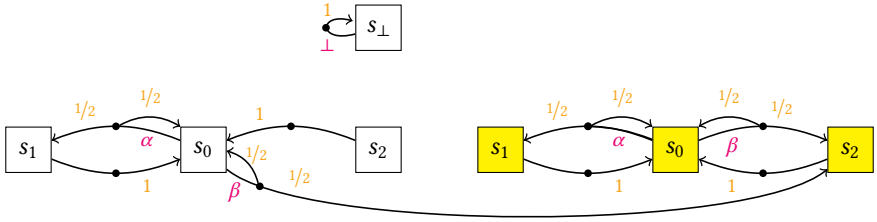
- $\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}}[\hat{s}, \alpha, \hat{s}'] \coloneqq \mathcal{R}^{un}[\hat{s}, \alpha, \hat{s}']$ for all $\hat{s}' \in post(\hat{s}, \alpha) \setminus \{s_\bot\}$,



(a) $ep(M, \langle n_1, n_2 \rangle) = ep(M, \langle n_1, \oslash \rangle)$



(b) $ep(M, \langle \oslash, n_2 \rangle)$



(c) $ep(M, \langle \oslash, \oslash \rangle)$

Figure 6.4: Different epoch MDPs for epochs $\langle n_1, n_2 \rangle$, $\langle n_1, \oslash \rangle$, $\langle \oslash, n_2 \rangle$, $\langle \oslash, \oslash \rangle \in \mathbb{E}_d$ with $n_1, n_2 \in \mathbb{N}$ (cf. Example 6.9).

- $\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}}[\hat{s}, \alpha, s_\perp] \;\coloneqq\; \sum_{s'_{un} \in (S_{un} \setminus S_{\boldsymbol{e}})} \left(\frac{\mathbf{P}_{un}(\hat{s}, \alpha, s'_{un})}{\mathbf{P}_{\boldsymbol{e}}(\hat{s}, \alpha, s_\perp)} \cdot \left(\mathcal{R}^{un}[\hat{s}, \alpha, s'_{un}] + f(s'_{un}) \right) \right)$, if $s_\perp \in \mathit{post}(\hat{s}, \alpha)$, and
- $\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}}[s_\perp, \perp, s_\perp] \;\coloneqq\; 0$.

Intuitively, for transitions that stay within the epoch $\boldsymbol{e}$, the reward assignment $\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}}$ mimicks the original rewards $\mathcal{R}^{un}$ from the unfolding MPD $un(M)$. Transitions that exit the epoch are—in the epoch MDP $ep(M, \boldsymbol{e})$—redirected to the dedicated sink state $s_\perp$. For those transitions, we consider the rewards that are assigned to the corresponding transitions by $\mathcal{R}^{un}$ plus the values given by the function f. Since a *single* transition in $ep(M, \boldsymbol{e})$ from a state $\hat{s} \in S_{\boldsymbol{e}} \setminus \{s_\perp\}$ to $s_\perp$ can represent *multiple* transitions in $un(M)$, we scale the assigned rewards with the probability that the rewards are collected in $un(M)$. The term $\frac{\mathbf{P}_{un}(\hat{s}, \alpha, s'_{un})}{\mathbf{P}_{\boldsymbol{e}}(\hat{s}, \alpha, s_\perp)}$ in Definition 6.5 yields the conditional probability to move from $\hat{s}$ to s'_{un}, given that we exit the epoch at $\hat{s}$ (via action α).

We now establish a connection between expected total rewards of the individual epoch MDPs and expected total rewards of the unfolding MDP. We fix a reward assignment $\mathcal{R}^{un}$ for $un(M)$ and a *memoryless* strategy $\sigma \in \Sigma^{un(M)}$. Let

$$S^{\mathrm{fin}}_{\sigma} \;\coloneqq\; \left\{ s_{un} \in S_{un} \;\middle|\; \mathrm{Ex}^{un(M)}_{\sigma, s_{un}}(\mathit{tot}(\mathcal{R}^{un})) \notin \{-\infty, +\infty\} \right\}.$$

Lemma 6.9 If $f \colon S_{un} \to \mathbb{R}$ satisfies $f(s_{un}) = \mathrm{Ex}^{un(M)}_{\sigma, s_{un}}(\mathit{tot}(\mathcal{R}^{un}))$ for all $s_{un} \in S^{\mathrm{fin}}_{\sigma}$, then

$$\mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\sigma, \hat{s}}(\mathit{tot}(\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}})) \;=\; \mathrm{Ex}^{un(M)}_{\sigma, \hat{s}}(\mathit{tot}(\mathcal{R}^{un}))$$

holds for all $\hat{s} \in (S_{\boldsymbol{e}} \cap S^{\mathrm{fin}}_{\sigma}) \subseteq S_{un}$.

Proof. Observe that SCC models and their reward assignments used for topological value iteration (see Definition 4.5 on page 127) are defined similarly to epoch MDPs. Lemma 6.9 can be shown similarly to Theorem 4.18 on page 128. ∎

Lemma 6.9 also applies for maximizing expected total rewards. Let

$$S^{\mathrm{fin}}_{\max} \;\coloneqq\; \left\{ s_{un} \in S_{un} \;\middle|\; \mathrm{Ex}^{un(M)}_{\max, s_{un}}(\mathit{tot}(\mathcal{R}^{un})) \notin \{-\infty, +\infty\} \right\}.$$

Lemma 6.10 If $f \colon S_{un} \to \mathbb{R}$ satisfies $f(s_{un}) = \mathrm{Ex}^{un(M)}_{\max, s_{un}}(\mathit{tot}(\mathcal{R}^{un}))$ for all $s_{un} \in S^{\mathrm{fin}}_{\max}$, then

$$\mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\max, \hat{s}}(\mathit{tot}(\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}})) \;=\; \mathrm{Ex}^{un(M)}_{\max, \hat{s}}(\mathit{tot}(\mathcal{R}^{un}))$$

holds for all $\hat{s} \in (S_{\boldsymbol{e}} \cap S^{\mathrm{fin}}_{\max}) \subseteq S_{un}$.

Proof. Let $\sigma_{\max} \in \Sigma^{un(M)}$ be a memoryless strategy that maximizes the expected total rewards in $un(M)$, i.e.,

$$\forall\, s_{un} \in S_{un}\colon\ \mathrm{Ex}^{un(M)}_{\max,s_{un}}(tot(\mathcal{R}^{un})) \;=\; \mathrm{Ex}^{un(M)}_{\sigma_{\max},s_{un}}(tot(\mathcal{R}^{un})).$$

Applying Lemma 6.9 to $\sigma_{\max}$ and $\hat{s} \in (S_{\boldsymbol{e}} \cap S^{\mathrm{fin}}_{\max})$ yields

$$\begin{aligned}\mathrm{Ex}^{un(M)}_{\max,\hat{s}}(tot(\mathcal{R}^{un})) \;=\; \mathrm{Ex}^{un(M)}_{\sigma_{\max},\hat{s}}(tot(\mathcal{R}^{un})) \;&=\; \mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\sigma_{\max},\hat{s}}(tot(\langle \mathcal{R}^{un}, f\rangle_{\boldsymbol{e}})) \\ &\leq\; \mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\max,\hat{s}}(tot(\langle \mathcal{R}^{un}, f\rangle_{\boldsymbol{e}})).\end{aligned}$$

Similarly, let $\hat{\sigma}_{\max} \in \Sigma^{ep(M,\boldsymbol{e})}$ be a memoryless strategy that maximizes the expected total rewards in $ep(M,\boldsymbol{e})$, i.e.,

$$\mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\max,\hat{s}}(tot(\langle \mathcal{R}^{un}, f\rangle_{\boldsymbol{e}})) \;=\; \mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\hat{\sigma}_{\max},\hat{s}}(tot(\langle \mathcal{R}^{un}, f\rangle_{\boldsymbol{e}})).$$

We lift $\hat{\sigma}_{\max}$ to the unfolding MDP by considering memoryless $\sigma \in \Sigma^{un(M)}$ with

$$\forall\, s_{un} \in S_{un}\colon\ \sigma(s_{un}) = \begin{cases}\hat{\sigma}_{\max}(s_{un}) & \text{if } s_{un} \in S_{\boldsymbol{e}} \\ \sigma_{\max}(s_{un}) & \text{otherwise.}\end{cases}$$

Applying Lemma 6.9 with strategy σ yields

$$\begin{aligned}\mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\max,\hat{s}}(tot(\langle \mathcal{R}^{un}, f\rangle_{\boldsymbol{e}})) \;&=\; \mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\hat{\sigma}_{\max},\hat{s}}(tot(\langle \mathcal{R}^{un}, f\rangle_{\boldsymbol{e}})) \\ &=\; \mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\sigma,\hat{s}}(tot(\langle \mathcal{R}^{un}, f\rangle_{\boldsymbol{e}})) \\ &=\; \mathrm{Ex}^{un(M)}_{\sigma,\hat{s}}(tot(\mathcal{R}^{un})) \\ &\leq\; \mathrm{Ex}^{un(M)}_{\max,\hat{s}}(tot(\mathcal{R}^{un})).\end{aligned}$$

■

6.2.2 Epoch Dependencies

Lemmas 6.9 and 6.10 establish a way to reason about expected total rewards for the unfolding MDP by only considering expected total rewards for (potentially much smaller) epoch MDPs. However, appropriate values for the function $f\colon S_{un} \to \mathbb{R}$ need to be computed *before* an epoch $\boldsymbol{e}$ can be analyzed. Fortunately, the values $f(s_{un})$ are only relevant for unfolding states s_{un} that are direct successors of states $\hat{s} \in S_{\boldsymbol{e}}$.

Our approach is to solve the given WSO instance for the unfolding MDP using a decomposition into the different epoch MDPs. We exploit that states $\langle s, \boldsymbol{e}, \mathfrak{g}\rangle \in S_{un}$ of the unfolding can only reach states $\langle s', \boldsymbol{e}', \mathfrak{g}'\rangle \in S_{un}$, where the entries of the epoch $\boldsymbol{e}'$ becomes either smaller or $\oslash$—avoiding circular dependencies between epoch MDPs.

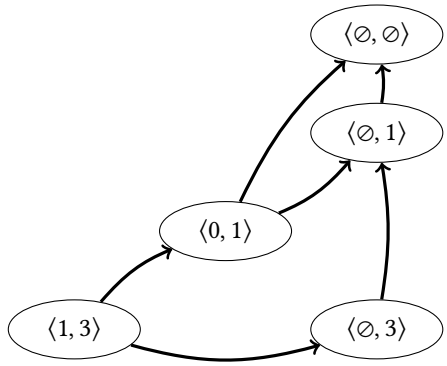


Figure 6.5: Reachable fragment of example epoch graph (cf. Example 6.10).

We now formalize those dependencies.

epoch graph

Definition 6.6 (Epoch Graph) The *epoch graph* for unfolding MDP $un(M)$ is the graph $(\mathbb{E}_d, \hookrightarrow)$ with vertices $\mathbb{E}_d$ and edges $\hookrightarrow \subseteq \mathbb{E}_d \times \mathbb{E}_d$ such that for $\boldsymbol{e}, \boldsymbol{e}' \in \mathbb{E}_d$: $\boldsymbol{e} \hookrightarrow \boldsymbol{e}'$ iff $\boldsymbol{e} \neq \boldsymbol{e}'$ and

$$\exists\, s, s' \in S,\ \mathfrak{g}, \mathfrak{g}' \in \mathbb{G}_m,\ \alpha \in \Delta_{un}(\langle s, \boldsymbol{e}, \mathfrak{g}\rangle)\colon\ \mathbf{P}_{un}\big(\langle s, \boldsymbol{e}, \mathfrak{g}\rangle, \alpha, \langle s', \boldsymbol{e}', \mathfrak{g}'\rangle\big) > 0.$$

Intuitively, there is an edge from one epoch $\boldsymbol{e}$ to another epoch $\boldsymbol{e}'$ in the epoch graph iff there is a transition in the unfolding MDP from a state with epoch $\boldsymbol{e}$ to a state with different epoch $\boldsymbol{e}'$.

Example 6.10 Figure 6.5 shows the fragment of the epoch graph for the unfolding MDP in Figure 6.3 reachable from $\boldsymbol{e}^{\mathcal{B}}_{init} = \langle 1, 3\rangle$.

Consider two states $s_{un} = \langle s, \boldsymbol{e}, \mathfrak{g}\rangle$ and $s'_{un} = \langle s', \boldsymbol{e}', \mathfrak{g}\rangle$ of the unfolding MDP such that there is a path $\hat{\pi} \in Paths^{un(M)}_{\text{fin}}(s_{un})$ from s_{un} to $last(\hat{\pi}) = s'_{un}$. Thus, the total reward accumulated from state s_{un} depends on the total reward accumulated from state s'_{un}. Observe that in this case, there is also a path $\boldsymbol{e}_1 \hookrightarrow \ldots \hookrightarrow \boldsymbol{e}_n$ from epoch $\boldsymbol{e} = \boldsymbol{e}_1$ to epoch $\boldsymbol{e}' = \boldsymbol{e}_n$ in the epoch graph. This path intuitively reflects that we have to analyze the epoch $\boldsymbol{e}'$ *before* we can deal with epoch $\boldsymbol{e}$. On the other hand, if there were no path from $\boldsymbol{e}$ to $\boldsymbol{e}'$ in the epoch graph, the rewards accumulated at states with epoch $\boldsymbol{e}'$ would not affect the total rewards at epoch $\boldsymbol{e}$. We say that epoch $\boldsymbol{e}$ *depends* on epoch $\boldsymbol{e}'$ if there is a path from $\boldsymbol{e}$ to $\boldsymbol{e}'$ in the epoch graph.

Lemma 6.11 The epoch graph for an unfolding MDP is acyclic.

Proof. Recall that the reward assignments $\mathcal{R}_1, \ldots, \mathcal{R}_d$ for the reward bounds are required to be non-negative (cf. Assumption 6.1). Therefore, if $\boldsymbol{e} \hookrightarrow \boldsymbol{e}'$ for $\boldsymbol{e}, \boldsymbol{e}' \in \mathbb{E}_d$, then

$$\forall\, i \in \{1..d\} \colon\ \boldsymbol{e}'(i) = \oslash \text{ or } \boldsymbol{e}(i) \geq \boldsymbol{e}'(i).$$

We also have $\boldsymbol{e} \neq \boldsymbol{e}'$. Thus, a path $\boldsymbol{e}_1 \hookrightarrow \ldots \hookrightarrow \boldsymbol{e}_n$ of the epoch graph cannot visit the same vertex twice. ■

Due to the acyclicity of the epoch graph, we can analyze epochs in a reverse topological order which—when analyzing $\boldsymbol{e} \in \mathbb{E}_d$—ensures that all epochs $\boldsymbol{e}'$ on which $\boldsymbol{e}$ depends have been analyzed before.

Definition 6.7 (Proper Epoch Sequence) A *proper epoch sequence* is a sequence of pairwise distinct epochs $\mathfrak{E} = \boldsymbol{e}_1 \ldots \boldsymbol{e}_n$ such that

proper epoch sequence

$$\forall\, i \in \{1..n\} \colon\ \forall\, \boldsymbol{e} \in \mathbb{E}_d \colon\ \boldsymbol{e}_i \hookrightarrow \boldsymbol{e} \text{ implies } \exists\, j < i \colon\ \boldsymbol{e} = \boldsymbol{e}_j.$$

For an epoch $\boldsymbol{e}_i$ occurring in a proper epoch sequence $\mathfrak{E} = \boldsymbol{e}_1 \ldots \boldsymbol{e}_n$, we have that all epochs on which $\boldsymbol{e}_i$ depends (i.e., which are reachable from $\boldsymbol{e}_i$ in the epoch graph) have to occur in $\mathfrak{E}$ at a position *before* $\boldsymbol{e}_i$. For a given epoch $\boldsymbol{e}$, we can compute a proper epoch sequence $\mathfrak{E}$ ending with epoch $\boldsymbol{e}$ by first building the fragment of the epoch graph reachable from $\boldsymbol{e}$ and then computing a reverse topological ordering for that graph.

Example 6.11 An example proper epoch sequence with respect to the epoch graph from Figure 6.5 is given by

$$\mathfrak{E} \;=\; \langle \oslash, \oslash \rangle\ \langle \oslash, 1 \rangle\ \langle \oslash, 3 \rangle\ \langle 0, 1 \rangle\ \langle 1, 3 \rangle.$$

6.2.3 Unfolding Implicitly

We now present the *sequential epoch analysis approach*. We first consider the case where $\varepsilon_{\texttt{WSO}} = 0$, i.e., all computations have to be exact. This case is simpler as there are no approximation errors that could potentially propagate through the epochs. Furthermore, we assume—for now—that all considered expected total rewards are finite, i.e.,

sequential epoch analysis

$$\forall\, \sigma \in \Sigma^{un(M)},\ \hat{s} \in S_{un},\ j \in \{1..\ell\} \colon\ \mathrm{Ex}^{un(M)}_{\sigma,\hat{s}}(tot(\mathcal{R}^{un}_j)) \notin \{+\infty, -\infty\}.$$

In Section 6.2.4, we extend our algorithm to deal with both: propagation of approximation errors and non-finite expected values.

Algorithm 6.1 outlines the dynamic programming-based procedure. For each epoch $\boldsymbol{e}$ in the proper epoch sequence $\mathfrak{E}$ computed in Line 2, we create an in-

solve WSO

1 **function** solveWso($M, s_I, \Phi, \boldsymbol{w}, \varepsilon_{\texttt{WSO}}$)
  // Φ consists of ℓ reward-bounded objectives
  // Assume $\varepsilon_{\texttt{WSO}} = 0$ and $\forall\, \sigma \in \Sigma^{un(M)} \colon\ \forall\, \hat{s} \in S_{un} \colon\ \mathrm{Ex}^{un(M)}_{\sigma,\hat{s}}(\Phi^{un}) \in \mathbb{R}^{\ell}$
2 Compute proper epoch sequence $\mathfrak{E} = \boldsymbol{e}_1 \dots \boldsymbol{e}_n$ ending with $\boldsymbol{e}_n = \boldsymbol{e}^{\mathcal{B}}_{init}$
3 Arbitrarily initialize functions $f_1, \dots, f_\ell \colon S_{un} \to \mathbb{R}$ *// Values are set in Line 8*
4 **for** $\boldsymbol{e} = \boldsymbol{e}_1, \dots, \boldsymbol{e}_n$ **do**
5 Build epoch MDP $ep(M, \boldsymbol{e}) = \left\langle S_{\boldsymbol{e}}, Act \uplus \{\bot\}, \Delta_{\boldsymbol{e}}, \mathbf{P}_{\boldsymbol{e}} \right\rangle$
6 $\Phi^{\boldsymbol{e}} \leftarrow \left\langle tot(\langle \mathcal{R}^{un}_1, f_1 \rangle_{\boldsymbol{e}}), \dots, tot(\langle \mathcal{R}^{un}_\ell, f_\ell \rangle_{\boldsymbol{e}}) \right\rangle$
7 $\left\langle \langle v_{\hat{s}}, \boldsymbol{p}_{\hat{s}} \rangle \right\rangle_{\hat{s} \in S_{\boldsymbol{e}}} \leftarrow \texttt{solveWso}^{\forall}(ep(M, \boldsymbol{e}), \Phi^{\boldsymbol{e}}, \boldsymbol{w}, 0)$ *// As in Chapter 4*
8 **for** $\hat{s} \in S_{\boldsymbol{e}} \cap S_{un}$, $j \in \{1..\ell\}$ **do** $f_j(\hat{s}) \leftarrow \boldsymbol{p}_{\hat{s}}(j)$
9 $\boldsymbol{p} \leftarrow \left\langle f_j(s^{un}_I) \right\rangle_{j \in \{1..\ell\}}$; $v \leftarrow \boldsymbol{p} \cdot \boldsymbol{w}$
10 **return** $\langle v, \boldsymbol{p} \rangle$

Algorithm 6.1: Solving bounded reward WSO instances (simplified)

stance for WSO considering the epoch MDP $ep(M, \boldsymbol{e})$ and total reward objectives $\Phi^{\boldsymbol{e}} = \left\langle tot(\langle \mathcal{R}^{un}_1, f_1 \rangle_{\boldsymbol{e}}), \dots, tot(\langle \mathcal{R}^{un}_\ell, f_\ell \rangle_{\boldsymbol{e}}) \right\rangle$ as outlined in Section 6.2.1 (Lines 4 to 6). When creating the reward assignments $\langle \mathcal{R}^{un}_j, f_j \rangle_{\boldsymbol{e}}$ for $j \in \{1..\ell\}$, we only access the values $f_j(s_{un})$ for unfolding MDP states $s_{un} = \langle s, \boldsymbol{e}', \mathfrak{g} \rangle \in S_{un}$ such that $\boldsymbol{e}$ depends on the epoch $\boldsymbol{e}'$ of s_{un} i.e., $\boldsymbol{e}'$ occurs before $\boldsymbol{e}$ in the proper epoch sequence $\mathfrak{E}$ and thus $\boldsymbol{e}'$ has been processed before the current epoch $\boldsymbol{e}$. We solve the created WSO instance in Line 7. Here, we use a slightly modified version $\texttt{solveWso}^{\forall}$ of the function solveWso from Algorithm 4.2 on page 108 that returns a solution $\langle v_{\hat{s}}, \boldsymbol{p}_{\hat{s}} \rangle$ for *every* possible initial state $\hat{s} \in S_{\boldsymbol{e}}$. This modification typically only has a minor effect on runtime since common algorithms for computing expected total rewards—including (sound) value iteration, strategy iteration, and linear programming—produce a result for every state as a by-product. Algorithm 4.2 is always applicable since—by assumption—infinite expected values do not occur. In Line 8 we set the values $f_j(\hat{s})$ for unfolding MDP states from the current epoch $\hat{s} \in S_{\boldsymbol{e}} \cap S_{un}$ so that the values are available when analyzing future epochs that depend on $\boldsymbol{e}$. Finally, we create an answer $\langle v, \boldsymbol{p} \rangle$ for the reward-bounded WSO instance in Line 9 by considering the values stored in functions $f_1, \dots, f_\ell$ for the initial state s^{un}_I of the unfolding MDP.

Theorem 6.12 The value v and the point $\boldsymbol{p}$ returned by Algorithm 6.1 yield a solution to the provided reward-bounded WSO instance.

Proof. We argue using the respective WSO instance on the unfolding MDP $un(M)$ with objectives $\Phi^{un} = \langle tot(\mathcal{R}_1^{un}), \ldots, tot(\mathcal{R}_\ell^{un}) \rangle$, which is valid due to Theorem 6.3. Recall our simplifying assumption, i.e., $\varepsilon_{\texttt{WSO}} = 0$ and all expected values are finite. To show that the returned pair $\langle v, \boldsymbol{p} \rangle$—where $v = \boldsymbol{w} \cdot \boldsymbol{p}$—is a solution to WSO, it suffices to show that

$$\boldsymbol{p} = \mathrm{Ex}^{un(M)}_{\sigma, s_I^{un}}(\Phi^{un}) \quad \text{for} \quad \sigma \in \operatorname*{arg\,max}_{\sigma' \in \Sigma^{un(M)}} \boldsymbol{w} \cdot \mathrm{Ex}^{un(M)}_{\sigma', s_I^{un}}(\Phi^{un}). \tag{6.2}$$

When solving the total-reward WSO instances in Line 7 of Algorithm 6.1 using Algorithm 4.2, memoryless strategies $\sigma_1, \ldots, \sigma_n$ are constructed for each processed epoch MDP $ep(M, \boldsymbol{e}_1), \ldots, ep(M, \boldsymbol{e}_n)$. Let $\sigma \in \Sigma^{un(M)}$ be the combined memoryless strategy for the unfolding MDP with $\sigma(\langle s, \boldsymbol{e}_i, \mathfrak{g} \rangle) \coloneqq \sigma_i(\langle s, \boldsymbol{e}_i, \mathfrak{g} \rangle)$ for each $i \in \{1..n\}$ and $\langle s, \boldsymbol{e}_i, \mathfrak{g} \rangle \in S_{un}$. We show that upon termination of the algorithm for any $\hat{s} \in S_{un}$ we have

1. $\forall\, j \in \{1..\ell\}\colon\ f_j(\hat{s}) = \mathrm{Ex}^{un(M)}_{\sigma, \hat{s}}(tot(\mathcal{R}_j^{un}))$ and
2. $\boldsymbol{w} \cdot \langle f_1(\hat{s}), \ldots, f_\ell(\hat{s}) \rangle = \max_{\sigma' \in \Sigma^{un(M)}} \boldsymbol{w} \cdot \mathrm{Ex}^{un(M)}_{\sigma', \hat{s}}(\Phi^{un})$.

Equation (6.2) follows by noting that $\boldsymbol{p} = \left\langle f_j(s_I^{un}) \right\rangle_{j \in \{1..\ell\}}$.
Let us fix an epoch $\boldsymbol{e}$ occurring in the proper epoch sequence $\mathfrak{E}$. Without loss of generality we assume that Items 1 and 2 already hold for all states $\hat{s}' = \langle s', \boldsymbol{e}', \mathfrak{g}' \rangle$ such that $\boldsymbol{e}$ depends on $\boldsymbol{e}'$, i.e., epoch $\boldsymbol{e}'$ occurs before epoch $\boldsymbol{e}$ in $\mathfrak{E}$. In particular, this assumption is valid for the first epoch $\boldsymbol{e}_1$ in $\mathfrak{E}$ since $\boldsymbol{e}_1$ does not depend on any other epoch.

Item 1 | For $\hat{s} = \langle s, \boldsymbol{e}, \mathfrak{g} \rangle \in S_{un}$, the point $\boldsymbol{p}_{\hat{s}}$ computed in Line 7 of Algorithm 6.1 satisfies $\boldsymbol{p}_{\hat{s}} = \mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\sigma, \hat{s}}(\Phi^{\boldsymbol{e}})$ due to the correctness of Algorithm 4.2. Thus, for $j \in \{1..\ell\}$ we get

$$f_j(\hat{s}) = \boldsymbol{p}_{\hat{s}}(j) = \mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\sigma, \hat{s}}(\langle \mathcal{R}_j^{un}, f_j \rangle_{\boldsymbol{e}}) = \mathrm{Ex}^{un(M)}_{\sigma, \hat{s}}(tot(\mathcal{R}_j^{un}))$$

The last equality above is due to Lemma 6.9 and the assumption that Item 1 already holds for epochs $\boldsymbol{e}'$ on which $\boldsymbol{e}$ depends.

Item 2 | It remains to show that $\sigma \in \arg\max_{\sigma' \in \Sigma^{un(M)}} \boldsymbol{w} \cdot \mathrm{Ex}^{un(M)}_{\sigma', \hat{s}}(\Phi^{un})$, i.e., the strategy σ for the unfolding MDP maximizes the weighted sum over the total reward objectives when considering an initial state with epoch $\boldsymbol{e}$.
Consider $f\colon S_{un} \to \mathbb{R}$ with

$$\forall\, \hat{s}' \in S_{un}\colon\ f(\hat{s}') \coloneqq \boldsymbol{w} \cdot \langle f_1(\hat{s}'), \ldots, f_\ell(\hat{s}') \rangle.$$

If $\hat{s}' = \langle s', \boldsymbol{e}', \mathfrak{g}' \rangle$ for some epoch $\boldsymbol{e}'$ on which $\boldsymbol{e}$ depends, our assumption that Item 2 holds for $\hat{s}'$ yields—using $\mathcal{R}^{un} := \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \mathcal{R}_j^{un}$ and Lemma 4.4 on page 106:

$$f(\hat{s}') \;=\; \max_{\sigma' \in \Sigma^{un(M)}} \boldsymbol{w} \cdot \mathrm{Ex}^{un(M)}_{\sigma',\hat{s}'}(\Phi^{un}) \;=\; \mathrm{Ex}^{un(M)}_{\max,\hat{s}'}(tot(\mathcal{R}^{un})). \tag{6.3}$$

The following equalities conclude the proof by showing that σ indeed maximizes the weighted sum over the total reward objectives.

$$\begin{aligned}
&\boldsymbol{w} \cdot \mathrm{Ex}^{un(M)}_{\sigma,\hat{s}}(\Phi^{un}) \\
&= \boldsymbol{w} \cdot \mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\sigma,\hat{s}}(\Phi^{\boldsymbol{e}}) && \text{(Item 1 and Lemma 6.9)} \\
&= \max_{\sigma' \in \Sigma^{ep(M,\boldsymbol{e})}} \boldsymbol{w} \cdot \mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\sigma',\hat{s}}(\Phi^{\boldsymbol{e}}) && \text{(Correctness of Algorithm 4.2)} \\
&= \mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\max,\hat{s}}(tot(\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}})) && \text{(Lemma 4.4)} \\
&= \mathrm{Ex}^{un(M)}_{\max,\hat{s}}(tot(\mathcal{R}^{un})) && \text{(Equation (6.3) and Lemma 6.10)} \\
&= \max_{\sigma' \in \Sigma^{un(M)}} \boldsymbol{w} \cdot \mathrm{Ex}^{un(M)}_{\sigma',\hat{s}}(\Phi^{un}). && \text{(Lemma 4.4)} \qquad \blacksquare
\end{aligned}$$

6.2.4 Allowing Non-finite and Approximative Expected Values

We extend Algorithm 6.1 to (i) deal with objectives that potentially yield infinite expected values and to (ii) allow approximation errors when analyzing the individual epoch MDPs. The former allows for a broader class of input models whereas the latter enables the use of practically efficient methods for expected total rewards such as sound value iteration (Chapter 4). The new, extended procedure is outlined in Algorithm 6.2.

Infinite Expected Values

We drop the assumption that *all* occurring expected rewards at any epoch are finite. Instead, we require

1. $\mathrm{Ex}^{M}_{\max,s_I}\big(bnd((\mathcal{R}'_j)^+, \mathcal{I}_j)\big) < +\infty$ for all bounded reward objectives $bnd(\mathcal{R}'_j, \mathcal{I}_j)$, $j \in \{1..\ell_R\}$, and
2. $Ach^M(\Phi) \cap \mathbb{R}^\ell \neq \emptyset$.

The reward assignments $(\mathcal{R}'_j)^+$ considered in Item 1 arise from $\mathcal{R}'_j$ by setting all negative rewards to zero (cf. Section 2.2.5). Item 1 implies that the bounded reward objectives are convergent (cf. Definition 2.31 on page 57) and always yield values less than $+\infty$ (cf. Assumption 3.1 on page 78). Item 2 yields that there is at least one real-valued achievable point. These assumptions are similar to those in the total reward case

function solveWso($M, s_I, \Phi, \boldsymbol{w}, \varepsilon_{\texttt{WSO}}$) — solve WSO

// Φ consists of ℓ reward-bounded objectives

Compute proper epoch sequence $\mathfrak{E} = \boldsymbol{e}_1 \dots \boldsymbol{e}_n$ ending with $\boldsymbol{e}_n = \boldsymbol{e}^{\mathcal{B}}_{init}$

$\hat{\varepsilon} \leftarrow \varepsilon_{\texttt{WSO}} \,/\, (1 + \|\boldsymbol{e}^{\mathcal{B}}_{init}\|)$ *// Precision for epoch MDPs*

Arbitrarily initialize functions $f, f_1, \dots, f_\ell \colon S_{un} \to \mathbb{R}$

$S^{+\infty} \leftarrow \emptyset$; $S^{-\infty} \leftarrow \emptyset$; $S^{\text{fin}} \leftarrow \emptyset$

for $\boldsymbol{e} = \boldsymbol{e}_1, \dots, \boldsymbol{e}_n$ **do**

Build epoch MDP $ep(M, \boldsymbol{e}) = \left\langle S_{\boldsymbol{e}}, Act \uplus \{\bot\}, \Delta_{\boldsymbol{e}}, \mathbf{P}_{\boldsymbol{e}} \right\rangle$

// Compute and cut away states that yield non-finite expected values

$$S^{+\infty} \leftarrow S^{+\infty} \cup \left\{ \hat{s} \in S_{\boldsymbol{e}} \setminus \{s_\bot\} \;\middle|\; \exists\, \sigma\colon \left(\Pr^{un(M)}_{\sigma,\hat{s}}\left(\Diamond S^{+\infty}\right) > 0 \right.\right.$$
$$\left.\left. \text{or } \exists\, j\colon \operatorname{Ex}^{ep(M,\boldsymbol{e})}_{\sigma,\hat{s}}\left(tot(\langle \mathcal{R}^{un}_j, f_j \rangle^{+}_{\boldsymbol{e}})\right) = +\infty \right) \right\}$$

$$S^{-\infty} \leftarrow S^{-\infty} \cup \left\{ \hat{s} \in S_{\boldsymbol{e}} \setminus \{s_\bot\} \;\middle|\; \forall\, \sigma\colon \left(\Pr^{un(M)}_{\sigma,\hat{s}}\left(\Diamond S^{-\infty}\right) > 0 \right.\right.$$
$$\left.\left. \text{or } \exists\, j\colon \operatorname{Ex}^{ep(M,\boldsymbol{e})}_{\sigma,\hat{s}}\left(tot(\langle \mathcal{R}^{un}_j, f_j \rangle_{\boldsymbol{e}})\right) = -\infty \right) \right\}$$

$S^{\text{fin}}_{\boldsymbol{e}} \leftarrow S_{\boldsymbol{e}} \setminus \left(S^{+\infty} \cup S^{-\infty} \cup \{s_\bot\}\right)$; $S^{\text{fin}} \leftarrow S^{\text{fin}} \cup S^{\text{fin}}_{\boldsymbol{e}}$

$C_{\text{fin}} \leftarrow \left\{ \langle \hat{s}, \alpha \rangle \in S^{\text{fin}}_{\boldsymbol{e}} \times Act \;\middle|\; \alpha \in \Delta_{\boldsymbol{e}}(\hat{s}) \text{ and } post^{un(M)}(\hat{s}, \alpha) \subseteq S^{\text{fin}} \right\}$

$M_{\boldsymbol{e}} \leftarrow ep(M, \boldsymbol{e})[\![C_{\text{fin}} \cup \{\langle s_\bot, \bot \rangle\}]\!]$

// Analyze total rewards for submodel $M_{\boldsymbol{e}}$ of $ep(M, \boldsymbol{e})$

$\Phi^{\boldsymbol{e}} \leftarrow \left\langle tot(\langle \mathcal{R}^{un}_1, f_1 \rangle_{\boldsymbol{e}}), \dots, tot(\langle \mathcal{R}^{un}_\ell, f_\ell \rangle_{\boldsymbol{e}}) \right\rangle$

$\mathcal{R}_{\boldsymbol{e}} \leftarrow \langle \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \mathcal{R}^{un}_j, f \rangle_{\boldsymbol{e}}$

$\left\langle \langle v_{\hat{s}}, \boldsymbol{p}_{\hat{s}} \rangle \right\rangle_{\hat{s} \in S^{\text{fin}}_{\boldsymbol{e}}} \leftarrow \texttt{solveWso}^{\forall}_{+}(M_{\boldsymbol{e}}, \Phi^{\boldsymbol{e}}, \mathcal{R}_{\boldsymbol{e}}, \boldsymbol{w}, \hat{\varepsilon})$ *// As in Chapter 4*

for $\hat{s} \in S^{\text{fin}}_{\boldsymbol{e}}$ **do**

$f(\hat{s}) \leftarrow v_{\hat{s}}$

for $j \in \{1..\ell\}$ **do** $f_j(\hat{s}) \leftarrow \boldsymbol{p}_{\hat{s}}(j)$

if $s^{un}_I \in S^{\text{fin}}$ **then**

$v \leftarrow f(s^{un}_I)$; $\boldsymbol{p} \leftarrow \left\langle f_j(s^{un}_I) \right\rangle_{j \in \{1..\ell\}}$

return $\langle v, \boldsymbol{p} \rangle$

else if $s^{un}_I \in S^{+\infty}$ **then**

return ‘ERROR: $\exists\, \varphi_j\colon$ φ_j *not convergent or* $\operatorname{Ex}_{\max}(\varphi_j) = +\infty$’

else if $s^{un}_I \in S^{-\infty}$ **then return** ‘ERROR: $Ach(\Phi) \cap \mathbb{R}^\ell = \emptyset$’

Algorithm 6.2: Solving bounded reward WSO instances (extended)

discussed in Section 4.1.3—which is natural since total reward objectives are subsumed by bounded reward objectives. Our assumptions make sure that a solution to the provided WSO instance exists (cf. Lemmas 3.13 and 4.3 on page 81 and on page 105).

A naive approach to enforce Items 1 and 2 above is to apply the procedure for total rewards from Section 4.1 on the unfolding MDP. This can be done by computing the partition $S_{un} = S^{+\infty} \uplus S^{-\infty} \uplus S^{\text{fin}}$ of unfolding MDP states, where

- $S^{+\infty} \coloneqq \left\{ s_{un} \in S_{un} \;\middle|\; \exists\, \sigma \colon\ \exists\, j \colon\ \mathrm{Ex}^{un(M)}_{\sigma, s_{un}}\left(tot((\mathcal{R}^{un}_j)^+)\right) = +\infty \right\}$,
- $S^{-\infty} \coloneqq \left\{ s_{un} \in S_{un} \;\middle|\; \forall\, \sigma \colon\ \exists\, j \colon\ \mathrm{Ex}^{un(M)}_{\sigma, s_{un}}\left(tot(\mathcal{R}^{un}_j)\right) = -\infty \right\}$, and
- $S^{\text{fin}} \coloneqq S_{un} \setminus \left(S^{+\infty} \cup S^{-\infty}\right)$.

If $s^{un}_I \in S^{+\infty}$ or $s^{un}_I \in S^{-\infty}$ holds for the initial state s^{un}_I of the unfolding MDP, we can infer that Item 1 or Item 2 does not hold, respectively. On the other hand, if $s^{un}_I \in S^{\text{fin}}$, our requirements are met. In this case, we can cut the states in $S^{+\infty} \cup S^{-\infty}$ by considering an appropriate submodel of the unfolding MDP. This yields a new MDP that only consists of the states in S^{fin}, where infinite expected values are no longer a concern.

To compute the above partition, we may analyze the end components of $un(M)$ and their reachability as outlined in Algorithm 4.1 on page 104. However, end components of the unfolding MDP do not stretch over multiple epochs due to the acyclicity of the epoch graph. We can therefore avoid considering the unfolding MDP as a whole by incorporating the end component analysis into the sequential epoch analysis. This is shown in Lines 8 to 12 of Algorithm 6.2. When processing an epoch $\boldsymbol{e}$, we decide for each state $\hat{s} = \langle s, \boldsymbol{e}, \mathfrak{g} \rangle \in S_{\boldsymbol{e}}$ whether it belongs to $S^{+\infty}$, $S^{-\infty}$, or S^{fin}. The sets $S^{+\infty}$ and $S^{-\infty}$ are extended accordingly in Lines 8 and 9 by

- considering reachability of states *outside* of the current epoch that are already known to be in $S^{+\infty}$ or $S^{-\infty}$, and
- investigating the rewards accumulated *within* the current epoch.

This step can be implemented using a combination of end component analysis and reachability analysis in the epoch MDP similar to Algorithm 4.1 on page 104. For the reachability analysis, it suffices to consider the states $\hat{s} \in S_{\boldsymbol{e}}$ of the current epoch as well as immediate successors[1] $\hat{s}' \in \bigcup \{ post(\hat{s}, \alpha) \setminus S_{\boldsymbol{e}} \mid \hat{s} \in S_{\boldsymbol{e}}, \alpha \in \Delta_{un}(\hat{s}) \}$. Once the sets $S^{+\infty}$ and $S^{-\infty}$ are extended by the corresponding states from the current epoch, we can cut away those problematic states from the epoch MDP by considering the submodel $M_{\boldsymbol{e}}$ of $ep(M, \boldsymbol{e})$ in Line 12, where states in (and transitions to) $S^{+\infty} \cup S^{-\infty}$ are omitted. For $M_{\boldsymbol{e}}$, non-finite expected values are no longer a concern.

[1] To see this, observe the following properties of $S^{+\infty}$ and $S^{-\infty}$: If $\hat{s}' \notin S^{+\infty}$, then $S^{+\infty}$ cannot be reached from $\hat{s}'$. Similarly, if $\hat{s}' \notin S^{-\infty}$, there is always a strategy that avoids reaching $S^{-\infty}$ from $\hat{s}'$.

In Lines 19 to 24, we check if the initial state of the unfolding MDP is contained in S^{fin}, $S^{+\infty}$, or $S^{-\infty}$. If $s_I^{un} \in S^{\text{fin}}$, an answer to the `WSO` query is returned. Otherwise, an error is raised to indicate that one of the assumptions is violated.

Approximative Computations

When an epoch $\boldsymbol{e}' \in \mathbb{E}_d$ is only analyzed approximatively, the approximation error affects all epochs $\boldsymbol{e}$ that depend on $\boldsymbol{e}'$, i.e., all epochs that have a path in the epoch graph leading to $\boldsymbol{e}'$. This is because the reward assignment $\langle \mathcal{R}_j^{un}, f_j \rangle_{\boldsymbol{e}}$ for $j \in \{1..\ell\}$ considered for epoch $\boldsymbol{e}$ depends on potentially imprecise values computed for $\boldsymbol{e}'$ which are stored in the function f_j. In general, it is not possible to make up for the approximation error at $\boldsymbol{e}'$—even if all other epochs were analyzed exactly (i.e., without approximation error). The following lemma provides bounds on the effect that a difference in the considered functions $f_1, \ldots, f_\ell$ can have.

Lemma 6.13 Let $f, f' \colon S_{un} \to \mathbb{R}$ be functions with

$$\forall\, s_{un} \in S_{un} \colon \quad \varepsilon_1 \;\leq\; f(s_{un}) - f'(s_{un}) \;\leq\; \varepsilon_2$$

for some $\varepsilon_1, \varepsilon_2 \in \overline{\mathbb{R}}$. Then, for strategy $\sigma \in \Sigma^{ep(M,\boldsymbol{e})}$ and reward assignment $\mathcal{R}^{un}$ for $un(M)$, we have

$$\forall\, \hat{s} \in S_{\boldsymbol{e}} \colon \quad \varepsilon_1 \;\leq\; \mathrm{Ex}_{\sigma,\hat{s}}^{ep(M,\boldsymbol{e})}\big(tot(\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}})\big) - \mathrm{Ex}_{\sigma,\hat{s}}^{ep(M,\boldsymbol{e})}\big(tot(\langle \mathcal{R}^{un}, f' \rangle_{\boldsymbol{e}})\big) \;\leq\; \varepsilon_2.$$

Proof. We have

$$\begin{aligned}
&\mathrm{Ex}_{\sigma,\hat{s}}^{ep(M,\boldsymbol{e})}\big(tot(\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}})\big) - \mathrm{Ex}_{\sigma,\hat{s}}^{ep(M,\boldsymbol{e})}\big(tot(\langle \mathcal{R}^{un}, f' \rangle_{\boldsymbol{e}})\big) \\
&= \mathrm{Ex}_{\sigma,\hat{s}}^{ep(M,\boldsymbol{e})}\big(tot(\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}}) - tot(\langle \mathcal{R}^{un}, f' \rangle_{\boldsymbol{e}})\big) \\
&= \mathrm{Ex}_{\sigma,\hat{s}}^{ep(M,\boldsymbol{e})}\big(tot(\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}} - \langle \mathcal{R}^{un}, f' \rangle_{\boldsymbol{e}})\big) \\
&= \mathrm{Ex}_{\sigma,\hat{s}}^{ep(M,\boldsymbol{e})}\big(tot(\langle \circ, f - f' \rangle_{\boldsymbol{e}})\big),
\end{aligned}$$

where $\circ$ is the reward assignment that sets all rewards to zero. Consider a path $\pi \in Paths_{\text{inf}}^{ep(M,\boldsymbol{e})}$. If π does not reach $s_\perp$, then $tot(\langle \circ, f - f' \rangle_{\boldsymbol{e}})(\pi) = 0$. Otherwise,

π contains a transition $\hat{s} \xrightarrow{\alpha} s_\perp$ for $\hat{s} \in S_{\boldsymbol{e}} \setminus \{s_\perp\}$ and we get (cf. Definition 6.5)

$$\begin{aligned} tot(\langle \circ, f - f' \rangle_{\boldsymbol{e}})(\pi) &= \langle \circ, f - f' \rangle_{\boldsymbol{e}}[\hat{s}, \alpha, s_\perp] \\ &= \sum_{s'_{un} \in (S_{un} \setminus S_{\boldsymbol{e}})} \left(\frac{\mathbf{P}_{un}(\hat{s}, \alpha, s'_{un})}{\mathbf{P}_{\boldsymbol{e}}(\hat{s}, \alpha, s_\perp)} \cdot \Big(f(s'_{un}) - f(s'_{un}) \Big) \right) \\ &\leq \sum_{s'_{un} \in (S_{un} \setminus S_{\boldsymbol{e}})} \left(\frac{\mathbf{P}_{un}(\hat{s}, \alpha, s'_{un})}{\mathbf{P}_{\boldsymbol{e}}(\hat{s}, \alpha, s_\perp)} \cdot \varepsilon_2 \right) = \frac{\varepsilon_2}{\mathbf{P}_{\boldsymbol{e}}(\hat{s}, \alpha, s_\perp)}. \end{aligned}$$

A lower bound using ε_1 can be established similarly. Since the objective value is multiplied with the path probabilities when considering expected values, we get

$$\varepsilon_1 \leq \mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\sigma,\hat{s}}\big(tot(\langle \circ, f - f' \rangle_{\boldsymbol{e}})\big) \leq \varepsilon_2$$

which concludes the proof. ∎

When all epochs are analyzed approximatively, each individual epoch adds a certain error to the already erroneous values of its successor epochs. For an epoch $\boldsymbol{e}$, let $\|\boldsymbol{e}\|$ be the length of the longest path in the epoch graph starting in $\boldsymbol{e}$. Intuitively, if each epoch adds an additional approximation error of $\hat{\varepsilon}$ to the approximation error of its successors, then the overall error at $\boldsymbol{e}$ accumulates to $(\|\boldsymbol{e}\| + 1) \cdot \hat{\varepsilon}$. This propagation of approximation errors is comparable to the observations we made for topological value iteration in Section 4.4.4 and has also been observed in [HH16] for the single reward bound case.

Our sequential epoch analysis algorithm is extended as follows. In addition to the functions $f_1, \ldots, f_\ell$—which are used to store the obtained achievable points $\boldsymbol{p}_{\hat{s}}$ for the solved total reward `WSO` instances—Algorithm 6.2 now also stores the obtained upper bounds $v_{\hat{s}}$ in Line 17 using the new function $f \colon S_{un} \to \mathbb{R}$. Moreover, an additional reward assignment $\mathcal{R}_{\boldsymbol{e}}$ is considered for each epoch MDP in Line 14. $\mathcal{R}_{\boldsymbol{e}}$ is based on the weighted sum of the reward assignments of the unfolding MDP as well as the values given by f. The idea is to use f for over-approximating the optimal weighted sum, whereas $f_1, \ldots, f_\ell$ are used to under-approximate achievable values. The precision parameter $\hat{\varepsilon}$ we set in Line 3 reflects the error allowed for each individual epoch. Finally, Line 15 considers a variant $\texttt{solveWso}^{\forall}_{+}$ of the function `solveWso` from Algorithm 4.2 on page 108 that

- returns a solution for every possible initial state (as already considered in Line 7 of Algorithm 6.1) and
- considers the given reward assignment $\mathcal{R}_{\boldsymbol{e}}$ when optimizing total rewards *instead* of the weighted sum of the other reward assignments, i.e., Line 7 of Algorithm 4.2 is omitted and the reward assignment $\mathcal{R}$ considered in Line 8 is provided as an input

parameter instead.

Lemma 6.14 When processing epoch $\boldsymbol{e} \in \mathbb{E}_d$, the invocation of $\texttt{solveWso}_+^{\forall}$ in Line 15 of Algorithm 6.2 yields $\left\langle \langle v_{\hat{s}}, \boldsymbol{p}_{\hat{s}} \rangle \right\rangle_{\hat{s} \in S_{\boldsymbol{e}}^{\text{fin}}}$ as well as a strategy $\sigma \in \Sigma^{M_{\boldsymbol{e}}}$, such that for all $\hat{s} \in S_{\boldsymbol{e}}^{\text{fin}}$:

1. $\forall\, j \in \{1..\ell\}\colon\ \boldsymbol{p}_{\hat{s}}(j) \ \leq\ \text{Ex}_{\sigma,\hat{s}}^{M_{\boldsymbol{e}}}\big(tot(\langle \mathcal{R}_j^{un}, f_j \rangle_{\boldsymbol{e}})\big)$
2. $\text{Ex}_{\max,\hat{s}}^{M_{\boldsymbol{e}}}\big(tot(\mathcal{R}_{\boldsymbol{e}})\big) \ \leq\ v_{\hat{s}}$
3. $v_{\hat{s}} \ \leq\ \boldsymbol{w} \cdot \boldsymbol{p}_{\hat{s}} + \varepsilon_{\texttt{WSO}} \cdot \dfrac{1 + \|\boldsymbol{e}\|}{1 + \|\boldsymbol{e}_{init}^{\mathcal{B}}\|} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}$

Proof. Items 1 and 2 follow as in the proof of Theorem 4.5 on page 110.
Similar as in the proof of Theorem 6.12, we can assume that Item 3 already holds for all epochs $\boldsymbol{e}'$ on which $\boldsymbol{e}$ depends.
Define $f'\colon S_{un} \to \mathbb{R}$ with $f'(\hat{s}) := \sum_{j=1}^{\ell} \boldsymbol{w}(j) f_j(\hat{s})$ and let $\boldsymbol{e}'$ be an epoch that was processed before $\boldsymbol{e}$. Due to our assumption that Lemma 6.14 already holds for $\boldsymbol{e}'$ and due to Lines 17 and 18, we get for all $\hat{s}' = \langle s', \boldsymbol{e}', \mathfrak{g}' \rangle \in S_{un}$:

$$f(\hat{s}') \ \leq\ f'(\hat{s}') + \varepsilon_{\texttt{WSO}} \cdot \frac{1 + \|\boldsymbol{e}'\|}{1 + \|\boldsymbol{e}_{init}^{\mathcal{B}}\|} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}} \ \leq\ f'(\hat{s}') + \varepsilon_{\texttt{WSO}} \cdot \frac{\|\boldsymbol{e}\|}{1 + \|\boldsymbol{e}_{init}^{\mathcal{B}}\|} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}.$$

For the second inequality, note that the longest path in the epoch graph starting at successor epoch $\boldsymbol{e}'$ of $\boldsymbol{e}$ has to be strictly shorter than the longest path starting at $\boldsymbol{e}$, i.e., $\|\boldsymbol{e}'\| < \|\boldsymbol{e}\|$.
We apply Lemma 6.13 to f and f', which for reward assignment

$$\mathcal{R}'_{\boldsymbol{e}} \ :=\ \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \langle \mathcal{R}_j^{un}, f_j \rangle_{\boldsymbol{e}} \ =\ \left\langle \sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot \mathcal{R}_j^{un}, f' \right\rangle_{\boldsymbol{e}}$$

yields

$$\text{Ex}_{\sigma,\hat{s}}^{M_{\boldsymbol{e}}}\big(tot(\mathcal{R}_{\boldsymbol{e}})\big) \ \leq\ \text{Ex}_{\sigma,\hat{s}}^{M_{\boldsymbol{e}}}\big(tot(\mathcal{R}'_{\boldsymbol{e}})\big) + \varepsilon_{\texttt{WSO}} \cdot \frac{\|\boldsymbol{e}\|}{1 + \|\boldsymbol{e}_{init}^{\mathcal{B}}\|} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}. \tag{6.4}$$

Let ε be defined as in Line 2 of Algorithm 4.2, i.e.,

$$\varepsilon \ :=\ \frac{1}{2} \cdot \hat{\varepsilon} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}} \ =\ \frac{1}{2} \cdot \varepsilon_{\texttt{WSO}} \cdot \frac{1}{1 + \|\boldsymbol{e}_{init}^{\mathcal{B}}\|} \cdot \sqrt{\boldsymbol{w} \cdot \boldsymbol{w}}.$$

We conclude the proof for Item 3 by incorporating Equation (6.4) into the reasoning

from the proof of Theorem 4.5 on page 110.

$$
\begin{aligned}
v_{\hat{s}} &\leq \mathrm{Ex}^{M_e}_{\sigma,\hat{s}}\big(tot(\mathcal{R}_{\boldsymbol{e}})\big) + \varepsilon && \text{(Line 8 of Algorithm 4.2)}\\
&\leq \left(\mathrm{Ex}^{M_e}_{\sigma,\hat{s}}\big(tot(\mathcal{R}'_{\boldsymbol{e}})\big) + \varepsilon_{\texttt{WSO}} \cdot \frac{\|\boldsymbol{e}\|}{1+\|\boldsymbol{e}^{\mathcal{B}}_{init}\|} \cdot \sqrt{\boldsymbol{w}\cdot\boldsymbol{w}}\right) + \varepsilon && \text{(Equation (6.4))}\\
&\leq \boldsymbol{w}\cdot\boldsymbol{p}_{\hat{s}} + \varepsilon_{\texttt{WSO}} \cdot \frac{\|\boldsymbol{e}\|}{1+\|\boldsymbol{e}^{\mathcal{B}}_{init}\|} \cdot \sqrt{\boldsymbol{w}\cdot\boldsymbol{w}} + 2\cdot\varepsilon && \text{(Proof of Theorem 4.5)}\\
&= \boldsymbol{w}\cdot\boldsymbol{p}_{\hat{s}} + \varepsilon_{\texttt{WSO}} \cdot \frac{1+\|\boldsymbol{e}\|}{1+\|\boldsymbol{e}^{\mathcal{B}}_{init}\|} \cdot \sqrt{\boldsymbol{w}\cdot\boldsymbol{w}}. && \text{(Substitute } \varepsilon\text{)}
\end{aligned}
$$

■

Theorem 6.15 The value v and the point $\boldsymbol{p}$ returned by Algorithm 6.2 yield a solution to the provided reward-bounded WSO instance.

Proof. We argue using the respective WSO instance on the unfolding MDP which is valid due to Theorem 6.3.
Applying Lemma 6.14 for the initial epoch $\boldsymbol{e}^{\mathcal{B}}_{init}$ and initial state s^{un}_I of $un(M)$ yields

$$v \;\leq\; \boldsymbol{w}\cdot\boldsymbol{p} + \varepsilon_{\texttt{WSO}} \cdot \sqrt{\boldsymbol{w}\cdot\boldsymbol{w}}.$$

It remains to show that $\boldsymbol{p} \in Ach^{un(M)}_{s^{un}_I}(\Phi^{un})$ and

$$\sup\left\{\boldsymbol{w}\cdot\mathrm{Ex}^{un(M)}_{\sigma,s^{un}_I}(\Phi^{un}) \;\middle|\; \sigma\in\Xi\right\} \;\leq\; v,$$

where $\Xi \coloneqq \left\{\sigma\in\Sigma^{un(M)} \;\middle|\; \mathrm{Ex}^{un(M)}_{\sigma,s^{un}_I}(\Phi^{un}) \in \mathbb{R}^{\ell}\right\}$ (cf. Problem 3.4 on page 81).
As in the proof of Theorem 6.12, we consider the memoryless strategy $\sigma\in\Sigma^{un(M)}$ that arises by composing the strategies obtained for each epoch MDP. We show for all $\hat{s}\in S_{un}$:

1. $\forall\, j\in\{1..\ell\}\colon\; f_j(\hat{s}) \;\leq\; \mathrm{Ex}^{un(M)}_{\sigma,\hat{s}}(tot(\mathcal{R}^{un}_j))$ and
2. $f(\hat{s}) \;\geq\; \sup\left\{\boldsymbol{w}\cdot\mathrm{Ex}^{un(M)}_{\sigma',\hat{s}}(\Phi^{un}) \;\middle|\; \sigma\in\Xi\right\}$.

Items 1 and 2 are similar to the two items from the proof of Theorem 6.12. Their proof is analogous except that Lemmas 6.13 and 6.14 are used to establish the corresponding inequalities. ■

Remark 6.2 (Acyclic Epoch MDPs) For practical applications, the individual epoch MDPs are often acyclic as rewards usually represent quantities like time or energy usage, for which the possibility to perform infinitely many "interesting" steps without accumulating any reward would be considered a modeling error. In the timed case, for example, such a model would allow Zeno behavior, which is generally considered unrealistic and undesirable. When epoch MDPs are acyclic, (sound-) value iteration converges to the exact result in a finite number of iterations. In this case, the tightening of the precision according to Algorithm 6.2 usually has no effect on runtime.

Remark 6.3 (Combinations with LRA Reward Objectives) Long-run average (LRA) reward objectives as discussed in Section 2.3.2 and Chapter 5 can be incorporated into our framework: given an arbitrary combination of reward-bounded and LRA reward objectives for MDP M, we can transform the reward-bounded objectives to total reward objectives for the unfolding MDP $un(M)$. Since unfolding preserves LRA reward objectives, techniques from Chapter 5—which mainly deal with end components (ECs)—can be applied to $un(M)$ and the resulting combination of total- and LRA reward objectives. Due to the acyclicity of the epoch graph, the ECs of $un(M)$ do not spread over multiple epochs. This also allows us to include the LRA computations into the sequential epoch analysis framework described above.

6.2.5 Runtime Complexity

In the following, we discuss the complexity of our approach relative to the size of a binary encoding of the reward bound values $b_1, \dots, b_d$ occurring in the considered bounds $\mathcal{B} = \langle \llbracket \mathcal{R}_1 \sim_1 b_1 \rrbracket, \dots, \llbracket \mathcal{R}_d \sim_d b_d \rrbracket \rangle$.

Algorithm 6.1 and its extended variant Algorithm 6.2 solve $|\mathfrak{E}| \leq \prod_{i=1}^{d}(b_i + 2)$ many total reward WSO instances on epoch MDPs $ep(M, \boldsymbol{e})$. Each of these computations can be done in polynomial time (in the size of $ep(M, \boldsymbol{e})$), e.g., via the LP encoding from Section 4.6.3. We conclude that Algorithms 6.1 and 6.2 can be implemented with an exponential runtime—similar to the explicit unfolding approach, where a single total reward WSO instance is solved on the exponentially large unfolding MDP.

In general, an exponential runtime cannot be avoided: [HK15] shows that deciding whether $\mathrm{Ex}^{M}_{\max}(\varphi) \geq a$ for a bounded reachability objective φ with non-negative rewards, and a threshold $a \in \mathbb{R}$ is EXPTIME-complete[2]. On the other hand, exper-

[2] [HK15] considers a slightly different setting with a single goal state set, a single reward assignment, but arbitrary Boolean combinations of reward bounds. However, the result also applies to our setting since the EXPTIME-hardness proof in [HK15, Theorem 8] only considers a conjunction of two reward bounds

iments with topological value iteration [BKLP+17] and with single reward-bounds (e.g., [BDDK+14; HH16]) have shown the practical benefits of analyzing several small sub-models instead of one large MDP. We make similar observations in Section 9.5 for the multi-objective, multi-reward bounded case.

6.2.6 Implementation Optimizations

We discuss two optimizations that—while not affecting theoretical complexity—can improve the practical performance of our approach.

Optimizing Memory | Algorithm 6.1 stores the values $f_j(\hat{s})$ for $j \in \{1..\ell\}$ and each state $\hat{s} \in S_{un}$ of the unfolding MDP. However, memory consumption can be optimized by erasing values $f_j(\hat{s})$ when they are no longer accessed by any of the remaining epoch MDPs. A similar approach can be applied for Algorithm 6.2, which also stores the function $f\colon S_{un} \to \mathbb{R}$ and the partition $S_{un} = S^{+\infty} \uplus S^{-\infty} \uplus S^{\text{fin}}$.

This optimization requires some bookkeeping to find erasable values. We keep the overhead at a moderate level by clustering together states with the same epoch. More precisely, all values for states with an epoch $\boldsymbol{e}$ are erased as soon as every direct predecessor $\boldsymbol{e}'$ of $\boldsymbol{e}$ (in the epoch graph) has been analyzed.

If $d = 1$ (i.e. there is only a single reward bound), such an optimization yields an algorithm that runs in polynomial space. In the general case ($d > 1$), the memory requirements remain exponential in the size of a binary encoding of the reward bound values.

Exploiting Epoch Similarities | Consider the equivalence relation

$$\bowtie \;:=\; \left\{ \langle \boldsymbol{e}, \boldsymbol{e}' \rangle \in \mathbb{E}_d \times \mathbb{E}_d \;\middle|\; \forall\, i \in \{1..d\} \colon\; \boldsymbol{e}(i) = \oslash \text{ iff } \boldsymbol{e}'(i) = \oslash \right\}.$$

For two epochs $\boldsymbol{e}, \boldsymbol{e}' \in \mathbb{E}_d$ with $\boldsymbol{e} \bowtie \boldsymbol{e}'$, the same reward bounds are active. We have that the epoch MDPs $ep(M, \boldsymbol{e})$ and $ep(M, \boldsymbol{e}')$ are isomorphic, i.e., equal up to renaming of states (cf. Example 6.9). Moreover, the reward assignments considered for the two epoch MDPs only differ in their assignments to transitions to $s_\perp$.

When analyzing epoch $\boldsymbol{e}_k$ such that $\boldsymbol{e}_{k-1} \bowtie \boldsymbol{e}_k$ for the previously checked epoch $\boldsymbol{e}_{k-1}$, these observations allow us to reuse the epoch MDP and (most of) the assigned rewards. To further amplify this effect, we always consider a proper epoch sequence $\mathfrak{E}$ that groups together epochs with the same $\bowtie$-equivalence class.

---

of the form $\wr \mathcal{R} \leq b \wr \wedge \wr \mathcal{R} \geq b \wr$.

6.3 Related Work

We provide an overview of related work on analyzing reward-bounded objectives.

Explicit Unfolding | [YLY98] considers bounded reachability probabilities in MDPs concerning a single reward bound of the form $\wr \mathcal{R} \leq b \wr$ and presents an approach that—roughly—corresponds to performing value iteration on the unfolding MDP. Similarly, [UB13] presents an LP that corresponds to the LP encoding from Section 4.6.3 when applied to the unfolding MDP. Explicitly unfolding the model by incorporating accumulated rewards into the state space has been suggested for DTMCs in [AHK03] and for MDPs in [Oht04]. In [RRS17], an explicit unfolding approach for multiple reward bounds—similar to our construction from Section 6.1—is used to decide multi-objective achievability (cf. Problem 3.1 on page 78) for multiple bounded reachability objectives, where each individual objective considers a single bounded goal with a single reward bound.

corollarySequential Analysis | [BDDK+14] improves the LP approach of [UB13]: instead of solving one large LP, several smaller LPs are solved sequentially—one LP for each reward epoch. In addition, [BDDK+14] considers bounded *reward* objectives and objectives with up to *two* reward bounds. The latter is achieved by unfolding the MDP in only one dimension—essentially reducing the query to a single reward bound. [HH16] further improves the practicability of the sequential approach by suggesting variants based on value iteration and state elimination. [HYV14] provides a similar approach based on an SCC decomposition of the unfolding MDP combined with topological value iteration. [KBCD+18] describes an implementation of the sequential approach using binary decision diagrams. *Our sequential epoch analysis approach from Section 6.2 lifts the ideas of [BDDK+14; HH16] from single- to multi-dimensional reward bounds without explicitly unfolding the model.* As we solve the weighted sum optimization problem (WSO), multi-objective queries can be answered using our methods from Chapter 3. A procedure for WSO considering simpler *step*-bounded reward objectives has been presented in [FKP12].

Variations of Reward-Bounded Objectives | Other related work considers variations of reward bounded objectives that are not (fully) compatible with the notions from this chapter. A Q-learning approach that can also deal with continuous rewards is given in [CGZO+17]. [HK15] considers arbitrary Boolean combinations of reward bounds and shows EXPTIME-completeness of the related single-objective decision problem[3]. [RRS17] introduces multi-constraint percentile queries which operate with more general types of reward bounds that, e.g., constrain the long-run average reward

[3] The EXPTIME-hardness proof also carries over to our setting, as mentioned in Section 6.2.5.

instead of the accumulated reward of a path. [BFRR17] considers beyond worst-case queries which guarantee that a reward bound holds for *all* executions while simultaneously optimizing an expected value. [BGMR18] combines the percentile queries of [RRS17] with the beyond worst-case analysis of [BFRR17]. Reward bounds with mixtures of positive- and negative rewards for DTMCs are considered in [HKL17]. [BBEK13; BKKN+14] extend MDPs with unbounded counters that can be incremented, decremented, and compared to zero. Consumption MDPs [BBNO+20; BCNO+21] allow to monitor a resource that is consumed during an execution but can also be reset to a given capacity by visiting dedicated reload states. [CC13] considers bounded reachability objectives in a stochastic network of nodes whose states can change over time.

Chapter Summary

- ➜ Bounded reward objectives and bounded reachability objectives can be reduced to total reward objectives by incorporating the accumulated rewards into the state space—yielding the *unfolding MDP*.
- ➜ Building the complete unfolding MDP can be avoided by *sequentially* analyzing reward epochs.
- ➜ This allows solving instances of the weighted sum optimization problem (`WSO`) concerning reward-bounded objectives in a practically efficient way.

—Chapter 7—

Multi-Dimensional Quantiles

Outlook | Reward-bounded objectives as discussed in Chapter 6 consider bounds that impose fixed thresholds to the accumulated rewards. Our main interest so far has been the calculation and optimization of expected values obtained for such objectives. We now study the opposite question: What are the reward bound values required to satisfy a given threshold on the expected value of a reward-bounded objective? This question thus asks for computing *quantiles* as considered in, e.g., [UB13; BDDK+14; KBCD+18; HKL17]. We lift quantiles to multiple reward bounds. In particular, we present an efficient procedure to answer questions like:

- How much time and energy is required to fulfill a task with at least probability 0.8?
- How many products can be manufactured with how many workers within an expected time of 8 hours?
- How much energy is needed to complete how many tasks with probability 0.9?

Section 7.1 introduces multi-reward bounded quantile queries to formalize these questions. We sketch our approach to solve them in Section 7.2, and provide a more extensive treatment of quantiles with only upper or only lower reward bounds in Section 7.3. Finally, we address more complex forms of quantiles in Section 7.4 and discuss related work in Section 7.5.

Origins | The results presented in this chapter originate from [10]. We extend the original work towards more detailed proofs as well as the treatment of bounded *reward* objectives and more expressive bounded reachability objectives as given in Definitions 2.31 and 2.33 on page 57 and on page 58.

Notations | Let $M = \langle S, Act, \Delta, \mathbf{P} \rangle$ be an MDP. We consider reward bounds (cf. Definition 2.30 on page 56) without a fixed bound value.

indefinite reward bound

Definition 7.1 (Indefinite Reward Bound) An *indefinite (reward) bound* for MDP M is a structure $\wr\mathcal{R} \sim ?\wr$ with reward assignment $\mathcal{R}$ for M and relation $\sim \in \{<, \leq, \geq, >\}$.

For a tuple of d indefinite reward bounds

$$\mathcal{B} \coloneqq \left\langle \wr\mathcal{R}_1 \sim_1 ?\wr, \ldots, \wr\mathcal{R}_d \sim_d ?\wr \right\rangle$$

we consider bounded *reward* objectives of the form $bnd^{\mathcal{B}}(\mathcal{R}', \mathcal{I})$ and bounded *reachability* objectives of the form $bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})$, which—syntactically—are defined similar to their counterparts for definite reward bounds (cf. Definitions 2.31 and 2.33 on page 57 and on page 58). To evaluate these objectives, concrete reward bound values have to be known. The *instantiation* for indefinite bounds $\mathcal{B}$ as above and a vector $\boldsymbol{b} \coloneqq \langle b_1, \ldots, b_d \rangle \in \mathbb{R}^d$ is given by

$$\mathcal{B}[\![\boldsymbol{b}]\!] \coloneqq \left\langle \wr\mathcal{R}_1 \sim_1 b_1\wr, \ldots, \wr\mathcal{R}_d \sim_d b_d\wr \right\rangle.$$

reward limits

We refer to the vector $\boldsymbol{b}$ as *reward limits* (for $\mathcal{B}$) and lift instantiations to objectives, where

$$\varphi[\![\boldsymbol{b}]\!] \coloneqq \begin{cases} bnd^{\mathcal{B}[\![\boldsymbol{b}]\!]}(\mathcal{R}', \mathcal{I}) & \text{if } \varphi = bnd^{\mathcal{B}}(\mathcal{R}', \mathcal{I}) \\ bnd^{\mathcal{B}[\![\boldsymbol{b}]\!]}_{\mathcal{G}}(c, \mathcal{J}) & \text{if } \varphi = bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J}). \end{cases}$$

7.1 Quantiles in Multiple Dimensions

Definition 7.2 (Multi-dimensional Quantile Query) A d-dimensional quantile query for MDP M with dedicated initial state s_I and $d \in \mathbb{N}$ indefinite reward bounds $\mathcal{B}$ is given by $Qu^M_{s_I}(\varphi \bowtie a)$, with

- bounded (reward or reachability) objective $\varphi \in \left\{bnd^{\mathcal{B}}(\mathcal{R}', \mathcal{I}),\ bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})\right\}$,
- threshold relation $\bowtie \in \{<, \leq, \geq, >\}$, and
- threshold value $a \in \mathbb{R}$.

For the remainder, we fix a d-dimensional quantile query $\Psi = Qu^M_{s_I}(\varphi \bowtie a)$ for a bounded (reward or reachability) objective $\varphi \in \left\{bnd^{\mathcal{B}}(\mathcal{R}', \mathcal{I}),\ bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})\right\}$ with indefinite bounds $\mathcal{B} = \left\langle \wr\mathcal{R}_1 \sim_1 ?\wr, \ldots, \wr\mathcal{R}_d \sim_d ?\wr \right\rangle$ and bounded goals $\mathcal{G} = \left\langle \langle \mathcal{I}_1, G_1 \rangle, \ldots, \langle \mathcal{I}_m, G_m \rangle \right\rangle$, such that the following assumptions hold (cf. Assumption 6.1 on page 172).

Assumption 7.1 For all $i \in \{1..d\}$ we have

(i) $\mathcal{R}_i(s, \alpha, s') \in \mathbb{N}$ for all $s, s' \in S$ and $\alpha \in \Delta(s) \subseteq Act$ and

(ii) $\sim_i \in \{\leq, >\}$.

Furthermore, if $\varphi = bnd^{\mathcal{B}}(\mathcal{R}', \mathcal{I})$ is a bounded reward objective, then

(iii) either $\mathcal{R}' \geq 0$ or $\mathcal{R}' \leq 0$, i.e., the reward assignment $\mathcal{R}'$ only considers non-negative or only considers non-positive rewards,

and if $\varphi = bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})$ is a bounded reachability objective, then (cf. Assumption 6.2 on page 173)

(iv) $\exists\, j \in \mathcal{J}\colon \big(s_I \notin G_j\big)$ or $\big(\exists\, i \in \mathcal{J}_j\colon \sim_i = >\big)$.

Items (i), (ii) and (iv) of Assumption 7.1 make sure that our procedures from Chapter 6—in particular the sequential epoch analysis—can be applied to quantiles as described in the next section. Items (i) and (ii) can sometimes be made valid through scaling of the reward assignments and the reward limits as outlined on page 172. Due to Item (i) it suffices to consider reward limits $\boldsymbol{b} \in \mathbb{N}^d$ over *natural* numbers. Item (iii) ensures that the bounded reward objective $\varphi[\![\boldsymbol{b}]\!]$ for reward limits $\boldsymbol{b}$ is convergent (and thus well-defined) and monotone in the sense that increasing a reward limit $\boldsymbol{b}(i)$ for $i \in \{1..d\}$ also increases (or decreases) the accumulated reward for $\mathcal{R}'$. We make use of this property in Lemma 7.1 below.

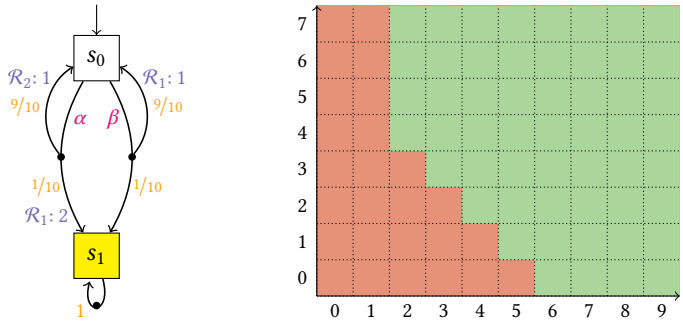
The solution of a quantile query is a set of reward limits $\boldsymbol{b} \in \mathbb{N}^d$ under which the maximal expected value for $\varphi[\![\boldsymbol{b}]\!]$ satisfies the imposed threshold.

Definition 7.3 (Satisfying Reward Limits) The set of *satisfying reward limits* for d-dimensional quantile query $\Psi = Qu^M_{s_I}(\varphi \bowtie a)$ is given by

$$Sat(\Psi) \;:=\; \big\{\boldsymbol{b} \in \mathbb{N}^d \mid \mathrm{Ex}^M_{\max,s_I}(\varphi[\![\boldsymbol{b}]\!]) \bowtie a\big\}.$$

Given $\Psi = Qu^M_{s_I}(\varphi \bowtie a)$, we write $\overline{\Psi} := Qu^M_{s_I}(\varphi \not\bowtie a)$ for the complementary query, where the threshold relation is inverted (i.e., $x \not\bowtie y$ iff $\neg(x \bowtie y)$). Observe that $Sat(\overline{\Psi}) = \mathbb{N}^d \setminus Sat(\Psi)$ or—equivalently—$Sat(\Psi) \uplus Sat(\overline{\Psi}) = \mathbb{N}^d$.

Remark 7.1 Quantile queries consider strategies that *maximize* the resulting expected value. This convention is without loss of generality: minimizing strategies are considered by negating the objective φ (cf. Lemma 2.12 on page 59) and the threshold value a, and flipping the relation $\bowtie$ accordingly.



(a) MDP M_1 (b) $Sat(\Psi_1)$ in green and $Sat(\overline{\Psi_1})$ in red

Figure 7.1: Example MDP and satisfying reward limits (cf. Examples 7.1 and 7.2)

Example 7.1 Consider the MDP M_1 given in Figure 7.1a and the quantile query

$$\Psi_1 = Qu_{s_0}^{M_1}\left(bnd_{\mathcal{G}}^{\mathcal{B}}(1,\{1\}) > 0.5\right), \text{ where}$$
$$\mathcal{B} = \left\langle \wr\mathcal{R}_1 \leq ?\wr, \wr\mathcal{R}_2 \leq ?\wr \right\rangle \quad \text{and} \quad \mathcal{G} = \left\langle \langle \{1,2\}, \{s_1\} \rangle \right\rangle.$$

Intuitively, our goal is to reach the state s_1 in more than 50 % of the cases. The query Ψ_1 asks how much reward for $\mathcal{R}_1$ and $\mathcal{R}_2$ needs to be incurred to achieve this goal. The (upper-right) green area in Figure 7.1b indicates the set of satisfying reward limits for Ψ_1, given by

$$Sat(\Psi_1) = \left\{ \boldsymbol{c} \in \mathbb{R}^2 \mid \exists \boldsymbol{b} \in \{\langle 2,4\rangle, \langle 3,3\rangle, \langle 4,2\rangle, \langle 5,1\rangle, \langle 6,0\rangle\} \colon \boldsymbol{c} \geq \boldsymbol{b} \right\}.$$

Concretely, the set describes a form of closure of a set of points on the frontier. We discuss why this is the satisfying set. First, consider reward limits $\langle 1, b_2 \rangle$ for arbitrary b_2. This vector indicates a limit of 1 for indefinite bound $\wr\mathcal{R}_1 \leq ?\wr$. In particular, the action α at state s_0 is then never helpful in satisfying the objective as it takes reward 2 for $\mathcal{R}_1$. Thus, we have to take the action β. When taking this action, we may return to s_0 at most once before violating the reward limit. Thus, the probability to reach the target is $0.1 + 0.9 \cdot 0.1 = 0.19 < 0.5$, and these reward limits violate the query. Now, consider reward limits $\langle 6, 0 \rangle$. Using similar reasoning as above, only action β is relevant. We may take the self-loop at most 6 times, which yields a probability to reach the target within the reward limit of $\sum_{i=0}^{6} 0.1 \cdot 0.9^i \approx 0.52 > 0.5$, and thus these reward limits satisfy the query. Finally, consider reward limits $\langle 2, 4 \rangle$. Now, action α helps: We can take α at most 4 times.

If we still have not reached the target, we have $\langle 2, 0\rangle$ reward remaining, which can be spend trying action β up to 2 times. The probability of reaching the target under this strategy is again $\sum_{i=0}^{6} 0.1 \cdot 0.9^i \approx 0.52 > 0.5$.

In Example 7.1, the infinite set of satisfying reward limits is concisely described using the finitely many reward limits generating its frontier. We now lift this type of representation to general quantile queries. We first define an order $\trianglerighteq^{\Psi} \subseteq \overline{\mathbb{N}}^d \times \overline{\mathbb{N}}^d$ with respect to the quantile query $\Psi = Qu^M_{s_I}(\varphi \bowtie a)$, where, $\overline{\mathbb{N}} := \mathbb{N} \cup \{\infty\}$ denotes the set of *extended natural numbers*[1].

We say that the objective φ is *non-negative* if either

non-negative objective

- $\varphi = bnd^{\mathcal{B}}(\mathcal{R}', \mathcal{I})$ and $\mathcal{R}'$ only assigns non-negative rewards, or
- $\varphi = bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})$ and $c \geq 0$.

Otherwise, φ is *non-positive*. Intuitively, for non-negative (non-positive) objectives, considering less restrictive reward bounds increases (decreases) the obtained value. Formally, we say that φ is *increasing* in dimension $i \in \{1..d\}$ if either

non-positive objective

- φ is non-negative and $\sim_i = \leq$, or
- φ is non-positive and $\sim_i = >$.

Otherwise, φ is *decreasing* in i. Finally, we define the order $\trianglerighteq^{\Psi} \subseteq \overline{\mathbb{N}}^d \times \overline{\mathbb{N}}^d$ for reward limits $\boldsymbol{b}, \boldsymbol{c} \in \overline{\mathbb{N}}^d$, where

$$\boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c} \quad \text{iff} \quad \forall\, i \in \{1..d\} \colon \boldsymbol{b}(i) \trianglerighteq^{\Psi}_i \boldsymbol{c}(i)\,, \quad \text{with}$$

$$\trianglerighteq^{\Psi}_i \;:=\; \begin{cases} \leq & \text{if } \varphi \text{ is increasing in } i \text{ and } \bowtie \in \{>, \geq\} \\ & \text{or } \varphi \text{ is decreasing in } i \text{ and } \bowtie \in \{<, \leq\} \\ \geq & \text{if } \varphi \text{ is increasing in } i \text{ and } \bowtie \in \{<, \leq\} \\ & \text{or } \varphi \text{ is decreasing in } i \text{ and } \bowtie \in \{>, \geq\}. \end{cases}$$

Intuitively, $\boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c}$ means that $\boldsymbol{c}$ specifies "improved" reward limits than $\boldsymbol{b}$ with respect to satisfaction of the quantile. If $\boldsymbol{b}$ is satisfying, then $\boldsymbol{c}$ is also satisfying.

Lemma 7.1 If $\boldsymbol{b} \in Sat(\Psi)$ and $\boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c}$, then $\boldsymbol{c} \in Sat(\Psi)$.

Proof. Let $\boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c}$ and assume[a] $\bowtie \in \{>, \geq\}$. Then, for $i \in \{1..d\}$ we have

- $\boldsymbol{b}(i) \leq \boldsymbol{c}(i)$ if φ is increasing in i and

[1] Arithmetic operations and orders for $\overline{\mathbb{N}}$ are as for the extended reals $\overline{\mathbb{R}}$, outlined on page 16.

- $\boldsymbol{b}(i) \geq \boldsymbol{c}(i)$ if φ is decreasing in i.

It follows that $\varphi[\![\boldsymbol{c}]\!](\pi) \geq \varphi[\![\boldsymbol{b}]\!](\pi)$ for all $\pi \in Paths^{M}_{\text{inf}}$. With $\boldsymbol{b} \in Sat(\Psi)$ we get

$$\text{Ex}^{M}_{\max,s_I}(\varphi[\![\boldsymbol{c}]\!]) \;\geq\; \text{Ex}^{M}_{\max,s_I}(\varphi[\![\boldsymbol{b}]\!]) \;\bowtie\; a,$$

i.e., $\boldsymbol{c} \in Sat(\Psi)$. ■

[a] The case $\bowtie \in \{<, \leq\}$ is similar.

closure w.r.t. quantile query

Definition 7.4 (Closure) The *closure* of a set $B \subseteq \overline{\mathbb{N}}^d$ with respect to a quantile query Ψ is given by

$$cl^{\Psi}(B) \;:=\; \{\boldsymbol{c} \in \mathbb{N}^d \mid \exists \boldsymbol{b} \in B : \boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c}\},$$

The closure is restricted to natural numbers—not including ∞.

Lemma 7.2 $Sat(\Psi) = cl^{\Psi}(Sat(\Psi))$.

Proof. The "$\subseteq$" direction follows since $\trianglerighteq^{\Psi}$ is reflexive. Now assume $\boldsymbol{c} \in cl^{\Psi}(Sat(\Psi))$. By definition of closure, there is $\boldsymbol{b} \in Sat(\Psi)$ with $\boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c}$. $\boldsymbol{c} \in Sat(\Psi)$ follows from Lemma 7.1. ■

From Lemma 7.2, we get that the set of satisfying reward limits can always be characterized by *some* set $B \subseteq Sat(\Psi)$ with $cl^{\Psi}(B) = cl^{\Psi}(Sat(\Psi))$. The smallest such set is called the *generator* of $Sat(\Psi)$.

generator

Definition 7.5 (Generator) The *generator* $Gen^{\Psi}(B)$ of $B \subseteq \overline{\mathbb{N}}^d$ is the smallest set $G \subseteq \overline{\mathbb{N}}^d$ such that $cl^{\Psi}(G) = cl^{\Psi}(B)$, i.e., $cl^{\Psi}(G') \neq cl^{\Psi}(B)$ for every proper subset $G' \subsetneq G$.

We now show that each set $B \subseteq \overline{\mathbb{N}}^d$ has exactly one generator. For this, we make use of the following well-known lemma (see, e.g., [FFSS11]).

Lemma 7.3 (Dickson's lemma) Every *infinite* subset $B \subseteq \overline{\mathbb{N}}^d$ contains $\boldsymbol{b}, \boldsymbol{c} \in B$ with $\boldsymbol{b} \gneq \boldsymbol{c}$, i.e., $\boldsymbol{b} \neq \boldsymbol{c}$ and $\boldsymbol{b} \geq \boldsymbol{c}$.

Lemma 7.4 If the generator of some $B \subseteq \overline{\mathbb{N}}^d$ exists, it is unique.

Proof. For the sake of contradiction, assume $B \subseteq \overline{\mathbb{N}}^d$ has two generators G_1, G_2 with $G_1 \neq G_2$. Without loss of generality assume there is $\boldsymbol{b}_1 \in G_1 \setminus G_2$. We have $\boldsymbol{b}_1 \in cl^{\Psi}(G_1) = cl^{\Psi}(G_2)$, thus there is $\boldsymbol{b}_2 \in G_2$ with $\boldsymbol{b}_2 \trianglerighteq^{\Psi} \boldsymbol{b}_1$. Similarly, $\boldsymbol{b}_2 \in cl^{\Psi}(G_1)$ implies $\boldsymbol{b}_3 \trianglerighteq^{\Psi} \boldsymbol{b}_2$ for some $\boldsymbol{b}_3 \in G_1$. Transitivity of $\trianglerighteq^{\Psi}$ yields $\boldsymbol{b}_3 \trianglerighteq^{\Psi} \boldsymbol{b}_1$. It follows

that $cl^{\Psi}(G_1 \setminus \{\boldsymbol{b}_1\}) = cl^{\Psi}(G_1)$ for the proper subset $G_1 \setminus \{\boldsymbol{b}_1\}$ of G_1. This contradicts our assumption that G_1 is a generator of B. ■

Lemma 7.5 Every $B \subseteq \overline{\mathbb{N}}^d$ has a generator.

Proof. We first observe that the generator of B is not necessary a subset of B (or $cl^{\Psi}(B)$). For example, with a query Ψ such that $d = 1$ and $\trianglerighteq^{\Psi} = \trianglerighteq^{\Psi}_1 = \geq$, we have $Gen^{\Psi}(\mathbb{N}) = \{\infty\}$. We thus consider the *extended* closure of a set B. For $b \in \mathbb{N}$, let $\mathcal{N}(b) \coloneqq \big\{\{b\}\big\}$ and let $\mathcal{N}(\infty) \coloneqq \bigcup_{k\in\mathbb{N}} \{k' \in \mathbb{N} \mid k' \geq k\}$. We lift the notation to $\boldsymbol{b} \in \overline{\mathbb{N}}^d$, where $\mathcal{N}(\boldsymbol{b}) \coloneqq \bigtimes_{i=1}^{d} \mathcal{N}(\boldsymbol{b}(i))$—yielding a (discretized) notion of neighborhoods of $\boldsymbol{b}$ (cf. Definition 2.2 on page 17). We now set

$$cl^{\Psi}_{\infty}(B) \;\coloneqq\; \Big\{\boldsymbol{b} \in \overline{\mathbb{N}}^d \;\Big|\; \forall \langle N_1, \dots, N_d\rangle \in \mathcal{N}(\boldsymbol{b}) \colon\; \exists \boldsymbol{c} \in cl^{\Psi}(B) \colon\; \boldsymbol{c} \in \textstyle\bigtimes_{i=1}^{d} N_i\Big\}.$$

We have $cl^{\Psi}(B) \subseteq cl^{\Psi}_{\infty}(B)$. Intuitively, $cl^{\Psi}_{\infty}(B)$ also contains the boundary of $cl^{\Psi}(B)$, which yields $Gen^{\Psi}(B) \subseteq cl^{\Psi}_{\infty}(B)$ for every Ψ. More specifically, we show that $Gen^{\Psi}(B) = G$ for

$$G \;\coloneqq\; \Big\{\boldsymbol{b} \in cl^{\Psi}_{\infty}(B) \;\Big|\; \big(\forall \boldsymbol{c} \in \overline{\mathbb{N}}^d \colon\; \boldsymbol{c} \trianglerighteq^{\Psi} \boldsymbol{b} \text{ implies } \boldsymbol{c} = \boldsymbol{b} \text{ or } \boldsymbol{c} \notin cl^{\Psi}_{\infty}(B)\big) \text{ and } \big(\forall i \in \{1..d\} \colon\; \boldsymbol{b}(i) = \infty \text{ implies } \trianglerighteq^{\Psi}_i = \geq\big)\Big\}.$$

Intuitively, $\boldsymbol{b} \in cl^{\Psi}_{\infty}(B)$ is contained in G, if $\boldsymbol{b}$ is not *dominated* (w.r.t. $\trianglerighteq^{\Psi}$) by any other $\boldsymbol{c} \in cl^{\Psi}_{\infty}(B)$. Furthermore, if $\boldsymbol{b}(i) = \infty$ for $i \in \{1..d\}$ with $\trianglerighteq^{\Psi}_i = \leq$, there are no reward limits $\boldsymbol{c} \in \mathbb{N}^d$ with $\boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c}$, i.e., $cl^{\Psi}(\{\boldsymbol{b}\}) = \emptyset$. We thus also exclude those vectors from G. We now conclude the proof by showing

1. $cl^{\Psi}(G) \subseteq cl^{\Psi}(\boldsymbol{b})$,
2. $cl^{\Psi}(G) \supseteq cl^{\Psi}(\boldsymbol{b})$,
3. $\forall G' \subsetneq G \colon\; cl^{\Psi}(G') \neq cl^{\Psi}(G)$, and
4. $\forall \boldsymbol{c} \in \mathbb{N}^d \colon\; |\{\boldsymbol{b} \in G \mid \boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c}\}| < \infty$, which is used in the proof of Item 3.

Item 1 | Let $\boldsymbol{c} \in cl^{\Psi}(G) \subseteq \mathbb{N}^d$ i.e., there is $\boldsymbol{b} \in G$ with $\boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c}$. Consider neighborhood $\langle N_1, \dots, N_d\rangle \in \mathcal{N}(\boldsymbol{b})$, where

$$\forall i \in \{1..d\} \colon\quad N_i \;=\; \begin{cases} \{\boldsymbol{b}(i)\} & \text{if } \boldsymbol{b}(i) \in \mathbb{N} \\ \{c \in \mathbb{N} \mid c \geq \boldsymbol{c}(i)\} & \text{if } \boldsymbol{b}(i) = \infty. \end{cases}$$

Since $\boldsymbol{b} \in cl^{\Psi}_{\infty}(B)$ by definition of G, there is $\boldsymbol{b}' \in cl^{\Psi}(B)$ in this neighborhood, i.e., $\boldsymbol{b}' \in \bigtimes_{i=1}^{d} N_i$. We get $\boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{b}' \trianglerighteq^{\Psi} \boldsymbol{c}$, yielding $\boldsymbol{c} \in cl^{\Psi}(B)$.

Item 2 | Let $\boldsymbol{c} \in cl^{\Psi}(B)$ i.e., there is $\boldsymbol{b} = \langle b_1, \ldots, b_d \rangle \in B$ with $\boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c}$. We construct $\boldsymbol{b}' = \langle b'_1, \ldots, b'_d \rangle \in G$ as follows. For $i \in \{1..d\}$, we set

$$b'_i \;=\; \operatorname{opt}\Big\{ b \in \overline{\mathbb{N}} \;\Big|\; \langle b'_1, \ldots, b'_{i-1}, b, b_{i+1}, \ldots, b_d \rangle \in cl^{\Psi}_{\infty}(B) \Big\}$$

with opt = min if $\trianglerighteq^{\Psi}_i = \leq$ and opt = max if $\trianglerighteq^{\Psi}_i = \geq$. The maxima above exist (and might be ∞) due to the consideration of the *extended* closure $cl^{\Psi}_{\infty}(B)$. We have $\boldsymbol{b}' \in cl^{\Psi}_{\infty}(B)$ and $\boldsymbol{b}'' \notin cl^{\Psi}_{\infty}(B)$ for any $\boldsymbol{b}'' \in \overline{\mathbb{N}}^d \setminus \{\boldsymbol{b}'\}$ with $\boldsymbol{b}'' \trianglerighteq^{\Psi} \boldsymbol{b}'$. Moreover, $\boldsymbol{b}' \trianglerighteq^{\Psi} \boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c} \in \mathbb{N}^d$. Hence, $\boldsymbol{b}(i) \leq \boldsymbol{c}(i) \in \mathbb{N}$ for $i \in \{1..d\}$ with $\trianglerighteq^{\Psi}_i = \leq$—yielding $\boldsymbol{b}' \in G$ and $\boldsymbol{c} \in cl^{\Psi}(G)$.

Item 3 | Let $\boldsymbol{c} \in G$. We show $cl^{\Psi}(G) \neq cl^{\Psi}(G \setminus \{\boldsymbol{c}\})$. Let $\boldsymbol{c}_0 \in \mathbb{N}^d$ be given such that for $i \in \{1..d\}$:

$$\boldsymbol{c}_0(i) \;\coloneqq\; \begin{cases} \boldsymbol{c}(i) & \text{if } \boldsymbol{c}(i) \in \mathbb{N} \\ 0 & \text{if } \boldsymbol{c}(i) = \infty. \end{cases}$$

We have $\boldsymbol{c} \trianglerighteq^{\Psi} \boldsymbol{c}_0$ and thus $\boldsymbol{c}_0 \in cl^{\Psi}(G)$.
Now consider $G' \coloneqq \big\{ \boldsymbol{b} \in G \;\big|\; \boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c}_0 \big\}$. The proof of Item 4 below shows that G' is finite. Let $\boldsymbol{b} \in \mathbb{N}^d$, where for $i \in \{1..d\}$:

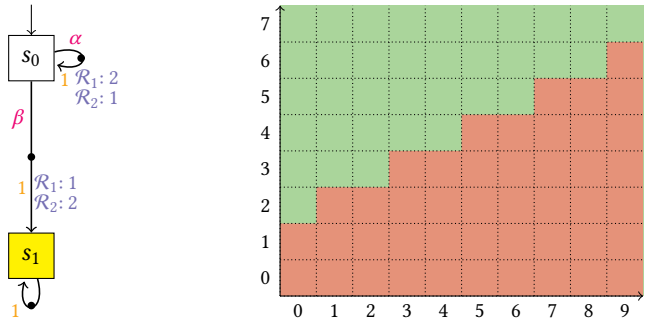
$$\boldsymbol{b}(i) \;\coloneqq\; \begin{cases} \boldsymbol{c}(i) & \text{if } \boldsymbol{c}(i) \in \mathbb{N} \\ \max\{\boldsymbol{b}'(i) + 1 \mid \boldsymbol{b}' \in G_{\boldsymbol{b}} \text{ and } \boldsymbol{b}'(i) \in \mathbb{N}\} & \text{if } \boldsymbol{c}(i) = \infty. \end{cases}$$

We get $\boldsymbol{c} \trianglerighteq^{\Psi} \boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c}_0$ which—in particular—yields $\boldsymbol{b} \in cl^{\Psi}(G)$. We now argue that $\boldsymbol{b} \notin cl^{\Psi}(G \setminus \{\boldsymbol{c}\})$. For the sake of contradiction, assume there is $\boldsymbol{c}' \in G \setminus \{\boldsymbol{c}\}$ with $\boldsymbol{c}' \trianglerighteq^{\Psi} \boldsymbol{b}$. Then also $\boldsymbol{c}' \trianglerighteq^{\Psi} \boldsymbol{c}_0$, i.e., $\boldsymbol{c}' \in G'$. By definition of $\boldsymbol{b}$, we thus get that $\forall\, i \in \{1..d\}\colon\ \boldsymbol{c}'(i) = \infty$ iff $\boldsymbol{c}(i) = \infty$ and therefore $\boldsymbol{c}' \trianglerighteq^{\Psi} \boldsymbol{c}$—contradicting that $\boldsymbol{c}, \boldsymbol{c}' \in G$.

Item 4 | We show that for a fixed $\boldsymbol{c} \in \mathbb{N}^d$ the set $G' = \big\{ \boldsymbol{b} \in G \;\big|\; \boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c} \big\}$ is finite. Consider the partition $G' = \biguplus_{\mathcal{L} \subseteq \{1..d\}} G_{\mathcal{L}}$, with

$$G_{\mathcal{L}} \;\coloneqq\; \big\{ \boldsymbol{b} \in G' \;\big|\; \forall\, i \in \{1..d\}\colon\ \boldsymbol{b}(i) = \infty \text{ iff } i \in \mathcal{L} \big\}.$$

Since there are $2^d < \infty$ many subsets $\mathcal{L} \subseteq \{1..d\}$, it suffices to show that $G_{\mathcal{L}}$ is finite for arbitrary, fixed $\mathcal{L} \subseteq \{1..d\}$.



(a) MDP M_2 (b) $Sat(\Psi_2)$ in green and $Sat(\overline{\Psi_2})$ in red

Figure 7.2: Example MDPs and satisfying reward limits (cf. Example 7.2)

Towards a contradiction, assume that $G_{\mathcal{L}}$ is infinite. For $\boldsymbol{b} \in G_{\mathcal{L}}$ let $[\boldsymbol{b}]_{\leq} \in \mathbb{N}^{|\mathcal{L}_{\leq}|}$ be the restriction of the vector $\boldsymbol{b}$ to those entries with index in $\mathcal{L}_{\leq} := \{i \in \{1..d\} \mid \trianglerighteq_i^{\Psi} = \leq\}$. We define $[\boldsymbol{b}]_{\geq}$ similarly. Observe $[\boldsymbol{b}]_{\leq} \leq [\boldsymbol{c}]_{\leq}$ for all $\boldsymbol{b} \in G_{\mathcal{L}}$ and thus the set $\left\{[\boldsymbol{b}]_{\leq} \,\middle|\, \boldsymbol{b} \in G_{\mathcal{L}}\right\}$ is finite. Therefore, at least one $[\boldsymbol{b}]_{\leq}$ must occur infinitely often, i.e. there is an infinite sequence $\boldsymbol{b}_1\boldsymbol{b}_2\boldsymbol{b}_3\ldots$ with $\boldsymbol{b}_j \in G_{\mathcal{L}}$, $\boldsymbol{b}_j \neq \boldsymbol{b}_k$, and $[\boldsymbol{b}_j]_{\leq} = [\boldsymbol{b}_k]_{\leq}$ for all $j,k \in \mathbb{N}$ with $j \neq k$. Two distinct vectors $\boldsymbol{b}_j, \boldsymbol{b}_k$ from this sequence only differ at entries $i \in \{1..d\}$ with $\trianglerighteq_i^{\Psi} = \geq$. We apply Lemma 7.3 to the infinite set $\left\{[\boldsymbol{b}_1]_{\geq}, [\boldsymbol{b}_2]_{\geq}, [\boldsymbol{b}_3]_{\geq}, \ldots\right\}$, which yields that there are $j,k \in \mathbb{N}$ with $[\boldsymbol{b}_j]_{\geq} \gtrsim [\boldsymbol{b}_k]_{\geq}$. We get that $\boldsymbol{b}_j \trianglerighteq^{\Psi} \boldsymbol{b}_k$ which contradicts the assumption that both $\boldsymbol{b}_j$ and $\boldsymbol{b}_k$ are contained in G. ■

We also refer to $Gen^{\Psi}(Sat(\Psi))$ as *the* generator of quantile query Ψ. A generator is called *natural* if it only contains points in $\mathbb{N}^d$. The following example shows that quantile queries can have (non-)natural (in-)finite generators.

natural generator

Example 7.2 Query Ψ_1 from Example 7.1 has a finite natural generator:

$$Gen^{\Psi_1}(Sat(\Psi_1)) = \{\langle 2,4\rangle, \langle 3,3\rangle, \langle 4,2\rangle, \langle 5,1\rangle, \langle 6,0\rangle\}.$$

The generator of the complementary query $\overline{\Psi_1}$ is still finite but not natural:

$$Gen^{\overline{\Psi_1}}(Sat(\overline{\Psi_1})) = \{\langle 1,\infty\rangle, \langle 2,3\rangle, \langle 3,2\rangle, \langle 4,1\rangle, \langle 5,0\rangle\}.$$

Next, consider the MDP M_2 from Figure 7.2a and the quantile query

$$\Psi_2 = Qu_{s_0}^{M_2}\left(bnd_{\mathcal{G}}^{\mathcal{B}}(1,\{1\}) > 0.5\right), \text{ where}$$
$$\mathcal{B} = \left\langle \wr \mathcal{R}_1 > ? \wr, \wr \mathcal{R}_2 \leq ? \wr \right\rangle \quad \text{and} \quad \mathcal{G} = \left\langle \langle \{1,2\}, \{s_1\} \rangle \right\rangle.$$

Taking action α at state s_0 for $n \in \mathbb{N}$ times while reaching s_1 afterwards results in an accumulated reward of $2 \cdot n + 1$ for $\mathcal{R}_1$ and an accumulated reward of $n + 2$ for $\mathcal{R}_2$. Thus, the reward bounds $\wr \mathcal{R}_1 > 2 \cdot n \wr$ and $\wr \mathcal{R}_2 \leq n + 2 \wr$ are active upon reaching s_1. This yields the satisfying reward limits as shown in Figure 7.2b. Ψ_2 does not have a finite generator:

$$Gen^{\Psi_2}(Sat(\Psi_2)) = \{\langle 2 \cdot n, 2 + n \rangle \mid n \in \mathbb{N}\}.$$

The rest of this chapter deals with the following problem.

Problem 7.1: Multi-dimensional Quantile Problem

Input Quantile query $\Psi = Qu_{s_I}^{M}(\varphi \bowtie a)$ for MDP M satisfying Assumption 7.1
Output Generator $Gen^{\Psi}(Sat(\Psi))$

7.2 Computing Finite Natural Generators

Given a quantile query Ψ as in Problem 7.1, we now present an algorithm to compute its generator—i.e., a representation of the set of satisfying reward limits. For now, we assume that both Ψ and its complementary query $\overline{\Psi}$ each have a finite and natural generator. In Section 7.3, we present suitable preprocessing steps to lift this algorithm to a broader class of quantiles.

Our approach is sketched in Algorithm 7.1. It analyzes epoch MDPs successively, similar to Algorithm 6.1 on page 194. However, the sequence of analyzed reward epochs is extended in an on-the-fly manner—considering more and more candidate epochs $\boldsymbol{e}$ with an increasing maximum entry $k = \max\{\boldsymbol{e}(i) \mid 1 \leq i \leq d\}$. Already processed epochs are stored in a set $E_{\checkmark}$. We use this set in Line 11 to ensure that each epoch MDP is analyzed at most once. Whenever the algorithm finds an epoch $\boldsymbol{e}$ that yields valid reward limits (i.e., $\boldsymbol{e} \in \mathbb{N}^d$), the epoch is added to either B^+ or B^-, depending on whether the probability threshold is satisfied in $\boldsymbol{e}$ or not (Lines 15 to 17). In this way, B^+ and B^- gather satisfying reward limits for the quantile query Ψ and its complementary query $\overline{\Psi}$, respectively. In Line 18, we update the set E_{cand} of remaining candidate epochs to avoid analyzing epochs which are already known to be satisfying for either Ψ or $\overline{\Psi}$. The procedure stops as soon as we find a $k \in \mathbb{N}$ such that all new

Input: d-dimensional quantile $\Psi = Qu^{M}_{s_I}(\varphi \bowtie a)$ for MDP M
Output: The (finite and natural) generator of Ψ

```
1  function solveQuantile(Ψ)
       // Using un(M) = ⟨S_un, ...⟩ and rewards R^un for φ (cf. Section 6.1.3)
2      B⁺ ← ∅; B⁻ ← ∅  // Collect satisfying reward limits for Ψ and Ψ̄
3      E_✓ ← ∅ // Collect epochs processed so far
4      Arbitrarily initialize function f : S_un → ℝ
5      for k = 0, 1, 2, ... do
6          E_cand ← {e ∈ {1..k}^d | ∃ i : e(i) = k} \ (cl^Ψ(B⁺) ∪ cl^Ψ̄(B⁻))
7          if E_cand = ∅ then break                    // No more epoch candidates
8          while E_cand ≠ ∅ do      // Analyze epoch candidates as in Algorithm 6.1
9              Get proper epoch sequence 𝔈 = e_1 ... e_n with e_n ∈ E_cand
10             for e = e_1, ..., e_n do
11                 if e ∈ E_✓ then continue              // Skip already processed e
12                 E_✓ ← E_✓ ∪ {e}
13                 Build epoch MDP ep(M, e) = ⟨S_e, ...⟩ and ⟨R^un, f⟩_e
14                 for ŝ ∈ S_e ∩ S_un do f(ŝ) ← Ex^{ep(M,e)}_{max,ŝ}(tot(⟨R^un, f⟩_e))
15                 if e ∈ ℕ^d then                       // ⊘ does not occur in e
16                     if f(⟨s_I, e, sat^B_G(e, s_I)⟩) ⋈ a then B⁺ ← B⁺ ∪ {e}
17                     else B⁻ ← B⁻ ∪ {e}
18             E_cand ← E_cand \ (cl^Ψ(B⁺) ∪ cl^Ψ̄(B⁻))
19     return Gen^Ψ(B⁺)
```

Algorithm 7.1: Multi-dimensional quantile computation

candidate epochs with maximum entry k are already in $cl^{\Psi}(B^+) \cup cl^{\overline{\Psi}}(B^-)$.

> **Remark 7.2** To keep the presentation simple, Algorithm 7.1 omits technical details on non-finite as well as approximative expected values. For the evaluation of reward-bounded objectives, these details have been discussed in Section 6.2.4. We briefly describe how this can be lifted to the computation of quantiles.
> If $obj = bnd^{\mathcal{B}}(\mathcal{R}', I)$, the expected value computed in Line 14 might be either $+\infty$ or $-\infty$. Due to Item (iii) of Assumption 7.1, we only have to deal with either the $+\infty$ case or the $-\infty$ case. As in Algorithm 6.2 on page 197, we can incorporate

the corresponding case by also maintaining a set

$$S^{\infty} = \left\{ \hat{s} = \langle s, \boldsymbol{e}, \mathfrak{g} \rangle \;\middle|\; \boldsymbol{e} \in E_{\checkmark} \text{ and } \mathrm{Ex}^{ep(M,\boldsymbol{e})}_{\max,\hat{s}}\left(tot(\langle \mathcal{R}^{un}, f \rangle_{\boldsymbol{e}})\right) = \pm\infty \right\}.$$

If the expected values in Line 14 are only computed approximatively (e.g., when using sound value iteration), the approximation error propagates through the epochs. Our results from Section 6.2.4 yield sound lower- and upper bounds on the expected value in a given epoch $\boldsymbol{e}$. These bounds can be used to decide if $\boldsymbol{e}$ can be added to B^+, B^-, or neither (when the threshold $\bowtie a$ of the query Ψ can not conclusively be decided based on the available bounds for $\boldsymbol{e}$). A pragmatic approach to deal with the latter case is to restart the algorithm with improved accuracy.

Lemma 7.6 During the execution of Algorithm 7.1 the following invariant holds:

$$cl^{\Psi}(B^+) \subseteq cl^{\Psi}(Sat(\Psi)) \quad \text{and} \quad cl^{\overline{\Psi}}(B^-) \subseteq cl^{\overline{\Psi}}(Sat(\overline{\Psi})).$$

Proof. As in the proof of Theorem 6.12 on page 194, it can be shown that in Line 14, the algorithm computes $\mathrm{Ex}^{un(M)}_{\max,\hat{s}}\left(tot(\mathcal{R}^{un})\right)$ for the unfolding MDP $un(M)$. With Theorem 6.3 on page 182 (applied to a single objective φ) and using $\hat{s} = \langle s_I, \boldsymbol{e}, sat^{\mathcal{B}}_{\mathcal{G}}(\boldsymbol{e}, s_I) \rangle$, we get

$$f(\hat{s}) \;=\; \mathrm{Ex}^{un(M)}_{\max,\hat{s}}\left(tot(\mathcal{R}^{un})\right) = \mathrm{Ex}^{M}_{\max,s_I}\left(\varphi[\![\boldsymbol{e}]\!]\right).$$

The algorithm checks whether this expected value meets the threshold $\bowtie a$ given in Ψ and inserts it in either B^+ or B^-, accordingly. Hence, B^+ (B^-) only contains satisfying reward limits for Ψ ($\overline{\Psi}$). ∎

Lemma 7.7 If Algorithm 7.1 terminates, it returns the generator of Ψ.

Proof. We first show that upon termination all reward limits $\boldsymbol{b} \in \mathbb{N}^d$ are contained in either $cl^{\Psi}(B^+)$ or $cl^{\overline{\Psi}}(B^-)$. The proof is based on the following observation: when the algorithm terminates, there is a $k \in \mathbb{N}$ such that for reward limits $\boldsymbol{b} \in \mathbb{N}^d$ we have

$$\max_i(\boldsymbol{b}(i)) = k \text{ implies } \boldsymbol{b} \in cl^{\Psi}(B^+) \cup cl^{\overline{\Psi}}(B^-) \text{ and } \boldsymbol{b} \notin B^+ \cup B^-.$$

In particular, for $\boldsymbol{b} \in B^+ \cup B^-$ we have $\max_i(\boldsymbol{b}(i)) < k$. Consider the smallest such k (i.e., the value of the variable k upon termination). For arbitrary $\boldsymbol{b} \in \mathbb{N}^d$ we distinguish two cases.

- If $\max_i(\boldsymbol{b}(i)) < k$, $\boldsymbol{b}$ has already been considered as a candidate epoch $\boldsymbol{b} = \boldsymbol{e}_n$ in

Line 9, therefore $\boldsymbol{b} \in cl^{\Psi}(B^+) \cup cl^{\overline{\Psi}}(B^-)$.

- Otherwise, $\max_i(\boldsymbol{b}(i)) \geq b$. Let $\boldsymbol{b}' \in \mathbb{N}^d$ be given such that $\boldsymbol{b}'(i) = \min(\boldsymbol{b}(i), k)$ for all i. Since $\max_i(\boldsymbol{b}'(i)) = k$, we have $\boldsymbol{b}' \in cl^{\Psi}(B^+) \cup cl^{\overline{\Psi}}(B^-)$. Assume $\boldsymbol{b}' \in cl^{\Psi}(B^+)$ (the case where $\boldsymbol{b}' \in cl^{\overline{\Psi}}(B^-)$ is completely analogous). According to the definition of closure, there has to be a $\boldsymbol{b}'' \in B^+$ with $\boldsymbol{b}'' \trianglerighteq^{\Psi} \boldsymbol{b}'$, where $\trianglerighteq^{\Psi}$ is as on page 222. We also have $\boldsymbol{b}' \trianglerighteq^{\Psi} \boldsymbol{b}$ since for all $i \in \{1..d\}$ one of the following cases holds:
  - If $\boldsymbol{b}(i) < k$, we have $\boldsymbol{b}(i) = \boldsymbol{b}'(i)$.
  - If $\boldsymbol{b}(i) \geq k$, we have $\boldsymbol{b}'(i) = k$ and $k > \boldsymbol{b}''(i)$. The latter follows from $\boldsymbol{b}'' \in B^+$, yielding $\max_j(\boldsymbol{b}''(j)) < k$. As $\boldsymbol{b}'' \trianglerighteq^{\Psi} \boldsymbol{b}'$ and $\boldsymbol{b}''(i) < \boldsymbol{b}'(i)$, we have to be in a case where $\trianglerighteq^i = \leq$. With $\boldsymbol{b}'(i) \leq k \leq \boldsymbol{b}(i)$ we get $\boldsymbol{b}'(i) \trianglerighteq^i \boldsymbol{b}(i)$.

  To summarize this case, we have $\boldsymbol{b}'' \trianglerighteq^{\Psi} \boldsymbol{b}'$ and $\boldsymbol{b}' \trianglerighteq^{\Psi} \boldsymbol{b}$. By transitivity of $\trianglerighteq^{\Psi}$, we get $\boldsymbol{b}'' \trianglerighteq^{\Psi} \boldsymbol{b}$ and thus $\boldsymbol{b} \in cl^{\Psi}(\{\boldsymbol{b}''\}) \subseteq cl^{\Psi}(B^+)$.

It follows that upon termination we have $\boldsymbol{b} \in cl^{\Psi}(B^+) \cup cl^{\Psi}(B^-)$ for any $\boldsymbol{b} \in \mathbb{N}^d$. Next, we show that $cl^{\Psi}(B^+) = cl^{\Psi}(Sat(\Psi))$. Lemma 7.7 follows as sets with the same closure must have the same unique generator. $cl^{\Psi}(B^+) \subseteq cl^{\Psi}(Sat(\Psi))$ follows already from the invariant established in Lemma 7.6. For the other direction, let $\boldsymbol{b} \in cl^{\Psi}(Sat(\Psi))$, i.e., there exists $\boldsymbol{b}' \in Sat(\Psi)$ with $\boldsymbol{b}' \trianglerighteq^{\Psi} \boldsymbol{b}$. As $\boldsymbol{b}'$ is satisfying for Ψ, $\boldsymbol{b}' \notin cl^{\overline{\Psi}}(B^-) \subseteq cl^{\overline{\Psi}}(Sat(\overline{\Psi}))$ holds. With $\boldsymbol{b}' \in \mathbb{N}^d \subseteq cl^{\Psi}(B^+) \cup cl^{\overline{\Psi}}(B^-)$ (as shown above), this yields $\boldsymbol{b}' \in cl^{\Psi}(B^+)$. It follows that there is $\boldsymbol{b}'' \in B^+$ with $\boldsymbol{b}'' \trianglerighteq^{\Psi} \boldsymbol{b}' \trianglerighteq^{\Psi} \boldsymbol{b}$. Hence, $\boldsymbol{b} \in cl^{\Psi}(B^+)$. ■

Intuitively, Algorithm 7.1 terminates when the boundary between $Sat(\Psi)$ and $Sat(\overline{\Psi})$ is fully contained in $B^+ \cup B^-$. However, since $B^+, B^- \subseteq \mathbb{N}^d$, termination after finitely many iterations can only occur if both Ψ and $\overline{\Psi}$ have finite natural generators.

Lemma 7.8 Algorithm 7.1 terminates if Ψ and $\overline{\Psi}$ have finite natural generators.

Proof. Eventually all reward limits of the finite set $Gen^{\Psi}(Sat(\Psi)) \subseteq \mathbb{N}^d$ are considered in Line 15 and inserted in B^+, yielding $cl^{\Psi}(B^+) = cl^{\Psi}(Sat(\Psi))$. Similarly, $cl^{\overline{\Psi}}(B^-) = cl^{\overline{\Psi}}(Sat(\overline{\Psi}))$ holds eventually. Hence, all $\boldsymbol{b} \in \mathbb{N}^d$ are contained in $cl^{\Psi}(B^+) \cup cl^{\overline{\Psi}}(B^-)$, which leads to termination of the procedure. ■

Lemmas 7.7 and 7.8 yield correctness.

Theorem 7.9 If Ψ and $\overline{\Psi}$ have finite natural generators, then Algorithm 7.1 returns with the generator of Ψ.

Lemma 7.10 If Algorithm 7.1 terminates, it analyzes at most $(k_{\max}+2)^d$ epoch MDPs, where

$$k_{\max} = \max\left\{\boldsymbol{b}(i) \;\middle|\; i \in \{1..d\} \text{ and } \boldsymbol{b} \in Gen^{\Psi}(Sat(\Psi)) \cup Gen^{\overline{\Psi}}(Sat(\overline{\Psi}))\right\}.$$

Proof. Upon termination (i.e., $E_{cand} = \emptyset$ in Line 7), variable k has value $k_{\max}+1$. Therefore, all epochs analyzed so far are contained in the set $\left(\{0..k_{\max}\} \cup \{\oslash\}\right)^d$ which has cardinality $(k_{\max}+2)^d$. ∎

Remark 7.3 Algorithm 7.1 can be extended to quantiles over mixtures of indefinite and definite reward bounds, i.e., d-dimensional quantiles of the form $Qu^M_{s_I}(\varphi \bowtie a)$ with $\varphi \in \left\{bnd^{\mathcal{B}}(\mathcal{R}', \mathcal{I}),\ bnd^{\mathcal{B}}_{\mathcal{G}}(c, \mathcal{J})\right\}$, such that

$$\mathcal{B} = \left\langle \wr\mathcal{R}_1 \sim_1 ?\wr, \ldots, \wr\mathcal{R}_d \sim_d ?\wr, \wr\mathcal{R}_{d+1} \sim_{d+1} b_{d+1}\wr, \ldots, \wr\mathcal{R}_{d+d'} \sim_{d+d'} b_{d+d'}\wr \right\rangle.$$

This can be achieved by considering a proper epoch sequence $\mathfrak{E} = \boldsymbol{e}'_1 \ldots \boldsymbol{e}'_n$ in Line 9 ending with

$$\boldsymbol{e}'_n = \langle \boldsymbol{e}_n(1), \ldots, \boldsymbol{e}_n(d), b_{d+1}, \ldots, b_{d+d'} \rangle$$

for the considered candidate $\boldsymbol{e}_n \in E_{cand}$.

Finite natural generators are only required for Lemma 7.8, i.e., to guarantee termination of the algorithm. For arbitrary queries, the algorithm still correctly identifies the satisfying reward limits (due to the invariant from Lemma 7.6) but might not terminate. In the next section, we consider a class of quantiles with finite but potentially non-natural generators. We provide an extension of our procedure to correctly handle such queries as well. A practical way to overcome the issue of non-termination of Algorithm 7.1 for *arbitrary* quantiles is to restrict the search to a finite set of reward limits. For example, one can consider a hyper parameter $k_{\max} \in \mathbb{N}$ and stop the algorithm as soon as $k = k_{\max}$.

7.3 Quantiles with Monotone Bounds

We now consider d-dimensional quantile queries $\Psi = Qu^M_{s_I}(\varphi \bowtie a)$, where φ considers indefinite reward bounds of the form

$$\mathcal{B} = \left\langle \wr\mathcal{R}_1 \sim ?\wr, \ldots, \wr\mathcal{R}_d \sim ?\wr \right\rangle,$$

i.e., all indefinite reward bounds $\wr\mathcal{R}_i \sim ?\wr$ consider the same relation $\sim \in \{\leq, >\}$. Under this assumption, the objective φ is either increasing in all dimensions $i \in \{1..d\}$ or

decreasing in all dimensions—depending on whether φ is non-negative or non-positive, and whether $\sim = \leq$ or $\sim = \geq$. Furthermore, the order $\trianglerighteq^{\Psi} \subseteq \overline{\mathbb{N}}^d \times \overline{\mathbb{N}}^d$ on reward limits coincides with either $\leq$ or $\geq$. For clarity of presentation, we assume[2] that $\trianglerighteq^{\Psi} = \geq$ and thus $\trianglerighteq^{\overline{\Psi}} = \leq$. For $\boldsymbol{b}, \boldsymbol{b}' \in \mathbb{N}^d$ we therefore get

$$\boldsymbol{b} \in Sat(\Psi) \quad \text{and} \quad \boldsymbol{b} \geq \boldsymbol{b}' \quad \text{implies} \quad \boldsymbol{b}' \in Sat(\Psi).$$

For single dimensional quantiles, the above assumption is trivially true. For this case, [UB13; BDDK+14] suggest preprocessing steps to check whether $Sat(\Psi) \neq \emptyset$ and $Sat(\overline{\Psi}) \neq \emptyset$ holds before invoking their variant of sequential epoch analysis. The preprocessing steps ensures that the (single-dimensional) sequential approach always finds a $k \in \mathbb{N}$ with $Gen^{\Psi}(Sat(\Psi)) = \{k\}$. We lift these ideas to multi-dimensional quantiles.

The 2-dimensional query $\overline{\Psi_1}$ from Example 7.2 satisfies the above assumptions and shows that the generator can be non-natural in this case. To guarantee termination of Algorithm 7.1, we show that it suffices to initialize the set B^+ in Line 2 to

$$B^+_{\text{init}} \coloneqq \left\{\boldsymbol{b} \in Gen^{\Psi}(Sat(\Psi)) \;\middle|\; \exists\, i \in \{1..d\} \colon \boldsymbol{b}(i) = \infty\right\} = Gen^{\Psi}(Sat(\Psi)) \setminus \mathbb{N}^m.$$

Lemma 7.11 Algorithm 7.1 terminates with the generator of Ψ, if initially $B^+ = B^+_{\text{init}}$.

Proof. We first show finiteness of $Gen^{\Psi}(Sat(\Psi))$ by contradiction. Suppose that Ψ has an infinite generator $G \subseteq \overline{\mathbb{N}}^d$. Due to Lemma 7.3, the infinite set G contains two different elements $\boldsymbol{b}$ and $\boldsymbol{c}$ with $\boldsymbol{b} \geq \boldsymbol{c}$ i.e., $\boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{c}$. Since $\boldsymbol{b} \in G$, we get $\boldsymbol{c} \in cl^{\Psi}(G \setminus \{\boldsymbol{c}\})$ and therefore $cl^{\Psi}(G \setminus \{\boldsymbol{c}\}) = cl^{\Psi}(G)$ which contradicts the assumption that G is a generator. It follows that $Gen^{\Psi}(Sat(\Psi))$ must be finite.
Next, let $G' = Gen^{\overline{\Psi}}(Sat(\overline{\Psi}))$ be the generator of $\overline{\Psi}$. Using a similar argument as above, we can show that G' must be finite. Furthermore, since $\trianglerighteq^{\overline{\Psi}} = \leq$, we get

$$\begin{aligned} cl^{\overline{\Psi}}(G') &= \{\boldsymbol{c} \in \mathbb{N}^d \mid \exists \boldsymbol{b} \in G' : \boldsymbol{b} \leq \boldsymbol{c}\} = \{\boldsymbol{c} \in \mathbb{N}^d \mid \exists \boldsymbol{b} \in G' \cap \mathbb{N}^d : \boldsymbol{b} \leq \boldsymbol{c}\} \\ &= cl^{\overline{\Psi}}(G' \cap \mathbb{N}^d). \end{aligned}$$

Since generators are unique (cf. Lemma 7.4), it holds that $G' = G' \cap \mathbb{N}^d$, i.e., $\overline{\Psi}$ has a natural generator.
It follows that all reward limits in $Gen^{\Psi}(Sat(\Psi)) \setminus B^+_{\text{init}} = Gen^{\Psi}(Sat(\Psi)) \cap \mathbb{N}^m$ and all reward limits in $Gen^{\overline{\Psi}}(Sat(\overline{\Psi}))$ are eventually considered in Line 15 of Algorithm 7.1

[2] If we have $\trianglerighteq^{\Psi} = \leq$ instead, $\trianglerighteq^{\overline{\Psi}} = \geq$ holds for the complementary query $\overline{\Psi}$. In this case the roles of Ψ and $\overline{\Psi}$ can be swapped in the subsequent explanations.

and inserted into either $B^+ \supseteq B^+_{\text{init}}$ or B^-. ∎

The remainder of this section addresses the computation of B^+_{init}. Intuitively, $\boldsymbol{b} \in B^+_{\text{init}}$ means that for $i \in \{1..d\}$ with $\boldsymbol{b}(i) = \infty$, we can instantiate the reward limit to arbitrarily high $b_i \in \mathbb{N}$ without changing satisfaction of the quantile threshold $\bowtie a$. The following lemma characterizes the set B^+_{init} in terms of reward limits that approach infinity.

Lemma 7.12 If either

- $\bowtie \in \{\leq, >\}$ and φ is *increasing* in all dimensions, or
- $\bowtie \in \{<, \geq\}$ and φ is *decreasing* in all dimensions,

it holds that

$$B^+_{\text{init}} = Gen^{\Psi}(\{\boldsymbol{b} \in \overline{\mathbb{N}}^d \setminus \mathbb{N}^d \mid \lim_{k\to\infty} \mathrm{Ex}^M_{\max,s_I}(\varphi[\![\boldsymbol{b}_k]\!]) \bowtie a\}),$$

where for $\boldsymbol{b} \in \overline{\mathbb{N}}$ and $i \in \{1..d\}$ we set

$$\boldsymbol{b}_k(i) := \begin{cases} \boldsymbol{b}(i) & \text{if } \boldsymbol{b}(i) \in \mathbb{N} \\ k & \text{if } \boldsymbol{b}(i) = \infty. \end{cases}$$

Proof. For $\boldsymbol{b} \in \overline{\mathbb{N}}^d \setminus \mathbb{N}^d$ we have $\boldsymbol{b} \geq \boldsymbol{b}_k$, i.e., $\boldsymbol{b} \trianglerighteq^{\Psi} \boldsymbol{b}_k$ for all $k \in \mathbb{N}$. Thus, $\boldsymbol{b} \in B^+_{\text{init}}$ implies $\boldsymbol{b}_k \in Sat(\Psi)$. Intuitively, this reflects a situation in which the reward-bounded objective value always meets the threshold $\bowtie a$ of the quantile query—no matter how the indefinite bounds $⁇\mathcal{R}_i \sim_i ?⁇$ for $i \in \{1..d\}$ with $\boldsymbol{b}(i) = \infty$ are instantiated. On the other hand, if $\boldsymbol{b}$ satisfies $\forall k \in \mathbb{N}\colon \boldsymbol{b}_k \in Sat(\Psi)$, then there must be $\boldsymbol{b}' \in B^+_{\text{init}}$ with $\boldsymbol{b}' \trianglerighteq^{\Psi} \boldsymbol{b}$. We can characterize B^+_{init} as follows:

$$\begin{aligned} B^+_{\text{init}} &= Gen^{\Psi}(\{\boldsymbol{b} \in \overline{\mathbb{N}}^d \setminus \mathbb{N}^d \mid \forall k \in \mathbb{N}\colon \boldsymbol{b}_k \in Sat(\Psi)\}) \\ &= Gen^{\Psi}(\{\boldsymbol{b} \in \overline{\mathbb{N}}^d \setminus \mathbb{N}^d \mid \forall k \in \mathbb{N}\colon \mathrm{Ex}^M_{\max,s_I}(\varphi[\![\boldsymbol{b}_k]\!]) \bowtie a\}). \end{aligned}$$

$\mathrm{Ex}^M_{\max,s_I}(\varphi[\![\boldsymbol{b}_k]\!])$ is monotone increasing or monotone decreasing in $k \in \mathbb{N}$—depending on whether φ is increasing or decreasing. Due the assumptions imposed on $\bowtie$, we may verify the following equivalence that concludes the proof.

$$\forall k \in \mathbb{N}\colon \mathrm{Ex}^M_{\max,s_I}(\varphi[\![\boldsymbol{b}_k]\!]) \bowtie a \quad \text{iff} \quad \lim_{k\to\infty} \mathrm{Ex}^M_{\max,s_I}(\varphi[\![\boldsymbol{b}_k]\!]) \bowtie a.$$ ∎

Remark 7.4 Without the assumptions imposed for $\bowtie$ in Lemma 7.12, the two statements

$$\forall k \in \mathbb{N}\colon\ \mathrm{Ex}^{M}_{\max,s_I}(\varphi[\![\boldsymbol{b}_k]\!]) \bowtie a \quad \text{and} \quad \lim_{k\to\infty} \mathrm{Ex}^{M}_{\max,s_I}(\varphi[\![\boldsymbol{b}_k]\!]) \bowtie a$$

are no longer equivalent. To see this, assume that $\lim_{k\to\infty} \mathrm{Ex}^{M}_{\max,s_I}(\varphi[\![\boldsymbol{b}_k]\!]) = a$. Then, we might need to check whether the limit can be attained, i.e., whether there is a $k \in \mathbb{N}$ such that the reward limits $\boldsymbol{b}_k$ induce the objective value a. To distinguish those two cases, we can use a result from [UB13, Lemma 10] which yields that if $\mathrm{Ex}^{M}_{\max,s_I}(\varphi[\![\boldsymbol{b}_k]\!]) = a$ for *some* $k \in \mathbb{N}$, then the smallest such k satisfies $k \leq |S| \cdot r_{\max}$. Here, $r_{\max}$ is the largest reward assigned to any transition in MDP M.
It thus suffices to consider finitely many reward limits $\boldsymbol{b}_k$ to determine whether $\boldsymbol{b} \in B^{+}_{\text{init}}$. For clarity of presentation, we omit further details on handling this edge case.

Using Lemma 7.12, our approach for computing B^{+}_{init} is to identify the vectors $\boldsymbol{b} \in \overline{\mathbb{N}}^{d} \setminus \mathbb{N}^{d}$ with $\lim_{k\to\infty} \mathrm{Ex}^{M}_{\max,s_I}(\varphi[\![\boldsymbol{b}_k]\!]) \bowtie a$. The idea is to compute for each $i \in \{1..d\}$ the set

$$B^{+}_{\text{init}}{\restriction}i \;:=\; \{\boldsymbol{b} \in B^{+}_{\text{init}} \mid \boldsymbol{b}(i) = \infty\}.$$

Then, B^{+}_{init} is given by $\bigcup_{i=1}^{d} B^{+}_{\text{init}}{\restriction}i$. We now argue that the set $B^{+}_{\text{init}}{\restriction}i$ for $i \in \{1..d\}$ can be obtained by solving a $(d-1)$-dimensional quantile query. More precisely, we specify a quantile query $\Psi{\restriction}i$ that satisfies the following lemma.

Lemma 7.13 Under the assumptions from Lemma 7.12 and using $\Psi{\restriction}i$ as given below, we have

$$\boldsymbol{b} \in B^{+}_{\text{init}}{\restriction}i \quad \text{iff} \quad \boldsymbol{b}_{\setminus i} \in \mathit{Gen}^{\Psi}(\mathit{Sat}(\Psi{\restriction}i)),$$

where $\boldsymbol{b}_{\setminus i} \in \overline{\mathbb{N}}^{d-1}$ is obtained from $\boldsymbol{b} = \langle b_1, \ldots, b_d \rangle \in \overline{\mathbb{N}}^{d}$ by dropping the i^{th} entry b_i of $\boldsymbol{b}$.

The derived query $\Psi{\restriction}i$ depends on whether the original query Ψ considers upper or lower reward bounds. We now distinguish these two cases and justify correctness of Lemma 7.13. In both cases, a more formal treatment of our arguments can also be found in [BDDK+14] for the case of single-dimensional quantiles.

Upper Reward Bounds

If Ψ only considers upper reward bounds (i.e., $\sim\ =\ \leq$), then $\Psi{\restriction}i$ arises from Ψ by *dropping* the i^{th} indefinite reward bound $(\mathcal{R}_i \leq ?)$. Lemma 7.13 holds by applying

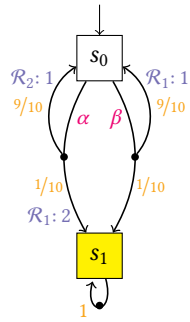


Figure 7.3: Example MDP M_1 as already shown in Figure 7.1a (cf. Example 7.3)

Lemma 7.12 and by noting that if reward limit b_i approaches infinity, the reward bound $⦇\mathcal{R}_i \leq b_i⦈$ is always active. We sketch this idea in the following example.

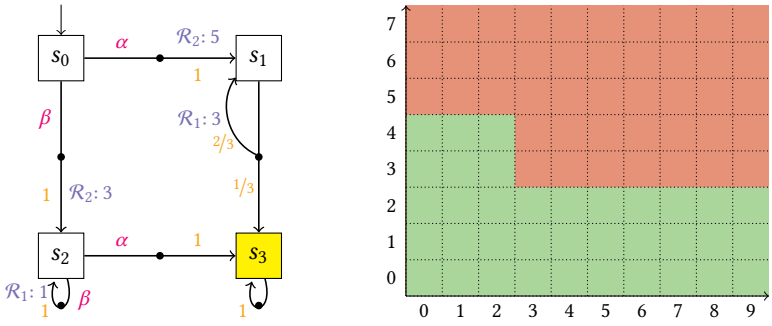
Example 7.3 Consider the 2-dimensional quantile query

$$\Psi_3 = Qu^{M}_{s_I}\big(bnd^{\mathcal{B}}_{\mathcal{G}}(1,\{1\}) \leq 0.5\big)\,, \text{ where}$$
$$\mathcal{B} = \big\langle ⦇\mathcal{R}_1 \leq ?⦈, ⦇\mathcal{R}_2 \leq ?⦈ \big\rangle \quad \text{and} \quad \mathcal{G} = \big\langle \langle \{1,2\}, \{s_1\} \rangle \big\rangle$$

for MDP M_1 from Figure 7.1a, depicted again in Figure 7.3. Ψ_3 coincides with the complementary query $\overline{\Psi_1}$ of Ψ_1 from Example 7.1 which we have already discussed in Example 7.2.
If $\boldsymbol{b} = \langle b_1, b_2 \rangle \in Sat(\Psi_3)$, then the (maximal) probability to reach s_1 while collecting at most b_1 reward for $\mathcal{R}_1$ and b_2 reward for $\mathcal{R}_2$ is less or equal to 0.5. This bounded reachability probability is increasing in the reward limits b_1 and b_2.
To compute the set $B^{+}_{\text{init}}⌈1$, we intuitively assume that b_1 approaches infinity—as argued in Lemma 7.12. Then, the reward bound $⦇\mathcal{R}_1 \leq b_1⦈$ becomes superfluous as it is always active. We can compute $B^{+}_{\text{init}}⌈1$ by solving the 1-dimensional quantile query $\Psi⌈1$ that arises from Ψ by dropping the first reward bound. For $b_2 \in \overline{\mathbb{N}}$ we have

$$\langle \infty, b_2 \rangle \in B^{+}_{\text{init}}⌈1 \quad \text{iff} \quad \langle b_2 \rangle \in Gen^{\Psi⌈1}(Sat(\Psi⌈1)).$$

When the first reward bound $⦇\mathcal{R}_1 \leq ?⦈$ is dropped, a strategy can enforce that s_1 is reached almost surely without collecting any reward for $\mathcal{R}_2$ by taking action β in s_0. Thus, the quantile threshold ≤ 0.5 is never met. We have $Gen^{\Psi⌈1}(Sat(\Psi⌈1)) = \emptyset$ and thus $B^{+}_{\text{init}}⌈1 = \emptyset$.
When dropping the second reward bound $⦇\mathcal{R}_2 \leq ?⦈$, we can only reach s_1 with



(a) MDP M_3 (b) $Sat(\Psi_4)$ in green and $Sat(\overline{\Psi_4})$ in red

Figure 7.4: Example MDP and satisfying reward limits (cf. Example 7.4)

probability below 0.5 iff the first bound is instantiated to $\wr\mathcal{R}_1 \leq b_1\wr$ with $b_1 < 2$. We get $Gen^{\Psi\restriction 2}(Sat(\Psi\restriction 2)) = \{\langle 1\rangle\}$ and thus $B^+_{\text{init}} = B^+_{\text{init}}\restriction 2 = \{\langle 1, \infty\rangle\}$. This coincides with our observations from Examples 7.1 and 7.2.

Lower Reward Bounds

If Ψ only considers lower reward bounds (i.e., $\sim = >$), we additionally require that φ is *non-negative*—intuitively meaning that a strategy strives to meet the reward bounds in order to maximize the objective value[3]. To define the quantile query $\Psi\restriction i$ for Lemma 7.13, we lift ideas from the single-dimensional case [BDDK+14] based on end components (ECs, cf. Definition 2.19 on page 40).

The key observation is that any visit of an EC can be extended to accumulate more rewards within that EC. Doing this can only increase the overall objective value. More precisely, if a path in MDP M reaches an EC $C \in MEC(M)$ in which non-zero reward for an assignment $\mathcal{R}_i$ can be collected, a strategy for M can choose to stay inside C to collect any arbitrary amount of reward for $\mathcal{R}_i$, i.e., for any $b_i \in \mathbb{N}$, the strategy can enforce that the reward bound $\wr\mathcal{R}_i > b_i\wr$ becomes active at some point. This can only have an increasing effect on the overall objective value since φ is non-negative and there are no upper reward bounds involved. On the other hand, if the reward limit b_i becomes arbitrarily large, the probability that a reward bound $\wr\mathcal{R}_i > b_i\wr$ becomes active at some point *without* reaching such an EC before approaches 0.

[3]We briefly discuss *non-positive* objectives in Section 7.4.

Example 7.4 Consider the MDP M_3 given in Figure 7.4a and the quantile query

$$\Psi_4 \;=\; Qu^{M_1}_{s_0}\big(bnd^{\mathcal{B}}_{\mathcal{G}}(1,\{1\}) \geq 0.5\big)\,, \text{ where}$$
$$\mathcal{B} \;=\; \big\langle \wr\mathcal{R}_1 > ?\wr, \wr\mathcal{R}_2 > ?\wr \big\rangle \quad \text{and} \quad \mathcal{G} \;=\; \big\langle \langle \{1,2\}, \{s_3\}\rangle \big\rangle.$$

If a strategy selects β at s_0, 3 reward will be collected for $\mathcal{R}_2$ and the EC $\{\langle s_2, \beta\rangle\}$ is reached. For arbitrary $n \in \mathbb{N}$, the strategy can collect n reward for $\mathcal{R}_1$ in this EC by taking β at s_2 n times. After enough reward is collected, the strategy can continue—moving to s_3 via α. We thus have $\langle b_1, b_2\rangle \in Sat(\Psi_4)$ for any $b_1, b_2 \in \mathbb{N}$ with $b_2 < 3$.
On the other hand, if a strategy selects α at s_0, state s_3 will be reached almost surely while collecting 5 reward for $\mathcal{R}_2$ and

- 0 reward for $\mathcal{R}_1$ with probability $1/3$,
- at least 3 reward for $\mathcal{R}_1$ with probability $2/3$, and
- more than 3 reward for $\mathcal{R}_1$ with probability $2/3 \cdot 2/3 = 4/9$.

Since the threshold ≥ 0.5 imposed by Ψ_4 is only met in the second case, we obtain the set of satisfying reward limits as indicated in the (lower-left) green area in Figure 7.4b. Its generator is given by

$$Gen^{\Psi_4}(Sat(\Psi_4)) = \big\{\langle \infty, 2\rangle, \langle 2, 4\rangle\big\}.$$

For $i \in \{1..d\}$, let $\mathcal{R}_i^{\mathrm{EC}}$ be the restriction of reward assignment $\mathcal{R}_i$ to transitions within end components, i.e.,

$$\mathcal{R}_i^{\mathrm{EC}}(s,\alpha,s') \;:=\; [\![\exists\, C \in MEC(M)\colon \langle s,\alpha\rangle \in C]\!] \cdot \mathcal{R}_i(s,\alpha,s').$$

If the reward bound $\wr\mathcal{R}_i^{\mathrm{EC}} > 0\wr$ becomes active on a given path, an EC with positive reward for $\mathcal{R}_i$ has been reached and for any given $b \in \mathbb{N}$, a strategy can choose to stay inside that EC to also make the reward bound $\wr\mathcal{R}_i^{\mathrm{EC}} > b\wr$ and thus $\wr\mathcal{R}_i > b\wr$ active. On the other hand, if b approaches infinity, it becomes less and less likely to collect more than b reward for $\mathcal{R}_i$ while not visiting such an EC, i.e.,

$$\lim_{b\to\infty} \Pr^M_{\max,s_I}\big(\big\{\pi \in Paths_{\inf}(M) \;\big|\; \mathcal{R}_i[\pi] > b \text{ and } \mathcal{R}_i^{\mathrm{EC}}[\pi] = 0\big\}\big) \;=\; 0.$$

The $(d-1)$-dimensional quantile query $\Psi\restriction i$ for Lemma 7.13 arises from Ψ (with lower reward bounds) by *replacing* the i^{th} indefinite reward bound $\wr\mathcal{R}_i > ?\wr$ by the definite reward bound $\wr\mathcal{R}_i^{\mathrm{EC}} > 0\wr$. The quantile $\Psi\restriction i$ considers mixtures of indefinite and definite reward bounds and can be analyzed as mentioned in Remark 7.3.

Input: d-dimensional quantile $\Psi = Qu^M_{s_I}(\varphi \bowtie a)$ for MDP M with $\trianglerighteq^\Psi = \geq$, $\bowtie$ as in Lemma 7.12, and either (i) only upper reward bounds or (ii) only lower reward bounds and non-negative φ
Output: The generator of Ψ

```
1 function solveQuantileRecursive(Ψ)
2     B⁺ ← ∅
3     if d = 1 then                                   // Deal with 1-dimensional queries
4         if Ex^M_{max,s_I}(φ↾1) ⋈ a then B⁺ ← {⟨∞⟩}     // Using objective φ↾1 of Ψ↾1
5     else
6         foreach i ∈ {1..d} do                       // Solve (d − 1)-dimensional queries
7             G_i ← solveQuantileRecursive(Ψ↾i)        // ... recursively
8             B⁺ ← B⁺ ∪ {b ∈ ℕ^d | b_{\{i}} ∈ G_i and b(i) = ∞}
9     return solveQuantile(Ψ, B⁺)                     // Algorithm 7.1 with initial B⁺
```

Algorithm 7.2: Quantile computation with monotone reward bounds

An Algorithm for Monotone Bounds

Algorithm 7.2 outlines the procedure for computing B^+_{init} using a recursive approach. If we have a 1-dimensional quantile Ψ, we compute the maximal expected value for the objective $\varphi\restriction 1$ of the "0-dimensional" quantile $\Psi\restriction 1$ and check whether the threshold $\bowtie a$ holds (Line 4). Since $\Psi\restriction 1$ does not consider indefinite reward bounds, it can be evaluated using techniques from Chapters 4 and 6. This is in line with already existing approaches for single-dimensional quantiles [UB13; BDDK+14].

For quantiles with $d \geq 2$ dimensions, we find each $\boldsymbol{b} \in B^+_{\text{init}}$ by computing the set $B^+_{\text{init}}\restriction i$ for each $i \in \{1..d\}$ using the generator of $\Psi\restriction i$ (Lines 6 and 8). This is done by calling Algorithm 7.2 recursively. Once we have initialized B^+ to B^+_{init}, we call Algorithm 7.1 in Line 9.

Results of the recursive calls of Algorithm 7.2 are cached so that each of the occurring quantiles is only processed once. The individual queries consider different subsets of the d indefinite reward bounds from the original input query Ψ. This results in roughly 2^d different calls to Algorithm 7.1. However, the recursive calls consider a simpler quantile with only $d' < d$ dimensions.

Theorem 7.14 Algorithm 7.2 yields a generator of the quantile query Ψ.

Proof. Due to Lemma 7.11, it remains to show that before calling Algorithm 7.1 in

Line 9, $B^+ = B^+_{\text{init}}$ holds. With Lemma 7.13 we have

$$\begin{aligned} \boldsymbol{b} \in B^+ \quad &\text{iff} \quad \exists\, i \in \{1..d\}\colon \boldsymbol{b}_{\setminus i} \in \mathit{Gen}^{\Psi\restriction i}(\mathit{Sat}(\Psi\restriction i)) \text{ and } \boldsymbol{b}(i) = \infty \\ &\text{iff} \quad \exists\, i \in \{1..d\}\colon \boldsymbol{b} \in B^+_{\text{init}}\restriction i \\ &\text{iff} \quad \boldsymbol{b} \in B^+_{\text{init}}. \end{aligned}$$

■

7.4 Intricate Quantile Queries

Above, we have considered only a subset of the possible quantile queries. Many more combinations are possible. These combinations do not easily fit into the framework provided above. We illustrate this mismatch with some cases.

Lower Reward-bounded Quantiles with Non-positive Objectives

Algorithm 7.2 from the previous section cannot deal with queries considering (only) *lower* reward bounds of the form $⟅\mathcal{R}_i > ?⟆$ and *non-positive* objective φ. In this case, a strategy intuitively strives to diminish the reward collected for $\mathcal{R}_i$ for each i so that the corresponding reward bound does not become active. For single-dimensional quantiles with a bounded reachability objective, [BDDK+14] proposes a transformation that instead computes the maximal probability to reach an end component in which either no goal state is visited or no reward is accumulated. However, in the multi-dimensional case, such a transformation does not preserve the other reward bounds.

Mixtures of lower and upper reward bounds

Quantile queries that consider mixtures of lower- and upper reward bounds might have infinite generators, as shown in Example 7.2. In this case, a finite representation of the satisfying reward limits cannot be achieved with our explicit construction of the generator. A procedure to check whether a given query has a finite generator is left for future work. However, when the set of considered reward limits is restricted to a finite domain, our approach can also be applied to general quantiles as mentioned at the end of Section 7.2.

Quantiles over multi-objective trade-offs | We considered quantile queries $Qu^M_{s_I}(\varphi \bowtie a)$ with a single objective φ. An extension inspired by multi-objective queries considers quantiles[4] over several conflicting objectives:

$$Qu^M_{s_I}\big(\varphi_1 \bowtie_1 a_1 \;\wedge\; \ldots \;\wedge\; \varphi_\ell \bowtie_\ell a_\ell\big).$$

[4] Technically, these objects are not quantiles in the classical sense.

Such a query asks for the reward limits for which there is a strategy that satisfies all quantile thresholds. [BDK14] considers a similar type of query, where—however—all objectives φ_j share the same indefinite reward bound $\wr\mathcal{R} \sim ?\wr$. With multiple indefinite reward bounds, two sources of trade-offs are introduced: a trade-off between different strategies and a trade-off between different reward limits. Handling such queries likely involves analyzing the trade-offs at each epoch model and propagating the results through the epochs. We left this for future work.

7.5 Related Work

Solving quantiles is tightly connected to analyzing objectives with definite reward bounds. Therefore, most works mentioned in Section 6.3 are also related to the contents of this chapter. We further discuss related work specific to the computation of quantiles.

[UB13] considers single-dimensional quantiles and provides a graph-based algorithm for *qualitative* queries ($Qu^M_{s_I}(\varphi \bowtie a)$ with $a \in \{0, 1\}$) that runs in polynomial time. For non-qualitative queries, an exponential time algorithm is given that performs a linear search for an optimal reward limit which relies on repeatedly computing reward-bounded reachability probabilities. In [HK15; HKL17], single-dimensional quantiles[5] are solved using a *binary* search on the reward limit, instead. A similar approach for *continuous-time* Markov chains is given in [BDKK+14]. [KSBD15] considers qualitative quantiles over mixtures of positive and negative rewards. [BDDK+14; KBCD+18] introduce energy-utility quantiles that consider one definite and one indefinite reward bound and propose a solution based on sequential epoch analysis. [BDKD+14; SDKB+19] provide applications of energy-utility quantiles in the context of software product line engineering and task scheduling, respectively.

To the best of our knowledge, this is the first work that addresses *multi*-dimensional quantiles for MDPs as in Definition 7.2.

[5] While [HKL17] considers multiple reward bounds, their approach for quantiles fixes all but one reward limit—yielding a 1-dimensional query according to Definition 7.2.

Chapter Summary

- ➜ Multi-dimensional quantiles reveal potential trade-offs between different types of reward (or cost) constraints.
- ➜ The set of satisfying reward limits can uniquely be described by its generator.
- ➜ Quantile queries can be solved based on a sequential epoch analysis (cf. Chapter 6).
- ➜ Depending on the query, termination of the procedure is ensured via dedicated preprocessing steps.

—Chapter 8—

Restriction to Simple Strategies

Outlook | This chapter considers the verification of multiple total reachability reward objectives under *simple*—pure and memoryless—strategies. We formalize the multi-objective achievability problem under such strategies and show NP-completeness in Section 8.1. Section 8.2 provides some necessary background for expected visiting times. Section 8.3 presents a decision procedure for pure memoryless achievability based on mixed integer linear programming (`MILP`). An alternative `MILP` encoding dedicated to total reward objectives is given in Section 8.4. Section 8.5 lifts the approach to Pareto front approximation and Section 8.6 incorporates strategies with finite memory into our framework. Section 8.7 summarizes related work.

Origins | The foundations of this chapter are taken from [9]. The approach and its presentation has been improved and extended towards reward assignments with positive and negative rewards. Various explanations and correctness arguments have been added, in particular in Section 8.2. Section 8.3.2 contains a new encoding inspired by [WWJB20].

Set-up & Notations | Throughout the chapter we assume an MDP $M = \langle S, Act, \Delta, \mathbf{P} \rangle$ with dedicated initial state $s_I \in S$ and $\ell > 0$ *convergent* total reachability reward objectives $\Phi = \langle tot(\mathcal{R}_1, G_1), \ldots, tot(\mathcal{R}_\ell, G_\ell) \rangle$. Markov automata are supported via the underlying MDP (cf. Theorem 4.2 and [13]).

8.1 Pure Memoryless Achievability

Recall the multi-objective achievability problem (`MOA`, Problem 3.1 on page 78) that asks whether a given point $\boldsymbol{p} \in \mathbb{R}^\ell$ is achievable, i.e., whether $\boldsymbol{p} \in Ach(\Phi)$. The point can be achieved by a general strategy that potentially performs randomized decisions based

on the entire execution history. Such types of strategies are not suitable for various applications. For example, they limit reproducibility which complicates debugging. Randomized strategies are also often despised for medical applications [LBM12] and product design—all products should have the same design, not a random one. In this chapter, we study a variant of the achievability problem that only considers *pure memoryless strategies*[1].

pure memoryless achievable point

Definition 8.1 (Pure Memoryless Achievable Point) A point $\boldsymbol{p} \in \overline{\mathbb{R}}^{\ell}$ is called *pure memoryless achievable* for objectives Φ iff $\exists\, \sigma \in \Sigma^{M}_{\mathrm{PM}} \colon \mathrm{Ex}^{M}_{\sigma}(\Phi) \geq \boldsymbol{p}$. The set of pure memoryless achievable points for Φ is denoted by $Ach_{\mathrm{PM}}(\Phi) \subseteq \overline{\mathbb{R}}^{\ell}$.

The set $Ach_{\mathrm{PM}}(\Phi)$ is *closed* (cf. Definition 2.3 on page 18) as the number of pure memoryless strategies of M is finite.

Problem 8.1: Pure Memoryless Multi-objective Achievability (PM-MOA)

PM-MOA, pure memoryless achievability

Input: MDP M, initial state s_I, $\ell > 0$ objectives Φ, point $\boldsymbol{p} \in \mathbb{R}^{\ell}$

Output: $\begin{cases} \text{'YES'} & \text{if } \boldsymbol{p} \in Ach_{\mathrm{PM}}(\Phi) \\ \text{'NO'} & \text{if } \boldsymbol{p} \notin Ach_{\mathrm{PM}}(\Phi) \end{cases}$

We discuss the theoretical complexity of both MOA and PM-MOA for total reachability reward objectives. The following lemma addresses the special case of multi-objective almost-sure reachability.

Lemma 8.1 ([RRS17, Theorem 2]) Answering MOA for reachability objectives $\Phi = \langle [\Diamond G_1], \ldots, [\Diamond G_\ell] \rangle$ and $\boldsymbol{p} = \mathbf{1}$ is PSPACE-complete.

For the hardness proof, [RRS17] uses a reduction from the quantified Boolean formula problem (QBF). Lemma 8.1 implies that MOA for total reachability reward objectives is at least PSPACE-hard[2]. The consideration of goal state sets $G_j \subseteq S$ is crucial for the complexity. In fact, the MOA problem becomes significantly easier when restricted to total reward objectives.

Lemma 8.2 Answering MOA for total reward objectives $\Phi = \langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_\ell) \rangle$ is P-complete.

Proof. The LP-based algorithms of [EKVY08; FKNP+11] can be implemented with polynomial runtime. P-hardness follows from the single-objective case as shown in Theorem 4.7 on page 113. ∎

[1] Section 8.6 also addresses pure strategies that have access to some (limited) information on the history, e.g., whether a goal state set has been reached already.

[2] To the best of our knowledge, a PSPACE upper bound for the complexity of MOA is unknown.

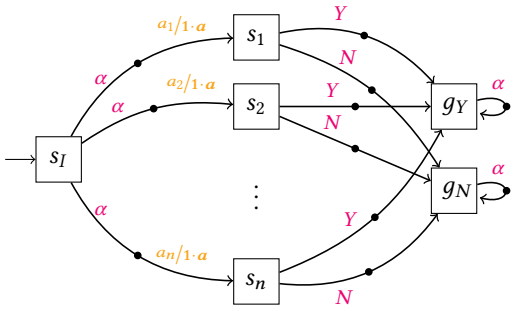


Figure 8.1: Sketch of MDP M from the proof of Lemma 8.3

As discussed in Section 4.1.1, total reachability reward objectives are reducible to total reward objectives via the *goal unfolding*. This increases the size of the considered MDP by a factor of up to 2^ℓ—yielding an exponential-time algorithm for MOA under total reachability reward objectives.

However, if the number of objectives is fixed—say e.g., $\ell = 2$—the blow-up from the goal unfolding becomes a constant factor, allowing us to answer MOA queries with "few" objectives in polynomial time. We now show that PM-MOA, on the other hand, is NP-complete—*even if restricted to only two objectives.* We first show NP-hardness.

Lemma 8.3 Answering PM-MOA for two reachable objectives $\langle [\Diamond G_1], [\Diamond G_2] \rangle$ is NP-hard.

For the proof of Lemma 8.3 we use a reduction from the subset sum problem which is known to be NP-complete [CLRS22, Theorem 34.15]. Our construction is inspired by similar ideas from [CMH06; RRS17].

Problem 8.2: Subset Sum (SUM)

Input: $n \in \mathbb{N}$, $\boldsymbol{a} \in \mathbb{N}^n$, and $z \in \mathbb{N}$.

Output: $\begin{cases} \text{'YES'} & \text{if } \exists \boldsymbol{v} \in \{0,1\}^n \colon \boldsymbol{v} \cdot \boldsymbol{a} = z \\ \text{'NO'} & \text{otherwise} \end{cases}$

Proof (of Lemma 8.3). For a given instance $n \in \mathbb{N}$, $\boldsymbol{a} = \langle a_1, \ldots, a_n \rangle \in \mathbb{N}^n$, and $z \in \mathbb{N}$ of SUM, we construct the MDP $M = \langle S, Act, \Delta, \mathbf{P} \rangle$ as sketched in Figure 8.1 and

formally given by $S = \{s_I, s_1, \ldots, s_n, g_Y, g_N\}$, $Act = \{\alpha, Y, N\}$,

$$\Delta(s) = \begin{cases} \{\alpha\} & \text{if } s \in \{s_I, g_Y, g_N\} \\ \{Y, N\} & \text{if } s \in \{s_1, \ldots, s_n\} \end{cases}$$

for $s \in S$, and for all $Z \in \{Y, N\}$ and all $i \in \{1..n\}$ the non-zero transition probabilities are given by

- $\mathbf{P}(s_I, \alpha, s_i) = \frac{a_i}{\mathbf{1} \cdot \boldsymbol{a}}$, where $\mathbf{1} \cdot \boldsymbol{a} = \sum_{j=1}^{n} a_j$,
- $\mathbf{P}(s_i, Z, g_Z) = 1$, and
- $\mathbf{P}(g_Z, \alpha, g_Z) = 1$.

There is a one-to-one correspondence between pure memoryless strategies of M and vectors $\boldsymbol{v} \in \{0,1\}^n$ given by the bijective mapping $f \colon \Sigma^M_{\mathrm{PM}} \to \{0,1\}^n$ with

$$f(\sigma)(i) = [\![\sigma(s_i){=}Y]\!]$$

for all $\sigma \in \Sigma^M_{\mathrm{PM}}$ and $i \in \{1..n\}$. We claim that the answer for the PM-MOA instance with M, initial state s_I,

$$\Phi = \left\langle [\![\Diamond\{g_Y\}]\!], [\![\Diamond\{g_N\}]\!] \right\rangle, \quad \text{and} \quad \boldsymbol{p} = \left\langle \tfrac{z}{\mathbf{1}\cdot\boldsymbol{a}}, 1 - \tfrac{z}{\mathbf{1}\cdot\boldsymbol{a}} \right\rangle$$

is 'YES' iff there is a vector $\boldsymbol{v} \in \{0,1\}^n$ with $\boldsymbol{v} \cdot \boldsymbol{a} = z$. To see this, observe for $\sigma \in \Sigma^M_{\mathrm{PM}}$:

$$\Pr^M_{\sigma,s_I}(\Diamond\{g_Y\}) = \sum_{i=1}^{n} \frac{a_i}{\mathbf{1}\cdot\boldsymbol{a}} \cdot [\![\sigma(s_i){=}Y]\!] = \frac{f(\sigma)\cdot\boldsymbol{a}}{\mathbf{1}\cdot\boldsymbol{a}}$$

$$\Pr^M_{\sigma,s_I}(\Diamond\{g_N\}) = \sum_{i=1}^{n} \frac{a_i}{\mathbf{1}\cdot\boldsymbol{a}} \cdot [\![\sigma(s_i){=}N]\!] = \frac{(\mathbf{1} - f(\sigma))\cdot\boldsymbol{a}}{\mathbf{1}\cdot\boldsymbol{a}} = 1 - \frac{f(\sigma)\cdot\boldsymbol{a}}{\mathbf{1}\cdot\boldsymbol{a}}.$$

Therefore, $\boldsymbol{p}$ is pure memoryless achievable iff $f(\sigma)\cdot\boldsymbol{a} = z$ for some $\sigma \in \Sigma^M_{\mathrm{PM}}$. ■

[CMH06, Theorem 6] provides a similar result for total reward objectives:

Lemma 8.4 Answering PM-MOA for two total reward objectives $\left\langle tot(\mathcal{R}_1), tot(\mathcal{R}_2) \right\rangle$ is NP-hard

Proof. We restate the proof from [CMH06] for the sake of completeness. Given a SUM instance $n \in \mathbb{N}$, $\boldsymbol{a} = \langle a_1, \ldots, a_n \rangle \in \mathbb{N}^n$, and $z \in \mathbb{N}$, consider the MDP $M = \langle S, Act, \Delta, \mathbf{P} \rangle$ as sketched in Figure 8.2 and formally given by $S = \{s_1, \ldots, s_n, s_{n+1}\}$,

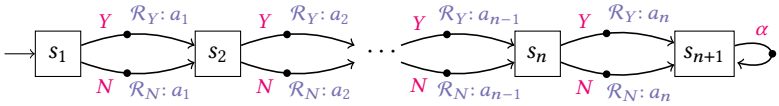


Figure 8.2: Sketch of MDP M from the proof of Lemma 8.4

$Act = \{\alpha, Y, N\}$,

$$\Delta(s) = \begin{cases} \{\alpha\} & \text{if } s = s_{n+1} \\ \{Y, N\} & \text{if } s \in \{s_1, \ldots, s_n\} \end{cases}$$

for $s \in S$, and for all $Z \in \{Y, N\}$ and all $i \in \{1..n\}$ the non-zero transition probabilities are given by

- $\mathbf{P}(s_i, Z, s_{i+1}) = 1$ and
- $\mathbf{P}(s_{n+1}, \alpha, s_{n+1}) = 1$.

Furthermore, we consider two reward assignments $\mathcal{R}_Y$ and $\mathcal{R}_N$ for M such that for all $Z \in \{Y, N\}$ and all $i \in \{1..n\}$ we have $\mathcal{R}_Z[s_i, Z, s_{i+1}] = a_i$. The remaining rewards are zero.
Similar to our construction from the proof of Lemma 8.3, there is a one-to-one correspondence between pure memoryless strategies of M and vectors $\boldsymbol{v} \in \{0,1\}^n$ given by the bijective mapping $f\colon \Sigma^M_{\mathrm{PM}} \to \{0,1\}^n$ with

$$f(\sigma)(i) = [\sigma(s_i)=Y]$$

for all $\sigma \in \Sigma^M_{\mathrm{PM}}$ and $i \in \{1..n\}$. The answer for the PM-MOA instance with M, initial state s_1,

$$\Phi = \left\langle tot(\mathcal{R}_Y),\ tot(\mathcal{R}_N) \right\rangle, \quad \text{and} \quad \boldsymbol{p} = \left\langle z,\ \mathbf{1}\cdot\boldsymbol{a} - z \right\rangle$$

is 'YES' iff there is a vector $\boldsymbol{v} \in \{0,1\}^n$ with $\boldsymbol{v}\cdot\boldsymbol{a} = z$. To see this, observe for $\sigma \in \Sigma^M_{\mathrm{PM}}$:

$$\mathrm{Ex}^M_{\sigma,s_1}(tot(\mathcal{R}_Y)) = \sum_{i=1}^{n} a_i \cdot [\sigma(s_i)=Y] = f(\sigma)\cdot\boldsymbol{a}$$

$$\mathrm{Ex}^M_{\sigma,s_1}(tot(\mathcal{R}_N)) = \sum_{i=1}^{n} a_i \cdot [\sigma(s_i)=N] = (\mathbf{1} - f(\sigma))\cdot\boldsymbol{a} = \mathbf{1}\cdot\boldsymbol{a} - f(\sigma)\cdot\boldsymbol{a}.$$

Therefore, $\boldsymbol{p}$ is pure memoryless achievable iff $f(\sigma)\cdot\boldsymbol{a} = z$ for some $\sigma \in \Sigma^M_{\mathrm{PM}}$. ■

Theorem 8.5 PM-MOA as well as the variant of PM-MOA restricted to only two objectives are both NP-complete.

Proof. Containment in NP follows by guessing a pure memoryless strategy $\sigma \in \Sigma^M_{\text{PM}}$ and evaluating the individual objectives on the induced submodel $M[\![\sigma]\!]$—which is a DTMC. This can be done in polynomial time [BK08]. Hardness follows by either Lemma 8.3 or Lemma 8.4. ∎

To summarize, deciding multi-objective achievability for total reachability reward objectives under general strategies is PSPACE-hard. The restriction to pure memoryless strategies reduces the complexity to an NP-complete problem. *However*, when a fixed number of objectives is assumed (e.g., $\ell = 2$), achievability under general strategies can be decided in polynomial time, *while the complexity for pure memoryless strategies remains NP-complete.*

8.2 Intermezzo: Expected Visiting Times

In subsequent sections, we present our approach for PM-MOA based on mixed integer linear programming (MILP). Some of our MILP formulations—more specifically the constraints in Figures 8.6 and 8.10—are based on *expected visiting times*. Before we present our encodings, we establish some foundations in this section.

expected visiting times

Definition 8.2 (Visiting Times) For MDP $M = \langle S, Act, \Delta, \mathbf{P}\rangle$ with $s_I \in S$ and $\sigma \in \Sigma^M$, the *expected number of visits* of a state $s \in S$ under σ is given by the expected value $\text{Ex}^M_{\sigma,s_I}(visits(s))$ of the objective $visits(s) : Paths^M_{\text{inf}} \to \mathbb{N} \cup \{\infty\}$ with

$$visits(s)(s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \dots) \coloneqq \sum_{i=0}^{\infty} [\![s_i = s]\!].$$

8.2.1 Relation to Total Reward Objectives

We first outline the connection between expected visiting times and expected total rewards. For $s \in S$, consider the reward assignment $\mathcal{R}_s$ for M with

$$\forall\, s', s'' \in S\colon\ \forall\, \alpha \in \Delta(s')\colon \qquad \mathcal{R}_s[s', \alpha, s''] \coloneqq [\![s{=}s']\!].$$

Lemma 8.6 $\forall\, \sigma \in \Sigma^M\colon\ \text{Ex}^M_{\sigma,s_I}(visits(s)) = \text{Ex}^M_{\sigma,s_I}(tot(\mathcal{R}_s))$.

Proof. For $\pi \in Paths^M_{\text{inf}}$ we have $visits(s)(\pi) = tot(\mathcal{R}_s)(\pi)$. ∎

Lemma 8.6 yields that expected visiting times can be expressed in terms of expected

total rewards. One implication is that maximal (or minimal) expected visiting times $\mathrm{Ex}^M_{\max,s_I}(\mathit{visits}(s))$ of a fixed state s are always attained by some pure memoryless strategy. However, in general there is not a single strategy that maximizes the visiting times of *all* states.

The next lemma shows that expected total rewards can also be expressed in terms of expected visiting times. Consider an expected total reward objective $\mathit{tot}(\mathcal{R})$ with a reward assignment $\mathcal{R}$ for M. For $s \in S$ and $\alpha \in \Delta(s)$ let

$$r_{\langle s,\alpha\rangle} \coloneqq \sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot \mathcal{R}[s,\alpha,s'].$$

Lemma 8.7 $\forall\, \sigma \in \Sigma^M_{\mathrm{PM}}\colon\ \mathrm{Ex}^M_{\sigma,s_I}(\mathit{tot}(\mathcal{R})) = \sum_{s\in S} r_{s,\sigma(s)} \cdot \mathrm{Ex}^M_{\sigma,s_I}(\mathit{visits}(s))$.

Proof. Using Lemma 8.6 and a similar reasoning as in the proof of Lemma 4.4 on page 106, we get

$$\sum_{s\in S} r_{s,\sigma(s)} \cdot \mathrm{Ex}^M_{\sigma,s_I}(\mathit{visits}(s)) = \sum_{s\in S} r_{s,\sigma(s)} \cdot \mathrm{Ex}^M_{\sigma,s_I}(\mathit{tot}(\mathcal{R}_s)) = \mathrm{Ex}^M_{\sigma,s_I}(\mathit{tot}(\mathcal{R}')),$$

with reward assignment $\mathcal{R}' = \sum_{s\in S} r_{s,\sigma(s)} \cdot \mathcal{R}_s$. Since σ is pure and memoryless, we can use the linear equation system characterization of total rewards on Markov chains (as in Theorem 8.11 below) to show that $\mathrm{Ex}^M_{\sigma,s_I}\big(\mathit{tot}(\mathcal{R})\big) = \mathrm{Ex}^M_{\sigma,s_I}\big(\mathit{tot}(\mathcal{R}')\big)$, which concludes the proof. ∎

8.2.2 Equation System Characterization

The following theorem yields an equation system characterization for the expected number of times each state $s \in S$ is visited in the Markov chain induced by MDP M and a strategy $\sigma \in \Sigma^M_{\mathrm{PM}}$. The theorem is based on similar observations from the literature [Put94; FKNP+11; Vel19].

Theorem 8.8 Let $\sigma \in \Sigma^M_{\mathrm{PM}}$ be a strategy and let $s_I \in S' \subseteq S$ such that under σ the set S' is not reached from any $s \in S \setminus S'$, i.e., $\mathit{post}(s,\sigma(s)) \cap S' = \emptyset$. The equation system over variables $\mathit{Vars} = \{z_s \mid s \in S'\}$

$$\forall\, s \in S'\colon\ z_s = [s{=}s_I] + \sum_{s'\in S'} \mathbf{P}\big(s',\sigma(s'),s\big) \cdot z_{s'} \tag{8.1}$$

$$1 = \sum_{s\in(S\setminus S')} \sum_{s'\in S'} \mathbf{P}\big(s',\sigma(s'),s\big) \cdot z_{s'} \tag{8.2}$$

has a non-negative solution $val\colon Vars \to \mathbb{R}_{\geq 0}$ iff $\Pr^M_{\sigma,s_I}(\Diamond(S \setminus S')) = 1$. If a non-negative solution val exists, the value $val(z_s)$ is unique for all $s \in T$ with

$$T \;=\; \big\{ s \in S' \;\big|\; \mathrm{Ex}^M_{\sigma,s}(visits(s)) < \infty \big\}.$$

and we have

$$\forall\, s \in T\colon \;\; val(s) \;=\; \mathrm{Ex}^M_{\sigma,s_I}(visits(s)).$$

Proof. We first show that Equation (8.2) is implied by Equation (8.1). Let $val\colon Vars \to \mathbb{R}_{\geq 0}$ be a solution for Equation (8.1). Taking the sum over all $s \in S'$ yields

$$\begin{aligned}
\sum_{s\in S'} val(z_s) \;&=\; \sum_{s\in S'} \Big([s{=}s_I] + \sum_{s'\in S'} \mathbf{P}\big(s',\sigma(s'),s\big)\cdot val(z_{s'}) \Big) \\
&=\; \Big(\sum_{s\in S'} [s{=}s_I] \Big) + \Big(\sum_{s'\in S'} val(z_{s'}) \cdot \sum_{s\in S'} \mathbf{P}\big(s',\sigma(s'),s\big) \Big) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(using } s_I \in S'\text{)} \\
&=\; 1 + \Big(\sum_{s'\in S'} val(z_{s'}) \cdot \sum_{s\in S'} \mathbf{P}\big(s',\sigma(s'),s\big) \Big) \\
&\qquad\qquad \big(\text{using } \textstyle\sum_{s\in S'} \mathbf{P}\big(s',\sigma(s'),s\big) + \sum_{s\in (S\setminus S')} \mathbf{P}\big(s',\sigma(s'),s\big) = 1\big) \\
&=\; 1 + \Big(\sum_{s'\in S'} val(z_{s'}) \cdot \Big(1 - \sum_{s\in (S\setminus S')} \mathbf{P}\big(s',\sigma(s'),s\big)\Big) \Big) \\
&=\; 1 + \Big(\sum_{s'\in S'} val(z_{s'}) \Big) - \Big(\sum_{s'\in S'} val(z_{s'}) \cdot \sum_{s\in (S\setminus S')} \mathbf{P}\big(s',\sigma(s'),s\big) \Big) \\
&=\; 1 + \Big(\sum_{s'\in S'} val(z_{s'}) \Big) - \Big(\sum_{s\in (S\setminus S')} \sum_{s'\in S'} \mathbf{P}\big(s',\sigma(s'),s\big)\cdot val(z_{s'}) \Big).
\end{aligned}$$

We thus get

$$\sum_{s\in S'} val(z_s) \;=\; 1 + \Big(\sum_{s'\in S'} val(z_{s'}) \Big) - \Big(\sum_{s\in (S\setminus S')} \sum_{s'\in S'} \mathbf{P}\big(s',\sigma(s'),s\big)\cdot val(z_{s'}) \Big)$$

$$\text{iff} \quad \sum_{s\in (S\setminus S')} \sum_{s'\in S'} \mathbf{P}\big(s',\sigma(s'),s\big)\cdot val(z_{s'}) \;=\; 1.$$

Next, we distinguish the two cases in which $S \setminus S'$ is either reached almost-surely from s_I or not.

Case 1 | Let $\Pr^M_{\sigma,s_I}\big(\Diamond(S \setminus S')\big) = 1$. We first argue that once the set $S' \setminus T$ is reached, it can not be left again. To see this, consider a state $s \in S' \setminus T$, i.e., $\mathrm{Ex}^M_{\sigma,s}(\mathit{visits}(s)) = \infty$. This also means that all successors of s (under strategy σ) are visited infinitely often. Our assumptions for S' imply that we can not reach $s \in S' \setminus T$ via a state in $S \setminus S'$. Therefore, all successors of s (under strategy σ) must be in $S' \setminus T$.
Since we consider the case where the set S' is almost surely exited when starting in the initial state $s_I \in S'$, the trapping set $S' \setminus T$ must not be reachable from s_I, i.e., we get for any state $s \in S'$ that

- s is not reachable from s_I, i.e., $\Pr^M_{\sigma,s_I}\big(\Diamond\{s\}\big) = 0$ or
- $s \notin S' \setminus T$, i.e., $s \in T$.

Setting z_s to 0 for all $s \in S'$ that are not reachable from s_I yields a trivial solution for Equation (8.1) which also coincides with the expected number of visits for those states. The states in T are so-called *transient* states of $M[\![\sigma]\!]$. It follows from [KS76, Theorem 3.2.4] that the equation system

$$\forall\, s \in T\colon \quad z_s \;=\; [s{=}s_I] + \sum_{s' \in T} \mathbf{P}\big(s', \sigma(s'), s\big) \cdot z_{s'}$$

has a unique solution that coincides with the expected number of visits for each $s \in T$. This equation system coincides with Equation (8.1) when restricted to states in T since—as argued above—$\mathbf{P}(s', \sigma(s'), s) = 0$ for $s' \in S' \setminus T$ and $s \in T$.

Case 2 | If $\Pr^M_{\sigma,s_I}\big(\Diamond(S \setminus S')\big) < 1$, the induced Markov chain $M[\![\sigma]\!]$ must contain a *bottom strongly connected component (BSCC)* $B \subseteq S'$ that is reachable from s_I, i.e.,

BSCC, bottom strongly connected component

$$\exists\, \hat{\pi} \in \mathit{Paths}^{M[\![\sigma]\!]}_{\mathrm{fin}}(s_I)\colon \ \mathit{last}(\hat{\pi}) \in B.$$

For this case, we have to show that the equation system from Theorem 8.8 has no non-negative solution. Towards a contradiction, assume that a solution $\mathit{val}\colon \mathit{Vars} \to \mathbb{R}_{\geq 0}$

exists. By taking the sum of the values for all states $s \in B$ we get

$$\begin{aligned}\sum_{s\in B} val(z_s) &= \sum_{s\in B}\Big([s{=}s_I] + \sum_{s'\in S'}\Big(\mathbf{P}\big(s',\sigma(s'),s\big)\cdot val(z_{s'})\Big)\Big)\\ &= [s_I \in B] + \sum_{\langle s,s'\rangle\in B\times S'}\Big(\mathbf{P}\big(s',\sigma(s'),s\big)\cdot val(z_{s'})\Big)\\ &= v_1 + v_2\,,\end{aligned}$$

where

$$\begin{aligned}v_1 &\coloneqq [s_I \in B] + \sum_{\langle s,s'\rangle\in B\times(S'\setminus B)}\Big(\mathbf{P}\big(s',\sigma(s'),s\big)\cdot val(z_{s'})\Big), \text{ and}\\ v_2 &\coloneqq \sum_{\langle s,s'\rangle\in B\times B}\Big(\mathbf{P}\big(s',\sigma(s'),s\big)\cdot val(z_{s'})\Big).\end{aligned}$$

We now show that $v_1 > 0$ and $v_2 = \sum_{s\in B} val(z_s)$. This yields a contradiction as

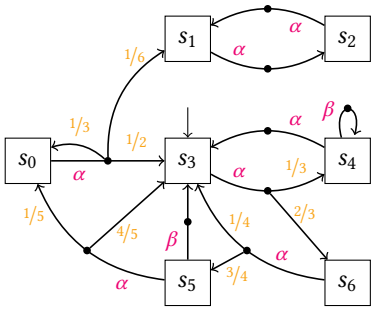
$$\sum_{s\in B} val(z_s) \;=\; v_1 + v_2 \;>\; \sum_{s\in B} val(z_s).$$

- If $s_I \in B$, we immediately get $v_1 \geq 1 > 0$ due to the non-negativity of the solution *val*. Otherwise, there is a finite path $\hat{\pi} = s_0 \xrightarrow{\sigma(s_0)} \dots \xrightarrow{\sigma(s_{n-1})} s_n \in Paths^{M[\![\sigma]\!]}_{\text{fin}}(s_I)$ of length $n \in \mathbb{N}$ with $s_0 = s_I$, $s_n \in B$, and $s_i \notin B$ for all $i \in \{0..n{-}1\}$. We first show that $val(z_{s_i}) > 0$ for all $i \in \{0..n{-}1\}$:
  - For $i = 0$ the claim follows as $[s_0{=}s_I] = 1$ and the solution *val* is non-negative.
  - For $i > 0$ we can assume that $val(z_{s_{i-1}}) > 0$. The solution *val* satisfies

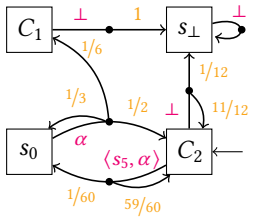
$$val(z_{s_i}) \;\geq\; \underbrace{\mathbf{P}\big(s_{i-1},\sigma(s_{i-1}),s_i\big)}_{>0}\cdot\underbrace{val(z_{s_{i-1}})}_{>0} \;>\; 0.$$

  With $s_{n-1} \in S' \setminus B$ and $s_n \in B$ we conclude that

$$v_1 \;\geq\; \mathbf{P}\big(s_{n-1},\sigma(s_{n-1}),s_n\big)\cdot val(z_{s_{n-1}}) \;>\; 0. \qquad \blacksquare$$



(a) MDP M



(b) Quotient $\overline{M_{\backslash \mathfrak{C}}}$

s_j	s_0	s_1	s_2	s_3	s_4	s_5	s_6
$\mathrm{Ex}^M_{\sigma_{\alpha\alpha},s_I}(visits(s_j))$	6	∞	∞	40	$40/3$	20	$80/3$
$\mathrm{Ex}^M_{\sigma_{\alpha\beta},s_I}(visits(s_j))$	0	0	0	∞	∞	∞	∞
$\mathrm{Ex}^M_{\sigma_{\beta\alpha},s_I}(visits(s_j))$	$18/43$	∞	∞	$120/43$	∞	$60/43$	$80/43$
$\mathrm{Ex}^M_{\sigma_{\beta\beta},s_I}(visits(s_j))$	0	0	0	3	∞	$3/2$	2

(c) Expected visiting times for M under simple strategies

Figure 8.3: Example MDP, its (modified) quotient, and resulting expected visiting times (cf. Examples 8.1 and 8.2)

- The BSCC B satisfies $post(\langle s, \sigma(s) \rangle) \subseteq B$ for any $s \in B$. We get

$$v_2 = \sum_{s' \in B} \Big(val(z_{s'}) \cdot \underbrace{\sum_{s \in B} \mathbf{P}(s', \sigma(s'), s)}_{=1} \Big) = \sum_{s' \in B} val(z_{s'}).$$

Example 8.1 Consider the MDP M from Figure 8.3a with initial state s_3. There are exactly four pure memoryless strategies for M:

$$\Sigma^M_{\mathrm{PM}} = \{\sigma_{\alpha\alpha}, \sigma_{\alpha\beta}, \sigma_{\beta\alpha}, \sigma_{\beta\beta}\},$$

where for $\gamma, \delta \in \{\alpha, \beta\}$ we set $\sigma_{\gamma\delta}(s_4) = \gamma$ and $\sigma_{\gamma\delta}(s_5) = \delta$. Figure 8.3c shows the expected visiting times under all four strategies. We outline the corresponding

computations for $\sigma_{\alpha\alpha}$ and $\sigma_{\beta\beta}$ using Theorem 8.8.
For $\sigma_{\alpha\alpha}$, the states s_1 and s_2 are almost surely reached from any other state. They form the only BSCC of $M[\![\sigma_{\alpha\alpha}]\!]$. We get $\mathrm{Ex}^{M}_{\sigma_{\alpha\alpha},s_I}(\mathit{visits}(s_1)) = \mathrm{Ex}^{M}_{\sigma_{\alpha\alpha},s_I}(\mathit{visits}(s_2)) = \infty$. With $S' = T = \{s_0, s_3, s_4, s_5, s_6\}$, the equation system in Theorem 8.8 is given by

$$z_{s_0} = 1/3 \cdot z_{s_0} + 1/5 \cdot z_{s_5} \qquad z_{s_3} = 1 + 1/2 \cdot z_{s_0} + 1 \cdot z_{s_4} + 4/5 \cdot z_{s_5} + 1/4 \cdot z_{s_6}$$
$$z_{s_4} = 1/3 \cdot z_{s_3} \qquad z_{s_5} = 3/4 \cdot z_{s_6} \qquad z_{s_6} = 2/3 \cdot z_{s_3}.$$

The unique solution of this equation system is given by

$$z_{s_0} = 6 \qquad z_{s_3} = 40 \qquad z_{s_4} = 40/3 \qquad z_{s_5} = 20 \qquad \text{and} \qquad z_{s_6} = 80/3,$$

which yields the expected visiting times as in Figure 8.3c.
Next, we consider the strategy $\sigma_{\beta\beta}$ which selects β at both s_4 and s_5. Under this strategy, the states s_0, s_1, and s_2 are not reachable from the initial state s_3 and therefore will not be visited at all. Moreover, s_4 forms a (reachable) BSCC in the induced Markov chain. With $S' = S \setminus \{s_4\}$ and $T = \{s_0, s_3, s_5, s_6\}$, the resulting equation system is given by

$$z_{s_0} = 1/3 \cdot z_{s_0} \qquad z_{s_1} = 1/6 \cdot z_{s_0} + 1 \cdot z_{s_2} \qquad z_{s_2} = 1 \cdot z_{s_1}$$
$$z_{s_3} = 1 + 1/2 \cdot z_{s_0} + 1 \cdot z_{s_5} + 1/4 \cdot z_{s_6} \qquad z_{s_5} = 3/4 \cdot z_{s_6} \qquad z_{s_6} = 2/3 \cdot z_{s_3}.$$

This equation system has multiple solutions for the states $s_1, s_2 \notin T$ but a unique solution for the states $s_0, s_3, s_5, s_6 \in T$ given by

$$z_{s_0} = 0 \qquad z_{s_3} = 3 \qquad z_{s_5} = 3/2 \qquad \text{and} \qquad z_{s_6} = 2.$$

8.2.3 Upper Bounds for Expected Visiting Times

For the considered `MILP` encodings, we also require upper bounds for the expected number of visits for any given state $s \in S$. For this, we only consider pure memoryless strategies under which this number is finite. Let S^{fin} be the set of states for which at least one strategy induces a finite expected number of visits, i.e.,

$$S^{\mathrm{fin}} := \left\{ s \in S \;\middle|\; \exists\, \sigma \in \Sigma^{M}_{\mathrm{PM}} \colon \mathrm{Ex}^{M}_{\sigma,s_I}(\mathit{visits}(s)) < \infty \right\}.$$

We want to compute for each $s \in S^{\text{fin}}$ an upper bound for the possible *finite* expected visiting times, i.e., we want to compute a value $V_s \in \mathbb{R}_{\geq 0}$ for each $s \in S^{\text{fin}}$ such that

$$V_s \;\geq\; \max\left(\left\{\mathrm{Ex}^{M}_{\sigma,s_I}(\mathit{visits}(s)) \;\middle|\; \sigma \in \Sigma^{M}_{\text{PM}}\right\} \cap \mathbb{R}\right). \tag{8.3}$$

For this, we assume that there is no transition from a state $s \notin S^{\text{fin}}$ to a state $s' \in S^{\text{fin}}$:

Assumption 8.1 $\forall\, s \in S \setminus S^{\text{fin}} \colon\ \forall\, \alpha \in \Delta(s) \colon\ \mathit{post}(s,\alpha) \cap S^{\text{fin}} \;=\; \emptyset$.

Assumption 8.1 is without loss of generality: given a strategy $\sigma \in \Sigma^{M}_{\text{PM}}$ and a state $s \in S \setminus S^{\text{fin}}$, we have—by definition of S^{fin}—that $\mathrm{Ex}^{M}_{\sigma,s_I}(\mathit{visits}(s)) = \infty$ which yields that $\mathrm{Ex}^{M}_{\sigma,s_I}(\mathit{visits}(s')) = \infty$ holds for all states $s' \in S^{\text{fin}}$ reached from s with positive probability. Thus, for every $s' \in S^{\text{fin}}$ we either have

- $\Pr^{M}_{\sigma,s}(\Diamond\{s'\}) = 0$ for all $s \in S \setminus S^{\text{fin}}$ or
- $\mathrm{Ex}^{M}_{\sigma,s_I}(\mathit{visits}(s')) = \infty$.

Only the former case is relevant when computing values $V_{s'}$ as in Equation (8.3). Therefore, if Assumption 8.1 is violated by a state s and an action α, we can safely remove α from $\Delta(s)$.

Following ideas of [BKLP+17], we further make use of the following lemma which links expected visiting times of a state $s \in S$ to the probability that—when starting from s—we do not reach s a second time. For $i \in \mathbb{N}$ let

$$\Pi_{\#s=i} \;:=\; \left\{\pi \in \mathit{Paths}^{M}_{\text{inf}} \;\middle|\; \mathit{visits}(s)(\pi) = i\right\}$$

be the set of paths that visit s exactly i times.

Lemma 8.9 Let $\sigma \in \Sigma^{M}_{\text{PM}}$ and let $s \in S$ with $\Pr^{M}_{\sigma,s_I}(\Diamond\{s\}) > 0$. If $\Pr^{M}_{\sigma,s}(\Pi_{\#s=1}) = 0$ then $\mathrm{Ex}^{M}_{\sigma,s_I}(\mathit{visits}(s)) = \infty$. Otherwise,

$$\mathrm{Ex}^{M}_{\sigma,s_I}(\mathit{visits}(s)) \;=\; \frac{\Pr^{M}_{\sigma,s_I}(\Diamond\{s\})}{\Pr^{M}_{\sigma,s}(\Pi_{\#s=1})}\,.$$

Proof. The lemma is a straightforward reformulation of [BKLP+17, Lemma 3.5]. ■

We use Lemma 8.9 to obtain upper bounds V_s for $s \in S^{\text{fin}}$ as in Equation (8.3). More precisely, the approach is to find for each $s \in S^{\text{fin}}$ a non-zero lower bound d_s for the probability that starting at s we never reach s again, i.e., $0 < d_s \leq \Pr^{M}_{\sigma,s}(\Pi_{\#s=1})$ for every strategy $\sigma \in \Sigma^{M}_{\text{PM}}$ with $0 < \Pr^{M}_{\sigma,s}(\Pi_{\#s=1})$. Using Lemma 8.9 and the trivial upper bound $\Pr^{M}_{\sigma,s_I}(\Diamond\{s\}) \leq 1$, an upper bound for the expected visiting times at $s \in S^{\text{fin}}$ is given by $V_s = 1/d_s$. We consider two cases for the MDP M.

Input: MDP M, set $S^{\text{fin}} \subseteq S$ with $\forall C \in EC(M)\colon\ states(C) \not\subseteq S^{\text{fin}}$
Output: $\langle \mathsf{V}_s \rangle_{s \in S^{\text{fin}}}$ with V_s as in Equation (8.3)

1 $S_{done} \leftarrow S \setminus S^{\text{fin}}$
2 **foreach** $s \in S_{done}$ **do** $d_s \leftarrow 1$
3 **while** $S_{done} \neq S$ **do**
4 Pick $s \in S \setminus S_{done}$ such that $\forall\, \alpha \in \Delta(s)\colon\ post(s,\alpha) \cap S_{done} \neq \emptyset$
5 $d_s \leftarrow \min_{\alpha \in \Delta(s)} \sum_{s' \in S_{done}} \mathbf{P}(s,\alpha,s') \cdot \begin{cases} d_{s'} & \text{if } \exists\, \text{SCC } C\colon\ \{s,s'\} \subseteq states(C) \\ 1 & \text{otherwise} \end{cases}$
6 $S_{done} \leftarrow S_{done} \cup \{s\}$
7 **return** $\left\langle \frac{1}{d_s} \right\rangle_{s \in S^{\text{fin}}}$

Algorithm 8.1: Computation of upper bounds for expected visiting times (cf. [BKLP+17])

Case 1 | First assume that there is no $C \in EC(M)$ with $states(C) \subseteq S^{\text{fin}}$. Thus, under any strategy we almost-surely leave the set S^{fin} (permanently, due to Assumption 8.1). This yields that the expected number of visits of each $s \in S^{\text{fin}}$ is finite for all $\sigma \in \Sigma^M_{\text{PM}}$. For this setting, [BKLP+17] provides an efficient algorithm for computing the values d_s and $\mathsf{V}_s = 1/d_s$. Algorithm 8.1 outlines the procedure. It maintains a set of states S_{done} for which values d_s are already known. In each iteration it then considers a state $s \in S \setminus S_{done}$ that—under all strategies—has a positive probability to reach S_{done} in a single transition step. Such a state must always exist since otherwise we would get an EC C with $states(C) \subseteq (S \setminus S_{done}) \subseteq S^{\text{fin}}$. Line 5 sets the value d_s for the chosen state s to a lower bound for the probability to move to S_{done} in one step and then stay in S_{done} forever—thus never reaching $s \notin S_{done}$ again. We refer to [BKLP+17] for further details.

Case 2 | If there is some $C \in EC(M)$ with $states(C) \subseteq S^{\text{fin}}$, Algorithm 8.1 eventually reaches a point where $S_{done} \neq S$ but no valid state s is found in Line 4. Hence, the approach of [BKLP+17] would fail to compute valid upper bounds. We overcome this issue by first eliminating all *maximal* ECs $C \in \mathfrak{C}$ with

$$\mathfrak{C} \coloneqq \left\{ C \in MEC(M) \;\middle|\; states(C) \subseteq S^{\text{fin}} \right\}$$

in a way that expected visiting times are over-approximated. Due to Assumption 8.1, ECs of M are either fully contained in S^{fin} or fully contained in $S \setminus S^{\text{fin}}$. Hence, every (potentially not maximal) $C' \in EC(M)$ with $states(C') \subseteq S^{\text{fin}}$ is contained in a MEC $C \in \mathfrak{C}$. Therefore, after eliminating each $C \in \mathfrak{C}$, Case 1 applies, i.e., we can use Algorithm 8.1. We perform the following steps:

1. For each $C \in \mathfrak{C}$, compute a value $l_C \in (0,1]$ such that—given any pair of distinct states $s, s' \in \mathit{states}(C)$ and a pure memoryless strategy under which s' is reachable from s—l_C is a lower bound for the probability that we reach s' from s without visiting s in between. Formally, we require for all $s, s' \in \mathit{states}(C)$ with $s \neq s'$ and for all $\sigma \in \Sigma_{\mathrm{PM}}^{M[\![C]\!]}$ with $\Pr_{\sigma,s}^{M[\![C]\!]}(\Diamond\{s'\}) > 0$ that

$$0 < l_C \leq \Pr_{\sigma,s}^{M[\![C]\!]}\Big(\big\{s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots \in \mathit{Paths}_{\mathrm{inf}}^{M[\![C]\!]}(s) \;\big|\; \exists\, j > 0 \colon\ s_j = s' \text{ and } \forall\, i \in \{1..j{-}1\} \colon\ s_i \neq s\big\}\Big).$$

   If C consists of a single state, we can set $l_C = 1$. Otherwise, consider two distinct states $s, s' \in \mathit{states}(C)$. Whenever s' is reachable from s, there is at least one finite path from s to s' that visits each state of C at most once. We can conservatively set l_C to a lower bound for the probability of such a single path. A naive lower bound for this is given by the product of the smallest probabilities occurring at each state in C—excluding self-loop probabilities—i.e.,

$$l_C = \prod_{s \in \mathit{states}(C)} \min\big\{\mathbf{P}(s,\alpha,s') \;\big|\; \langle s,\alpha\rangle \in C,\ s' \in \mathit{post}(s,\alpha) \setminus \{s\}\big\}.$$

2. Build the *quotient* (cf. Definition 2.21) $M_{\setminus\mathfrak{C}} = \langle S_{\setminus\mathfrak{C}}, \mathit{Act}_{\setminus\mathfrak{C}}, \Delta_{\setminus\mathfrak{C}}, \mathbf{P}_{\setminus\mathfrak{C}}\rangle$ of M with respect to $\mathfrak{C}$.

3. Obtain MDP $\overline{M_{\setminus\mathfrak{C}}} = \langle S_{\setminus\mathfrak{C}}, \mathit{Act}_{\setminus\mathfrak{C}}, \Delta_{\setminus\mathfrak{C}}, \overline{\mathbf{P}_{\setminus\mathfrak{C}}}\rangle$ from $M_{\setminus\mathfrak{C}}$ by setting the outgoing transition probabilities for each collapsed EC state $C \in \mathfrak{C} \subseteq S_{\setminus\mathfrak{C}}$ of $M_{\setminus\mathfrak{C}}$, each enabled action $\alpha \in \Delta_{\setminus\mathfrak{C}}(C)$, and each successor state $s' \in S_{\setminus\mathfrak{C}}$ to

$$\overline{\mathbf{P}_{\setminus\mathfrak{C}}}(C,\alpha,s') \;\coloneqq\; l_C \cdot \mathbf{P}_{\setminus\mathfrak{C}}(C,\alpha,s') + (1{-}l_C) \cdot [\![s'{=}C]\!].$$

   In all other cases $\overline{\mathbf{P}_{\setminus\mathfrak{C}}}$ and $\mathbf{P}_{\setminus\mathfrak{C}}$ coincide. Hence, the self-loop probability for each collapsed EC state C is increased. Roughly speaking, this reflects the possibility to cycle through the EC C in M which potentially yields multiple visits of states $s \in \mathit{states}(C)$.

4. Compute upper bounds $\overline{\mathsf{V}_s}$ for the expected visiting times in $\overline{M_{\setminus\mathfrak{C}}}$ using Algorithm 8.1.

5. For the original MDP M consider for $s \in S^{\mathrm{fin}}$

$$\mathsf{V}_s = \begin{cases} \overline{\mathsf{V}_C} & \text{if } s \in \mathit{states}(C) \text{ for } C \in \mathfrak{C} \\ \overline{\mathsf{V}_s} & \text{otherwise.} \end{cases}$$

Example 8.2 We compute upper bounds for the expected visiting times for the MDP from Figure 8.3a. From Example 8.1 we get $S^{\text{fin}} = S$. We therefore have

$$\mathfrak{C} = MEC(M) = \Big\{ \underbrace{\{\langle s_1, \alpha\rangle, \langle s_2, \alpha\rangle\}}_{=:C_1}, \underbrace{\{\langle s_3, \alpha\rangle \langle s_4, \alpha\rangle \langle s_4, \beta\rangle \langle s_5, \beta\rangle \langle s_6, \alpha\rangle\}}_{=:C_2} \Big\}.$$

We compute $l_{C_1}, l_{C_2} \in (0, 1]$ as in Item 1 above by considering the product of the smallest probabilities occurring at each state of the corresponding MEC, yielding

$$l_{C_1} = 1 \cdot 1 = 1 \quad \text{and} \quad l_{C_2} = 1/3 \cdot 1 \cdot 1/4 \cdot 1 = 1/12.$$

The MDP $\overline{M_{\setminus \mathfrak{C}}}$ from Item 3 is depicted in Figure 8.3b. With respect to this MDP we have $\overline{S^{\text{fin}}} = \{s_0, C_1, C_2\}$. Algorithm 8.1 then computes

$$d_{s_\perp} = d_{C_1} = 1, \quad d_{s_0} = 1/6, \quad \text{and} \quad d_{C_2} = \min(1/60 \cdot 1/6, 1/12 \cdot 1) = 1/360.$$

From this, we get the following upper bounds V_s satisfying Equation (8.3) for each state s of the original MDP M:

$$\mathsf{V}_{s_j} = \begin{cases} 6 & \text{if } j = 0 \\ 1 & \text{if } j \in \{1, 2\} \\ 360 & \text{if } j \in \{3, 4, 5, 6\}. \end{cases}$$

Lemma 8.10 The values V_s computed for each state $s \in S^{\text{fin}}$ via the steps above satisfy Equation (8.3).

Proof. Let $s \in S^{\text{fin}}$ and $\sigma \in \Sigma^M_{\text{PM}}$ such that $0 < \text{Ex}^M_{\sigma,s_I}(visits(s)) < \infty$. We have to show that $\mathsf{V}_s \geq \text{Ex}^M_{\sigma,s_I}(visits(s))$.

- If $s \notin states(C)$ for each $C \in \mathfrak{C}$, considering the quotient with respect to $\mathfrak{C}$ preserves the maximal expected number of visits of s. To see this, recall that visiting times can be expressed as total reward objectives (cf. Lemma 8.6) and that quotients preserve total rewards (cf. Section 4.1.4 and [Alf97, Chapter 7]). The scaling of the probabilities at the states $C \in \mathfrak{C}$ of $\overline{M_{\setminus \mathfrak{C}}}$ (Item 3) does not affect visiting times of the remaining states of the quotient.
- If $s \in states(C)$ for some $C \in \mathfrak{C}$, we consider the following cases.
  - If $\Pr^M_{\sigma,s}\big(\Diamond(S \setminus states(C))\big) = 0$, there must be a BSCC $B \subseteq states(C)$ in $M[\![\sigma]\!]$ that is reached from s with positive probability. $\text{Ex}^M_{\sigma,s_I}(visits(s)) < \infty$

implies $s \notin B$. Moreover, it is not possible to visit s again once we are in B. Thus, the probability to reach B from s without visiting s a second time is also positive and lower-bounded by the value l_C computed in Item 1 above. Using the set $\Pi_{\#s=1}$ of paths that visit s exactly once as in Lemma 8.9, we get

$$\Pr^M_{\sigma,s}(\Pi_{\#s=1}) \geq \Pr^M_{\sigma,s}(\Pi_{\#s=1} \cap \Diamond B) \geq l_C. \qquad \blacksquare$$

Let σ' be a strategy for $\overline{M_{\setminus \mathfrak{C}}}$ that maximizes the probability to reach C and selects the action $\bot$ at C which leads to the dedicated sink state $s_\bot$ of $\overline{M_{\setminus \mathfrak{C}}}$ with probability l_C and back to C with probability $(1-l_C)$. With Lemma 8.9 we conclude

$$\begin{aligned}\mathrm{Ex}^M_{\sigma,s_I}(\mathit{visits}(s)) &= \frac{\Pr^M_{\sigma,s_I}(\Diamond\{s\})}{\Pr^M_{\sigma,s}(\Pi_{\#s=1})} \leq \frac{\Pr^M_{\sigma,s_I}(\Diamond\{s\})}{l_C} \\ &\leq \frac{\Pr^{\overline{M_{\setminus \mathfrak{C}}}}_{\sigma',s_I}(\Diamond\{C\})}{l_C} = \mathrm{Ex}^{\overline{M_{\setminus \mathfrak{C}}}}_{\sigma',s_I}(\mathit{visits}(C)) \leq \overline{\mathsf{V}_C} = \mathsf{V}_s.\end{aligned}$$

– If $\Pr^M_{\sigma,s}(\Diamond(S \setminus \mathit{states}(C))) > 0$ there must be a reachable state-action pair $\langle t, \alpha\rangle \in \mathit{exits}(C)$ with $\sigma(t) = \alpha$. The value l_C is a lower bound for the probability to reach $t \in \mathit{states}(C)$ from s without leaving $\mathit{states}(C)$ and without visiting s a second time. This yields

$$\begin{aligned}\Pr^M_{\sigma,s}(\Pi_{\#s=1}) &\geq \Pr^M_{\sigma,s}(\Pi_{\#s=1} \cap \Diamond\{t\}) \\ &\geq l_C \cdot \sum_{s' \in (S \setminus \mathit{states}(C))} \mathbf{P}(t,\alpha,s') \cdot \Pr^M_{\sigma,s'}(\Pi_{\#s=0}), \\ &\geq l_C \cdot \sum_{s' \in (S \setminus \mathit{states}(C))} \mathbf{P}(t,\alpha,s') \cdot \Pr^M_{\sigma,s'}(\Pi_{\#C=0}) > 0,\end{aligned}$$

where $\Pi_{\#C=0}$ denotes the set of paths that never visit a state in C. Since C is maximal, there must be at least one $s' \in \mathit{post}(t,\alpha)$ from which C is not almost surely reached, i.e., for which $\Pr^M_{\sigma,s'}(\Pi_{\#C=0}) > 0$. Consider a strategy σ' for $\overline{M_{\setminus \mathfrak{C}}}$ that selects the action $\langle t, \alpha\rangle$ at C and maximizes the probability to reach C for any other state. A similar reasoning as in the previous case yields

$$\mathrm{Ex}^M_{\sigma,s_I}(\mathit{visits}(s)) \leq \mathrm{Ex}^{\overline{M_{\setminus \mathfrak{C}}}}_{\sigma',s_I}(\mathit{visits}(C)) \leq \overline{\mathsf{V}_C} = \mathsf{V}_s.$$

8.3 A Mixed Integer Linear Programming Approach

We present a procedure for answering PM-MOA for MDP $M = \langle S, Act, \Delta, \mathbf{P} \rangle$ with $s_I \in S$, $\ell > 0$ total reachability reward objectives $\Phi = \langle tot(\mathcal{R}_1, G_1), \ldots, tot(\mathcal{R}_\ell, G_\ell) \rangle$, and point $\boldsymbol{p} \in \mathbb{R}^\ell$. There are exactly $|\Sigma^M_{\text{PM}}| = \prod_{s \in S} |\Delta(s)|$ many pure memoryless strategies for M. A naive algorithm to solve PM-MOA enumerates all strategies $\sigma \in \Sigma^M_{\text{PM}}$ and checks whether σ achieves the given point $\boldsymbol{p}$, i.e., whether $\text{Ex}^M_{\sigma,s_I}(\Phi) \geq \boldsymbol{p}$ holds. The latter can be checked using standard algorithms on the induced DTMC $M[\![\sigma]\!]$. In practice, however, such a brute-force approach is not feasible. For the MDPs that we consider in our experiments in Section 9.6, the number of pure memoryless strategies often exceeds 10^{1000}.

Instead, our approach is to encode the PM-MOA query as a mixed integer linear program (MILP). Recall from Section 2.1.3 that an MILP instance consists of

- a finite number of linear constraints, and
- a linear optimization function

over integral as well as real-valued variables. The task is to decide whether the constraints are feasible and if so, find a solution that maximizes the optimization function.

We provide a translation of the given PM-MOA instance to an instance for MILP that has a feasible solution iff $\boldsymbol{p} \in Ach_{\text{PM}}(\Phi)$. More specifically, in Section 8.3.1, we present an encoding that is applicable to MDPs with just a single end component and for which the expected total reward is finite under every strategy. In Section 8.3.2, we extend our encoding towards MDPs with multiple end components. Finally, in Section 8.3.3 we also lift the restriction to finite total rewards, thus dealing with arbitrary MDPs. Section 8.4 further presents an alternative to the encoding which is smaller but restricted to *total* reward objectives. The general approach in both MILP formulations is to consider integer variables to encode a pure memoryless strategy $\sigma \in \Sigma^M_{\text{PM}}$. Other variables and constraints then encode the expected total reachability reward for each objective on the induced DTMC $M[\![\sigma]\!]$. Special care needs to be taken to deal with the BSCCs of $M[\![\sigma]\!]$.

> **Remark 8.1 (Quantitative Achievability)** Our MILP encodings presented below only rely on checking feasibility of a set of linear inequalities and do not make use of an optimization function. This enables a straightforward extension towards multi-objective *quantitative* achievability queries (MOQA, cf. Problem 3.2 on page 78) under *simple*—i.e., pure and memoryless—strategies. Section 8.5 further uses the MILP approach to approximate the set $Ach_{\text{PM}}(\Phi)$.

8.3.1 MDPs with a Single End Component and Finite Rewards

For simplicity, we first explain our encoding under the following assumption which we lift in subsequent sections.

Assumption 8.2 The MDP M has exactly one end component, i.e., $|EC(M)| = |\{C_0\}| = 1$. Furthermore, no reward is collected in this EC, i.e., $\forall\, j \in \{1..\ell\}\colon \mathcal{R}_j[\![C_0]\!] = 0$.

The unique EC $C_0 \in EC(M)$ must be a bottom component, i.e., $exits(C_0) = \emptyset$. Furthermore, C_0 is also the unique BSCC of the induced DTMC $M[\![\sigma]\!]$ for any $\sigma \in \Sigma^M_{\mathrm{PM}}$. Using [BK08, Theorem 10.27]—or alternatively Lemma 2.4 on page 41—we get for all $\sigma \in \Sigma^M_{\mathrm{PM}}$ and $s \in S$ that

$$\Pr^M_{\sigma,s}\big(\Diamond states(C_0)\big) \;=\; \Pr^{M[\![\sigma]\!]}_s\big(\Diamond states(C_0)\big) \;=\; 1,$$

i.e., the EC C_0 is reached almost surely from every state. Let $j \in \{1..\ell\}$. As no further rewards are collected once the bottom component C_0 is reached, the above also implies $\mathrm{Ex}^M_{\sigma,s}(tot(\mathcal{R}_j, G_j)) \in \mathbb{R}$. Our `MILP` encoding considers the sets of states

$$S^j_0 \;\coloneqq\; \big\{s \in S \;\big|\; \forall\, \sigma \in \Sigma^M\colon\; \mathrm{Ex}^M_{\sigma,s}\big(tot(\mathcal{R}_j, G_j)\big) = 0\big\} \quad \text{and}$$
$$S^j_? \;\coloneqq\; \big\{s \in S \setminus S^j_0 \;\big|\; s \text{ can be reached from } s_I \text{ without visiting a state in } S^j_0\big\}.$$

These sets can be obtained by simple reachability analysis on the graph structure of MDP M. We have $G_j \subseteq S^j_0$ and $states(C_0) \subseteq S^j_0$. The latter yields that S^j_0 is reached almost surely from every state and under every strategy. There is no need to explicitly encode the value for objective $tot(\mathcal{R}_j, G_j)$ at states $s \notin S^j_?$:

- if $s \in S^j_0$, the objective value at s is known to be 0, and
- if $s \in S \setminus (S^j_0 \cup S^j_?)$, the value at s is irrelevant for the total reward accumulated from the initial state s_I.

Our `MILP` encoding further applies the well-known characterization of expected total rewards for DTMCs as a *linear equation system* (e.g., [BK08, Section 10.5]).

Theorem 8.11 For every $j \in \{1..\ell\}$ and $\sigma \in \Sigma^M_{\mathrm{PM}}$, the equation system

$$\forall\, s \in S^j_0\colon\; x_s \;=\; 0$$
$$\forall\, s \in S^j_?\colon\; x_s = \sum_{s' \in S} \mathbf{P}\big(s, \sigma(s), s'\big) \cdot \big(x_{s'} + \mathcal{R}_j[s, \sigma(s), s']\big).$$

$\forall s \in S$: ▷ *Select a unique action at each state*

$\quad \forall \alpha \in \Delta(s)$: $a_{s,\alpha} \in \{0,1\}$ (1)

$\qquad \sum_{\alpha \in \Delta(s)} a_{s,\alpha} = 1$ (2)

$\forall j \in \{1..\ell\}$: ▷ *Compute expected reachability reward values*

$\quad \forall s \in S_0^j$: $x_s^j = 0$ (3)

$\quad \forall s \in S_?^j$: $x_s^j \in [\mathsf{L}_s^j, \mathsf{U}_s^j]$ (4)

$\quad\quad \forall \alpha \in \Delta(s)$: $x_s^j \leq (1 - a_{s,\alpha}) \cdot (\mathsf{U}_s^j - \mathsf{L}_s^j) + \sum_{s' \in post(s,\alpha)} \mathbf{P}(s,\alpha,s') \cdot (x_{s'}^j + \mathcal{R}_j[s,\alpha,s'])$ (5)

$\quad\quad x_{s_I}^j \geq \boldsymbol{p}(j)$ ▷ *Assert objective value at initial state* (6)

Figure 8.4: MILP encoding for MDPs with one EC and finite rewards

has a unique solution $val\colon \{x_s \mid s \in S_0^j \cup S_?^j\} \to \mathbb{R}$ satisfying $val(x_s) = \mathrm{Ex}_{\sigma,s}^M(tot(\mathcal{R}_j, G_j))$.

Proof. As outlined above, we have

$$\forall \sigma \in \Sigma^M\colon\ \forall s \in S\colon\ \Pr_{\sigma,s}^M(\Diamond S_0^j) = 1.$$

The set S_0^j is therefore *attracting* (cf. Definition 4.2 on page 112). By noting that $tot(\mathcal{R}_j, G_j) = tot(\mathcal{R}_j, S_0^j)$, the theorem is a special instance of Theorem 4.8 on page 116 when applied to the DTMC $M[\![\sigma]\!]$. ■

Figure 8.4 shows the MILP encoding over variables

$$Vars = \left\{ a_{s,\alpha},\ x_s^j \;\middle|\; s \in S, \alpha \in \Delta(s) \right\}$$

in case Assumption 8.2 holds. We discuss the intuition of each constraint. Let $val\colon Vars \to \mathbb{R}$ be an assignment of variables to values. val is a *solution* of the constraint system if all (in)equations are satisfied upon replacing all variables $v \in Vars$ by $val(v)$.

Strategy Encoding | Constraints 1 and 2 encode a strategy $\sigma \in \Sigma_{\mathrm{PM}}^M$ by considering a binary variable $a_{s,\alpha}$ for each state s and enabled action α such that for a solution val we have

$$val(a_{s,\alpha}) = 1 \quad \text{iff} \quad \sigma(s) = \alpha.$$

Due to Constraint 2, exactly one action is chosen at each state.

Objective Values Encoding | Constraints 3 to 5 encode for each objective $\Phi(j) = tot(\mathcal{R}_j, G_j)$ the expected reachability reward obtained for the encoded strategy σ. For every $s \in S$, the variable x_s^j represents a lower bound of the value at s, i.e., for a solution *val* the constraints ensure

$$val(x_s^j) \;\leq\; \mathrm{Ex}_{\sigma,s}^{M}\big(tot(\mathcal{R}_j, G_j)\big).$$

Constraint 3 sets the value for all $s \in S_0^j$ to zero—reflecting the analogous case from Theorem 8.11. Constraint 4 bounds the variables x_s^j for $s \in S_?^j$ using *lower- and upper bounds* $\mathsf{L}_s^j, \mathsf{U}_s^j \in \mathbb{R}$ that satisfy

$$\forall\, \sigma \in \Sigma^M\colon\;\; \mathrm{Ex}_{\sigma,s}^{M}\big(tot(\mathcal{R}_j, G_j)\big) \;\in\; [\mathsf{L}_s^j,\; \mathsf{U}_s^j].$$

Such bounds are obtained via a pre-computation using single-objective model checking techniques for total reachability reward objectives as outlined in Chapter 4. In Constraint 5, we consider two cases for $\alpha \in \Delta(s)$.

- If the encoded strategy σ selects α (i.e., $val(a_{s,\alpha}) = 1$), the constraint asserts—using the equation system characterization from Theorem 8.11—that the value of x_s^j can not be greater than the expected reward at s, that is

$$val(x_s^j) \;\leq\; \mathrm{Ex}_{\sigma,s}^{M}\big(tot(\mathcal{R}_j, G_j)\big).$$

- Otherwise, the constraint yields

$$(\mathsf{U}_s^j - \mathsf{L}_s^j) + \underbrace{\sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot \big(val(x_{s'}^j) + \mathcal{R}_j[s,\alpha,s']\big)}_{\geq \mathsf{L}_s^j} \;\geq\; \mathsf{U}_s^j \geq val(x_s^j)$$

  which is already implied by Constraint 4.

Achievability Encoding | Constraint 6 and our observations above yield

$$\forall\, j \in \{1..\ell\}\colon\; \boldsymbol{p}(j) \;\leq\; val(x_{s_I}^j) \;\leq\; \mathrm{Ex}_{\sigma,s_I}^{M}\big(tot(\mathcal{R}_j, G_j)\big).$$

Therefore, $\boldsymbol{p}$ is achievable if a solution *val* for the constraints in Figure 8.4 exists. On the other hand, if $\boldsymbol{p}$ is achievable by some $\sigma \in \Sigma_{\mathrm{PM}}^M$, then $val\colon Vars \to \mathbb{R}$ is a solution for the constraints, where for $s \in S$ and $\alpha \in \Delta(s)$

- $val(a_{s,\alpha}) \;=\; \big[\sigma(s) = \alpha\big]$ and

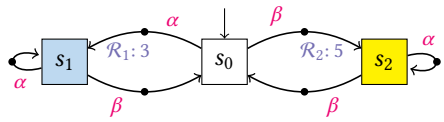


Figure 8.5: MDP M with multiple end components (cf. Examples 8.3 and 8.4)

- $val(x_s^j) = \mathrm{Ex}^M_{\sigma,s}\big(tot(\mathcal{R}_j, G_j)\big)$.

Theorem 8.12 If Assumption 8.2 holds, the constraints in Figure 8.4 are feasible iff $\boldsymbol{p} \in Ach_{\mathrm{PM}}(\Phi)$.

Proposition 8.13 The number of variables in the MILP encoding in Figure 8.4 satisfies

$$|Vars| \in \mathcal{O}(|S| \cdot |Act| \cdot \ell).$$

8.3.2 Dealing with Multiple End Components

EC, end component

We now lift the restriction to a single end component, i.e., we allow that $|EC(M)| > 1$ holds. The key challenge for extending our MILP encoding is that the equation system in Theorem 8.11 does not necessarily have a unique solution anymore.

Example 8.3 The MDP in Figure 8.5 has multiple end components

$$EC(M) = \big\{\{\langle s_1, \alpha\rangle\},\ \{\langle s_0, \alpha\rangle, \langle s_1, \beta\rangle\},\ \{\langle s_0, \alpha\rangle, \langle s_1, \alpha\rangle, \langle s_1, \beta\rangle\},\ \ldots\big\}.$$

Consider the objectives $\Phi = \big\langle tot(\mathcal{R}_1, \{s_1\}), tot(\mathcal{R}_2, \{s_2\})\big\rangle$. We have $S_0^1 = \{s_1\}$, $S_?^1 = \{s_0, s_2\}$, $S_0^2 = \{s_2\}$, and $S_?^2 = \{s_0, s_1\}$. For strategy $\sigma \in \Sigma^M_{\mathrm{PM}}$ with $\sigma(s_i) = \alpha$ for $i \in \{0, 1, 2\}$ we get

$$\mathrm{Ex}^M_{\sigma,s_0}\big(tot(\mathcal{R}_1, \{s_1\})\big) = 3 \quad \text{and} \quad \mathrm{Ex}^M_{\sigma,s_0}\big(tot(\mathcal{R}_2, \{s_2\})\big) = 0.$$

However, for the second objective the equation system in Theorem 8.11 is given by

$$x_{s_0} = 1 \cdot x_{s_1} \qquad x_{s_1} = 1 \cdot x_{s_1} \qquad x_{s_2} = 0.$$

For every $v \in \mathbb{R}$, $val\colon \{x_{s_0}, x_{s_1}, x_{s_2}\} \to \mathbb{R}$ with $val(x_{s_0}) = val(x_{s_1}) = v$ and $val(x_{s_2}) = 0$ is a solution for this equation system. As a consequence, the MILP from Figure 8.4 applied to this example has a solution for, e.g., point $\boldsymbol{p} = \langle 3, v\rangle$ with $v > 0$—even though $\boldsymbol{p}$ is not achievable.

Recall the sets S_0^j and $S_?^j$ for some objective $tot(\mathcal{R}_j, G_j)$ as given on page 249. If the MDP has multiple ECs, it can be the case that for some strategy σ the set S_0^j is not reached with probability 1, i.e., there is a positive probability to stay in the set $S_?^j$ forever. For the induced Markov chain $M[\![\sigma]\!]$, this means that there is a reachable BSCC consisting only of states in $S_?^j$, i.e., there is a set $B \subseteq S_?^j$ such that $\forall\, s \in B\colon\ post(\langle s, \sigma(s)\rangle) \subseteq B$.

Since the potential BSCCs of $M[\![\sigma]\!]$ coincide with ECs of M, we need to inspect the ECs that only consist of $S_?^j$-states, given by

$$EC_{\mathsf{p}}(M,\Phi) \;\coloneqq\; \left\{C \in EC(M) \;\middle|\; \exists\, j \in \{1..\ell\}\colon\ states(C) \subseteq S_?^j\right\}. \tag{8.4}$$

We call the ECs $C \in EC_{\mathsf{p}}(M,\Phi)$ *problematic* and denote by $MEC_{\mathsf{p}}(M,\Phi)$ the set of *maximal* problematic ECs

problematic EC

$$MEC_{\mathsf{p}}(M,\Phi) \;\coloneqq\; \{C \in EC_{\mathsf{p}}(M,\Phi) \mid \nexists\, C' \in EC_{\mathsf{p}}(M,\Phi)\colon\ C \subsetneq C'\}.$$

The set $MEC_{\mathsf{p}}(M,\Phi)$ can be obtained by computing for each $j \in \{1..\ell\}$ the MECs of an MDP that corresponds to M after dropping transitions from $S \setminus S_?^j$ to $S_?^j$. A state $s \in S$ can be contained in multiple problematic MECs originating from different objectives.

Example 8.4 For MDP M in Figure 8.5 and objectives Φ as in Example 8.3, the maximal problematic ECs are given by

$$MEC_{\mathsf{p}}(M,\Phi) \;=\; \big\{\{\langle s_0,\alpha\rangle, \langle s_1,\alpha\rangle, \langle s_1,\beta\rangle\},\ \{\langle s_0,\beta\rangle, \langle s_2,\alpha\rangle, \langle s_2,\beta\rangle\}.\big\}$$

The strategy $\sigma \in \Sigma_{\mathrm{PM}}^M$ with $\sigma(s_i) = \alpha$ for $i \in \{0,1,2\}$ from Example 8.3 induces the BSCC $B = \{s_1\} \subseteq S_?^2$.

For now, we require that *expected total reachability rewards are always finite*. The following assumption ensures this, since infinite expected reward for the j^{th} objective can only be accumulated within an EC consisting of states in $S_?^j$. We lift the assumption in the next section.

Assumption 8.3 For $j \in \{1..\ell\}$, $C \in EC(M)$: if $states(C) \subseteq S_?^j$ then $\mathcal{R}_j[\![C]\!] = 0$.

Our goal is to discard `MILP` solutions that correspond to unachievable values—such as the solution from Example 8.3. For this, we provide two alternative approaches which both extend the `MILP` encoding from Figure 8.4. Similar encodings in different contexts have been presented in, e.g., [WJÁK+14; Vel19; WWJB20].

$$\forall\, C \in MEC_{\mathsf{p}}(M,\Phi)\colon \qquad \triangleright \textit{ Handle BSCCs } B \subseteq \textit{states}(C) \textit{ of } M[\![\sigma]\!]$$

$$\triangleright \textit{ Assert ``If there is a BSCC } B \textit{ then } \exists\, s \in B\colon\ b_s^C{=}1\textit{''}$$

$$\forall\, s \in \textit{states}(C)\colon \qquad b_s^C \in \{0,1\} \tag{7}$$

$$z_{s,\bot}^C \in [0, \mathsf{V}_s \cdot b_s^C] \tag{8}$$

$$\forall\, \alpha \in \Delta(s)\colon \qquad z_{s,\alpha}^C \in [0, \mathsf{V}_s \cdot a_{s,\alpha}] \tag{9}$$

$$z_{s,\bot}^C + \sum_{\alpha\in\Delta(s)} z_{s,\alpha}^C = \frac{1}{|\textit{states}(C)|} + \sum_{\langle s',\alpha'\rangle \in \textit{pre}(s)\cap C} \mathbf{P}(s',\alpha',s)\cdot z_{s',\alpha'}^C \tag{10}$$

$$1 = \sum_{s\in\textit{states}(C)} \left(z_{s,\bot}^C + \sum_{\alpha\in\Delta(s),\langle s,\alpha\rangle\notin C} z_{s,\alpha}^C \right) \tag{11}$$

$$\forall\, j \in \{1..\ell\} \text{ with } \textit{states}(C) \subseteq S_?^j\colon \qquad \triangleright \textit{ Assert ``} s \in \textit{BSCC} \implies x_s^j \le 0\textit{''}$$

$$\forall\, s \in \textit{states}(C)\colon \qquad x_s^j \le \mathsf{U}_s^j \cdot (1 - b_s^C) \tag{12}$$

Figure 8.6: MILP encoding based on visiting times for detection of BSCCs

Visiting Times Approach

The first approach is based on the equation system for expected visiting times given in Theorem 8.8. More specifically, we exploit that the equation system only has a solution if a given set S' of states is left almost surely. This is used to detect whether the strategy from the MILP encoding induces a BSCC $B \subseteq \textit{states}(C)$ for some maximal problematic $C \in MEC_{\mathsf{p}}(M,\Phi)$. To this end, we consider the constraints from Figure 8.4 in conjunction with the constraints from Figure 8.6. Let $\textit{val}$ be a solution of these constraints and let $\sigma \in \Sigma_{\text{PM}}^M$ be the encoded strategy with

$$\textit{val}(a_{s,\alpha}) \;=\; 1 \quad \text{iff} \quad \sigma(s) \;=\; \alpha.$$

Detecting BSCCs | Constraints 7 to 11 ensure that if a BSCC $B \subseteq \textit{states}(C)$ exists in $M[\![\sigma]\!]$, then $\textit{val}(b_s^C) = 1$ holds for at least one $s \in B$. The idea is based on Theorem 8.8 and the observation that if such a BSCC B exists, there is a state $s \in B \subseteq \textit{states}(C)$ that does not reach the set $S \setminus \textit{states}(C)$ almost surely. For each $C \in MEC_{\mathsf{p}}(M,\Phi)$, we consider the equation system from Theorem 8.8 for an auxiliary MDP $M_C = \langle S_C, \textit{Act}_C, \Delta_C, \mathbf{P}_C\rangle$ which intuitively extends the submodel $M[\![C]\!]$ with two new states: a state s_I^C (which serves as initial state and from which all states $s \in \textit{states}(C)$ are reached with positive probability) and a state $s_\bot^C$ (to which transitions exiting the EC are redirected). Additionally, all states $s \in \textit{states}(C)$ get an extra action $\alpha_\bot^C$ leading to $s_\bot^C$ which we use in our encoding to detect BSCCs within C. Formally, we set

- $S_C = \textit{states}(C) \uplus \{s_I^C, s_\bot^C\}$,

- $Act_C = Act \uplus \{\alpha^C_\bot\}$,
- $\Delta_C(s^C_I) = \Delta_C(s^C_\bot) = \{\alpha^C_\bot\}$,
- for $s \in states(C)$: $\Delta_C(s) = \Delta(s) \cup \{\alpha^C_\bot\}$, and
- for $s, s' \in S_C$ and $\alpha \in \Delta_C(s)$:

$$\mathbf{P}_C(s,\alpha,s') = \begin{cases} \mathbf{P}(s,\alpha,s') & \text{if } \{s,s'\} \subseteq states(C), \alpha \in Act \\ \sum_{s'' \in S \setminus states(C)} \mathbf{P}(s,\alpha,s'') & \text{if } s \in states(C), s' = s^C_\bot, \alpha \in Act \\ 1/|states(C)| & \text{if } s = s^C_I, s' \in states(C), \alpha = \alpha^C_\bot \\ 1 & \text{if } s \in states(C) \cup \{s^C_\bot\}, s' = s^C_\bot, \alpha = \alpha^C_\bot \\ 0 & \text{otherwise.} \end{cases}$$

The MILP encoding considers variables $z^C_{s,\alpha}$ for each pair of state $s \in states(C)$ and action $\alpha \in \Delta(s)$. The values V_s in Constraints 8 and 9 are upper bounds for the maximal expected number of visits of each $s \in states(C)$ in the auxiliary MDP M_C. Their computation is outlined in Section 8.2.3. Constraint 9 ensures that $z^C_{s,\alpha}$ can only have a non-zero value if α is the selected action at s. It further discards negative solutions. Constraints 10 and 11 then reflect the equation system from Theorem 8.8. Here, additional variables $z^C_{s,\bot}$ are considered that—according to Constraint 8—can only be set to a non-zero value if the binary variable b^C_s is set to 1.

Assume a solution *val* such that $\forall\, s \in states(C)\colon\ val(z^C_{s,\bot}) = 0$. Then, *val* also yields a solution for the equation system from Theorem 8.8 with respect to the auxiliary MDP M_C and the encoded strategy σ (which can straightforwardly be applied to M_C). Consequently, we have that $\Pr^{M_C}_{\sigma,s^C_I}(\Diamond\{s^C_\bot\}) = 1$, i.e., there is no BSCC $B \subseteq states(C)$ in $M[\![\sigma]\!]$.

On the other hand, the constraints still allow encodings of arbitrary strategies that potentially induce BSCCs $B \in states(C)$. This, however, requires to set b^C_s to 1 for some $s \in B$, enabling non-zero values for $z^C_{s,\bot}$ which in M_C reflects a strategy that selects $\alpha^C_\bot$ at $s \in B$. Intuitively speaking, setting b^C_s to 1 "deflates" the expected number of visits of a BSCC state $s \in B$—which would otherwise accumulate to infinity.

Discarding Unachievable Solutions | Consider $j \in \{1..\ell\}$ and a BSCC B of $M[\![\sigma]\!]$ contained in a problematic EC C such that $B \subseteq states(C) \subseteq S^j_?$. Due to Assumption 8.3, C can not contain a transition with non-zero reward with respect to $\mathcal{R}_j$, yielding $\mathrm{Ex}^M_{\sigma,s}(tot(\mathcal{R}_j, G_j)) = 0$ for all $s \in B$. The constraints discussed so far imply that some $s \in B$ with $val(b^C_s) = 1$ exists. Constraint 12 ensures that x^j_s can not be set to a value larger than 0 in this case. Since all states $s' \in B$ almost surely reach $s \in B$ with

$val(b_s^C) = 1$, we get

$$\forall s \in B\colon\ val(x_s^j) \ \leq\ 0 \ =\ \mathrm{Ex}^M_{\sigma,s}(tot(\mathcal{R}_j, G_j)).$$

Theorem 8.14 If Assumption 8.3 holds, the constraints from Figures 8.4 and 8.6 are feasible iff $\boldsymbol{p} \in Ach_{\mathrm{PM}}(\Phi)$.

Reachability Order Approach

The second approach for detecting BSCCs induced by the encoded strategy $\sigma \in \Sigma^M_{\mathrm{PM}}$ is based on the following lemma. A similar result is used in [WJÁK+14].

Lemma 8.15 For $S' \subseteq S$ we have $\forall s \in S\colon\ \Pr^M_{\sigma,s}(\Diamond S') > 0$ iff there is a strict partial order $\prec \subseteq S \times S$ such that

$$\forall s \in S \setminus S'\colon\ \exists s' \in post(s, \sigma(s))\colon\ s \prec s'.$$

Proof. For $s \in S$ we have

$$\Pr^M_{\sigma,s}(\Diamond S') > 0 \quad \text{iff} \quad \exists\, \hat{\pi} \in Paths^{M[\![\sigma]\!]}_{\mathrm{fin}}(s)\colon\ last(\hat{\pi}) \in S'.$$

Let d_s be the length of the shortest such path for s. Then, $\prec = \{\langle s, s'\rangle \mid d_s > d_{s'}\}$ satisfies the condition from the lemma.
On the other hand, if such an order $\prec$ exists, we can construct for every $s \in S$ a path from s to a state $s' \in S'$ by considering a chain

$$s = s_0 \prec s_1 \prec \ldots \prec s_{n-1} \prec s_n = s'$$

with $n \leq |S \setminus S'|$, $s_i \in S \setminus S'$, and $s_{i+1} \in post(s_i, \sigma(s_{i+1}))$ for $i \in \{0..n{-}1\}$. Such a chain must exist due to the strictness of $\prec$ and the well-known pigeonhole principle. ■

The constraints in Figure 8.7 use Lemma 8.15 to detect if the encoded strategy induces a BSCC inside a problematic EC. Only Constraints 14 to 17 are different from Figure 8.6. As in the approach based on visiting times, the constraints ensure that for each BSCC $B \in states(C)$ there is one state $s \in B$ for which b_s^C is set to 1. To see this, assume a solution val for the constraints which defines a strategy $\sigma \in \Sigma^M_{\mathrm{PM}}$ as before as well as a partial order

$$\prec \ =\ \left\{\langle s, s'\rangle \in states(C) \times states(C) \;\middle|\; val(r_s^C) < val(r_{s'}^C)\right\}$$

$\forall\, C \in MEC_{\mathsf{p}}(M,\Phi)\colon$ ▷ *Handle BSCCs $B \subseteq states(C)$ of $M[\![\sigma]\!]$*

▷ *Assert "If there is a BSCC B then $\exists\, s \in B\colon\ b_s^C{=}1$"*

$\forall\, s \in states(C)\colon \quad b_s^C \in \{0,1\}$ (13)

$r_s^C \in [0,1]$ (14)

$\forall\, \langle s,\alpha\rangle \in C\colon \quad a_{s,\alpha} \le b_s^C + \sum_{s' \in post(s,\alpha)} d_{s,\alpha,s'}^C$ (15)

$\forall\, s' \in post(s,\alpha)\colon \quad d_{s,\alpha,s'}^C \in \{0, a_{s,\alpha}\}$ (16)

$r_s^C \le \frac{-1}{|states(C)|} + r_{s'}^C + (1 - d_{s,\alpha,s'}^C)$ (17)

$\forall\, j \in \{1..\ell\}$ with $states(C) \subseteq S_?^j\colon$ ▷ *Assert "$s \in BSCC \implies x_s^j \le 0$"*

$\forall\, s \in states(C)\colon \quad x_s^j \le \mathsf{U}_s^j \cdot (1 - b_s^C)$ (18)

Figure 8.7: `MILP` encoding based on reachability order for detection of BSCCs

and a set of states

$$S' \;=\; \big\{ s \in states(C) \;\big|\; \langle s,\sigma(s)\rangle \in exits(C) \text{ or } val(b_s^C) = 1 \big\}.$$

Constraints 15 and 16 imply that for any $s \in states(C) \setminus S'$ there is some $s' \in post(s,\sigma(s))$ with $val(d_{s,\sigma(s),s'}^C) = 1$. Constraint 17 ensures that for such states s and s' we have[3] $val(r_s^C) < val(r_{s'}^C)$, i.e., $s \prec s'$. The order $\prec$ and the set S' satisfy all conditions from Lemma 8.15, yielding $\Pr_{\sigma,s}^M(\Diamond S') > 0$ for all $s \in states(C) \setminus S'$. Consequently, every BSCC $B \subseteq states(C)$ of $M[\![\sigma]\!]$ must contain a state $s \in S'$ to ensure that reaching S' from B is possible, i.e., there is $s \in B \cap S'$. Since s lies on a BSCC, we have $\langle s,\sigma(s)\rangle \notin exits(C)$. From $s \in S'$ we then conclude that $val(b_s^C) = 1$ must hold. We obtain correctness of the encoding by arguing as for the previously discussed variant based on expected visiting times.

Theorem 8.16 If Assumption 8.3 holds, the constraints from Figures 8.4 and 8.7 are feasible iff $\boldsymbol{p} \in Ach_{\mathrm{PM}}(\Phi)$.

8.3.3 Encoding for Arbitrary MDPs with Infinite Rewards

We finally consider *pure memoryless* achievability under arbitrary MDPs, i.e., we also drop Assumption 8.3. The only remaining assumption is that the total reachability

[3] The term $-1 / |states(C)|$ in Constraint 17 yields that $val(r_s^C)$ is *strictly* smaller than $val(r_{s'}^C)$. This value is small enough to allow an encoding of every possible total order $\prec$ on $states(C)$. Strict inequalities are technically not allowed in an `MILP` constraint.

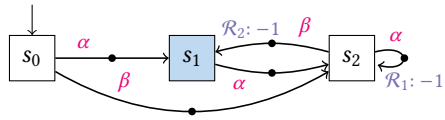


Figure 8.8: MDP M with potentially infinite total reachability reward (cf. Example 8.5)

reward objectives are convergent, yielding that for each objective $tot(\mathcal{R}_j, G_j)$ ($j \in \{1..\ell\}$) we either have (cf. Section 4.1.3)

- Case 1: $\forall\, C \in EC(M)\colon\ \mathcal{R}_j[\![C]\!] \leq 0$ or
- Case 2: $\forall\, C \in EC(M)\colon\ \mathcal{R}_j[\![C]\!] \geq 0$.

In the following, we consider the two cases for a given objective $tot(\mathcal{R}_j, G_j)$.

Case 1 | Assume a strategy $\sigma \in \Sigma^M_{\text{PM}}$ such that there is a BSCC $B \subseteq S^j_?$ of $M[\![\sigma]\!]$ that contains a transition with *negative* reward. For this BSCC, we can show that the inequality system

$$\forall\, s \in B\colon\quad x_s \;\leq\; \sum_{s' \in B} \mathbf{P}\big(s, \sigma(s), s'\big) \cdot \big(x_{s'} + \mathcal{R}_j[s, \sigma(s), s']\big) \tag{8.5}$$

over real-valued variables $\{x_s \mid s \in B\}$ has no solution. Consequently, the MILP constraints in Figure 8.4 have no solution that encodes the strategy σ. If (in $M[\![\sigma]\!]$) the BSCC B is reachable from a state $s \in S$ without visiting a state in G_j before (i.e., $\Pr^M_{\sigma,s}\big(S^j_? \;\mathcal{U}\; B\big) > 0$), we get $\text{Ex}^M_{\sigma,s}\big(tot(\mathcal{R}_j, G_j)\big) = -\infty$. When this is true for the initial state s_I, discarding the strategy σ in the MILP encoding is correct as the value $\boldsymbol{p}(j) \in \mathbb{R}$ is not achieved. However, if $\Pr^M_{\sigma,s}\big(S^j_? \;\mathcal{U}\; B\big) = 0$, the strategy σ might still achieve the given point, i.e., it should not be discarded[4].

Example 8.5 Consider the MDP M from Figure 8.8 with objectives $\Phi = \big\langle tot(\mathcal{R}_1, \{s_1\}), tot(\mathcal{R}_2)\big\rangle$. We have $S^1_? = \{s_0, s_2\}$ and $S^2_? = \{s_0, s_1, s_2\}$. The strategy $\sigma \in \Sigma^M_{\text{PM}}$ with $\sigma(s_0) = \sigma(s_1) = \sigma(s_2) = \alpha$ is the only strategy that achieves the point $\boldsymbol{p} = \langle 0, 0\rangle$. However, for $j = 1$ and the BSCC $B = \{s_2\}$ of $M[\![\sigma]\!]$, the inequality system in Equation (8.5) has no solution since

$$x_{s_2} \;\leq\; 1 \cdot (x_{s_2} - 1) \qquad \text{iff} \qquad 0 \;\leq\; -1.$$

Consequently, the constraints from Figure 8.4 for M, Φ, and $\boldsymbol{p} = \langle 0, 0\rangle$ are infeasi-

[4] This case was missing in the originating paper [9].

$$\forall\, j \in \{1..\ell\} \text{ with } s_I \in S^j_{-\infty?} \colon\ \forall\, s \in S^j_{-\infty?} \colon$$

$$c^j_s \in \{0,1\} \qquad \triangleright \textit{ Indicate reachability of } s \qquad (19)$$

$$\forall\, \alpha \in \Delta(s) \colon \qquad x^j_s \le \big(1 - a_{s,\alpha}\big) \cdot \big(\mathsf{U}^j_s - \mathsf{L}^j_s\big) + \sum_{s' \in post(s,\alpha)} \mathbf{P}(s,\alpha,s') \cdot \big(x^j_{s'} + c^j_s \cdot \mathcal{R}_j[s,\alpha,s']\big) \qquad (20)$$

$$\forall\, s' \in post(s,\alpha) \cap S^j_{-\infty?} \colon \quad c^j_s \le c^j_{s'} + 1 - a_{s,\alpha} \qquad \triangleright \textit{ Successor reachability} \qquad (21)$$

$$c^j_{s_I} = 1 \qquad \triangleright \textit{ Initial state reachability} \qquad (22)$$

Figure 8.9: Additional constraints for objectives with $s_I \in S^j_{-\infty?}$—replacing Constraint 5 from Figure 8.4

ble for this case, even though $\boldsymbol{p} \in Ach_{\mathrm{PM}}(\Phi)$.

Let

$$S^j_{-\infty?} \;\coloneqq\; \big\{ s \in S^j_? \;\big|\; \mathrm{Ex}^M_{\min,s}\big(tot(\mathcal{R}_j, G_j)\big) = -\infty \big\}.$$

The definition of $S^j_?$ on page 249 yields that either $S^j_{-\infty?} = \emptyset$ or $s_I \in S^j_{-\infty?}$. The above-mentioned issue can only occur in the latter case. If $s_I \in S^j_{-\infty?}$, we replace Constraint 5 from Figure 8.4 for all $s \in S^j_{-\infty?}$ by the constraints from Figure 8.9.

The constraints introduce binary variables c^j_s for each state $s \in S^j_{-\infty?}$. The idea is that reward collected at state s can be ignored by setting c^j_s to 0. For a BSCC B containing negative reward, this enables solutions that set both x^j_s and c^j_s to 0 for any $s \in B$. However, Constraints 21 and 22 enforce c^j_s to be set to 1 for all states that under the encoded strategy are reached from s_I via states in $S^j_{-\infty?}$. In particular, Constraint 21 is equivalent to the statement

$$\text{if } c^j_s = 1 \text{ then } (c^j_{s'} = 1 \text{ or } a_{s,\alpha} = 0).$$

It follows that the reward collected at a state $s \in S^j_{-\infty?}$ can be ignored *if* it is not relevant for the value at the initial state.

It remains to find appropriate constants $\mathsf{L}^j_s, \mathsf{U}^j_s \in \mathbb{R}$ used in the MILP constraints in Figure 8.4. If $s \in S^j_{-\infty?}$, we set

$$\mathsf{U}^j_s \;=\; \max\Big\{0,\ \mathrm{Ex}^M_{\sigma,s}\big(tot(\mathcal{R}_j, G_j)\big)\Big\}$$

to allow for the above-mentioned trivial solution that sets both x^j_s and c^j_s to 0 for any state s in a BSCC. For the lower bounds L^j_s, we must consider all strategies that yield

finite expected reward at s, i.e., we require

$$\mathsf{L}_s^j \;\leq\; \min\Big(\Big\{\mathrm{Ex}_{\sigma,s}^{M}\big(\mathit{tot}(\mathcal{R}_j, G_j)\big) \;\Big|\; \sigma \in \Sigma_{\mathrm{PM}}^{M}\Big\} \cap \mathbb{R}\Big).$$

To compute appropriate values, we make use of our observations from Section 8.2. Let $\mathfrak{C}_j$ be the set of maximal end components in $\{C \in EC(M) \mid \mathit{states}(C) \subseteq S_?^j\}$. We define a reward assignment $\mathcal{R}_{\mathfrak{C}_j}$ for the quotient model $M_{\setminus \mathfrak{C}_j}$, where for $C \in \mathfrak{C}_j$, enabled action $\hat{\alpha} \in \mathit{exits}(C) \cup \{\bot\}$, and state $\hat{s}$ of $M_{\setminus \mathfrak{C}_j}$ we set

$$\mathcal{R}_{\mathfrak{C}_j}[C, \hat{\alpha}, \hat{s}] \;\coloneqq\; \sum_{s \in \mathit{states}(C)} \mathsf{V}_{C,s} \cdot \min\Big\{ \sum_{s' \in S} \mathbf{P}(s, \alpha, s') \cdot \mathcal{R}_j[s, \alpha, s'] \;\Big|\; \langle s, \alpha\rangle \in C\Big\}.$$

All other rewards for $\mathcal{R}_{\mathfrak{C}_j}$ are zero. Here, $\mathsf{V}_{C,s}$ is an upper bound for the largest, finite expected number of times we visit the state $s \in \mathit{states}(C)$ while staying in C, more specifically,

$$\mathsf{V}_{C,s} \;\geq\; \max\Big(\Big\{\mathrm{Ex}_{\sigma,s}^{M_C}(\mathit{visits}(s)) \;\Big|\; \sigma \in \Sigma_{\mathrm{PM}}^{M_C}\Big\} \cap \mathbb{R}\Big),$$

where M_C is the auxiliary MDP discussed on page 254. These values can be computed by upper-bounding the expected number of visits in M_C. Applying Lemma 8.7 for M_C intuitively yields that $\mathcal{R}_{\mathfrak{C}_j}[C, \hat{\alpha}, \hat{s}]$ is a lower bound for the smallest expected reward that can potentially be accumulated while being in C.

Recall the reward assignment $\mathcal{R}_{\setminus \mathfrak{C}_j}$ for $M_{\setminus \mathfrak{C}_j}$ obtained from the original assignment $\mathcal{R}_j$ defined on page 44. Intuitively, $\mathcal{R}_{\setminus \mathfrak{C}_j}$ captures all rewards collected *outside* of ECs whereas $\mathcal{R}_{\mathfrak{C}_j}$ provides a lower bound for the rewards collected *within* ECs. The objective $\mathit{tot}(\mathcal{R}_{\setminus \mathfrak{C}_j} + \mathcal{R}_{\mathfrak{C}_j}, G_j)$ for $M_{\setminus \mathfrak{C}_j}$ under-approximates the finite expected rewards in M. We therefore set

$$\mathsf{L}_s^j \;=\; \mathrm{Ex}_{\min,\hat{s}}^{M_{\setminus \mathfrak{C}_j}}\big(\mathit{tot}(\mathcal{R}_{\setminus \mathfrak{C}_j} + \mathcal{R}_{\mathfrak{C}_j}, G_j)\big),$$

where $s \in S_{-\infty?}^j$ and $\hat{s} = C$ if some $C \in \mathfrak{C}_j$ with $s \in \mathit{states}(C)$ exists and $\hat{s} = s$ otherwise. These values can be computed using standard model checking techniques on the quotient $M_{\setminus \mathfrak{C}_j}$.

Case 2 | For a strategy $\sigma \in \Sigma_{\mathrm{PM}}^{M}$ such that there is a BSCC $B \subseteq S_?^j$ of $M[\![\sigma]\!]$ that contains a *positive* reward, the inequality system from Equation (8.5) has infinitely many solutions—including solutions that for some fixed value $v \in \mathbb{R}$ set x_s to v for all $s \in B$. The `MILP` constraints from Figure 8.4 already handle this situation in an adequate way: If B is not reachable under the encoded strategy (or only reachable via states in G_j), the values assigned to x_s^j for $s \in B$ have no effect on the value of $x_{s_I}^j$ for initial state s_I. On the other hand, if B is reachable with a positive probability

$\Pr^M_{\sigma,s_I}(S^j_? \,\mathcal{U}\, B) > 0$, an MILP solver can assign an arbitrarily large value to the states $s \in B$, e.g., it can set x^j_s to

$$v^j_{\sigma,s} \;:=\; \frac{\boldsymbol{p}(j) - \mathrm{Ex}^M_{\sigma,s_I}\big(tot(\mathcal{R}^-_j, G_j)\big)}{\Pr^M_{\sigma,s_I}(S^j_? \,\mathcal{U}\, \{s\})},$$

where $\mathcal{R}^-_j$ is obtained from $\mathcal{R}_j$ by setting all positive rewards to 0. With this assignment, the value at the initial state satisfies

$$x^j_{s_I} \;\geq\; \Pr^M_{\sigma,s_I}(S^j_? \,\mathcal{U}\, \{s\}) \cdot v^j_{\sigma,s} + \mathrm{Ex}^M_{\sigma,s_I}\big(tot(\mathcal{R}^-_j, G_j)\big) \;=\; \boldsymbol{p}(j)\,.$$

The intuition is that instead of accumulating infinite reward from $s \in B$, a finite but sufficiently large amount (such as $v^j_{\sigma,s}$) is collected to ensure that the objective value threshold $\boldsymbol{p}(j) \in \mathbb{R}$ is exceeded.

Appropriate constants $\mathsf{L}^j_s, \mathsf{U}^j_s \in \mathbb{R}$ for the MILP constraints in Figure 8.4 are computed as follows. We set $\mathsf{L}^j_s = \mathrm{Ex}^M_{\min,s}\big(tot(\mathcal{R}_j, G_j)\big)$ for $s \in S^j_?$. Upper bound values U^j_s for $s \in S^j_?$ are computed on a quotient model similar to the lower bounds from the previous case. As before, let $\mathfrak{C}_j$ be the set of maximal end components in $\{C \in EC(M) \mid states(C) \subseteq S^j_?\}$. We consider a reward assignment $\mathcal{R}_{\mathfrak{C}_j}$ for the quotient model $M_{\backslash \mathfrak{C}_j}$ to reflect the rewards collected *inside* an EC. Given $C \in \mathfrak{C}_j$, enabled action $\hat{\alpha} \in exits(C) \cup \{\bot\}$, and state $\hat{s}$ we set

$$\begin{aligned}\mathcal{R}_{\mathfrak{C}_j}[C, \hat{\alpha}, \hat{s}] \;:=\; &\sum_{s \in states(C)} \mathsf{V}_{C,s} \cdot \max\Big\{ \sum_{s' \in S} \mathbf{P}(s,\alpha,s') \cdot \mathcal{R}_j[s,\alpha,s'] \;\Big|\; \langle s, \alpha\rangle \in C \Big\} \\ &+ [\hat{\alpha}{=}\bot \text{ and } \hat{s}{=}s_\bot] \cdot \mathsf{W}_C,\end{aligned}$$

where W_C has to satisfy $\mathsf{W}_C \geq v^j_{\sigma,s}$ for all $\sigma \in \Sigma^M_{\mathrm{PM}}$ and $s \in states(C)$ with $\Pr^M_{\sigma,s_I}(S^j_? \,\mathcal{U}\, \{s\}) > 0$. To obtain W_C, we can compute $\mathrm{Ex}^M_{\min,s_I}\big(tot(\mathcal{R}^-_j, G_j)\big)$ as well as a lower bound for

$$\min\Big(\big\{\Pr^M_{\sigma,s_I}(S^j_? \,\mathcal{U}\, \{s\}) \;\big|\; \sigma \in \Sigma^M_{\mathrm{PM}}, s \in states(C)\big\} \setminus \{0\}\Big).$$

The latter is obtained by the product of the smallest transition probabilities that potentially occur on a path from s_I to C, which yields

$$\mathsf{W}_C \;=\; \frac{\boldsymbol{p}(j) - \mathrm{Ex}^M_{\min,s_I}\big(tot(\mathcal{R}^-_j, G_j)\big)}{\prod_{s \in S_C} \min\big\{\mathbf{P}(s,\alpha,s') \;\big|\; \alpha \in \Delta(s) \text{ and } s' \in post(s,\alpha) \cap S_C\big\}},$$

where $S_C \subseteq S^j_?$ is the set of states that lie on a path from s_I to some state $s \in states(C)$. For $s \in S^j_?$, let $\hat{s} = C$ if some $C \in \mathfrak{C}_j$ with $s \in states(C)$ exists and $\hat{s} = s$ otherwise. We

set

$$\mathsf{U}_s^j \;=\; \mathrm{Ex}_{\max,\hat{s}}^{M\setminus\mathfrak{C}_j}\big(tot(\mathcal{R}_{\setminus\mathfrak{C}_j}+\mathcal{R}_{\mathfrak{C}_j},G_j)\big)\,.$$

Problematic ECs | In Section 8.3.2, we introduced MILP constraints that deal with what we called problematic ECs of M. The more general setting considered in this section requires similar means. To be more specific, we refine the definition of the set of problematic ECs from Equation (8.4) as follows:

$$EC_{\mathsf{p}}(M,\Phi) \;\coloneqq\; \big\{C\in EC(M)\;\big|\;\exists\, j\in\{1..\ell\}\colon\ \mathcal{R}_j[\![C]\!]=0 \text{ and } states(C)\subseteq S_?^j\big\},$$

i.e., only ECs in which no reward is collected (for the respective objective) are considered problematic. As before, the constraints from either Figure 8.6 or Figure 8.7 are imposed for each maximal EC in $EC_{\mathsf{p}}(M,\Phi)$. Constraints 12 and 18—which imply the upper bound $x_s^j \le 0$ for $s\in states(C)$ with $b_s^C=1$—are only imposed for objective indices $j\in\{1..\ell\}$ with $\mathcal{R}_j[\![C]\!]=0$ and $states(C)\subseteq S_?^j$.

Theorem 8.17 The MILP constraints described above are feasible iff $\boldsymbol{p}\in Ach_{\mathrm{PM}}(\Phi)$.

Remark 8.2 (Indicator constraints) Several constraints of our MILP encoding consider constants L_s^j, U_s^j, or V_s that have to be "large enough" so that valid solutions are not excluded. Such kind of constraints are often referred to as big-M constraints and—if the considered constants become very large—may introduce numerical instabilities during the MILP solving process [BLTW15]. As a practical solution to this, many MILP solvers such as Gurobi [Gur22] and CPLEX [Stu21] also support so-called *indicator constraints* which are constraints of the form

$$\text{if } b=i \text{ then } \langle constr\rangle$$

where b is a binary variable, $i\in\{0,1\}$ is a constant, and $\langle constr\rangle$ is a linear inequality constraint which only has to hold if the value of b equals i. The big-M constraints in our encoding can alternatively be replaced by equivalent indicator constraints. For example, Constraint 8 from Figure 8.6 is equivalent to

$$\forall\, s\in states(C)\colon\quad z_{s,\bot}^C\ge 0 \quad\text{ and }\quad \text{if } b_s^C=0 \text{ then } z_{s,\bot}^C\le 0\,.$$

Our implementation discussed in Chapter 9 can consider both encoding variants.

8.4 An Alternative Encoding for Total Rewards

We now consider PM-MOA instances for MDP $M = \langle S, Act, \Delta, \mathbf{P} \rangle$ with $s_I \in S$ and point $\boldsymbol{p} \in \mathbb{R}^\ell$, where all $\ell > 0$ objectives are expected *total* reward objectives, i.e., the objectives are given by $\Phi = \langle tot(\mathcal{R}_1), \ldots, tot(\mathcal{R}_\ell) \rangle$. For such instances, we can employ an alternative MILP encoding based on expected visiting times that is inspired by the approach of [FKNP+11] for multi-objective achievability under general strategies. In Section 4.1.1 we discussed a conversion from more general total reachability reward objectives to total reward objectives using the *goal unfolding*. In many cases, this transformation preserves achievability under pure memoryless strategies, e.g., if the different sets of goal states can not be exited or if all objectives consider the same goal states. Hence, we can often preprocess total *reachability* reward objectives so that the MILP encoding from this section can be applied. There are, however, cases where the goal unfolding does *not* preserve achievability under memoryless strategies. We further impose the following two assumptions.

Assumption 8.4 $\forall\, j \in \{1..\ell\} \colon\ \forall\, C \in EC(M) \colon\ \mathcal{R}_j[\![C]\!] \leq 0.$

Assumption 8.5 $\forall\, s \in S \colon\ \exists\, \sigma \in \Sigma^M_{\mathrm{PM}} \colon\ \forall\, j \in \{1..\ell\} \colon\ \mathrm{Ex}^M_{\sigma,s}\big(tot(\mathcal{R}_j)\big) > -\infty.$

Both assumptions are similar to what we assumed for the analysis under general strategies (cf. Section 4.1.3). Assumption 8.4 ensures that $\mathrm{Ex}^M_{\sigma,s}\big(tot(\mathcal{R}_j)\big) < +\infty$ for all $s \in S$, $\sigma \in \Sigma^M$, and $j \in \{1..\ell\}$. Assumption 8.5 yields that there is always *some* strategy inducing a finite expected reward for all objectives. The latter is without loss of generality: we can consider an appropriate submodel of M that discards all states that violate Assumption 8.5. If this discards the initial state, we directly get that no $\boldsymbol{p} \in \mathbb{R}^\ell$ is achievable.

Our MILP encoding considers the set of states from which a non-zero reward for at least one objective can be accumulated, formally given by

$$S_? \;\coloneqq\; \Big\{ s \in S \;\Big|\; \exists\, j \in \{1..\ell\} \colon\ \exists\, \sigma \in \Sigma^M_{\mathrm{PM}} \colon\ \mathrm{Ex}^M_{\sigma,s}\big(tot(\mathcal{R}_j)\big) \neq 0 \Big\}.$$

No reward can be accumulated from states $s \in S \setminus S_?$ which is why those states can safely be discarded from the encoding. Next, we define the set of ECs within the set $S_?$ in which we can reside without accumulating infinite (negative) reward:

$$EC_?(M, \Phi) \;\coloneqq\; \Big\{ C \in EC(M) \;\Big|\; states(C) \subseteq S_? \text{ and } \forall\, j \in \{1..\ell\} \colon\ \mathcal{R}_j[\![C]\!] = 0 \Big\}.$$

The set of maximal ECs in $EC_?(M, \Phi)$ is given by $MEC_?(M, \Phi)$.

$$\forall s \in S_? \colon$$

$$\forall \alpha \in \Delta(s) \colon \quad a_{s,\alpha} \in \{0,1\} \qquad \triangleright \textit{ Select a unique action at each state} \qquad (23)$$

$$\sum_{\alpha \in \Delta(s)} a_{s,\alpha} = 1 \qquad (24)$$

$$\forall \alpha \in \Delta(s) \colon \quad y_{s,\alpha} \in [0, \mathsf{V}_s \cdot a_{s,\alpha}] \qquad \triangleright \textit{ Compute expected visits} \qquad (25)$$

$$y_{s,\bot} \in [0, \mathsf{V}_s \cdot b_s] \qquad (26)$$

$$b_s \in \left\{0, [s \in \textstyle\bigcup_{C \in MEC_?(M,\Phi)} states(C)]\right\} \qquad (27)$$

$$y_{s,\bot} + \sum_{\alpha \in \Delta(s)} y_{s,\alpha} = [s{=}s_I] + \sum_{\langle s',\alpha' \rangle \in pre(s)} \mathbf{P}(s',\alpha',s) \cdot y_{s',\alpha'} \qquad (28)$$

$$1 = \sum_{s \in S_?} y_{s,\bot} + \sum_{\alpha \in \Delta(s)} y_{s,\alpha} \cdot \sum_{s' \in S \setminus S_?} \mathbf{P}(s,\alpha,s') \qquad (29)$$

$$\forall C \in MEC_?(M,\Phi) \colon \qquad \triangleright \textit{ Assert ``}b_s{=}1 \implies s \textit{ reaches a BSCC almost-surely''}$$

$$\forall \langle s,\alpha \rangle \in C \colon$$

$$\forall s' \in post(s,\alpha) \colon \quad b_s \leq b_{s'} + 1 - a_{s,\alpha} \qquad (30)$$

$$\forall \langle s,\alpha \rangle \in exits(C) \colon \quad b_s \leq 1 - a_{s,\alpha} \qquad (31)$$

$$\forall j \in \{1..\ell\} \colon \quad x^j_{s_I} \geq \boldsymbol{p}(j) \qquad \triangleright \textit{ Assert objective value at initial state} \qquad (32)$$

$$x^j_{s_I} = \sum_{s \in S_?} \sum_{\alpha \in \Delta(s)} y_{s,\alpha} \cdot \sum_{s' \in S} \left(\mathbf{P}(s,\alpha,s') \cdot \mathcal{R}_j(s,\alpha,s')\right) \qquad (33)$$

Figure 8.10: Alternative MILP encoding for total reward objectives

Figure 8.10 shows the MILP constraints for the alternative encoding using variables

$$Vars = \{a_{s,\alpha}, b_s, y_{s,\alpha}, y_{s,\bot} \mid s \in S_?, \alpha \in \Delta(s)\} \cup \{x^j_{s_I} \mid j \in \{1..\ell\}\}.$$

Strategy Encoding | As in the previous encoding, Constraints 23 and 24 encode a strategy $\sigma \in \Sigma^M_{\mathrm{PM}}$ by considering a binary variable $a_{s,\alpha}$ for each state s and enabled action α such that $a_{s,\alpha}$ is set to 1 iff $\sigma(s) = \alpha$. The action selected at states in $S \setminus S_?$ is arbitrary and has no effect on the induced objective values.

Visiting Times | Constraints 25 to 31 encode the expected number of times each state is visited under the encoded strategy σ. The approach is based on Theorem 8.8 and similar to the encoding from Figure 8.6. For each state $s \in S_?$ and enabled action $\alpha \in \Delta(s)$, we consider a variable $y_{s,\alpha}$ that can only get a positive value if $a_{s,\alpha}$ is set to 1, i.e., $\sigma(s) = \alpha$. The idea is that $y_{s,\sigma(s)}$ holds the expected number of times that s is visited. However, special treatment of states that—under the encoded strategy—lie on a BSCC is necessary as those states can potentially be visited infinitely often. For

this purpose, we additionally consider variables $y_{s,\bot}$ for $s \in S_?$ that can get a positive value to intuitively "deflate" the visiting times in a BSCC. Positive values for $y_{s,\bot}$ are, however, only possible if the binary variable b_s is set to 1, which requires that

- s lies on some EC $C \in MEC_?(M, \Phi)$ (Constraint 27),
- $b_{s'}$ is set to 1 for all states $s' \in states(C)$ reachable from s under strategy σ (Constraint 30), and
- $\langle s, \alpha \rangle \in C$ for the selected action $\alpha = \sigma(s)$ (Constraint 31).

Hence, a positive value for $y_{s,\bot}$ implies that s is contained in an EC $C \in MEC_?(M, \Phi)$ and that it is not possible to leave the EC under σ. Constraints 25 and 26 consider constants V_s that upper bound the expected number of visits of s as discussed in Section 8.2.3.

Objective Values | Constraints 32 and 33 compute values for the individual objectives based on Lemma 8.7 and assert that the objective threshold $\boldsymbol{p}(j)$ is met. The following lemma yields correctness.

Lemma 8.18 Every solution val of the constraints in Figure 8.10 satisfies

$$\forall\, j \in \{1..\ell\}\colon \quad \boldsymbol{p}(j) \;\le\; val(x^j_{s_I}) \;\le\; \mathrm{Ex}^M_{\sigma,s_I}\big(tot(\mathcal{R}_j)\big),$$

where σ is the encoded strategy. Furthermore, for any strategy $\sigma \in \Sigma^M_{\mathrm{PM}}$ with $\mathrm{Ex}^M_{\sigma,s_I}(tot(\mathcal{R}_j)) \in \mathbb{R}$ for all $j \in \{1..\ell\}$, there is a solution val for Constraints 23 to 31 with

$$val(y_{s,\alpha}) \;=\; \begin{cases} \mathrm{Ex}^M_{\sigma,s_I}(visits(s)) & \text{if } \sigma(s) = \alpha \text{ and } \mathrm{Ex}^M_{\sigma,s_I}(visits(s)) < \infty \\ 0 & \text{otherwise.} \end{cases} \tag{8.6}$$

Proof. We consider a solution $val\colon Vars \to \mathbb{R}$ which induces a strategy $\sigma \in \Sigma^M_{\mathrm{PM}}$ and the sets $T \subseteq S' \subseteq S_? \subseteq S$ given by

$$S' \;:=\; \big\{s \in S_? \mid val(b_s) = 0\big\} \quad \text{and} \quad T \;:=\; \big\{s \in S' \;\big|\; \mathrm{Ex}^M_{\sigma,s}(visits(s)) < \infty\big\}.$$

We further fix some $j \in \{1..\ell\}$ and define for $s \in S$

$$r^j_s \;:=\; \sum_{s' \in S} \mathbf{P}(s, \sigma(s), s') \cdot \mathcal{R}_j[s, \sigma(s), s'].$$

Observe the following facts.

1. For $s \in S_?$ and $\alpha \in \Delta(s)$, $\sigma(s) \neq \alpha$ implies $val(y_{s,\alpha}) = 0$.

2. For $s \in S \setminus S'$ we have $r_s^j = 0$ and $post(s, \sigma(s)) \cap S' = \emptyset$ because of the following cases:
   - If $s \notin S_?$, the definition of $S_?$ yields $\mathrm{Ex}^M_{\sigma,s}(tot(\mathcal{R}_j)) = 0$ and thus in particular $r_s^j = 0$. Also, a state $s' \in S' \subseteq S_?$ can not be reached from $s \notin S_?$, i.e., $post(s, \sigma(s)) \cap S' = \emptyset$.
   - If $s \in S_?$, we infer from $s \notin S'$ that $val(b_s) = 1$. Our observations above yield that all reachable successors s' of s are contained in $S \setminus S'$—as they satisfy $val(b_{s'}) = 1$—and thus $post(s, \sigma(s)) \cap S' = \emptyset$. Furthermore, $r_s^j = 0$ follows from the fact that $\langle s, \sigma(s)\rangle \in C$ for some $C \in MEC_?(M, \Phi)$.

3. For $s \in T$ we have $val(y_{s,\sigma(s)}) = \mathrm{Ex}^M_{\sigma,s_I}(visits(s))$. This is because val yields a solution for the equation system from Theorem 8.8 when applied to M, σ, and S'.

4. For $s \in S' \setminus T$ we have $val(y_{s,\sigma(s)}) \geq \mathrm{Ex}^M_{\sigma,s_I}(visits(s)) = 0$ and $r_s^j \leq 0$. To see this, observe that $s \notin T$ yields $\mathrm{Ex}^M_{\sigma,s}(visits(s)) = \infty$, i.e., s lies on a BSCC B of $M[\![\sigma]\!]$. Since $B \subseteq states(C)$ for some $C \in EC(M)$, Assumption 8.4 implies $r_s^j \leq 0$. Moreover, $s \in S'$ yields that $val(b_{s'}) = 0$ must hold for all predecessors of s, which yields $B \subseteq S'$. Theorem 8.8 applied to M, σ, and S' yields $\Pr^M_{\sigma,s_I}(\Diamond(S \setminus S')) = 1$. Therefore the probability to reach the BSCC $B \subseteq S'$ from s_I must be zero, i.e., $\mathrm{Ex}^M_{\sigma,s_I}(visits(s)) = 0$.

Using these facts we get from Constraint 33 that

$$
\begin{aligned}
val(x^j_{s_I}) &\overset{\text{Fact 1}}{=} \sum_{s\in S_?} val(y_{s,\sigma(s)}) \cdot \underbrace{r_s^j}_{=0 \text{ if } s\notin S'} \overset{\text{Fact 2}}{=} \sum_{s\in S'} val(y_{s,\sigma(s)}) \cdot r_s^j \\
&= \Big(\sum_{s\in T} \underbrace{val(y_{s,\sigma(s)})}_{=\mathrm{Ex}^M_{\sigma,s_I}(visits(s))} \cdot r_s^j\Big) + \underbrace{\Big(\sum_{s\in S'\setminus T} \underbrace{val(y_{s,\sigma(s)})}_{\geq \mathrm{Ex}^M_{\sigma,s_I}(visits(s)) = 0} \cdot \underbrace{r_s^j}_{\leq 0}\Big)}_{\leq 0} \\
&\overset{\text{Facts 3,4}}{\leq} \Big(\sum_{s\in T} \mathrm{Ex}^M_{\sigma,s_I}(visits(s)) \cdot r_s^j\Big) + \underbrace{\Big(\sum_{s\in S'\setminus T} \mathrm{Ex}^M_{\sigma,s_I}(visits(s)) \cdot r_s^j\Big)}_{=0} \\
&= \sum_{s\in S'} \mathrm{Ex}^M_{\sigma,s_I}(visits(s)) \cdot r_s^j \overset{\text{Fact 2}}{=} \sum_{s\in S} \mathrm{Ex}^M_{\sigma,s_I}(visits(s)) \cdot r_s^j \\
&\overset{\text{Lem. 8.7}}{=} \mathrm{Ex}^M_{\sigma,s_I}(tot(\mathcal{R}_j)) \,.
\end{aligned}
$$

We obtain equality if for $\forall s \in S' \setminus T\colon\ val(y_{s,\sigma(s)}) = \mathrm{Ex}^M_{\sigma,s_I}(visits(s)) = 0$. This allows us to use similar arguments to show that a solution val as in Equation (8.6)

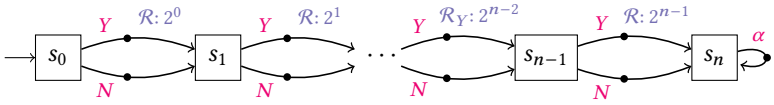


Figure 8.11: Example MDP M from Example 8.6

exists, where we set $val(b_s) = [\mathrm{Ex}^M_{\sigma,s_I}(visits(s)) = \infty]$ for $s \in S_?$. ■

Theorem 8.19 The constraints from Figure 8.10 are feasible iff $\boldsymbol{p} \in Ach_{\mathrm{PM}}(\Phi)$.

Proposition 8.20 The number of variables in the MILP encoding in Figure 8.10 satisfies

$$|Vars| \in O(|S| \cdot |Act| + \ell).$$

The MILP encoding presented in this section use fewer variables compared to the encoding from Section 8.3. In practice, this often leads to faster solving times as we discuss in Section 9.6.

8.5 Computing the Pareto Front

We consider a variant of the multi-objective Pareto Problem (MOP, cf. Problem 3.3 on page 79), where achievability is restricted to pure memoryless strategies.

Definition 8.3 (Pure Memoryless Pareto Front) The *pure memoryless Pareto front* for MDP M and objectives Φ is given by

pure memoryless Pareto front

$$Pareto_{\mathrm{PM}}(\Phi) \coloneqq \left\{ \boldsymbol{p} \in Ach_{\mathrm{PM}}(\Phi) \;\middle|\; \forall\, \boldsymbol{q} \in \mathbb{R}^\ell \colon\ \boldsymbol{q} \gtrsim \boldsymbol{p} \text{ implies } \boldsymbol{q} \notin Ach_{\mathrm{PM}}(\Phi) \right\}.$$

We have $Pareto_{\mathrm{PM}}(\Phi) \subseteq \{\mathrm{Ex}^M_{\sigma,s_I}(\Phi) \mid \sigma \in \Sigma^M_{\mathrm{PM}}\}$ and $Pareto_{\mathrm{PM}}(\Phi)$ is the smallest set $P \subseteq \overline{\mathbb{R}}^\ell$ such that

$$down(P) \;=\; down\big(\{\mathrm{Ex}^M_{\sigma,s_I}(\Phi) \mid \sigma \in \Sigma^M_{\mathrm{PM}}\}\big) \;=\; Ach_{\mathrm{PM}}(\Phi).$$

Similar to the corresponding notions under general strategies, we can thus interpret $Pareto_{\mathrm{PM}}(\Phi)$ as a representation for $Ach_{\mathrm{PM}}(\Phi)$ and vice-versa. The Pareto front is finite but can be very large—in particular if the objectives are strongly conflicting with many different trade-offs. The following example shows that in the worst case every pure memoryless strategy induces a different point $\boldsymbol{p} \in Pareto_{\mathrm{PM}}(\Phi)$.

Example 8.6 For $n \in \mathbb{N}$ let M be the MDP from Figure 8.11 with reward assignment $\mathcal{R}$. We have for $\sigma \in \Sigma^M_{\text{PM}}$

$$\text{Ex}^M_{\sigma,s_0}(tot(\mathcal{R})) \;=\; \sum_{i=0}^{n-1} [\sigma(s_i){=}Y] \cdot 2^i.$$

Thus, each of the 2^n possible strategies σ induces a different expected total reward. It follows that

$$\left|Pareto_{\text{PM}}(\langle tot(\mathcal{R}), tot(-\mathcal{R})\rangle)\right| \;=\; \left|\{\langle i, -i\rangle \mid 0 \le i < n\}\right| \;=\; 2^n \;=\; \left|\Sigma^M_{\text{PM}}\right|.$$

For the rest of this section, our goal is to obtain an *approximation* of $Pareto_{\text{PM}}(\Phi)$.

Problem 8.3: Pure Memoryless Multi-objective Pareto (PM-MOP)

PM-MOP, pure memoryless Pareto

Input: MDP M, initial state s_I, $\ell > 0$ total reachability reward objectives Φ, precision vector $\boldsymbol{\varepsilon} \in (\mathbb{R}_{>0})^\ell$
Output: $\langle L, U\rangle \in 2^{(\mathbb{R}^\ell)} \times 2^{(\mathbb{R}^\ell)}$ with $L \subseteq Ach_{\text{PM}}(\Phi) \cap \mathbb{R}^\ell \subseteq U$ and $\forall\, \boldsymbol{p} \in U\colon\ \exists\, \boldsymbol{q} \in L\colon\ |\boldsymbol{p} - \boldsymbol{q}| \le \boldsymbol{\varepsilon}$

We only consider inputs for Problem 8.3 that satisfy the following assumption.

Assumption 8.6 $\forall\, j \in \{1..\ell\}\colon\ \text{Ex}^M_{\max}(tot(\mathcal{R}_j, G_j)) \;<\; \infty.$

Assumption 8.6 avoids algorithmic issues with our framework. Our approach for general strategies requires a similar assumption (cf. Assumption 3.1 on page 78). An all-embracing treatment of objectives that potentially induce positive infinite expected reward is subject to future work.

Our approach for PM-MOP successively divides the set of points $\mathbb{R}^\ell$ into convex regions. For each region $R \subseteq \mathbb{R}^\ell$—given as polyhedron (cf. Definition 2.6 on page 21)—we use the MILP encoding from either Section 8.3 or Section 8.4 to either find a point $\boldsymbol{p} \in R \cap Ach_{\text{PM}}(\Phi)$ or conclude that no such point exists in this region. For this, we slightly modify the MILP encodings. Recall that both encodings consider for each $j \in \{1..\ell\}$ a variable $x^j_{s_I}$ for the expected value for the j^{th} objective at the initial state s_I. Instead of asserting that this value meets a certain threshold, we

- assert that $\left\langle x^1_{s_I}, \ldots, x^\ell_{s_I}\right\rangle \in R$ by using the representation of polyhedron R as intersection of halfspaces (cf. Definition 2.5 on page 20), and
- add the optimization function $\sum_{j=1}^{\ell} \boldsymbol{w}(j) \cdot x^\ell_{s_I}$ for a given heuristically determined weight vector $\boldsymbol{w} \in (\mathbb{R}_{\ge 0})^\ell \setminus \mathbf{0}$.

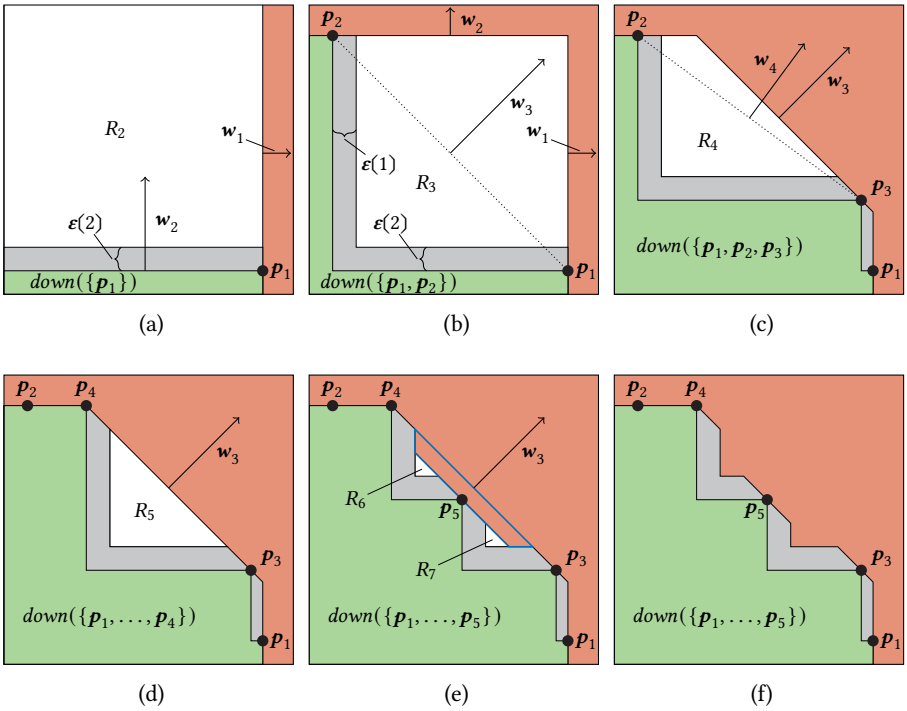


Figure 8.12: Example Exploration of achievable points (cf. Example 8.7)

Lemma 8.21 With the extensions above, a solution *val* of the respective MILP yields a point $\boldsymbol{p} = \langle val(x^1_{s_I}), \ldots, val(x^\ell_{s_I}) \rangle$ such that

$$\boldsymbol{p} \in \left(R \cap Ach_{\mathrm{PM}}(\Phi)\right) \subseteq halfsp(\boldsymbol{w}, \boldsymbol{w} \cdot \boldsymbol{p}).$$

Proof. $\boldsymbol{p} \in \left(R \cap Ach_{\mathrm{PM}}(\Phi)\right)$ follows immediately from the encoding. For every $\boldsymbol{q} \in \left(R \cap Ach_{\mathrm{PM}}(\Phi)\right)$ there is a (potentially not optimal) solution for the MILP constraints that induce achievability of $\boldsymbol{q}$. The optimality of the considered solution *val*, however, yields $\boldsymbol{w} \cdot \boldsymbol{q} \leq \boldsymbol{w} \cdot \boldsymbol{p}$ and thus $\boldsymbol{q} \in halfsp(\boldsymbol{w}, \boldsymbol{w} \cdot \boldsymbol{p})$. ■

Our approach for approximating $Ach_{\mathrm{PM}}(\Phi)$ takes strong inspiration from the sandwich algorithm discussed in [SAC93; RDH11; FKP12] and Section 3.4. We first demonstrate the procedure in terms of an example.

Example 8.7 Figure 8.12 sketches the approach for some (not further specified) MDP M and two objectives $\Phi = \langle tot(\mathcal{R}_1, G_1), tot(\mathcal{R}_2, G_2)\rangle$. We maintain a set of achievable points (green area) and a set of unachievable points (red area).
Initially, our candidate region corresponds to $R_1 = \mathbb{R}^2$ and we heuristically choose the weight vector $\boldsymbol{w}_1 = \langle 1, 0\rangle$. Using the MILP encoding from before, we find some point $\boldsymbol{p}_1 \in Ach_{\text{PM}}(\Phi) \cap R_1 = Ach_{\text{PM}}(\Phi)$ with $\boldsymbol{w}_1 \cdot \boldsymbol{p}_1 = \max\{\boldsymbol{w}_1 \cdot \boldsymbol{p} \mid \boldsymbol{p} \in Ach_{\text{PM}}(\Phi)\}$. With Lemma 8.21, we get that $down(\{\boldsymbol{p}_1\}) \subseteq Ach_{\text{PM}}(\Phi)$ whereas the points in $\{\boldsymbol{p} \in \mathbb{R}^2 \mid \boldsymbol{p}(1) > \boldsymbol{p}_1(1)\}$ are certainly not achievable—as indicated by the green and red areas in Figure 8.12a. The grey area does not have to be checked in order to obtain an approximation of $Ach_{\text{PM}}(\Phi)$. The white area indicates the next region R_2 we consider for analysis.
In the second step, we again solve an MILP, now searching for an achievable point in R_2 and optimizing for weight vector $\boldsymbol{w}_2 = \langle 0, 1\rangle$. This yields the point $\boldsymbol{p}_2$ shown in Figure 8.12b. Again, we obtain some (un-)achievable and irrelevant areas as well as a region R_3 for the next step.
For R_3, we consider the weight vector $\boldsymbol{w}_3$ which is orthogonal to the line connecting the two achievable points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$. Figure 8.12c shows the point $\boldsymbol{p}_3$ which results from solving the MILP for R_3 and $\boldsymbol{w}_4$. This yields the next region R_4 for which the considered direction vector $\boldsymbol{w}_4$ is orthogonal to the line connecting $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$.
The procedure continues through Figures 8.12d to 8.12f. The point $\boldsymbol{p}_5$ found in Figure 8.12e is dominated by the point $\frac{\boldsymbol{p}_3+\boldsymbol{p}_4}{2}$ which, however, is only achievable with a strategy that uses randomization. In this step, the trapezoidal area is added to the unachievable points. We also see that *two* subregions $R_6, R_7 \subseteq R_5$ need further analysis. In Figure 8.12f we see that neither R_6 nor R_7 contain an achievable point: the corresponding MILP constraints are infeasible. Since $\boldsymbol{p}_2$ is dominated by $\boldsymbol{p}_4$, we derive

$$P \coloneqq \{\boldsymbol{p}_1, \boldsymbol{p}_3, \boldsymbol{p}_4, \boldsymbol{p}_5\} \subseteq Pareto_{\text{PM}}(\Phi) \quad \text{and} \quad down(P) \subseteq Ach_{\text{PM}}(\Phi).$$

The red area in Figure 8.12f (excluding the boundary) does not contain any achievable point. The grey area might contain further points $\boldsymbol{p} \in Pareto_{\text{PM}}(\Phi)$. To find those, we have to consider smaller values for $\boldsymbol{\varepsilon} \in (\mathbb{R}_{>0})^2$.

Algorithm 8.2 outlines the precise procedure. It maintains a set $P \subseteq \mathbb{R}^\ell$ of achievable points that the algorithm has found so far as well as a region $R_\times \subseteq \mathbb{R}^\ell$ that contains all points that are known to be unachievable. Furthermore, the algorithm considers a set of regions $\mathfrak{R}$ that contains all regions that still need to be processed—initially starting with region $\mathbb{R}^\ell$.

As long as there are unprocessed regions, the algorithm picks some $R \in \mathfrak{R}$ and heuristically determines a weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell \setminus \{\boldsymbol{0}\}$ for R in Line 5. The details of the heuristic are discussed below. Then, the algorithm invokes an MILP solver to

Input: PM-MOP instance $\langle M, s_I, \Phi, \boldsymbol{\varepsilon} \rangle$ as in Problem 8.3
Output: $\langle L, U \rangle$ as in Problem 8.3

1 $P \leftarrow \emptyset$; $R_\times \leftarrow \emptyset$ *// Achievable points and unachievable region*
2 $\mathfrak{R} \leftarrow \{\mathbb{R}^\ell\}$ *// Set of unprocessed regions*
3 **while** $\mathfrak{R} \neq \emptyset$ **do**
4 Pick $R \in \mathfrak{R}$; $\mathfrak{R} \leftarrow \mathfrak{R} \setminus R$
5 $\boldsymbol{w} \leftarrow$getWeights$(R)$ *// Heuristically get some $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell \setminus \{\mathbf{0}\}$*
6 **if** $R \cap Ach_{\mathrm{PM}}(\Phi) \neq \emptyset$ **then** *// Use MILP encoding for PM-MOA ...*
7 $\boldsymbol{p} \leftarrow \arg\max_{\boldsymbol{q} \in R \cap Ach_{\mathrm{PM}}(\Phi)} (\boldsymbol{w} \cdot \boldsymbol{q})$ *// ...with optimization fct.*
8 $P \leftarrow P \cup \{\boldsymbol{p}\}$; $R_\times \leftarrow R_\times \cup \big(R \setminus halfsp(\boldsymbol{w}, \boldsymbol{w} \cdot \boldsymbol{p})\big)$
9 **foreach** $j \in \{1..\ell\}$ **do** *// Get convex, unprocessed subregions*
10 $R_j \leftarrow \big\{\boldsymbol{q} \in R \cap halfsp(\boldsymbol{w}, \boldsymbol{w} \cdot \boldsymbol{p}) \;\big|\; \boldsymbol{q}(j) \geq \boldsymbol{p}(j) + \boldsymbol{\varepsilon}(j)$ and $\forall\, i < j\colon \boldsymbol{q}(i) \leq \boldsymbol{p}(i) + \boldsymbol{\varepsilon}(i)\big\}$
11 **if** $R_j \neq \emptyset$ **then** $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{R_j\}$
12 **else** $R_\times \leftarrow R_\times \cup R$ *// Entire region R is unachievable*
13 **return** $\langle down(P),\ \mathbb{R}^\ell \setminus R_\times \rangle$

Algorithm 8.2: Approximating $Ach_{\mathrm{PM}}(\Phi)$

check whether an achievable point $\boldsymbol{p} \in R$ exists and, if possible, provide such a point that maximizes the dot product $\boldsymbol{w} \cdot \boldsymbol{p}$. With such a point, the (un-)achievable sets P and $R_\times$ are extended accordingly (Line 8) and the undetermined part of the region R is split into subregions $R_1, \ldots, R_\ell \subseteq R$ that only overlap on their boundary (Line 10). We have

$$\bigcup_{j=1}^{\ell} R_j \;=\; \big(R \cap halfsp(\boldsymbol{w}, \boldsymbol{w} \cdot \boldsymbol{p})\big) \setminus down(\boldsymbol{p} + \boldsymbol{\varepsilon}).$$

Non-empty subregions R_j are inserted into $\mathfrak{R}$ (Line 11). All regions inserted into $\mathfrak{R}$ are polyhedra, represented as a finite intersection of halfspaces.

If the MILP solver concludes that R does not contain any achievable point, the entire region R is added to the unachievable region $R_\times$ (Line 12). The algorithm returns the downward hull $down(P)$ of the found achievable points P as well as the complement of the unachievable region $R_\times$.

Theorem 8.22 Algorithm 8.2 terminates with a solution for PM-MOP.

Proof. As we require $\boldsymbol{\varepsilon} \in (\mathbb{R}_{>0})^\ell$, the considered regions $R \in \mathfrak{R}$ become strictly smaller and thus any two points $\boldsymbol{p}, \boldsymbol{p}' \in P$ found in different iterations of the algorithm must have a certain distance to each other: $|\boldsymbol{p} - \boldsymbol{p}'| \geq \boldsymbol{\varepsilon}$. Eventually, all points in $\mathit{Pareto}_{\mathrm{PM}}(\Phi)$ are either in P or sufficiently close to some point in P which necessarily leads to termination—independent of the weight vector selection heuristic in Line 5.
Upon termination, we have $\mathit{down}(P) \subseteq \mathit{Ach}_{\mathrm{PM}}(\Phi) \subseteq \mathbb{R}^\ell \setminus R_\times$. Furthermore, all points $\boldsymbol{p} \in \mathbb{R}^\ell \setminus R_\times$ must be in $\mathit{down}(\{\boldsymbol{p}' + \boldsymbol{\varepsilon}\})$ for some $\boldsymbol{p}' \in P$. Hence, $\boldsymbol{p}'' \coloneqq \boldsymbol{p} - \boldsymbol{\varepsilon} \in \mathit{down}(\{\boldsymbol{p}'\}) \subseteq \mathit{down}(P)$. We thus have

$$\forall\, \boldsymbol{p} \in \mathbb{R}^\ell \setminus R_\times \colon\ \exists\, \boldsymbol{p}'' \in \mathit{down}(P) \colon \quad |\boldsymbol{p} - \boldsymbol{p}''| = |\boldsymbol{p} - (\boldsymbol{p} - \boldsymbol{\varepsilon})| = \boldsymbol{\varepsilon}. \qquad \blacksquare$$

Selecting Weight Vectors

We now discuss the selection of weight vectors in Line 5 of Algorithm 8.2. While the heuristic has no impact on *whether* the algorithm terminates, it can have a strong impact on how many regions need to be analyzed. Inspired by the selection of weight vectors for the sandwich algorithm as in Section 3.4, our heuristic chooses vectors that are orthogonal to the convex hull (cf. Definition 2.4 on page 19) of the achievable points found so far. The precise selection of weight vectors is as follows.

- In the first ℓ iterations of the algorithm, we always analyze the weight vectors in $\left\{\mathbf{1}_i \in (\mathbb{R}_{\geq 0})^\ell \;\middle|\; i \in \{1..\ell\}\right\}$ that intuitively correspond to optimizing for each individual objective. Recall that $\mathbf{1}_i(j) = [i{=}j]$ for $i, j \in \{1..\ell\}$.
- For subsequent regions $R \in \mathfrak{R}$, let $\mathcal{H}_P$ be the unique set of halfspaces that represent the polyhedron $\mathit{down}_{\mathbb{R}}(\mathit{conv}(P))$. $\mathcal{H}_P$ can be obtained with a combination of the Quickhull algorithm [BDH96] and Lemma 2.3 on page 22. For each $\mathit{halfsp}(\boldsymbol{w}, b) \in \mathcal{H}_P$, the weight vector $\boldsymbol{w} \in (\mathbb{R}_{\geq 0})^\ell$ is orthogonal to a facet of $\mathit{down}_{\mathbb{R}}(\mathit{conv}(P))$.
  - If there is $H = \mathit{halfsp}(\boldsymbol{w}, b) \in \mathcal{H}_P$ such that $R \not\subseteq H$, we take the weight vector $\boldsymbol{w}$ of that halfspace when analyzing R.
  - Otherwise, we consider the same weight vector as for the parent region $R' \supseteq R$ which must have been analyzed in a previous step.

As an optimization, expensive `MILP` solver invocations can sometimes be avoided by solving an instance for the weighted sum optimization problem (`WSO`, Problem 3.4 on page 81) instead. More specifically, if the current region $R \in \mathfrak{R}$ is not fully contained in $\mathit{down}_{\mathbb{R}}(\mathit{conv}(P))$, it might contain points that are not achievable—even under general strategies. With Lemma 3.16 on page 84, solving `WSO` with the weight vector $\boldsymbol{w}$ for R yields a halfspace H with $\mathit{Ach}_{\mathrm{PM}}(\Phi) \subseteq \mathit{Ach}(\Phi) \subseteq H$. Therefore, the set $R \setminus H$ is

not achievable under general strategies, but in particular also under pure memoryless strategies. Consequently, the `MILP` solver only needs to analyze $R \cap H$. If we only consider total reward objectives, the points obtained from solving `WSO` are in fact achievable under pure and memoryless strategies. If such a point lies in the current region R, the algorithm can proceed with that point—entirely avoiding `MILP` solving for R.

8.6 Pure Achievability under Strategies with Memory

So far, we considered achievability under pure and memoryless strategies. While these strategies are simple, they also might be too restrictive. For example, when scheduling tasks it is reasonable to assume that strategies can memorize which tasks have already been accomplished and which tasks still need to be done. This section introduces *strategies with memory* and provides a product construction that links those memory strategies to memoryless strategies on a larger MDP.

8.6.1 Memory Strategies

memory structure

Definition 8.4 (Memory Structure) A *memory structure* for MDP M is a tuple $\mathcal{A} = \langle Q, \delta, q_I \rangle$ with

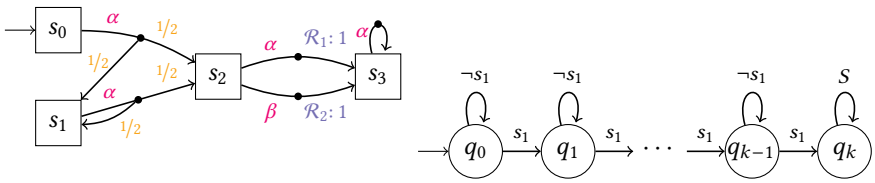
- finite set of *memory states* Q containing the *initial memory state* $q_I \in Q$ and
- transition function $\delta \colon S \times Q \to 2^Q \setminus \{\emptyset\}$.

Intuitively, a memory structure $\mathcal{A} = \langle Q, \delta, q_I \rangle$ runs in parallel to the associated MDP M. Assume that $s \in S$ is the current MDP state and $q \in Q$ is the current memory state. When the MDP takes a transition, the memory structure nondeterministically switches to some successor memory state $q' \in \delta(s, q)$. Formally, for a path $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots \in Paths^M_{\text{inf}}$, we call the sequence

$$\langle s_0, q_0 \rangle \xrightarrow{\langle \alpha_0, q_1 \rangle} \langle s_1, q_1 \rangle \xrightarrow{\langle \alpha_1, q_2 \rangle} \cdots$$

an $\mathcal{A}$*-annotation* of π if $q_0 = q_I$ and $q_{i+1} \in \delta(s_i, q_i)$ for all $i \in \mathbb{N}$. A memory structure is called *deterministic* if $|\delta(s, q)| = 1$ for all $s \in S$ and $q \in Q$. For deterministic $\mathcal{A}$, we denote by $\pi_{\mathcal{A}}$ the *unique* $\mathcal{A}$-annotation of π. $\mathcal{A}$-annotations of finite paths $\hat{\pi} \in Paths^M_{\text{fin}}$ are defined similarly. Notations for paths are lifted to $\mathcal{A}$-annotations in a straightforward way.

Definition 8.5 (Memory Strategy) Given a strategy $\sigma \in \Sigma^M$ and a deterministic



(a) MDP M (b) memory structure $\mathcal{A}$

Figure 8.13: MDP and memory structure from Example 8.8

memory strategy

memory structure $\mathcal{A}$ for M, we say that σ is an *$\mathcal{A}$-memory strategy* if

$$\forall\, \hat{\pi}, \hat{\pi}' \in Paths^M_{\text{fin}}\colon \quad last(\hat{\pi}_{\mathcal{A}}) = last(\hat{\pi}'_{\mathcal{A}}) \text{ implies } \sigma(\hat{\pi}) = \sigma(\hat{\pi}').$$

$\sigma \in \Sigma^M$ is a *k-memory strategy* if it is an $\mathcal{A}$-memory strategy for some deterministic memory structure $\mathcal{A}$ with $k > 0$ states.

Given a deterministic memory structure $\mathcal{A} = \langle Q, \delta, q_I \rangle$, we denote by $\Sigma^M_{\text{P},\mathcal{A}}$ the set of pure $\mathcal{A}$-memory strategies. A strategy $\sigma \in \Sigma^M_{\text{P},\mathcal{A}}$ can also be interpreted as a function $\sigma\colon S \times Q \to Act$. For $k > 0$, $\Sigma^M_{\text{P},k}$ denotes the set of pure k-memory strategies. We have $\Sigma^M_{\text{PM}} = \Sigma^M_{\text{P},1}$ and $\Sigma^M_{\text{P},k} \subseteq \Sigma^M_{\text{P},k+1}$ for each $k > 0$. $\mathcal{A}$-memory strategies can easily be implemented in a controller: the performed action as well as the memory update only depend on the current state of the controlled system (i.e., the MDP) and the current memory state. This is in contrast to general strategies $\sigma \in \Sigma^M$, where the entire execution history is potentially relevant for the next choice to be taken.

8.6.2 Pure Achievability with Memory

To decide achievability for $\ell \geq 2$ total reachability reward objectives under *general* strategies, it is necessary and sufficient to consider strategies that require memory exponential in the number of objectives [EKVY08; FKP12; RRS17] by storing which goal state set has been reached already. The goal unfolding discussed in Section 4.1.1 intuitively incorporates this information into the state space of the model. Achievability under *pure* (but not necessarily memoryless) strategies, on the other hand, imposes nontrivial memory requirements that can not be upper-bounded a priori.

Example 8.8 Let M be the MDP in Figure 8.13a and $\Phi = \big\langle tot(\mathcal{R}_1), tot(\mathcal{R}_2) \big\rangle$. The point $\boldsymbol{p}_k = \langle 0.5^k, 1{-}0.5^k \rangle$ for $k \in \mathbb{N}$ is achievable by the non-pure strategy that takes α at s_2 with probability 0.5^k. The point $\boldsymbol{p}_k$ is also achievable with the *pure* strategy $\sigma_k \in \Sigma^M$ that for $\hat{\pi} \in Paths^M_{\text{fin}}$ with $last(\hat{\pi}) = s_2$ sets $\sigma_k(\hat{\pi}) = \alpha$ iff

$|\hat{\pi}|_{s_1} \geq k$, where $|\hat{\pi}|_{s_1}$ denotes the number of times $\hat{\pi}$ has visited the state s_1. Strategy σ_k is an $\mathcal{A}$-memory strategy, where $\mathcal{A}$ is the memory structure from Figure 8.13b, implying that $\sigma_k \in \Sigma^M_{\mathrm{P},\mathcal{A}} \subseteq \Sigma^M_{\mathrm{P},(k+1)}$. Pure strategies with fewer memory states do not suffice. In particular, $\boldsymbol{p}_k$ is not achievable with a k'-memory strategy $\sigma \in \Sigma^M_{\mathrm{P},k'}$ for $k' \leq k$.

We consider multi-objective achievability under pure k-memory strategies for a *fixed* $k \in \mathbb{N}$.

Definition 8.6 (Pure Memory Achievable Point) For memory structure $\mathcal{A}$, the set of *pure $\mathcal{A}$-memory achievable points* is given by

pure memory achievable point

$$Ach_{\mathrm{P},\mathcal{A}}(\Phi) \coloneqq \left\{ \boldsymbol{p} \in \overline{\mathbb{R}}^{\ell} \;\middle|\; \exists\, \sigma \in \Sigma^M_{\mathrm{P},\mathcal{A}} \colon \mathrm{Ex}^M_\sigma(\Phi) \geq \boldsymbol{p} \right\}.$$

Similarly, for $k \in \mathbb{N}$, the set of *pure k-memory achievable points* is given by

$$Ach_{\mathrm{P},k}(\Phi) \coloneqq \left\{ \boldsymbol{p} \in \overline{\mathbb{R}}^{\ell} \;\middle|\; \exists\, \sigma \in \Sigma^M_{\mathrm{P},k} \colon \mathrm{Ex}^M_\sigma(\Phi) \geq \boldsymbol{p} \right\}.$$

It holds that $Ach_{\mathrm{PM}}(\Phi) = Ach_{\mathrm{P},1}(\Phi)$ and $Ach_{\mathrm{P},k} \subseteq Ach_{\mathrm{P},k+1}$ for all $k > 0$. Similar to what has been discussed in previous sections, our goal is to decide whether a given point $\boldsymbol{p} \in \mathbb{R}^{\ell}$ is in $Ach_{\mathrm{P},k}(\Phi)$ and to compute an approximation of the set $Ach_{\mathrm{P},k}(\Phi)$.

8.6.3 Reduction to Memoryless Strategies

We incorporate a memory structure of size k into the MDP M such that pure memoryless strategies of this product MDP coincide with pure k-memory strategies of M. A similar construction is given in, e.g., [4].

Definition 8.7 (Memory Product) The product of MDP $M = \langle S, Act, \Delta, \mathbf{P} \rangle$ and memory structure $\mathcal{A} = \langle Q, \delta, q_I \rangle$ is given by the MDP $M \otimes \mathcal{A} = \langle S \times Q, Act \times Q, \Delta', \mathbf{P}' \rangle$, where for $s \in S$ and $q \in Q$

memory product

$$\Delta'(\langle s, q \rangle) = \Delta(s) \times \delta(q)$$

and for $\langle \alpha, q' \rangle \in \Delta'(\langle s, q \rangle)$, $s' \in S$, and $q'' \in Q$

$$\mathbf{P}'(\langle s, q \rangle, \langle \alpha, q' \rangle, \langle s', q'' \rangle) = [q' = q''] \cdot \mathbf{P}(s, \alpha, s').$$

For simplicity, we directly apply total reachability reward objectives $tot(\mathcal{R}, G)$ for M to the memory product $M \otimes \mathcal{A}$—implicitly lifting the set of goal states $G \subseteq S$ to $G \times Q$ and the reward assignment $\mathcal{R}$ for M to $\mathcal{R}'$ with

$$\mathcal{R}'[\langle s, q \rangle, \langle \alpha, q' \rangle, \langle s', q' \rangle] = \mathcal{R}[s, \alpha, s']$$

for $s, s' \in S$, $\alpha \in Act$, and $q, q' \in Q$. The initial state of $M \otimes \mathcal{A}$ is derived from the initial state s_I of M and given by $\langle s_I, q_I \rangle$. We say that memory structure $\mathcal{A}' = \langle Q, \delta', q_I \rangle$ is a *concretization* of $\mathcal{A} = \langle Q, \delta, q_I \rangle$—written $\mathcal{A}' \sqsubseteq \mathcal{A}$—if $\mathcal{A}'$ is deterministic and $\delta'(s,q) \subseteq \delta(s,q)$ for all $s \in S$ and $q \in Q$.

concretization of memory structure

Lemma 8.23 $Ach_{\mathrm{PM}}^{M \otimes \mathcal{A}}(\Phi) = \bigcup_{\mathcal{A}' \sqsubseteq \mathcal{A}} Ach_{\mathrm{P},\mathcal{A}'}(\Phi)$.

Proof. Let $\mathcal{A} = \langle Q, \delta, q_I \rangle$. First assume $\boldsymbol{p} \in Ach_{\mathrm{PM}}^{M \otimes \mathcal{A}}(\Phi)$, i.e., there is a pure memoryless strategy $\sigma \in \Sigma_{\mathrm{PM}}^{M \otimes \mathcal{A}}$ that achieves $\boldsymbol{p}$. Consider the memory structure $\mathcal{A}' = \langle Q, \delta', q_I \rangle$ and the pure $\mathcal{A}'$-memory strategy σ' for M which for $s \in S$, $q \in Q$, and $\sigma(\langle s, q \rangle) = \langle \alpha, q' \rangle$ are defined by

$$\delta'(s,q) = q' \quad \text{and} \quad \sigma'(s,q) = \alpha.$$

Every σ-consistent path

$$\langle s_0, q_0 \rangle \xrightarrow{\langle \alpha_0, q_1 \rangle} \langle s_1, q_1 \rangle \xrightarrow{\langle \alpha_1, q_2 \rangle} \ldots \in Paths_{\mathrm{inf}}^{M \otimes \mathcal{A}}(\langle s_I, q_I \rangle)$$

coincides with the (unique) $\mathcal{A}'$-annotation $\pi'_{\mathcal{A}}$ of $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots \in Paths_{\mathrm{inf}}^{M}(s_I)$. Since π is σ'-consistent, we can show

$$\mathrm{Ex}_{\sigma', s_I}^{M}(\Phi) = \mathrm{Ex}_{\sigma, \langle s_I, q_I \rangle}^{M \otimes \mathcal{A}}(\Phi) \geq \boldsymbol{p}$$

yielding $\boldsymbol{p} \in Ach_{\mathrm{P},\mathcal{A}'}(\Phi)$.
For the other direction, assume a concretization $\mathcal{A}' = \langle Q, \delta', q_I \rangle$ of $\mathcal{A}$ and a point $\boldsymbol{p} \in Ach_{\mathrm{P},\mathcal{A}'}(\Phi)$. Let σ' be a pure $\mathcal{A}'$-memory strategy of M that achieves $\boldsymbol{p}$. We construct a pure memoryless strategy σ for $M \otimes \mathcal{A}$ as follows. For $s \in S$ and $q \in Q$ let $\sigma'(s,q) = \alpha$ and $\delta'(s,q) = \{q'\}$. Then we set

$$\sigma(\langle s, q \rangle) = \langle \alpha, q' \rangle.$$

With a similar argument as above, we can show $\boldsymbol{p} \in Ach_{\mathrm{PM}}^{M \otimes \mathcal{A}}(\Phi)$ since

$$\mathrm{Ex}_{\sigma, \langle s_I, q_I \rangle}^{M \otimes \mathcal{A}}(\Phi) = \mathrm{Ex}_{\sigma', s_I}^{M}(\Phi) \geq \boldsymbol{p}. \qquad \blacksquare$$

A memory structure $\mathcal{A} = \langle Q, \delta, q_I \rangle$ is called *free* if $\delta(s,q) = Q$ for all $s \in S$ and $q \in Q$. For $k \in \mathbb{N}$, let $\mathcal{A}_k$ be the free memory structure with k memory states. $\mathcal{A}_k$ is unique up to renaming of memory states.

free memory structure

Theorem 8.24 $\forall\, k \in \mathbb{N}\colon\ Ach_{\mathrm{P},k}(\Phi) = Ach_{\mathrm{PM}}^{M \otimes \mathcal{A}_k}(\Phi)$.

Proof. The theorem follows from Lemma 8.23 and the observation that the deterministic memory structures with k states are exactly the concretizations of $\mathcal{A}_k$. ■

Due to Theorem 8.24, we can answer achievability and Pareto queries for k-memory strategies by considering the corresponding query on the product $M \otimes \mathcal{A}_k$ with free memory structure $\mathcal{A}_k = \langle Q_k, \delta_k, q_I \rangle$ instead. The product $M \otimes \mathcal{A}_k$ has $|S| \cdot k$ many states and

$$\left|\Sigma_{\text{PM}}^{M \otimes \mathcal{A}_k}\right| = \prod_{s \in S} \prod_{q \in Q_k} \left(|\Delta(s)| \cdot \underbrace{|\delta_k(q)|}_{=k} \right) = \prod_{s \in S} \left(|\Delta(s)|^k \right) \cdot k^{|S| \cdot k} = \left(\left|\Sigma_{\text{PM}}^{M}\right| \cdot k^{|S|} \right)^k$$

many pure memoryless strategies.

Remark 8.3 (Memory Patterns) The set of considered strategies can be further refined by considering products with non-free memory structures instead. For example, we can consider a memory structure that only allows incrementing a counter or that only remembers visits of goal states. Such refined memory patterns yield smaller products, improving practical feasibility of the `MILP`-based approach but discarding strategies that do not follow the given pattern.

8.7 Related Work

Pure memoryless multi-objective achievability has been studied in various settings. NP-completeness of the decision problem for discounted (and total) reward objectives under pure memoryless strategies was shown in [CMH06]. [EG16] claims that this generalizes to PCTL objectives (including reachability probabilities) but no proof is given. [Vel19] shows NP-hardness for pure memoryless achievability over long-run average objectives. Pure memoryless Pareto optimal strategies for discounted rewards are obtained in [WJ07] using value-iteration but the approach is restricted to small MDPs where all transition probabilities are either 0 or 1. [CAH04] outlines the potential trade-offs between strategies with randomization and memorization. Objectives where infinite memory is required are given in, e.g., [BBCF+14; BFRR17; BRR17]. [MR22] discusses expressiveness of strategies with finite memory under different notions of randomization.

Linear programming is a popular tool for various aspects of (advanced) MDP analysis such as generating counter-examples [WJÁK+14; WJVÁ+15], maximizing the expected total reward in a cooperative multi-agent MDP [WD06], synthesizing pure memoryless strategies for partially observable MDPs [WWJB20], and deciding achievability or Pareto optimality in multi-objective MDPs [EKVY08; FKNP+11; PW10; BBCF+14; RRS15; RRS17; Vel19]. These (`MI`)`LP` encodings are commonly based on classical formulations

for single-objective expected values as found in [Put94]. The approach of [WWJB20] has a noteworthy overlap with the approach in this chapter. Their encoding for reachability probabilities is similar to our constraints from Figures 8.4 and 8.7. It seems promising to combine both approaches in an attempt to lift the results from this chapter to partially observable MDPs.

Chapter Summary

- ➜ Deciding pure memoryless multi-objective achievability (PM-MOA) for expected total reachability reward objectives is NP-complete—even if just two objectives are considered.
- ➜ PM-MOA can be decided using an MILP encoding. The crux lies in dealing with end components.
- ➜ For total reward objectives, an alternative MILP encoding with fewer variables can be used.
- ➜ The Pareto front under pure memoryless strategies can be approximated by repeatedly solving a slight variant of the MILP for PM-MOA.
- ➜ Achievability under pure strategies with (limited) memory can be reduced to achievability under pure memoryless strategies on a product of the MDP with a memory structure.

—Chapter 9—

Tool Support and Empirical Evaluation

Outlook | We consider practical aspects of multi-objective verification of Markov models. All algorithms presented in this thesis have been implemented into the probabilistic model checker Storm. In Section 9.1 we outline the capabilities of Storm as well as related tools for answering multi-objective queries. Section 9.2 compares different methods for computing (single-objective) expected total reachability rewards. Section 9.3 presents our benchmark set containing multi-objective queries. Section 9.4 empirically compares the performance of Storm and competing tools. Section 9.5 evaluates approaches for reward-bounded objectives and quantiles discussed in Chapters 6 and 7. Section 9.6 contains experiments for multi-objective model checking under simple strategies using the methods from Chapter 8.

Origins | This chapter is mainly based on the evaluations conducted in [5; 10; 9; 11]. The original implementation of the presented methods within Storm has been done in the context of the corresponding publications. An initial implementation of the sandwich algorithm (cf. [FKP12] and Section 3.4) has been implemented in Storm as part of the author's master's thesis [1]. However, the implementations evolved over time as Storm's source code is actively maintained and improved. In particular, the implementation for sound value iteration (cf. Chapter 4) and the `MILP` encoding for simple strategies (cf. Chapter 8) have been revised since their initial release. The experimental setup in Section 9.2 has been adapted from [6; 12; 14]. The benchmarks presented in Section 9.3 were taken from various sources (see Table 9.2). The experiments and their presentation in Sections 9.4 to 9.6 are mostly new.

Data Availability | All considered benchmarks, execution scripts, log files, and the version of Storm exercised for the presented experiments is available at

`https://doi.org/10.5281/zenodo.7766202`.

Technical Setup | All experiments in this chapter were conducted on a Linux CentOS 7.9 system using 8 cores of an Intel® Xeon® Platinum 8160 Processor. Runtime and memory were limited to 2 hours and 32 GB, respectively.

9.1 Multi-Objective Verification with STORM

STORM The probabilistic model checker STORM [12] is a feature-rich verification tool for Markov models written in C++. It has played a predominant role in both editions of the QComp tool comparison [6; 8]. DTMC, CTMC, MDP, and MA models can either be specified via the PRISM [KNP11] or JANI [BDHH+17] modeling languages, in an explicit format, or more specialized formalisms such as PGCL programs, dynamic fault trees, or generalized stochastic Petri nets. Supported verification queries include expected total reachability rewards, unbounded and bounded reachability probabilities, long-run average objectives, as well as multi-objective combinations thereof. STORM handles multi-objective (quantitative) achievability (MO(Q)A) as well as Pareto (MOP) queries, cf. Problems 3.1 to 3.3 on pages 78–79. To answer such queries, it implements the algorithms presented in the preceding chapters of this thesis.

Implementation Details

We discuss details of the implementation of multi-objective verification within STORM, reflecting its source code as included in the above-mentioned data package. While the implementation generally follows the algorithmic ideas as presented in this thesis, efficiency considerations and ease of implementation occasionally lead to minor deviations between implementation and algorithm description—which we do not always list here. All mentioned features are also available in the main release of STORM. Since STORM is actively developed, we also refer to its website for up-to-date information:

https://www.stormchecker.org

Multi-Objective Model Checking Framework | Multi-objective verification is implemented in STORM's *sparse engine*, using explicit data structures such as sparse matrices and vectors. STORM implements the sandwich algorithm as presented in Section 3.4. Geometric sets and operations (cf. Section 2.1.2) are implemented using arbitrary precision rational numbers as provided by the GMP library[1]. Furthermore, EIGEN [GJ+10] and Z3 [MB08] are used to exactly solve linear equation systems and LPs, respectively.

[1] https://gmplib.org

Before the sandwich algorithm is invoked, the input query is simplified. This includes, e.g., applying the *goal unfolding* (cf. Definition 4.1 on page 100) and collapsing states where all objective values are known to be zero.

Computing Expected Total Rewards | STORM implements various methods for total reachability objectives based on value iteration, strategy iteration, and linear programming. By default, these methods are used with topological optimizations, i.e., each SCC of the model is analyzed in isolation (cf. Section 4.4.4). The value iteration-based approaches are implemented in their Gauss-Seidel variants, using a single CPU core and double-precision floating point numbers.

Long-Run Average Values | Our approach for multiple LRA objectives requires to compute optimal expected LRA reward values within a given end component as in Problem 5.1 on page 155. For this, we use the value-iteration based approach of [BWH17; But20].

Reward-Bounded Objectives and Quantiles | We implemented the sequential epoch analysis of Section 6.2—including the optimizations from Section 6.2.6. However, the all-embracing treatment of infinite expected values and the incorporation of LRA reward objectives (Remark 6.3 on page 203) into this framework is subject to future work. Consequently, STORM is currently unable to analyze mixtures of reward-bounded and LRA reward objectives[2].

Multi-Dimensional Quantiles | STORM supports 2-dimensional quantile queries with monotone bounds as in Section 7.3. Additional *definite* reward bounds can be added to the query as suggested in Remark 7.3 on page 220.

Simple Strategies | STORM can approximate the pure memoryless Pareto front for total reachability reward objectives using the approach from Chapter 8. STORM also supports the incorporation of memory structures, optionally using memory patterns as in Remark 8.3 on page 277.

The occurring `MILP` encodings are solved using GUROBI [Gur22] in version 9.5. The computations within GUROBI might suffer from numerical instabilities. To diminish their impact, we use the *exact* model checking capabilities of STORM to confirm for each `MILP` solution that the encoded strategy achieves the encoded point. Hence, the reported *achievable* regions are always identified correctly. However, when GUROBI returns a sub-optimal solution, a region might incorrectly be reported as *unachievable*.

[2] Step-bounded (MDP) and time-bounded (MA) objectives are handled in a different code-branch that implements ideas of [FKP12] and [13], respectively. Hence, mixtures of those objectives with total reachability and LRA reward objectives are supported.

Table 9.1: Tool capabilities for multi-objective verification

	Storm	EPMC	Prism	MultiGain	Prism-games
models	MDP, MA	MDP	MDP	MDP	stoch. games
objectives					
• reach prob	yes	yes	yes	no	qualitative
• total reach rew	yes	yes	yes	no	yes
• LRA rew	yes	no	no	yes	yes
• rew bounded	yes	no	steps	no	no
• LTL prob	no	yes	yes	no	no
queries					
• `MOA`	yes	yes	yes	yes	yes
• `MOQA`	yes	yes	yes	yes	no
• `MOP`	yes	no	$\ell = 2$	$\ell = 2$	$\ell = 2$
main approach	SW	SW	SW, LP	LP	convex VI

Besides the encodings outlined in Chapter 8, the implementation also supports a straightforward variation using *indicator constraints* as in Remark 8.2 on page 262.

9.1.1 Related Tools

We list the most related tools for the verification of multi-objective Markov models. The information is obtained from publications, tool websites, and sometimes from examining the source code in its current version (December 2022). As such, the provided details are potentially incomplete or outdated.

Table 9.1 provides an overview of the tool capabilities in terms of multi-objective verification. The table lists the supported models, objective types, and query types as well as the main approach used for solving those queries. For the latter, we write "SW" to denote the sandwich algorithm as in [FKP12] and Section 3.4, "LP" for linear programming-based approaches, and "convex VI" for value iteration over convex sets.

EPMC **EPMC [FHLS+22]** | EPMC is an extensible model checking framework mostly written in `JAVA`. It supports multi-objective (qualitative and quantitative) achievability queries for MDPs over total reachability reward objectives as well as objectives specified in linear temporal logic (LTL) applied in [LTHS+22] to solve probabilistic preference-based planning problems. EPMC implements the sandwich algorithm based on [FKP12], similar to our approach from Section 3.4. The instances of `WSO` are solved using standard

(unsound) value iteration. Up to our knowledge, EPMC can not compute approximations of the Pareto front.

Prism [KNP11] | Prism is a well-established, user friendly probabilistic model checker that answers achievability and Pareto queries for MDPs over combinations of total and step-bounded reward objectives as well as LTL specifications. The tool is mostly written in JAVA. In addition to the sandwich algorithm of [FKP12], it also implements the LP-based characterization of [FKNP+11]. While the latter only works for (qualitative and quantitative) achievability queries, the former can also be used to approximate Pareto fronts over up to $\ell = 2$ objectives. Prism

MultiGain [BCFK15] | MultiGain is an extension of Prism that implements the LP-based approach of [BBCF+14] for multiple LRA objectives on MDP to answer qualitative and quantitative achievability as well as Pareto queries with up to $\ell = 2$ objectives. For the latter, it uses the sandwich approach similar to Storm but solves the occurring WSO instances using linear programming. For solving the LPs, MultiGain uses either lp_solve[3] or Gurobi. MultiGain

Prism-games [KNPS20; KPW18] | Prism-games is another extension of Prism focused on the verification of stochastic games (SGs). The tool implements value iteration over convex sets [BKTW15; BKW18] to analyze multiple total and LRA (ratio) reward objectives as well as almost-sure reachability constraints. In contrast to our setting, Prism-games also allows arbitrary Boolean combinations of objectives. Prism-games can be applied to MDPs by converting them to 1-player SGs. However, some experiments on 1-player SGs indicated that this approach is not competitive compared to the dedicated MDP implementations in the other tools. Prism-games

9.2 Expected Total Reward Algorithms

In subsequent sections, we evaluate the practical performance of verification algorithms for *multiple* objectives. In this section, however, the focus lies on algorithms that compute (or approximate) the expected value of a *single* total reachability reward objective. This type of computation is one of the core ingredients for probabilistic model checking in general, but in particular also for the various multi-objective verification approaches presented in this work—even for those that go beyond analyzing plain total reachability reward objectives. More specifically, our algorithm for LRA reward objectives requires to compute expected total rewards on a quotient model (cf.

[3] https://lpsolve.sourceforge.net

Section 5.1) and the sequential epoch analysis for reward-bounded objectives needs to compute expected total rewards for each epoch MDP.

Experimental Setup

We considered the Storm implementation of value iteration (VI) and its extensions sound value iteration (SVI), interval iteration (II), and optimistic value iteration (OVI). Furthermore, we consider two variants of strategy iteration, where the underlying equation systems are solved with the numerical algorithm gmres [SS86] implemented in gmm++[4] (SI_{gmres}) and using direct solving via sparse LU factorization implemented in Eigen [GJ+10] (SI_{LU}), respectively. Finally, we consider the computation via linear programming using Gurobi (LP). All algorithms are also considered in their topological variant (VI^{topo}, $\text{SI}^{\text{topo}}_{\text{gmres}}$, ...).

Unless stated otherwise, the algorithms were invoked with *relative* precision $\varepsilon = 10^{-6}$. SVI, II, and OVI provide ε-sound results—assuming that floating-point errors can be neglected. SI_{LU} uses arbitrary precision rational numbers and thus yields the exact expected value. VI and SI_{gmres} provide no guarantee on the accuracy of the reported solution.

All instances of the quantitative verification benchmark set (QVBS) [7] with a total reachability reward objective (including subsumed objectives such as reachability probabilities) were considered. We excluded instances where the model building step (i.e., the translation from the high-level model specification to the explicit representation of Storm) did not terminate within 1 hour. This yields a total of 587 concrete ⟨model, objective⟩-instances consisting of 130 DTMC, 72 CTMC, 292 MDP[5], and 93 MA queries. Mixtures of positive and negative rewards do *not* occur in this benchmark set.

We compare the reported solutions with reference values, if they are available either from the QVBS data set or as an exact solutions reported by SI_{LU} or $\text{SI}^{\text{topo}}_{\text{LU}}$. A solution is considered incorrect (INC) if it deviates from the reference value by more than $\delta = 10^{-3}$.

For this experiment, the reported runtimes correspond to the time it takes to analyze the explicitly represented Markov model, i.e., we discard the model building time. However, the total time limit (2 hours) applies to the entire runtime of Storm.

Quantile and Scatter Plots

Our results are mostly presented in the form of quantile plots and scatter plots.

quantile plot

Quantile plots as in Figure 9.1 show the number of benchmark instances that the algorithms can solve within a given runtime (in seconds). More specifically, a point

[4] https://getfem.org/gmm.html

[5] Including 20 probabilistic timed automata (PTA) instances that were converted to MDPs via digital clock semantics using The Modest Toolset [HH14].

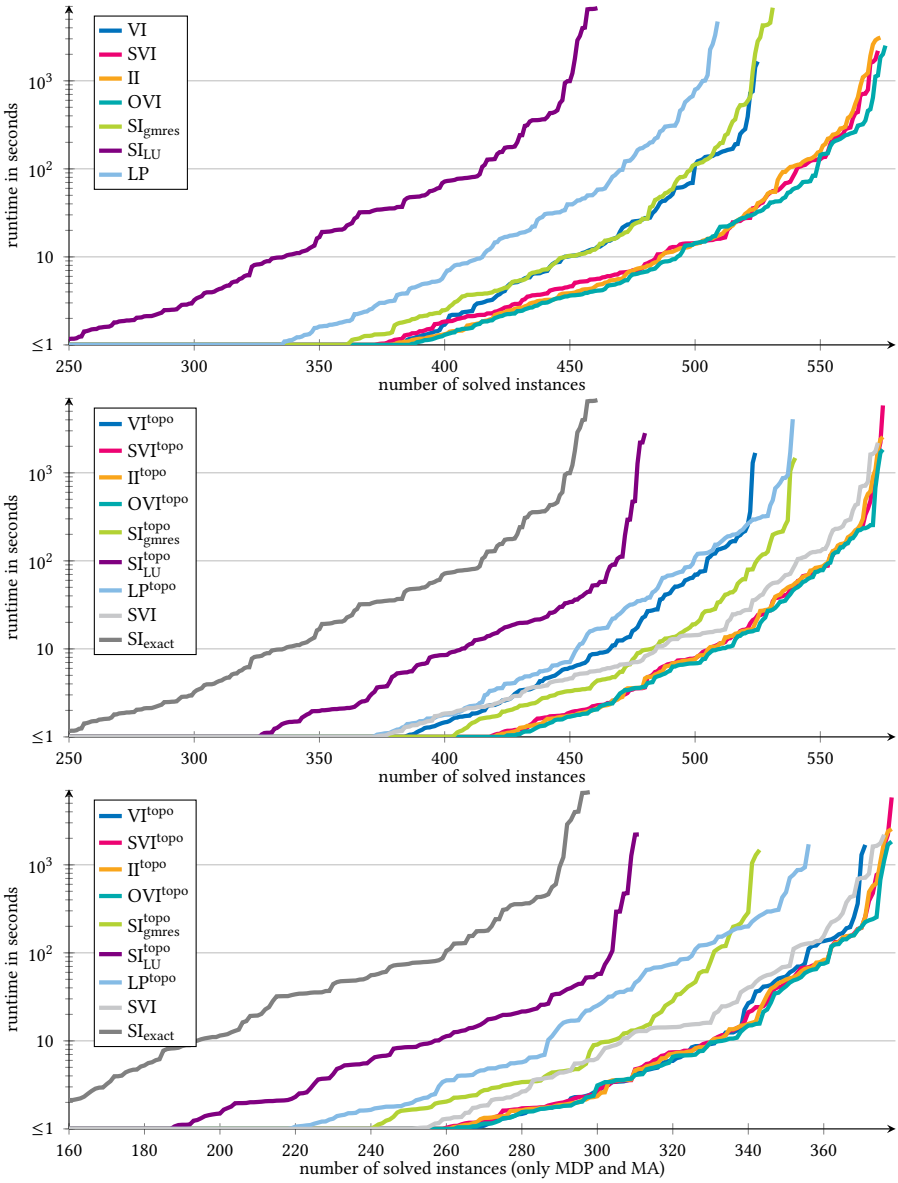


Figure 9.1: Runtime comparison for expected total reward algorithms without (top) and with (middle, bottom) topological optimizations, considering all (top, middle) and only MDP and MA (bottom) QVBS models

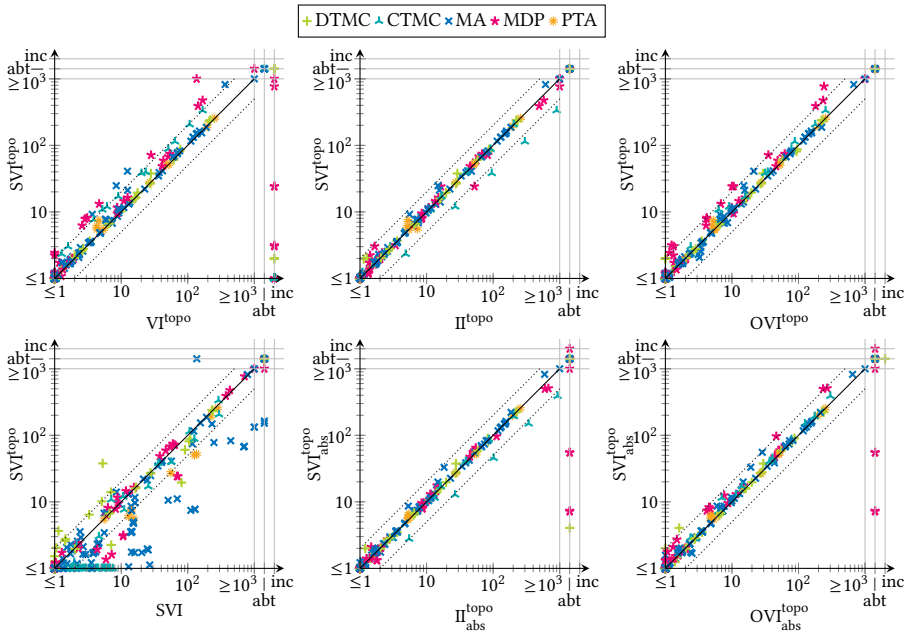


Figure 9.2: Runtime comparison for expected total reward algorithms

$\langle x, y \rangle$ on the line for Algorithm A indicates that there are x instances in our benchmark set such that A can solve any of these instances within y seconds. Consequently, a flat line in the plot indicates that there are many instances that the corresponding algorithm can solve with a low runtime. Instances for which an algorithm did not produce a valid result (e.g., due to reaching the time limit or because the reported result deviates too much from the reference result) are discarded from the plot.

scatter plot

We use *scatter plots* as in Figure 9.2 to compare the runtime (in seconds) of two algorithms on individual benchmark instances. Here, a point $\langle x, y \rangle$ means that for one of the instances we observed a runtime of x seconds when using the algorithm on the x-axis and y seconds when using the algorithm on the y-axis. Points above (below) the diagonal thus mean that the algorithm on the x-axis is faster (slower) compared to the algorithm on the y-axis. The dashed diagonal lines indicate that one algorithm is twice as fast. Whenever a point lies on the "abt"-line of the plot, the corresponding computation was aborted without producing a result, e.g., because the time limit of 2 hours was exceeded or because of an out-of-memory situation. Similarly, the "inc"-line indicates executions where the reported result is considered incorrect. The "$\geq 10^3$" line indicates that the execution successfully terminated *after* 1000 seconds but within the time limit.

Results and Discussion

The quantile plots in Figure 9.1 compare the performance of all exercised algorithms. The plot at the top considers the algorithms in their classic (non-topological) variant. We see that *for any given time limit the fastest algorithm is among the sound variants of value iteration, where OVI appears to be the fastest, followed by SVI and II.* The competition among those three algorithms is very tight, though. The unsound algorithms VI, SI_{gmres}, and LP clearly lag behind which—partially—can be explained by the fact that incorrectly solved instances (56 for VI, 28 for SI_{gmres}, 5 for LP) are discarded in the plot. While the only exact algorithm SI_{LU} has been the slowest approach in our experiment, it still managed to correctly solve the vast majority of the 587 benchmark instances.

The plot in the middle of Figure 9.1 shows the results obtained by the topological variants of the various approaches. For reference, we also include non-topological SVI and SI_{LU} in gray. *Topological solving improves the performance of all considered approaches. Still, the sound algorithms perform best.* The bottom plot also considers the topological algorithms but only for nondeterministic benchmark instances, i.e., MDP (including PTA) and MA models. VI is more competitive on this subset of benchmarks since the majority of the incorrectly solved VI instances were DTMCs and CTMCs. Moreover, SI_{gmres} now lags behind VI and shows similar performance compared to LP.

We show six scatter plots in Figure 9.2 that provide further insights. The three plots in the top row compare SVI against VI, II, and OVI using the topological variant of each. VI is generally slightly faster than SVI, but also incorrect on quite a few instances (there are 44 points overlapping at the coordinate $\langle \text{inc}, \leq 1 \rangle$). As in the quantile plot, *the runtimes of SVI, II, and OVI are very close with a slight trend towards OVI being the fastest.*

In the second row of Figure 9.2, we compare plain SVI with its topological variant and the sound algorithms when requesting an *absolute* precision of $\varepsilon = 10^{-6}$. For the former, we see that *topological optimizations improve the runtime for many instances, sometimes by several orders of magnitude.* On the other hand, the decomposition into SCCs also causes some overhead which for a few instances is quite significant. Computations with absolute precision yield a similar picture as their relative counterparts. However, incorrect results due to floating point issues appear for SVI and OVI.

In summary, SVI, II, and OVI all provide a good trade-off between accuracy and performance when it comes to computing expected total rewards. While plain VI is generally a bit faster, it is also less reliable as it yields quite a few incorrect results on our benchmark set. SI_{gmres} and LP lag behind VI and its sound variants. SI_{LU} is the slowest algorithm in our experiments, but useful for applications where floating-point inaccuracies can not be tolerated. We continue the experimental comparison of SVI, II, OVI, and SI_{LU} in a multi-objective context in Section 9.4.

Table 9.2: Multi-objective Markov models from the literature

application	name	description / reference
Network protocols	csn	Client-server mutex with N fails [KPC12; KM17]
	frw	FireWire root contention protocol [SV99; KNP12b; 7]
	sen	Sensor network [KPC12; KM17]
	tea	Team formation protocols [CKPS11; FKP12]
	wla	Wireless LAN [KNS02; KNP12b; 7]
Randomized algorithms	nnd	NAND multiplexing [NPKS05; KNP12b; 7]
	phi	Randomized dining philosophers [LR81; BCFK15; 7]
Queueing systems	pol	Polling system [TKPS12; 13; 7]
	rqs	Reentrant queueing system [HH12; 7]
	str	Video streaming client [13; 7]
Planning	res	Resource gathering [BN08; 7]
	rov	Mars rover [10]
	srv	Service robot [LPH17; 10]
	uav	UAV mission planning [FWHT15]
Scheduling	clu	Workstation cluster [HHK00; KNP12b; 7; 11]
	dpm	Dynamic power management [QWP99; 7]
	ejs	Energy-aware job scheduling [BDDK+14; 7]
	mut	Randomized mutex [TKPS12; 13]
	pow	Power management [FKNP+11]
	rab	Rabin randomized mutex [Rab82; BCFK15; 7]
Biology	vir	Network virus infection [KNPV09; BCFK15]

9.3 Markov Models with Multiple Objectives

For an extensive evaluation of our multi-objective verification algorithms, we gathered 14 MDP, 6 MA, and 1 DTMC[6] model families from the literature. Table 9.2 lists the individual model families grouped by their application areas. For each family, the table depicts a short name, a description, and a list of references. Table 9.3 shows further meta data such as the modeling formalism and the model type. The column "#inst / *tot*, *lra*, bnd_m, bnd_s, *qu*" gives the total number of concrete verification queries—i.e., combinations of a model $\mathcal{M}$ and a list of objectives Φ—and how those distribute over the following categories. We distinguish between multi-objective queries, involving

[6] The DTMC model is only relevant for the multi-reward bound analysis in Section 9.5.

Table 9.3: Benchmark instances

name	formalism	type	#inst / *tot*, *lra*, bnd_m, bnd_s, *qu*	$\lvert\Phi\rvert$	$\lvert S\rvert$
csn	PRISM	MDP	3 / 0, 3, 0, 0, 0	3..5	183..4863
frw	PRISM	MDP	7 / 2, 0, 2, 2, 1	1..2	776..$1{\cdot}10^7$
sen	PRISM	MDP	5 / 0, 5, 0, 0, 0	3	462..$4{\cdot}10^6$
tea	PRISM	MDP	8 / 8, 0, 0, 0, 0	2..3	865..$7{\cdot}10^5$
wla	PRISM	MDP	20 / 8, 0, 4, 4, 4	1..3	2954..$5{\cdot}10^6$
nnd	PRISM	DTMC	3 / 0, 0, 0, 3, 0	1	$2{\cdot}10^6$
phi	PRISM	MDP	3 / 0, 3, 0, 0, 0	2	9440..$2{\cdot}10^6$
pol	PRISM	MA	12 / 3, 9, 0, 0, 0	2	185..$1{\cdot}10^6$
rqs	JANI	MA	15 / 5, 10, 0, 0, 0	2	206..$3{\cdot}10^6$
str	PRISM	MA	4 / 4, 0, 0, 0, 0	2	175..$2{\cdot}10^6$
res	PRISM	MDP	21 / 3, 9, 3, 3, 3	1..3	94..$8{\cdot}10^5$
rov	PRISM	MDP	30 / 12, 0, 6, 6, 6	1..2	16..$5{\cdot}10^6$
srv	PRISM	MDP	4 / 2, 0, 1, 1, 0	1..2	$4{\cdot}10^4$..$3{\cdot}10^6$
uav	PRISM	MDP	12 / 4, 0, 4, 4, 0	1..2	3310..$8{\cdot}10^5$
clu	PRISM	MA	15 / 6, 9, 0, 0, 0	2..3	1756..$2{\cdot}10^6$
dpm	JANI	MA	9 / 0, 9, 0, 0, 0	2	2640..$1{\cdot}10^6$
ejs	PRISM	MDP	20 / 8, 0, 4, 4, 4	1..2	349..$1{\cdot}10^7$
mut	PRISM	MA	4 / 4, 0, 0, 0, 0	3	1399..$2{\cdot}10^5$
pow	PRISM	MDP	16 / 8, 0, 0, 0, 0	2..3	636..$3{\cdot}10^5$
rab	PRISM	MDP	3 / 0, 3, 0, 0, 0	2	$3{\cdot}10^4$..$1{\cdot}10^7$
vir	PRISM	MDP	3 / 0, 3, 0, 0, 0	2	80..$2{\cdot}10^4$

- only total reachability reward objectives (*tot*),
- at least one long-run average reward objective (*lra*), or
- at least one reward-bounded objective (bnd_m),

as well as single-objective queries, with a multi-dimensional

- reward-bounded objective (bnd_s), or
- quantile (*qu*).

The pow model family also considers 8 multi-objective queries with *step-bounded* expected rewards. All other considered queries belong to exactly one of the above-

mentioned categories. In total, our benchmark set consists of 172 multi-objective, 27 (single-objective) multi-reward bounded queries, and 18 quantile queries. The last two columns of Table 9.3 indicate the range of the number of considered objectives and the number of model states, respectively. We thus cover verification queries of various intricacy, where models can have millions of states.

9.4 Experiments for Multi-Objective Queries

We first consider the multi-objective verification queries in our benchmark set. More specifically, we look at the performance of Storm on these benchmarks and compare Storm with other tools afterwards.

9.4.1 Performance of Storm's methods

Experimental Setup

MOP, multi-objective Pareto

We compute an approximation of the Pareto front for each of the 172 multi-objective verification queries. More precisely, we solve the multi-objective Pareto (MOP) problem (cf. Problem 3.3 on page 79) with a precision parameter $\eta = 10^{-4}$—which is the default value in Storm. We consider the most promising expected total reward algorithms from Section 9.2, namely the sound variants of value iteration SVI, II, and OVI. For our multi-objective experiments, these algorithms are invoked with *absolute* precision $\varepsilon = 10^{-6}$. Additionally, we consider an *exact* configuration (EX), where all values are represented with arbitrary precision rational numbers and expected total rewards are computed with strategy iteration using sparse LU factorization (referred to as $\mathrm{SI}_{\mathrm{LU}}$ in Section 9.2). Hence, the WSO instances are solved exactly. All methods are used in their topological variant. The *exact* configuration is currently not implemented for queries involving long-run average objectives.

For each combination of verification query and method we measure the total runtime of Storm (including model building time). We do not check the reported Pareto front approximations against reference results.

Results and Discussion

Figures 9.3 and 9.4 summarize our results. The top quantile plot in Figure 9.3 compares all considered Storm methods. The plot only considers instances that are supported by *all* methods, which—due to the above-mentioned limitations of EX regarding LRA objectives—restricts the total number of instances to 109. The quantile plot in the middle omits EX and thus considers all 172 instances. The three scatter plots at the bottom of Figure 9.3 compare the runtime of SVI with the other methods II, OVI, and EX. The "$\geq 10^3$" and the "abt"-line are as in the scatter plots of Section 9.2 and indicate

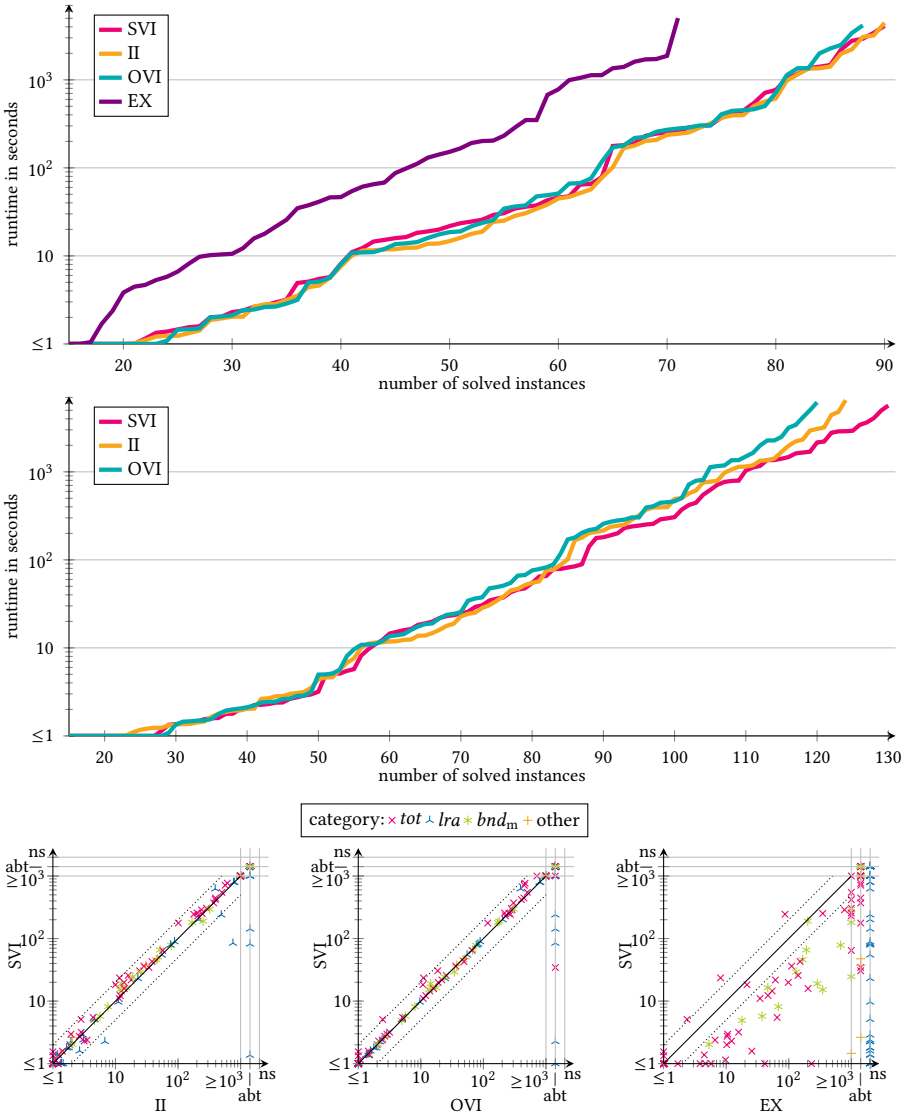


Figure 9.3: Runtime comparison of Storm's multi-objective verification methods

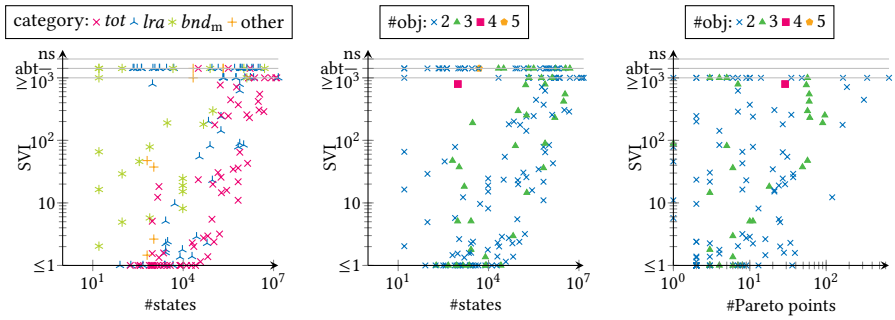


Figure 9.4: Correlation between runtime and model size / trade-off intricacy

that the computation finished after 10^3 seconds or was aborted, respectively. The "ns"-line shows queries that are not supported by the corresponding method. The different markers indicate the different query categories described in Section 9.3. The scatter plots in Figure 9.4 plot the runtime of SVI against the number of model states (left, middle) and the number of Pareto points in the resulting approximation (right). The latter gives a rough indication on the intricacity of the trade-offs between the objectives. For the middle and right plot, the markers indicate the number of objectives considered in the query.

As in Section 9.2, we see that *the runtimes of SVI, II, and OVI are very close to each other, while there is a noticeable overhead for the exact methods.* In the second quantile plot of Figure 9.3, we see that SVI is slightly ahead of II and OVI on this benchmark set. The scatter plots below reveal that this is mainly because both II and OVI fail to converge within the time limit for a few instances.

Figure 9.4 shows that Storm was able to solve instances with *millions of states*. However, for intricate queries—in particular reward-bounded objectives (bnd_m)—small models can give rise to high runtimes as well. The vast majority of queries in our benchmark set consider two or three objectives. *Queries with three objectives tend to have a higher runtime, but the difference is not very significant.* For some queries, a single Pareto optimal point already yields a precise enough approximation of the Pareto front. For others, hundreds of points are necessary—even if there are just two objectives.

9.4.2 Comparison With Related Tools

Experimental Setup

We compare Storm against the following tools.

- EPMC obtained from https://github.com/iscas-tis/ePMC, commit b1ba8ab.
- PRISM obtained from https://github.com/prismmodelchecker/prism, commit 3a632e2.
- MULTIGAIN obtained from http://qav.cs.ox.ac.uk/multigain/, version 1.0.2.

For PRISM, we consider both the sandwich algorithm (PRISM_{sw}) using Gauss-Seidel value iteration for expected total reward computations, and the linear programming-based method (PRISM_{lp}) using the LP solver LP_SOLVE [BEN22]. For MULTIGAIN, we use GUROBI [Gur22] to solve the underlying LPs. STORM is invoked with SVI as in Section 9.4.1. For all tools, we set an absolute precision of $\varepsilon = 10^{-6}$ for expected total reward computations (if applicable).

We consider the same 172 benchmark instances as before. However, many benchmark instances—in particular those with reward-bounded objectives as well as mixtures of LRA and total reward objectives—are only supported by STORM[7].

Approximating Pareto fronts is not supported for EPMC and PRISM_{lp}, and only supported for $\ell = 2$ objectives for PRISM_{sw} and MULTIGAIN. We therefore also consider multi-objective *achievability* (MOA) queries (cf. Problem 3.1 on page 78), which ask whether a given point $\boldsymbol{p} \in \mathbb{R}^{\ell}$ is achievable. Roughly, those points have been determined by setting the threshold $\boldsymbol{p}(j)$ for the j^{th}objective to 90% of the optimal expected value for that objective, computed via standard single-objective model checking. This way, each objective threshold alone (i.e., when discarding the other objectives) is achievable, but their multi-objective combination might or might not be achievable.

Results and Discussion

Figures 9.5 and 9.6 summarize our results. The quantile plots in Figure 9.5 compare the runtime of the various tools on achievability (MOA) queries. Each plot is restricted to the instances that are supported by *all* depicted tools, giving rise to 46 instances in the upper plot and 20 instances in the lower plot. The scatter plots in Figure 9.6 directly compare the runtime of STORM against the other tools and—in the case of PRISM_{sw} and MULTIGAIN—also consider Pareto (MOP) queries.

For the MOA queries, we observed inconsistencies of the results between the tools for 13 total reward instances from the model families pow, rov, and srv. After some investigation, these are likely caused by issues in the EPMC and PRISM implementations. For example, PRISM_{sw} and PRISM_{lp} sometimes reported different results and EPMC claimed some points to be achievable, even if the threshold of one of the objectives alone is already unachievable by a great margin. Since it is non-trivial to indubitably

[7] As STORM is the focus of this evaluation, our benchmark set intentionally does not contain queries that are currently unsupported by STORM.

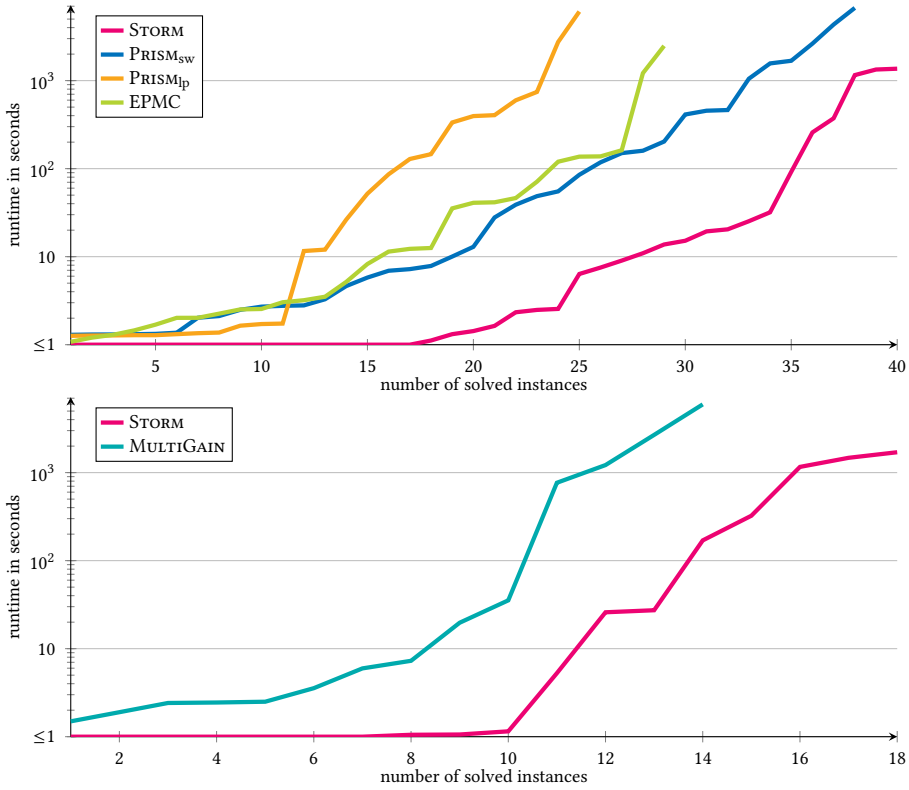


Figure 9.5: Runtime comparison of multi-objective verification tools (1/2)

identify the *correct* solution for all queries, those problematic cases are still included in our figures (regardless of the reported result).

We see that *Storm outperforms the other tools on the vast majority of instances—often by several orders of magnitudes.* In particular, the LP-based implementations Prism$_{lp}$ and MultiGain are significantly slower and less scalable. These observations hold for both `MOA` and `MOP`.

9.5 Experiments for Multi-Reward Bounded Objectives & Quantiles

We now consider verification queries involving multiple reward bounds.

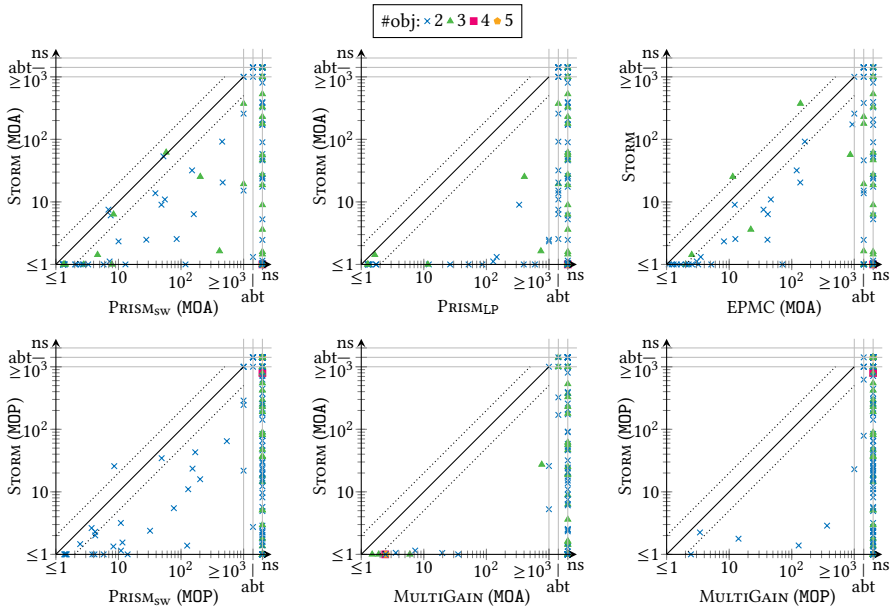


Figure 9.6: Runtime comparison of multi-objective verification tools (2/2)

Experimental Setup

We consider

- computing optimal expected values of a single objective with (multiple) reward bounds,
- approximating Pareto fronts for multiple objectives with reward bounds, and
- computing 2-dimensional quantiles.

For the former two, we compare the sequential approach (Seq) with the explicit unfolding (Unf) using sound value iteration (SVI) and exact methods (EX). The explicit unfolding described in Section 6.1 has been done "by hand" on the level of the modeling formalism (`PRISM` or `JANI`). This allows us to also use Storm's symbolic model checking techniques for the unfolded models, namely the dd engine (DD) and the hybrid engine (HYB). Both use multi-terminal binary decision diagrams [Ake78] to represent the model. While DD performs all computations on this representation, HYB switches to an explicit format for the numerical computations (i.e., for SVI). For DD, we use classical value iteration (VI) since SVI is not implemented. Neither DD nor

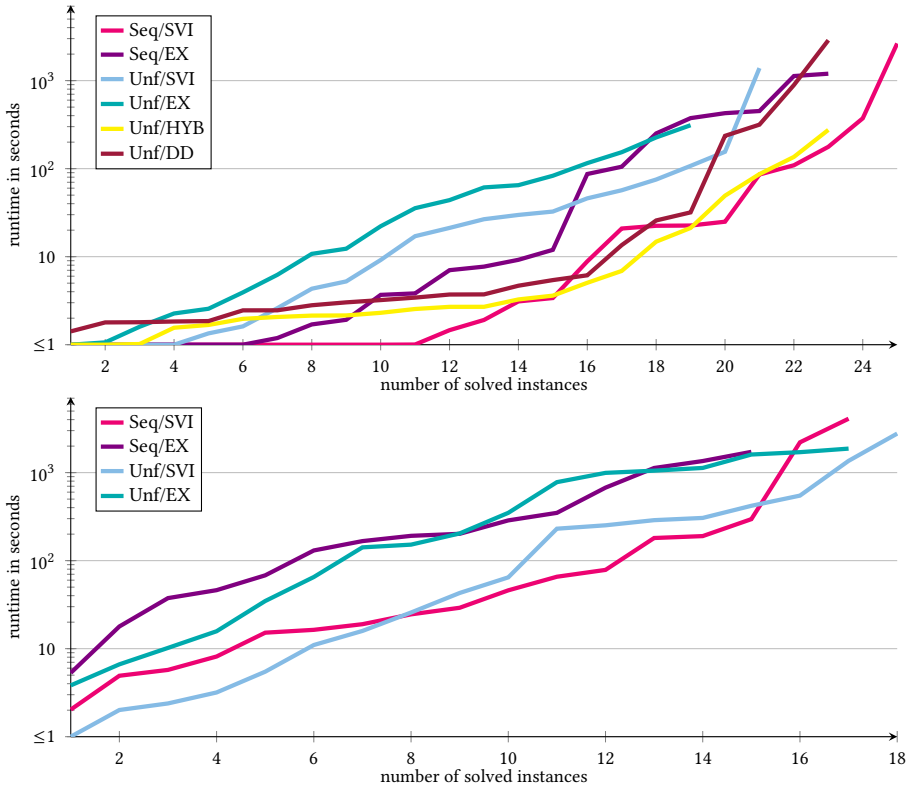


Figure 9.7: Runtime comparison of sequential and unfolding approach for reward-bounded objectives (1/2)

HYB support multi-objective queries. For the quantile queries, we distinguish between SVI and EX.

Results and Discussion

The quantile plots in Figure 9.7 compare the runtime of the considered methods on the 27 single-objective (upper) and 24 multi-objective (lower) instances in our benchmark set. The first two scatter plots in Figure 9.8 directly compare the sequential and the unfolding approach. *For the single-objective instances, we see that the sequential approach clearly outperforms the unfolding approach when using explicit data structures.* While symbolic model checking techniques yield a noticeable improvement for the unfolding approach, the sequential approach still takes the lead for most instances.

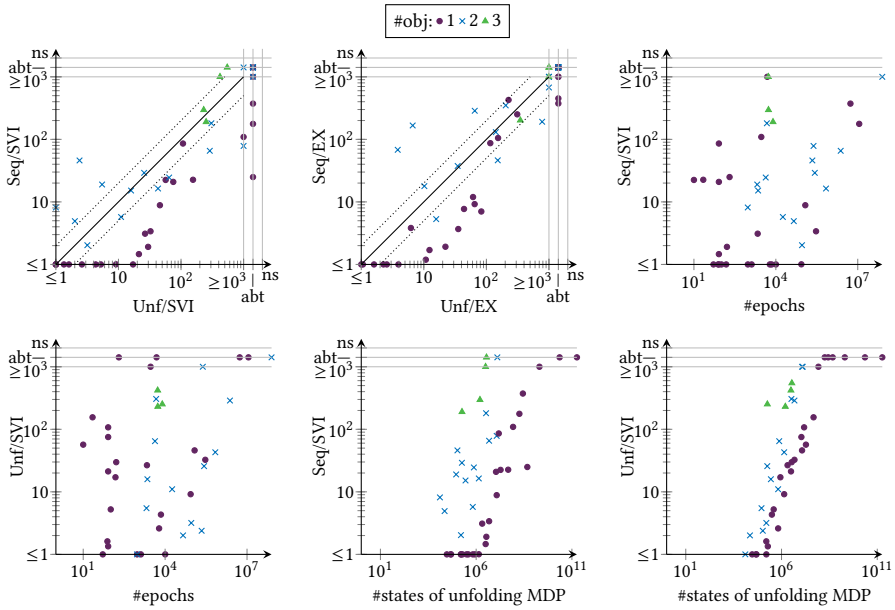
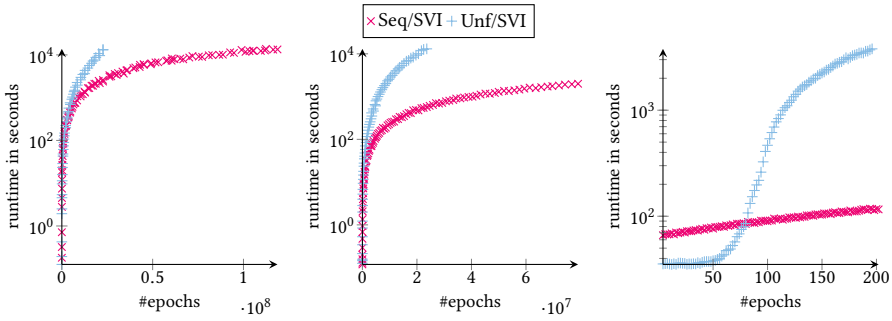


Figure 9.8: Runtime comparison and correlation with epoch- and state count of sequential and unfolding approach for reward-bounded objectives

In a *multi-objective* setting, the competition is much more tight. Here, *the unfolding approach actually solves more instances than the sequential approach.* However, the sequential approach is faster on 10 instances while the unfolding approach is faster on 9 instances (the remaining 5 instances in our benchmark set are unsolved by both). The most important reason that the unfolding approach performs better in some cases is that it can restrict the model analysis precisely to those states of the unfolding MDP that are reachable from the initial state. The sequential approach, on the other hand, analyzes the full epoch MDP for each epoch, even if some of the states are not relevant for the resulting value. This is more noticeable in a multi-objective setting, as the model analysis is more involved then.

The remaining four plots in Figure 9.8 show the correlation between runtime and the number of epochs that needed to be analyzed as well as the number of states in the unfolded model. The plots indicate that *a low (or high) number of epochs does not necessarily mean that the runtime for the two approaches is also low (or high).* The size of the epoch MDPs and the general structure of the model are important, as well. However, there is a more clear *correlation between the number of states of the unfolded model and the runtime* of the sequential and—in particular—the unfolding approach.



(a) res (multi-obj) (b) rov (multi-obj) (c) wla (single-obj)

Figure 9.9: Runtime for increasing cost bounds on three benchmark instances

Besides the benchmarks from Section 9.3, we also picked three instances from the res, rov, and wla families and measured the runtime of the sequential and the unfolding approach for varying reward bound values. Both approaches use SVI and explicit (non-symbolic) model representations. Only for this experiment, we increased the time- and memory limit to 4 hours and 64 GB, respectively. We plot the number of considered epochs (which reflects the magnitude of the reward bound values) against the runtime of the two approaches in Figure 9.9. *While for small bound values unfolding is sometimes faster, the sequential approach scales better with an increasing number of epochs.* Ultimately, the increased state space size and the accompanying memory consumption of the unfolding MDP becomes a bottleneck.

Figure 9.10 summarizes our results for the 18 2-dimensional quantile queries. We compare the runtime of SVI and EX and show how the runtime correlates with the size of the resulting generators and the number of analyzed epoch models—both giving an indication on the intricacy of the query. The markers in the scatter plot also show how many (indefinite plus definite) reward bounds the query considers. We successfully computed quantiles for 14 out of 18 instances, indicating the practicability of our approach. Naturally, *the runtime for quantiles largely depends on the number of analyzed epochs—which can be in the thousands or even millions.* The resulting generators are comparably small, allowing for a concise representation of the set of satisfying reward limits.

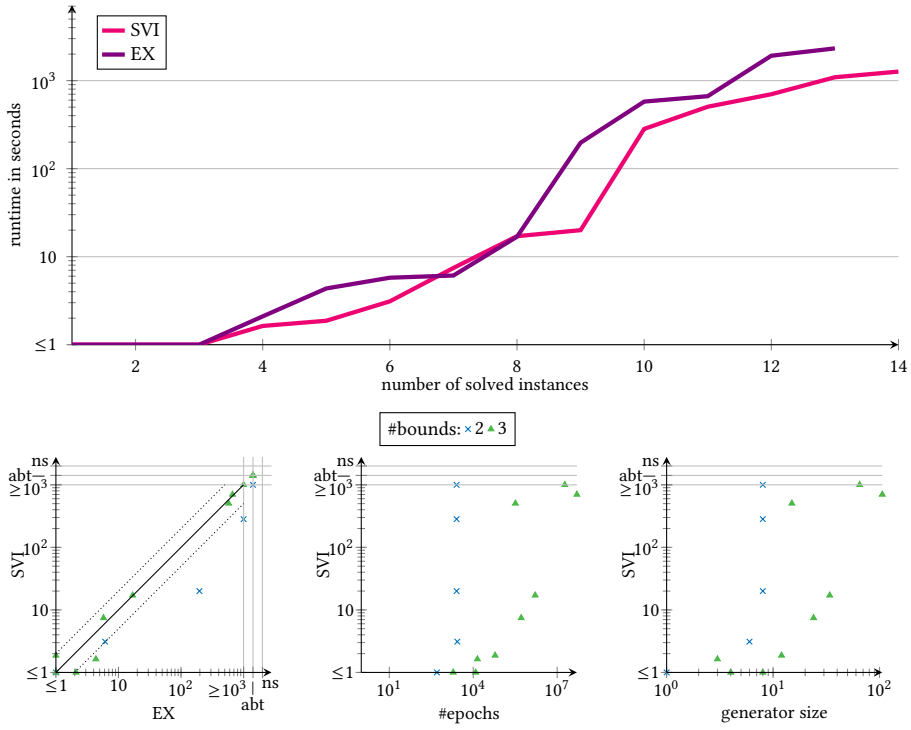


Figure 9.10: Results for the computation of 2-dimensional quantiles

9.6 Experiments for Multi-Objective Verification under Simple Strategies

Experimental Setup

We solve the pure memoryless multi-objective Pareto problem (cf. Problem 8.3 on page 268) for each of the 77 total reachability reward instances from Section 9.3. To obtain similarly challenging precision requirements across the different benchmark instances, we determine the precision vector $\boldsymbol{\varepsilon} \in (\mathbb{R}_{>0})^{\ell}$ as follows. For a given precision parameter $\varepsilon > 0$ and $j \in \{1..\ell\}$, we set

$$\boldsymbol{\varepsilon}(j) = \varepsilon \cdot \max\left\{ |\boldsymbol{p}(j) - \boldsymbol{q}(j)| \;\middle|\; \boldsymbol{p}, \boldsymbol{q} \in P \right\},$$

where P is the set of achievable points found after the first ℓ iterations of Algorithm 8.2, i.e., after analyzing the weight vectors in $\left\{ \mathbf{1}_i \in (\mathbb{R}_{\geq 0})^{\ell} \;\middle|\; i \in \{1..\ell\} \right\}$. We consider two

different precision parameters $\varepsilon \in \{10^{-2}, 10^{-3}\}$.

In our experiments, we consider 6 variants of the `MILP` encoding for pure stationary achievability. More specifically, we compare the encoding from Section 8.3 based on the *classic* equation system characterization for total reachability rewards, where BSCCs are detected via

- C_v: the visiting times approach and
- C_r: the reachability order approach

against the

- V: alternative encoding from Section 8.4 based on the equation system for *visiting times*.

Furthermore, we consider two variants for each of the three encodings above based on

- M: *Big-M constraints*, i.e., the constraints as outlined in Chapter 8, potentially dealing with very large coefficients and
- I: equivalent *indicator constraints*, i.e., constraints as discussed in Remark 8.2 on page 262.

Results and Discussion

Figure 9.11 shows the runtimes for solving the multi-objective Pareto problems using the various encodings, where we consider the precision parameter $\varepsilon = 10^{-2}$ for the quantile and the first four scatter plots. The quantile plot disregards 6 instances as they consider total *reachability* reward objectives that are not supported by the encodings based on expected visiting times (V/M and V/I). The scatter plots consider all 77 instances and provide a direct comparison between a selection of encodings and—for the last two plots—between the two considered precisions.

We see that the encodings based on expected visiting times (V/M and V/I) mostly outperform the other encodings on those benchmarks that they support. Dealing with indicator constraints instead of big-M constraints tends to be a bit slower. The reachability order approach for detecting BSCCs is slightly faster compared to the visiting times variant. *Overall, the encoding based on the visiting times characterization with big-M constraints (V/M) performed best in our experiments.* Still, it only solves 25 out of the 71 supported instances within the considered time- and memory limit. Increasing the precision requirements from $\varepsilon = 10^{-2}$ to $\varepsilon = 10^{-3}$ further reduces the number to 14 solved instances. When compared to the performance for general strategies shown in Section 9.4, these observations indicate that *multi-objective model checking under pure stationary strategies is less scalable which is in line with our observations regarding*

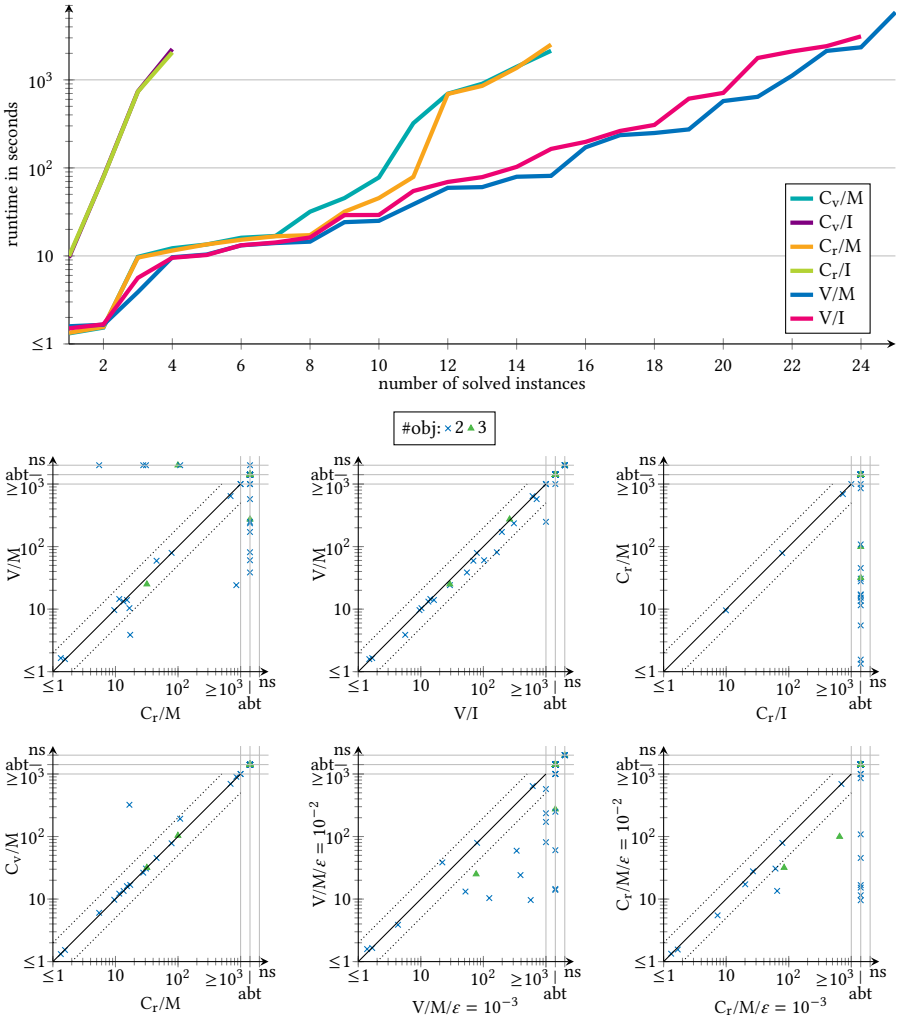


Figure 9.11: Comparison of encodings for pure stationary achievability

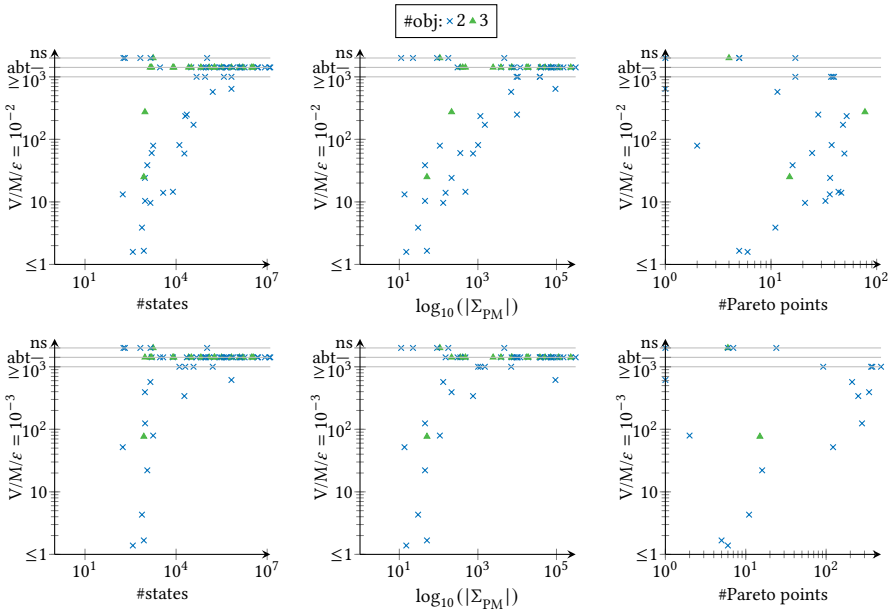


Figure 9.12: Correlation between runtime and model size / number of strategies / trade-off intricacy for $\varepsilon = 10^{-2}$ (upper) and $\varepsilon = 10^{-3}$ (lower)

the theoretical complexity of the general and the pure stationary achievability problem in Section 8.1.

Figure 9.12 plots the runtime obtained for the V/M encoding against the number of model states, the number of pure stationary strategies in the model, and the number of Pareto points in the obtained approximation. The three upper plots consider $\varepsilon = 10^{-2}$ while the three lower plots consider $\varepsilon = 10^{-3}$. Our implementation is able to compute pure stationary Pareto fronts for some models with over 100 000 states. At the same time, however, models with a few thousand states can already be a challenge. The two plots in the middle show the logarithm of the number of pure stationary strategies, i.e., a point with x-value 10^3 means that there are $10^{10^3} = 10^{1000}$ strategies. Hence, in most cases a naive brute-force enumeration and analysis of all strategies is infeasible. For $\varepsilon = 10^{-2}$, 10 to 100 Pareto optimal points (i.e., strategies) are often sufficient to represent the requested Pareto front approximation. For $\varepsilon = 10^{-3}$, much more Pareto points need to be found which often leads to timeouts. Similarly, for more than 2 objectives the desired accuracy can often not be achieved within the time limit.

In Figure 9.13 we plot the Pareto front for an instance of the pol family under differently restricted strategies. More specifically, we consider general strategies (Gen),

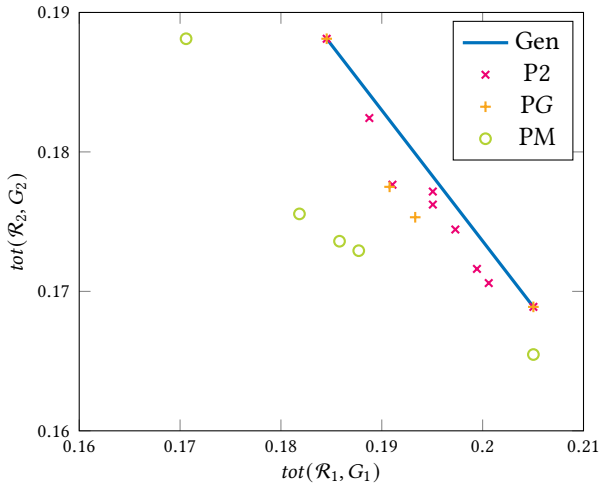


Figure 9.13: Impact of memory for a pol instance

pure 2-memory strategies that can change the memory state exactly once (P2), pure strategies that observe which goal state set G_j has been visited already (PG), and pure memoryless strategies (PM). Adding simple memory structures already leads to noticeable improvements in the quality of strategies. In particular, P2 strategies perform quite well, and even outperform PG strategies—which would be optimal if randomization were allowed.

Chapter Summary

- ➜ STORM is a comprehensive and powerful tool for multi-objective verification of Markov models.
- ➜ Sound variants of value iteration improve the accuracy of results on practical benchmark instances while only inducing little runtime overhead. The runtimes of the three approaches SVI, II, and OVI are close.
- ➜ The sandwich algorithm implementation in STORM for multi-objective verification scales to large models with millions of states—outperforming competing tools up to several orders of magnitude on the vast majority of benchmarks.
- ➜ The sequential epoch analysis for reward-bounded objectives often scales better than the naive unfolding approach, especially for large bound values.
- ➜ The `MILP` based multi-objective verification under pure memoryless strategies scales to medium-sized models with up to approximately 100 000 states.
- ➜ Experimental data is available at

  `https://doi.org/10.5281/zenodo.7766202`.

—Chapter 10—

Conclusion and Future Work

10.1 Conclusion

Multi-objective verification investigates the interplay between different, potentially conflicting objectives of a system and reveals the achievable trade-offs. We studied a flexible algorithmic framework based on a sandwich algorithm and a reduction to multiple instances of the *weighted sum optimization problem* (WSO) solved via single-objective algorithms. To improve practicality, we considered approximative approaches that provide correct and arbitrarily tight error bounds. We showed how these errors propagate through the entire verification procedure.

Our framework has been instantiated with (mixtures of) total reachability reward objectives, long-run average reward objectives, and reward-bounded objectives. For computing expected total rewards, we presented *sound value iteration* which bounds the error made after k iterations of classical value iteration by considering the probability that more reward can still be accumulated *after* k steps. *Mixtures* of total- and long-run average rewards were handled with a careful treatment of end components. Our approach for reward-bounded objectives analyzes the reward epochs *sequentially*, one-after-the-other. The latter idea can also be used to compute *multi-dimensional quantiles* which reveal the possible resource constraints under which an objective is achievable. Finally, we studied multi-objective verification under *simple strategies*, i.e., strategies with no randomization and limited memory usage.

An implementation of all these approaches is publicly available in the model checker Storm. Our experiments have shown that multi-objective verification scales to large practical models with millions of states.

10.2 Future Work

Multi-objective verification is a rich research field. The results in this work can be extended in several dimensions. We outline a few promising directions for future work.

More Objectives | The general framework for multi-objective analysis can be instantiated with additional types of objectives. This includes percentile objectives [RRS17], long-run ratio objectives [BDK14], discounted total reward objectives [CMH06], and ω-regular objectives, e.g., using LTL [EKVY08]. Since many of these objectives concern the system behavior in the long-run, a sophisticated treatment of end components when solving `WSO` queries appears to be the crux for such extensions—in particular when considering mixtures of different objective types. Another direction is the incorporation of the beyond-worst-case paradigm [BFRR17; BGMR18; BGR20], where some objectives impose constraints on *all* paths of the model—even those with probability zero. Finally, we point out that our `MILP` encoding for achievability under simple strategies can potentially be extended towards, e.g., LRA reward objectives.

Additional Models | Our techniques can potentially be lifted to richer modeling formalisms. An interesting direction is to consider multi-objective stochastic games (SGs) as in, e.g., [BKW18; ACKW+20]. We see the lifting of our `WSO` approaches as a major obstacle: in a two-player SG, Player 1 can only achieve a point if all objective values are achieved, no matter which actions Player 2 performs. Consequently, a result of a `WSO` query (i.e., a point on the Pareto front) is not necessarily induced by a single strategy for Player 2 as a different adversarial strategy for each objective has to be considered. Other related formalisms include probabilistic timed automata, parametric Markov models, or partially observable MDPs. Their single-objective analysis methods often rely on a reduction to plain single-objective MDP or SG analysis. There is hope that a similar connection exists in the multi-objective case which would enable a straightforward lifting of our methods.

Alternative Algorithms | Our Storm implementation can be further extended by implementing more algorithms from the literature. For sound approximations of expected total rewards, one can additionally consider bisection value iteration [LX22] and the approach of Hansen [Han17], as well as implementations with safe floating-point rounding as in [Har22]. For the computation of LRA rewards, a strategy iteration-based approach as in [KM17] can be considered, which is promising for computing exact solutions. Lastly, it would be interesting to explore the use of evolutionary algorithms in a multi-objective verification context as in [GCCA+21].

New Applications | We hope the theoretical insights and practical results from this work help pushing the state of the art—paving the way for new application areas for multi-objective verification. Due to the improved scalability, controllers for large-scale planning problems with intricate objective trade-offs can be synthesized. Another possible direction is to revisit the assume-guarantee framework of [EKVY08; FKNP+11] and investigate if this yields verification speed-ups when integrated in Storm.

Declaration of Authorship

The doctoral degree regulations at RWTH Aachen University require the following.

Eidesstattliche Erklärung

I, Tim Quatmann declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

Hiermit erkläre ich an Eides statt / I do solemnely swear that:

1. This work was done wholly or mainly while in candidature for the doctoral degree at this faculty and university;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others or myself, this is always clearly attributed;
4. Where I have quoted from the work of others or myself, the source is always given. This thesis is entirely my own work, with the exception of such quotations;
5. I have acknowledged all major sources of assistance;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. Parts of this work have been published before as detailed in Section 1.6 on page 13.

Tim Quatmann, October 2023, Aachen

Bibliography

References

[ABAV20] G. K. Atia, A. Beckus, I. Alkhouri, and A. Velasquez. "*Steady-State Policy Synthesis in Multichain Markov Decision Processes.*" *IJCAI*. ijcai.org, 2020, pages 4069–4075. DOI: 10.24963/ijcai.2020/563 (cited on page 168).

[ACDK+17] P. Ashok, K. Chatterjee, P. Daca, J. Křetínský, and T. Meggendorfer. "*Value Iteration for Long-Run Average Reward in Markov Decision Processes.*" *CAV (1)*. Volume 10426. LNCS. Springer, 2017, pages 201–221. DOI: 10.1007/978-3-319-63387-9_10 (cited on pages 41, 153, 155, 157, 168).

[ACKW+20] P. Ashok, K. Chatterjee, J. Křetínský, M. Weininger, and T. Winkler. "*Approximating Values of Generalized-Reachability Stochastic Games.*" *LICS*. ACM, 2020, pages 102–115. DOI: 10.1145/3373718.3394761 (cited on pages 12, 306).

[AD99] R. B. Ash and C. Doléans-Dade. "*Probability and Measure Theory.*" Harcourt/Academic Press, 1999 (cited on pages 15, 28, 35).

[AHK03] S. Andova, H. Hermanns, and J.-P. Katoen. "*Discrete-Time Rewards Model-Checked.*" *FORMATS*. Volume 2791. LNCS. Springer, 2003, pages 88–104. DOI: 10.1007/978-3-540-40903-8_8 (cited on pages 171, 205).

[Ake78] S. B. Akers. "*Binary Decision Diagrams.*" *IEEE Trans. Computers* 27.6 (1978), pages 509–516. DOI: 10.1109/TC.1978.1675141 (cited on page 295).

[Alf97] L. de Alfaro. "*Formal verification of probabilistic systems.*" PhD thesis. Stanford University, USA, 1997 (cited on pages 41, 107, 111, 153, 155, 168, 246).

[ASSB00] A. Aziz, K. Sanwal, V. Singhal, and R. K. Brayton. "*Model-checking continous-time Markov chains*." *ACM Trans. Comput. Log.* 1.1 (2000), pages 162–170. DOI: 10.1145/343369.343402 (cited on page 4).

[Aum64] R. J. . Aumann. "*28. Mixed and Behavior Strategies in Infinite Extensive Games*." *Advances in Game Theory. (AM-52), Volume 52*. Princeton University Press, 1964, pages 627–650. DOI: 10.1515/9781400882014-029 (cited on page 35).

[BAFK18] C. Baier, L. de Alfaro, V. Forejt, and M. Kwiatkowska. "*Model Checking Probabilistic Systems*." *Handbook of Model Checking*. Springer, 2018, pages 963–999. DOI: 10.1007/978-3-319-10575-8_28 (cited on page 4).

[BBCF+14] T. Brázdil, V. Brozek, K. Chatterjee, V. Forejt, and A. Kucera. "*Two Views on Multiple Mean-Payoff Objectives in Markov Decision Processes*." *Log. Methods Comput. Sci.* 10.1 (2014). DOI: 10.2168/LMCS-10(1:13)2014 (cited on pages 7, 12, 55, 70, 78, 155, 168, 277, 283).

[BBDG+18] C. Baier, N. Bertrand, C. Dubslaff, D. Gburek, and O. Sankur. "*Stochastic Shortest Paths and Weight-Bounded Properties in Markov Decision Processes*." *LICS*. ACM, 2018, pages 86–94. DOI: 10.1145/3209108.3209184 (cited on pages 41, 44, 50).

[BBEK13] T. Brázdil, V. Brozek, K. Etessami, and A. Kucera. "*Approximating the termination value of one-counter MDPs and stochastic games*." *Inf. Comput.* 222 (2013), pages 121–138. DOI: 10.1016/j.ic.2012.01.008 (cited on page 206).

[BBNO+20] F. Blahoudek, T. Brázdil, P. Novotný, M. Ornik, P. Thangeda, and U. Topcu. "*Qualitative Controller Synthesis for Consumption Markov Decision Processes*." *CAV (2)*. Volume 12225. LNCS. Springer, 2020, pages 421–447. DOI: 10.1007/978-3-030-53291-8_22 (cited on page 206).

[BCCF+14] T. Brázdil, K. Chatterjee, M. Chmelik, V. Forejt, J. Křetínský, M. Z. Kwiatkowska, D. Parker, and M. Ujma. "*Verification of Markov Decision Processes Using Learning Algorithms*." *ATVA*. Volume 8837. LNCS. Springer, 2014, pages 98–114. DOI: 10.1007/978-3-319-11936-6_8 (cited on pages 41, 146).

[BCFK15] T. Brázdil, K. Chatterjee, V. Forejt, and A. Kucera. "*MultiGain: A Controller Synthesis Tool for MDPs with Multiple Mean-Payoff Objectives*." *TACAS*. Volume 9035. LNCS. Springer, 2015, pages 181–187. DOI: 10.1007/978-3-662-46681-0_12 (cited on pages 12, 168, 283, 288).

[BCFK17] T. Brázdil, K. Chatterjee, V. Forejt, and A. Kucera. "*Trading performance for stability in Markov decision processes.*" *J. Comput. Syst. Sci.* 84 (2017), pages 144–170. DOI: 10.1016/j.jcss.2016.09.009 (cited on page 168).

[BCM18] D. A. Basin, C. Cremers, and C. A. Meadows. "*Model Checking Security Protocols.*" *Handbook of Model Checking*. Springer, 2018, pages 727–762. DOI: 10.1007/978-3-319-10575-8_22 (cited on page 1).

[BCNO+21] F. Blahoudek, M. Cubuktepe, P. Novotný, M. Ornik, P. Thangeda, and U. Topcu. "*Fuel in Markov Decision Processes (FiMDP): A Practical Approach to Consumption.*" *FM*. Volume 13047. LNCS. Springer, 2021, pages 640–656. DOI: 10.1007/978-3-030-90870-6_34 (cited on page 206).

[BDDK+14] C. Baier, M. Daum, C. Dubslaff, J. Klein, and S. Klüppelholz. "*Energy-Utility Quantiles.*" *NASA Formal Methods*. Volume 8430. LNCS. Springer, 2014, pages 285–299. DOI: 10.1007/978-3-319-06200-6_24 (cited on pages 9, 171, 187, 204, 205, 207, 221, 223, 225, 227–229, 288).

[BDH96] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. "*The Quickhull Algorithm for Convex Hulls.*" *ACM Trans. Math. Softw.* 22.4 (1996), pages 469–483. DOI: 10.1145/235815.235821 (cited on pages 21, 22, 89, 272).

[BDHH+17] C. E. Budde, C. Dehnert, E. M. Hahn, A. Hartmanns, S. Junges, and A. Turrini. "*JANI: Quantitative Model and Tool Interaction.*" *TACAS (2)*. Volume 10206. LNCS. 2017, pages 151–168. DOI: 10.1007/978-3-662-54580-5_9 (cited on pages 4, 280).

[BDK14] C. Baier, C. Dubslaff, and S. Klüppelholz. "*Trade-off analysis meets probabilistic model checking.*" *CSL-LICS*. ACM, 2014, 1:1–1:10. DOI: 10.1145/2603088.2603089 (cited on pages 13, 168, 229, 306).

[BDKD+14] C. Baier, C. Dubslaff, S. Klüppelholz, M. Daum, J. Klein, S. Märcker, and S. Wunderlich. "*Probabilistic Model Checking and Non-standard Multi-objective Reasoning.*" *FASE*. Volume 8411. LNCS. Springer, 2014, pages 1–16. DOI: 10.1007/978-3-642-54804-8_1 (cited on page 229).

[BDKK+14] C. Baier, C. Dubslaff, J. Klein, S. Klüppelholz, and S. Wunderlich. "*Probabilistic Model Checking for Energy-Utility Analysis.*" *Horizons of the Mind*. Volume 8464. LNCS. Springer, 2014, pages 96–123. DOI: 10.1007/978-3-319-06880-0_5 (cited on page 229).

[BDKK+17] C. Baier, C. Dubslaff, L. Korenciak, A. Kucera, and V. Rehák. "*Synthesis of Optimal Resilient Control Strategies.*" *ATVA*. Volume 10482. LNCS. Springer, 2017, pages 417–434. DOI: 10.1007/978-3-319-68167-2_27 (cited on page 168).

[BEGW03] S. Ben-David, C. Eisner, D. Geist, and Y. Wolfsthal. "*Model Checking at IBM.*" *Formal Methods Syst. Des.* 22.2 (2003), pages 101–108. DOI: 10.1023/A:1022905120346 (cited on page 1).

[Bel57] R. Bellman. "*A Markovian Decision Process.*" *Journal of Mathematics and Mechanics* 6 (4 1957), pages 679–684 (cited on pages 116, 146).

[BEN22] M. Berkelaar, K. Eikland, and P. Notebaert. "*lp_solve 5.5, Open source (Mixed-Integer) Linear Programming system.*" http://lpsolve.sourceforge.net/5.5/. 2022 (cited on page 293).

[Ber05] D. P. Bertsekas. "*Dynamic programming and optimal control.*" 3rd. Volume I. Athena Scientific, 2005. URL: http://athenasc.com/dpbook.html (cited on page 148).

[BF19] Y. Butkova and G. Fox. "*Optimal Time-Bounded Reachability Analysis for Concurrent Systems.*" *TACAS (2).* Volume 11428. LNCS. Springer, 2019, pages 191–208. DOI: 10.1007/978-3-030-17465-1_11 (cited on page 61).

[BFHW+15] B. Braitling, L. M. F. Fioriti, H. Hatefi, R. Wimmer, B. Becker, and H. Hermanns. "*Abstraction-Based Computation of Reward Measures for Markov Automata.*" *VMCAI.* Volume 8931. LNCS. Springer, 2015, pages 172–189. DOI: 10.1007/978-3-662-46081-8_10 (cited on page 61).

[BFRR17] V. Bruyère, E. Filiot, M. Randour, and J.-F. Raskin. "*Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games.*" *Inf. Comput.* 254 (2017), pages 259–295. DOI: 10.1016/j.ic.2016.10.011 (cited on pages 12, 206, 277, 306).

[BGMR18] P. Bouyer, M. González, N. Markey, and M. Randour. "*Multi-weighted Markov Decision Processes with Reachability Objectives.*" *GandALF.* Volume 277. EPTCS. 2018, pages 250–264. DOI: 10.4204/EPTCS.277.18 (cited on pages 12, 206, 306).

[BGR20] R. Berthon, S. Guha, and J.-F. Raskin. "*Mixing Probabilistic and non-Probabilistic Objectives in Markov Decision Processes.*" *LICS.* ACM, 2020, pages 195–208. DOI: 10.1145/3373718.3394805 (cited on pages 12, 306).

[BHH21] Y. Butkova, A. Hartmanns, and H. Hermanns. "*A Modest Approach to Markov Automata.*" *ACM Trans. Model. Comput. Simul.* 31.3 (2021), 14:1–14:34. DOI: 10.1145/3449355 (cited on page 154).

[BHHK03] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. "*Model-Checking Algorithms for Continuous-Time Markov Chains.*" *IEEE Trans. Software Eng.* 29.6 (2003), pages 524–541. DOI: 10.1109/TSE.2003.1205180 (cited on page 4).

[BHHK15] Y. Butkova, H. Hatefi, H. Hermanns, and J. Krcál. "*Optimal Continuous Time Markov Decisions*." *ATVA*. Volume 9364. LNCS. Springer, 2015, pages 166–182. DOI: 10.1007/978-3-319-24953-7_12 (cited on page 61).

[BHK19] C. Baier, H. Hermanns, and J.-P. Katoen. "*The 10, 000 Facets of MDP Model Checking*." *Computing and Software Science*. Volume 10000. LNCS. Springer, 2019, pages 420–451. DOI: 10.1007/978-3-319-91908-9_21 (cited on page 4).

[BK08] C. Baier and J.-P. Katoen. "*Principles of model checking*." MIT Press, 2008 (cited on pages 1, 15, 41, 51, 52, 105, 113, 115, 236, 249).

[BKKN+14] T. Brázdil, S. Kiefer, A. Kucera, P. Novotný, and J.-P. Katoen. "*Zero-reachability in probabilistic multi-counter automata*." *CSL-LICS*. ACM, 2014, 22:1–22:10. DOI: 10.1145/2603088.2603161 (cited on page 206).

[BKLP+17] C. Baier, J. Klein, L. Leuschner, D. Parker, and S. Wunderlich. "*Ensuring the Reliability of Your Model Checker: Interval Iteration for Markov Decision Processes*." *CAV (1)*. Volume 10426. LNCS. Springer, 2017, pages 160–180. DOI: 10.1007/978-3-319-63387-9_8 (cited on pages 121, 123, 135, 139, 146–149, 151, 204, 243, 244).

[BKNP+19] N. Balaji, S. Kiefer, P. Novotný, G. A. Pérez, and M. Shirmohammadi. "*On the Complexity of Value Iteration*." *ICALP*. Volume 132. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 102:1–102:15. DOI: 10.4230/LIPIcs.ICALP.2019.102 (cited on page 146).

[BKTW15] N. Basset, M. Z. Kwiatkowska, U. Topcu, and C. Wiltsche. "*Strategy Synthesis for Stochastic Games with Multiple Long-Run Objectives*." *TACAS*. Volume 9035. LNCS. Springer, 2015, pages 256–271. DOI: 10.1007/978-3-662-46681-0_22 (cited on pages 12, 168, 283).

[BKW18] N. Basset, M. Z. Kwiatkowska, and C. Wiltsche. "*Compositional strategy synthesis for stochastic games with multiple objectives*." *Inf. Comput.* 261 (2018), pages 536–587. DOI: 10.1016/j.ic.2017.09.010 (cited on pages 12, 168, 283, 306).

[BL17] D. Beyer and T. Lemberger. "*Software Verification: Testing vs. Model Checking - A Comparative Evaluation of the State of the Art*." *Haifa Verification Conference*. Volume 10629. LNCS. Springer, 2017, pages 99–114. DOI: 10.1007/978-3-319-70389-3_7 (cited on page 1).

[BLR11] T. Ball, V. Levin, and S. K. Rajamani. "*A decade of software model checking with SLAM*." *Commun. ACM* 54.7 (2011), pages 68–76. DOI: 10.1145/1965724.1965743 (cited on page 1).

[BLTW15] P. Bonami, A. Lodi, A. Tramontani, and S. Wiese. "*On mathematical programming with indicator constraints.*" *Math. Program.* 151.1 (2015), pages 191–223. DOI: 10.1007/s10107-015-0891-4 (cited on page 262).

[BMRL+18] P. Bouyer, N. Markey, M. Randour, K. G. Larsen, and S. Laursen. "*Average-energy games.*" *Acta Informatica* 55.2 (2018), pages 91–127. DOI: 10.1007/s00236-016-0274-1 (cited on page 169).

[BN08] L. Barrett and S. Narayanan. "*Learning all optimal policies with multiple criteria.*" *ICML*. Volume 307. ACM International Conference Proceeding Series. ACM, 2008, pages 41–47. DOI: 10.1145/1390156.1390162 (cited on pages 12, 288).

[BRR17] R. Berthon, M. Randour, and J.-F. Raskin. "*Threshold Constraints with Guarantees for Parity Objectives in Markov Decision Processes.*" *ICALP*. Volume 80. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 121:1–121:15. DOI: 10.4230/LIPIcs.ICALP.2017.121 (cited on page 277).

[BT89] D. P. Bertsekas and J. N. Tsitsiklis. "*Convergence rate and termination of asynchronous iterative algorithms.*" *ICS*. ACM, 1989, pages 461–470. DOI: 10.1145/318789.318894 (cited on page 146).

[BT91] D. P. Bertsekas and J. N. Tsitsiklis. "*An Analysis of Stochastic Shortest Path Problems.*" *Math. Oper. Res.* 16.3 (1991), pages 580–595. DOI: 10.1287/moor.16.3.580 (cited on pages 109, 113, 116).

[But20] Y. Butkova. "*Towards efficient analysis of Markov automata.*" PhD thesis. Saarland University, Saarbrücken, Germany, 2020 (cited on pages 155, 168, 281).

[BWH17] Y. Butkova, R. Wimmer, and H. Hermanns. "*Long-Run Rewards for Markov Automata.*" *TACAS (2)*. Volume 10206. LNCS. 2017, pages 188–203. DOI: 10.1007/978-3-662-54580-5_11 (cited on pages 45, 153–155, 168, 281).

[BWH18] Y. Butkova, R. Wimmer, and H. Hermanns. "*Markov Automata on Discount!*" *MMB*. Volume 10740. LNCS. Springer, 2018, pages 19–34. DOI: 10.1007/978-3-319-74947-1_2 (cited on page 51).

[CAH04] K. Chatterjee, L. de Alfaro, and T. A. Henzinger. "*Trading Memory for Randomness.*" *QEST*. IEEE Computer Society, 2004, pages 206–217. DOI: 10.1109/QEST.2004.1348035 (cited on page 277).

[CBGK08] F. Ciesinski, C. Baier, M. Größer, and J. Klein. "*Reduction Techniques for Model Checking Markov Decision Processes*." *QEST*. IEEE Computer Society, 2008, pages 45–54. DOI: 10.1109/QEST.2008.45 (cited on pages 127, 146, 187).

[CC13] A. Christman and J. Cassamano. "*Maximizing the Probability of Arriving on Time*." *ASMTA*. Volume 7984. LNCS. Springer, 2013, pages 142–157. DOI: 10.1007/978-3-642-39408-9_11 (cited on page 206).

[CD16] K. Chatterjee and L. Doyen. "*Perfect-Information Stochastic Games with Generalized Mean-Payoff Objectives*." *LICS*. ACM, 2016, pages 247–256. DOI: 10.1145/2933575.2934513 (cited on page 168).

[CE81] E. M. Clarke and E. A. Emerson. "*Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic*." *Logic of Programs*. Volume 131. LNCS. Springer, 1981, pages 52–71. DOI: 10.1007/BFb0025774 (cited on page 1).

[CGKM12] R. Calinescu, C. Ghezzi, M. Z. Kwiatkowska, and R. Mirandola. "*Self-adaptive software needs quantitative verification at runtime*." *Commun. ACM* 55.9 (2012), pages 69–77. DOI: 10.1145/2330667.2330686 (cited on page 4).

[CGZO+17] Z. Cao, H. Guo, J. Zhang, F. A. Oliehoek, and U. Fastenrath. "*Maximizing the Probability of Arriving on Time: A Practical Q-Learning Method*." *AAAI*. AAAI Press, 2017, pages 4481–4487 (cited on page 205).

[CH14] K. Chatterjee and M. Henzinger. "*Efficient and Dynamic Algorithms for Alternating Büchi Games and Maximal End-Component Decomposition*." *J. ACM* 61.3 (2014), 15:1–15:40. DOI: 10.1145/2597631 (cited on page 40).

[Cha07] K. Chatterjee. "*Markov Decision Processes with Multiple Long-Run Average Objectives*." *FSTTCS*. Volume 4855. LNCS. Springer, 2007, pages 473–484. DOI: 10.1007/978-3-540-77050-3_39 (cited on pages 12, 168).

[CJR22] M. Camilli, A. Janes, and B. Russo. "*Automated test-based learning and verification of performance models for microservices systems*." *J. Syst. Softw.* 187 (2022), page 111225. DOI: 10.1016/j.jss.2022.111225 (cited on page 4).

[CKK17] K. Chatterjee, Z. Kretínská, and J. Křetínský. "*Unifying Two Views on Multiple Mean-Payoff Objectives in Markov Decision Processes*." *Log. Methods Comput. Sci.* 13.2 (2017). DOI: 10.23638/LMCS-13(2:15)2017 (cited on pages 12, 168).

[CKPS11] T. Chen, M. Z. Kwiatkowska, D. Parker, and A. Simaitis. "*Verifying Team Formation Protocols with Probabilistic Model Checking*." *CLIMA*. Volume 6814. LNCS. Springer, 2011, pages 190–207. DOI: 10.1007/978-3-642-22359-4_14 (cited on page 288).

[CLLV+17] J. Cámara, R. de Lemos, N. Laranjeiro, R. Ventura, and M. Vieira. "*Robustness-Driven Resilience Evaluation of Self-Adaptive Software Systems*." *IEEE Trans. Dependable Secur. Comput.* 14.1 (2017), pages 50–64. DOI: 10.1109/TDSC.2015.2429128 (cited on page 4).

[CLRS22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. "*Introduction to Algorithms, 4th Edition*." MIT Press, 2022. URL: https://mitpress.mit.edu/books/introduction-algorithms-fourth-edition (cited on page 233).

[CMH06] K. Chatterjee, R. Majumdar, and T. A. Henzinger. "*Markov Decision Processes with Multiple Objectives*." *STACS*. Volume 3884. LNCS. Springer, 2006, pages 325–336. DOI: 10.1007/11672142_26 (cited on pages 5, 12, 233, 234, 277, 306).

[CY88] C. Courcoubetis and M. Yannakakis. "*Verifying Temporal Properties of Finite-State Probabilistic Programs*." *FOCS*. IEEE Computer Society, 1988, pages 338–345. DOI: 10.1109/SFCS.1988.21950 (cited on page 4).

[DH13] Y. Deng and M. Hennessy. "*On the semantics of Markov automata*." *Inf. Comput.* 222 (2013), pages 139–168. DOI: 10.1016/j.ic.2012.10.010 (cited on pages 30, 33).

[Dij72] E. W. Dijkstra. "*The Humble Programmer*." *Commun. ACM* 15.10 (1972), pages 859–866. DOI: 10.1145/355604.361591 (cited on page 1).

[DMWG11] P. Dai, Mausam, D. S. Weld, and J. Goldsmith. "*Topological Value Iteration Algorithms*." *J. Artif. Intell. Res.* 42 (2011), pages 181–209 (cited on pages 127, 128, 187).

[EG16] C. von Essen and D. Giannakopoulou. "*Probabilistic verification and synthesis of the next generation airborne collision avoidance system*." *Int. J. Softw. Tools Technol. Transf.* 18.2 (2016), pages 227–243. DOI: 10.1007/s10009-015-0388-8 (cited on page 277).

[EHKZ13] C. Eisentraut, H. Hermanns, J.-P. Katoen, and L. Zhang. "*A Semantics for Every GSPN*." *Petri Nets*. Volume 7927. LNCS. Springer, 2013, pages 90–109. DOI: 10.1007/978-3-642-38697-8_6 (cited on page 4).

[EHZ10] C. Eisentraut, H. Hermanns, and L. Zhang. "*On Probabilistic Automata in Continuous Time*." *LICS*. IEEE Computer Society, 2010, pages 342–351. DOI: 10.1109/LICS.2010.41 (cited on pages 2, 30, 32).

[EJ11] C. von Essen and B. Jobstmann. "*Synthesizing Systems with Optimal Average-Case Behavior for Ratio Objectives*." *iWIGP*. Volume 50. EPTCS. 2011, pages 17–32. DOI: 10.4204/EPTCS.50.2 (cited on page 168).

[EKVY08] K. Etessami, M. Z. Kwiatkowska, M. Y. Vardi, and M. Yannakakis. "*Multi-Objective Model Checking of Markov Decision Processes*." *Log. Methods Comput. Sci.* 4.4 (2008). DOI: 10.2168/LMCS-4(4:8)2008 (cited on pages 5, 7, 12, 70, 78, 232, 274, 277, 306, 307).

[FFSS11] D. Figueira, S. Figueira, S. Schmitz, and P. Schnoebelen. "*Ackermannian and Primitive-Recursive Bounds with Dickson's Lemma*." *LICS*. IEEE Computer Society, 2011, pages 269–278. DOI: 10.1109/LICS.2011.39 (cited on page 212).

[FHLS+22] C. Fu, E. M. Hahn, Y. Li, S. Schewe, M. Sun, A. Turrini, and L. Zhang. "*EPMC Gets Knowledge in Multi-agent Systems*." *VMCAI*. Volume 13182. LNCS. Springer, 2022, pages 93–107. DOI: 10.1007/978-3-030-94583-1_5 (cited on pages 4, 146, 282).

[Fix08] L. Fix. "*Fifteen Years of Formal Property Verification in Intel*." *25 Years of Model Checking*. Volume 5000. LNCS. Springer, 2008, pages 139–144. DOI: 10.1007/978-3-540-69850-0_8 (cited on page 1).

[FKNP+11] V. Forejt, M. Z. Kwiatkowska, G. Norman, D. Parker, and H. Qu. "*Quantitative Multi-objective Verification for Probabilistic Systems*." *TACAS*. Volume 6605. LNCS. Springer, 2011, pages 112–127. DOI: 10.1007/978-3-642-19835-9_11 (cited on pages 7, 12, 51, 63, 232, 237, 263, 277, 283, 288, 307).

[FKNP11] V. Forejt, M. Z. Kwiatkowska, G. Norman, and D. Parker. "*Automated Verification Techniques for Probabilistic Systems*." *SFM*. Volume 6659. LNCS. Springer, 2011, pages 53–113. DOI: 10.1007/978-3-642-21455-4_3 (cited on pages 4, 51, 52, 78, 122).

[FKP12] V. Forejt, M. Z. Kwiatkowska, and D. Parker. "*Pareto Curves for Probabilistic Model Checking*." *ATVA*. Volume 7561. LNCS. Springer, 2012, pages 317–332. DOI: 10.1007/978-3-642-33386-6_25 (cited on pages 7, 9, 12, 63, 70, 78, 81, 87, 88, 91, 99, 107, 205, 269, 274, 279, 281–283, 288).

[Fur80] N. Furukawa. "*Characterization of Optimal Policies in Vector-Valued Markovian Decision Processes*." *Math. Oper. Res.* 5.2 (1980), pages 271–279. DOI: 10.1287/moor.5.2.271 (cited on page 12).

[FWHT15] L. Feng, C. Wiltsche, L. R. Humphrey, and U. Topcu. "*Controller synthesis for autonomous systems interacting with human operators.*" *ICCPS*. ACM, 2015, pages 70–79. DOI: 10.1145/2735960.2735973 (cited on pages 4, 13, 288).

[GABC+20] G. Gamrath, D. Anderson, K. Bestuzheva, W.-K. Chen, L. Eifler, M. Gasse, P. Gemander, A. Gleixner, L. Gottwald, K. Halbig, G. Hendel, C. Hojny, T. Koch, P. Le Bodic, S. J. Maher, F. Matter, M. Miltenberger, E. Mühmer, B. Müller, M. E. Pfetsch, F. Schlösser, F. Serrano, Y. Shinano, C. Tawfik, S. Vigerske, F. Wegscheider, D. Weninger, and J. Witzig. "*The SCIP Optimization Suite 7.0.*" Technical Report. Optimization Online, 2020. URL: http://www.optimization-online.org/DB_HTML/2020/03/7705.html (cited on page 28).

[GB93] V. Gullapalli and A. G. Barto. "*Convergence of Indirect Adaptive Asynchronous Value Iteration Algorithms.*" *NIPS*. Morgan Kaufmann, 1993, pages 695–702 (cited on page 146).

[GCCA+21] S. Gerasimou, J. Cámara, R. Calinescu, N. Alasmari, F. Alhwikem, and X. Fang. "*Evolutionary-Guided Synthesis of Verified Pareto-Optimal MDP Policies.*" *ASE*. IEEE, 2021, pages 842–853. DOI: 10.1109/ASE51524.2021.9678727 (cited on pages 13, 306).

[GHHK+14] D. Guck, H. Hatefi, H. Hermanns, J.-P. Katoen, and M. Timmer. "*Analysis of Timed and Long-Run Objectives for Markov Automata.*" *Log. Methods Comput. Sci.* 10.3 (2014). DOI: 10.2168/LMCS-10(3:17)2014 (cited on pages 41, 45, 52, 55, 61, 153, 154, 157, 168).

[GHNR14] A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani. "*Probabilistic programming.*" *FOSE*. ACM, 2014, pages 167–181. DOI: 10.1145/2593882.2593900 (cited on page 4).

[Gir14] S. Giro. "*Optimal schedulers vs optimal bases: An approach for efficient exact solving of Markov decision processes.*" *Theor. Comput. Sci.* 538 (2014), pages 70–83. DOI: 10.1016/j.tcs.2013.08.020 (cited on page 150).

[GJ+10] G. Guennebaud, B. Jacob, et al. "*Eigen v3.*" http://eigen.tuxfamily.org. 2010 (cited on pages 150, 280, 284).

[GNU20] GNU Project. "*GNU Linear Programming Kit, Version 5.0.*" http://www.gnu.org/software/glpk/glpk.html. 2020 (cited on pages 28, 150).

[GTHR+14] D. Guck, M. Timmer, H. Hatefi, E. Ruijters, and M. Stoelinga. "*Modelling and Analysis of Markov Reward Automata.*" *ATVA*. Volume 8837. LNCS. Springer, 2014, pages 168–184. DOI: 10.1007/978-3-319-11936-6_13 (cited on pages 32, 52, 61, 155, 168).

[Gur22] Gurobi Optimization, LLC. "*Gurobi Optimizer Reference Manual*." http://www.gurobi.com. 2022 (cited on pages 28, 262, 281, 293).

[GZ18] M. Guo and M. M. Zavlanos. "*Probabilistic Motion Planning Under Temporal Tasks and Soft Constraints*." *IEEE Trans. Autom. Control.* 63.12 (2018), pages 4051–4066. DOI: 10.1109/TAC.2018.2799561 (cited on page 168).

[Han17] E. A. Hansen. "*Error bounds for stochastic shortest path problems*." *Math. Methods Oper. Res.* 86.1 (2017), pages 1–27. DOI: 10.1007/s00186-017-0581-5 (cited on pages 146, 148, 306).

[Har22] A. Hartmanns. "*Correct Probabilistic Model Checking with Floating-Point Arithmetic*." *TACAS (2)*. Volume 13244. LNCS. Springer, 2022, pages 41–59. DOI: 10.1007/978-3-030-99527-0_3 (cited on pages 147, 306).

[Hen83] M. I. Henig. "*Vector-Valued Dynamic Programming*." *SIAM Journal on Control and Optimization* 21.3 (1983), pages 490–499. DOI: 10.1137/0321030 (cited on page 12).

[HH12] H. Hatefi and H. Hermanns. "*Model Checking Algorithms for Markov Automata*." *Electron. Commun. Eur. Assoc. Softw. Sci. Technol.* 53 (2012). DOI: 10.14279/tuj.eceasst.53.783 (cited on pages 45, 61, 154, 288).

[HH14] A. Hartmanns and H. Hermanns. "*The Modest Toolset: An Integrated Environment for Quantitative Modelling and Verification*." *TACAS*. Volume 8413. LNCS. Springer, 2014, pages 593–598. DOI: 10.1007/978-3-642-54862-8_51 (cited on pages 4, 146, 284).

[HH15] A. Hartmanns and H. Hermanns. "*Explicit Model Checking of Very Large MDP Using Partitioning and Secondary Storage*." *ATVA*. Volume 9364. LNCS. Springer, 2015, pages 131–147. DOI: 10.1007/978-3-319-24953-7_10 (cited on page 146).

[HH16] E. M. Hahn and A. Hartmanns. "*A Comparison of Time- and Reward-Bounded Probabilistic Model Checking Techniques*." *SETTA*. Volume 9984. LNCS. 2016, pages 85–100. DOI: 10.1007/978-3-319-47677-3_6 (cited on pages 9, 130, 171, 187, 200, 204, 205).

[HHHK13] E. M. Hahn, A. Hartmanns, H. Hermanns, and J.-P. Katoen. "*A compositional modelling and analysis framework for stochastic hybrid systems*." *Formal Methods Syst. Des.* 43.2 (2013), pages 191–232. DOI: 10.1007/s10703-012-0167-z (cited on page 4).

[HHHL+19] E. M. Hahn, V. Hashemi, H. Hermanns, M. Lahijanian, and A. Turrini. "*Interval Markov Decision Processes with Multiple Objectives: From Robust Strategies to Pareto Curves*." *ACM Trans. Model. Comput. Simul.* 29.4 (2019), 27:1–27:31. DOI: 10.1145/3309683 (cited on page 12).

[HHK00] B. R. Haverkort, H. Hermanns, and J. Katoen. "*On the Use of Model Checking Techniques for Dependability Evaluation*." *SRDS*. IEEE Computer Society, 2000, pages 228–237. DOI: 10.1109/RELDI.2000.885410 (cited on page 288).

[HJ94] H. Hansson and B. Jonsson. "*A Logic for Reasoning about Time and Reliability*." *Formal Aspects Comput.* 6.5 (1994), pages 512–535. DOI: 10.1007/BF01211866 (cited on page 4).

[HK15] C. Haase and S. Kiefer. "*The Odds of Staying on Budget*." *ICALP (2)*. Volume 9135. LNCS. Springer, 2015, pages 234–246. DOI: 10.1007/978-3-662-47666-6_19 (cited on pages 203, 205, 229).

[HK20] A. Hartmanns and B. L. Kaminski. "*Optimistic Value Iteration*." *CAV (2)*. Volume 12225. LNCS. Springer, 2020, pages 488–511. DOI: 10.1007/978-3-030-53291-8_26 (cited on pages 146, 147, 152).

[HKL17] C. Haase, S. Kiefer, and M. Lohrey. "*Computing quantiles in Markov chains with multi-dimensional costs*." *LICS*. IEEE Computer Society, 2017, pages 1–12. DOI: 10.1109/LICS.2017.8005090 (cited on pages 206, 207, 229).

[HM18] S. Haddad and B. Monmege. "*Interval iteration algorithm for MDPs and IMDPs*." *Theor. Comput. Sci.* 735 (2018), pages 111–131. DOI: 10.1016/j.tcs.2016.12.003 (cited on pages 41, 117, 146).

[How60] R. A. Howard. "*Dynamic programming and Markov processes*." MIT Press, 1960 (cited on pages 32, 149).

[HWBF+17] H. Hatefi, R. Wimmer, B. Braitling, L. M. F. Fioriti, B. Becker, and H. Hermanns. "*Cost vs. time in stochastic games and Markov automata*." *Formal Aspects Comput.* 29.4 (2017), pages 629–649. DOI: 10.1007/s00165-016-0411-1 (cited on page 61).

[HYV14] P. Hou, W. Yeoh, and P. Varakantham. "*Revisiting Risk-Sensitive MDPs: New Algorithms and Results*." *ICAPS*. AAAI, 2014 (cited on page 205).

[Kat16] J.-P. Katoen. "*The Probabilistic Model Checking Landscape*." *LICS*. ACM, 2016, pages 31–45. DOI: 10.1145/2933575.2934574 (cited on page 4).

[KBCD+18] J. Klein, C. Baier, P. Chrszon, M. Daum, C. Dubslaff, S. Klüppelholz, S. Märcker, and D. Müller. "*Advances in probabilistic model checking with PRISM: variable reordering, quantiles and weak deterministic Büchi automata*." *Int. J. Softw. Tools Technol. Transf.* 20.2 (2018), pages 179–194. DOI: 10.1007/s10009-017-0456-3 (cited on pages 205, 207, 229).

[KHK21] M. H. Khan, O. Hassan, and S. Khan. "*Accelerating SpMV Multiplication in Probabilistic Model Checkers Using GPUs*." *ICTAC*. Volume 12819. LNCS. Springer, 2021, pages 86–104. DOI: 10.1007/978-3-030-85315-0_6 (cited on page 146).

[KKKW18] E. Kelmendi, J. Krämer, J. Křetínský, and M. Weininger. "*Value Iteration for Simple Stochastic Games: Stopping Criterion and Learning Algorithm*." *CAV (1)*. Volume 10981. LNCS. Springer, 2018, pages 623–642. DOI: 10.1007/978-3-319-96145-3_36 (cited on page 147).

[KM17] J. Křetínský and T. Meggendorfer. "*Efficient Strategy Iteration for Mean Payoff in Markov Decision Processes*." *ATVA*. Volume 10482. LNCS. Springer, 2017, pages 380–399. DOI: 10.1007/978-3-319-68167-2_25 (cited on pages 155, 168, 288, 306).

[KNP08] M. Z. Kwiatkowska, G. Norman, and D. Parker. "*Using probabilistic model checking in systems biology*." *SIGMETRICS Perform. Evaluation Rev.* 35.4 (2008), pages 14–21. DOI: 10.1145/1364644.1364651 (cited on page 4).

[KNP11] M. Z. Kwiatkowska, G. Norman, and D. Parker. "*PRISM 4.0: Verification of Probabilistic Real-Time Systems*." *CAV*. Volume 6806. LNCS. Springer, 2011, pages 585–591. DOI: 10.1007/978-3-642-22110-1_47 (cited on pages 4, 12, 146, 168, 280, 283).

[KNP12a] M. Z. Kwiatkowska, G. Norman, and D. Parker. "*Probabilistic verification of Herman's self-stabilisation algorithm*." *Formal Aspects Comput.* 24.4-6 (2012), pages 661–670. DOI: 10.1007/s00165-012-0227-6 (cited on page 4).

[KNP12b] M. Z. Kwiatkowska, G. Norman, and D. Parker. "*The PRISM Benchmark Suite*." *QEST*. IEEE Computer Society, 2012, pages 203–204. DOI: 10.1109/QEST.2012.14 (cited on page 288).

[KNPS20] M. Kwiatkowska, G. Norman, D. Parker, and G. Santos. "*PRISM-games 3.0: Stochastic Game Verification with Concurrency, Equilibria and Time*." *CAV (2)*. Volume 12225. LNCS. Springer, 2020, pages 475–487. DOI: 10.1007/978-3-030-53291-8_25 (cited on pages 169, 283).

[KNPV09] M. Z. Kwiatkowska, G. Norman, D. Parker, and M. G. Vigliotti. "*Probabilistic Mobile Ambients*." *Theor. Comput. Sci.* 410.12-13 (2009), pages 1272–1303. DOI: 10.1016/j.tcs.2008.12.058 (cited on page 288).

[KNS02] M. Z. Kwiatkowska, G. Norman, and J. Sproston. "*Probabilistic Model Checking of the IEEE 802.11 Wireless Local Area Network Protocol*." *PAPM-PROBMIV*. Volume 2399. LNCS. Springer, 2002, pages 169–187. DOI: 10.1007/3-540-45605-8_11 (cited on page 288).

[KPC12] A. Komuravelli, C. S. Pasareanu, and E. M. Clarke. "*Assume-Guarantee Abstraction Refinement for Probabilistic Systems*." *CAV*. Volume 7358. LNCS. Springer, 2012, pages 310–326. DOI: 10.1007/978-3-642-31424-7_25 (cited on page 288).

[KPW18] M. Kwiatkowska, D. Parker, and C. Wiltsche. "*PRISM-games: verification and strategy synthesis for stochastic multi-player games with multiple objectives*." *Int. J. Softw. Tools Technol. Transf.* 20.2 (2018), pages 195–210. DOI: 10.1007/s10009-017-0476-z (cited on pages 169, 283).

[Kře21] J. Křetínský. "*LTL-Constrained Steady-State Policy Synthesis*." *IJCAI*. ijcai.org, 2021, pages 4104–4111. DOI: 10.24963/ijcai.2021/565 (cited on page 168).

[KS76] J. G. Kemeny and J. L. Snell. "*Finite Markov Chains*." Springer undergraduate texts in mathematics. Sppringer, 1976. URL: https://link.springer.com/book/9780387901923 (cited on page 239).

[KSBD15] D. Krähmann, J. Schubert, C. Baier, and C. Dubslaff. "*Ratio and Weight Quantiles*." *MFCS (1)*. Volume 9234. LNCS. Springer, 2015, pages 344–356. DOI: 10.1007/978-3-662-48057-1_27 (cited on page 229).

[Kwi16] M. Z. Kwiatkowska. "*Model Checking and Strategy Synthesis for Stochastic Games: From Theory to Practice*." *ICALP*. Volume 55. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 4:1–4:18. DOI: 10.4230/LIPIcs.ICALP.2016.4 (cited on page 147).

[LBM12] D. J. Lizotte, M. Bowling, and S. A. Murphy. "*Linear fitted-Q iteration with multiple reward functions*." *J. Mach. Learn. Res.* 13 (2012), pages 3253–3295. DOI: 10.5555/2503308.2503346 (cited on page 232).

[Lev17] N. G. Leveson. "*The Therac-25: 30 Years Later*." *Computer* 50.11 (2017), pages 8–11. DOI: 10.1109/MC.2017.4041349 (cited on page 1).

[Lio96] J.-L. Lions. "*ARIANE 5 Flight 501 Failure: Report by the Enquiry Board*." http://esamultimedia.esa.int/docs/esa-x-1819eng.pdf. 1996 (cited on page 1).

[LMT04] R. Lanotte, A. Maggiolo-Schettini, and A. Troina. "*Automatic Analysis of a Non-Repudiation Protocol*." *QAPL*. Volume 112. Electronic Notes in Theoretical Computer Science. Elsevier, 2004, pages 113–129. DOI: 10.1016/j.entcs.2004.01.020 (cited on page 4).

[LPH17] B. Lacerda, D. Parker, and N. Hawes. "*Multi-Objective Policy Generation for Mobile Robots under Probabilistic Time-Bounded Guarantees*." *ICAPS*. AAAI Press, 2017, pages 504–512 (cited on pages 13, 288).

[LR81] D. Lehmann and M. O. Rabin. "*On the Advantages of Free Choice: A Symmetric and Fully Distributed Solution to the Dining Philosophers Problem*." *POPL*. ACM Press, 1981, pages 133–138. DOI: 10.1145/567532.567547 (cited on page 288).

[LTHS+22] M. Li, A. Turrini, E. M. Hahn, Z. She, and L. Zhang. "*Probabilistic Preference Planning Problem for Markov Decision Processes*." *IEEE Trans. Software Eng.* 48.5 (2022), pages 1545–1559. DOI: 10.1109/TSE.2020.3024215 (cited on pages 13, 282).

[LX22] J. Lu and M. Xu. "*Bisection Value Iteration*." *APSEC*. IEEE, 2022, pages 109–118. DOI: 10.1109/APSEC57359.2022.00023 (cited on pages 146, 149, 152, 306).

[Mat91] V. K. Mathur. "*How Well Do We Know Pareto Optimality?*" *The Journal of Economic Education* 22.2 (1991), pages 172–178. DOI: 10.2307/1182422 (cited on page 5).

[MB08] L. M. de Moura and N. S. Bjørner. "*Z3: An Efficient SMT Solver*." *TACAS*. Volume 4963. LNCS. Springer, 2008, pages 337–340. DOI: 10.1007/978-3-540-78800-3_24 (cited on page 280).

[MBCS+20] U. Mathur, M. S. Bauer, R. Chadha, A. P. Sistla, and M. Viswanathan. "*Exact quantitative probabilistic model checking through rational search*." *Formal Methods Syst. Des.* 56.1 (2020), pages 90–126. DOI: 10.1007/s10703-020-00348-y (cited on pages 146, 149).

[MKI20] M. Mohagheghi, J. Karimpour, and A. Isazadeh. "*Prioritizing Methods to Accelerate Probabilistic Model Checking of Discrete-Time Markov Models*." *Comput. J.* 63.1 (2020), pages 105–122. DOI: 10.1093/comjnl/bxz001 (cited on page 146).

[MLG05] H. B. McMahan, M. Likhachev, and G. J. Gordon. "*Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees*." *ICML*. Volume 119. ACM International Conference Proceeding Series. ACM, 2005, pages 569–576. DOI: 10.1145/1102351.1102423 (cited on page 146).

[MR22] J. C. A. Main and M. Randour. "*Different Strokes in Randomised Strategies: Revisiting Kuhn's Theorem Under Finite-Memory Assumptions*." *CONCUR*. Volume 243. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 22:1–22:18. DOI: 10.4230/LIPIcs.CONCUR.2022.22 (cited on pages 35, 277).

[MS20] M. Mohagheghi and K. Salehi. "*Accelerating Interval Iteration for Expected Rewards in Markov Decision Processes*." *ICSOFT*. ScitePress, 2020, pages 39–50. DOI: 10.5220/0009833700390050 (cited on page 148).

[Neu10] M. R. Neuhäußer. "*Model checking nondeterministic and randomly timed systems*." PhD thesis. RWTH Aachen University, 2010. URL: https://publications.rwth-aachen.de/record/51599 (cited on pages 28, 35).

[Nor98] J. R. Norris. "*Markov chains*." Cambridge series in statistical and probabilistic mathematics. Cambridge University Press, 1998 (cited on pages 47, 55).

[NPKS05] G. Norman, D. Parker, M. Z. Kwiatkowska, and S. K. Shukla. "*Evaluating the reliability of NAND multiplexing with PRISM*." *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 24.10 (2005), pages 1629–1637. DOI: 10.1109/TCAD.2005.852033 (cited on page 288).

[NS06] G. Norman and V. Shmatikov. "*Analysis of probabilistic contract signing*." *J. Comput. Secur.* 14.6 (2006), pages 561–589. DOI: 10.3233/jcs-2006-14604 (cited on page 4).

[Oht04] Y. Ohtsubo. "*Optimal threshold probability in undiscounted Markov decision processes with a target set*." *Appl. Math. Comput.* 149.2 (2004), pages 519–532. DOI: 10.1016/S0096-3003(03)00158-9 (cited on page 205).

[PDM17] A. D. Pia, S. S. Dey, and M. Molinaro. "*Mixed-integer quadratic programming is in NP*." *Math. Program.* 162.1-2 (2017), pages 225–240. DOI: 10.1007/s10107-016-1036-0 (cited on page 27).

[Pnu77] A. Pnueli. "*The Temporal Logic of Programs*." *FOCS*. IEEE Computer Society, 1977, pages 46–57. DOI: 10.1109/SFCS.1977.32 (cited on page 4).

[PT87] C. H. Papadimitriou and J. N. Tsitsiklis. "*The Complexity of Markov Decision Processes*." *Math. Oper. Res.* 12.3 (1987), pages 441–450. DOI: 10.1287/moor.12.3.441 (cited on pages 113, 115).

[PTHH20] K. Phalakarn, T. Takisaka, T. Haas, and I. Hasuo. "*Widest Paths and Global Propagation in Bounded Value Iteration for Stochastic Games*." *CAV (2)*. Volume 12225. LNCS. Springer, 2020, pages 349–371. DOI: 10.1007/978-3-030-53291-8_19 (cited on page 147).

[Put94] M. L. Puterman. "*Markov Decision Processes: Discrete Stochastic Dynamic Programming*." Wiley Series in Probability and Statistics. Wiley, 1994. DOI: 10.1002/9780470316887 (cited on pages 2, 9, 12, 45, 51, 54, 55, 99, 115, 125, 143, 153–155, 168, 237, 278).

[PW10] P. Perny and P. Weng. "*On Finding Compromise Solutions in Multiobjective Markov Decision Processes*." *ECAI*. Volume 215. Frontiers in Artificial Intelligence and Applications. IOS Press, 2010, pages 969–970. DOI: 10.3233/978-1-60750-606-5-969 (cited on pages 12, 277).

[QS82] J.-P. Queille and J. Sifakis. "*Specification and verification of concurrent systems in CESAR*." *Symposium on Programming*. Volume 137. LNCS. Springer, 1982, pages 337–351. DOI: 10.1007/3-540-11494-7_22 (cited on page 1).

[QWP99] Q. Qiu, Q. Wu, and M. Pedram. "*Stochastic modeling of a power-managed system: construction and optimization*." *ISLPED*. ACM, 1999, pages 194–199. DOI: 10.1145/313817.313923 (cited on page 288).

[Rab82] M. O. Rabin. "*N-Process Mutual Exclusion with Bounded Waiting by 4 Log_2 N-Valued Shared Variable*." *J. Comput. Syst. Sci.* 25.1 (1982), pages 66–75. DOI: 10.1016/0022-0000(82)90010-1 (cited on page 288).

[RDH11] G. Rennen, E. R. van Dam, and D. den Hertog. "*Enhancement of Sandwich Algorithms for Approximating Higher-Dimensional Convex Pareto Sets*." *INFORMS J. Comput.* 23.4 (2011), pages 493–517. DOI: 10.1287/ijoc.1100.0419 (cited on pages 7, 12, 81, 269).

[RRS15] M. Randour, J.-F. Raskin, and O. Sankur. "*Variations on the Stochastic Shortest Path Problem*." *VMCAI*. Volume 8931. LNCS. Springer, 2015, pages 1–18. DOI: 10.1007/978-3-662-46081-8_1 (cited on page 277).

[RRS17] M. Randour, J.-F. Raskin, and O. Sankur. "*Percentile queries in multi-dimensional Markov decision processes*." *Formal Methods Syst. Des.* 50.2-3 (2017), pages 207–248. DOI: 10.1007/s10703-016-0262-7 (cited on pages 7, 12, 60, 63, 168, 171, 174, 205, 206, 232, 233, 274, 277, 306).

[RVWD13] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. "*A Survey of Multi-Objective Sequential Decision-Making*." *J. Artif. Intell. Res.* 48 (2013), pages 67–113. DOI: 10.1613/jair.3987 (cited on page 13).

[SAC93] R. S. Solanki, P. A. Appino, and J. L. Cohon. "*Approximating the non-inferior set in multiobjective linear programming problems*." *European Journal of Operational Research* 68.3 (1993), pages 356–373. DOI: 10.1016/0377-2217(93)90192-P (cited on pages 7, 12, 81, 269).

[SBHH17] D. Scheftelowitsch, P. Buchholz, V. Hashemi, and H. Hermanns. "*Multi-Objective Approaches to Markov Decision Processes with Uncertain Transition Parameters*." *VALUETOOLS*. ACM, 2017, pages 44–51. DOI: 10.1145/3150928.3150945 (cited on page 12).

[Sch99] A. Schrijver. "*Theory of Linear and Integer Programming*." Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999 (cited on pages 27, 28).

[SDC99] K. J. Sullivan, J. B. Dugan, and D. Coppit. "*The Galileo Fault Tree Analysis Tool*." *FTCS*. IEEE Computer Society, 1999, pages 232–235. DOI: 10.1109/FTCS.1999.781056 (cited on page 4).

[SDKB+19] M. U. Sardar, C. Dubslaff, S. Klüppelholz, C. Baier, and A. Kumar. "*Performance Evaluation of Thermal-Constrained Scheduling Strategies in Multi-core Systems*." *EPEW*. Volume 12039. LNCS. Springer, 2019, pages 133–147. DOI: 10.1007/978-3-030-44411-2_9 (cited on page 229).

[Sol15] V. Soltan. "*Lectures on Convex Sets*." World Scientific, 2015. DOI: 10.1142/9508 (cited on pages 15, 16, 21, 24–26, 75).

[SS86] Y. Saad and M. H. Schultz. "*GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems*." *SIAM Journal on Scientific and Statistical Computing* 7.3 (1986), pages 856–869. DOI: 10.1137/0907058 (cited on page 284).

[ST04] D. A. Spielman and S.-H. Teng. "*Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time*." *J. ACM* 51.3 (2004), pages 385–463. DOI: 10.1145/990308.990310 (cited on page 28).

[Stu21] I. I. C. O. Studio. "*CPLEX User's Manual, Version 12 Release 8*." https://www.ibm.com/docs/en/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/usrcplex.pdf. 2021 (cited on pages 28, 262).

[SV99] M. Stoelinga and F. W. Vaandrager. "*Root Contention in IEEE 1394*." *ARTS*. Volume 1601. LNCS. Springer, 1999, pages 53–74. DOI: 10.1007/3-540-48778-6_4 (cited on page 288).

[Tar72] R. E. Tarjan. "*Depth-First Search and Linear Graph Algorithms*." *SIAM J. Comput.* 1.2 (1972), pages 146–160. DOI: 10.1137/0201010 (cited on pages 40, 130).

[TD17] J. Tinsley Oden and L. Demkowicz. "*Applied Functional Analysis*." CRC Press, Taylor & Francis Group, 2017. DOI: 10.1201/9781315119489 (cited on pages 16, 18).

[Tim13] M. Timmer. "*Efficient modelling, generation and analysis of Markov automata*." PhD thesis. University of Twente, Enschede, Netherlands, 2013 (cited on pages 2, 32).

[TKPS12] M. Timmer, J.-P. Katoen, J. van de Pol, and M. Stoelinga. "*Efficient Modelling and Generation of Markov Automata*." *CONCUR*. Volume 7454. LNCS. Springer, 2012, pages 364–379. DOI: 10.1007/978-3-642-32940-1_26 (cited on page 288).

[UB13] M. Ummels and C. Baier. "*Computing Quantiles in Markov Reward Models*." *FoSSaCS*. Volume 7794. LNCS. Springer, 2013, pages 353–368. DOI: 10.1007/978-3-642-37075-5_23 (cited on pages 205, 207, 221, 223, 227, 229).

[Vel19] A. Velasquez. "*Steady-State Policy Synthesis for Verifiable Control*." *IJCAI*. ijcai.org, 2019, pages 5653–5661. DOI: 10.24963/ijcai.2019/784 (cited on pages 237, 253, 277).

[WB12] A. Wijs and D. Bosnacki. "*Improving GPU Sparse Matrix-Vector Multiplication for Probabilistic Model Checking*." *SPIN*. Volume 7385. LNCS. Springer, 2012, pages 98–116. DOI: 10.1007/978-3-642-31759-0_9 (cited on page 146).

[WBBH+10] R. Wimmer, B. Braitling, B. Becker, E. M. Hahn, P. Crouzen, H. Hermanns, A. Dhama, and O. E. Theel. "*Symblicit Calculation of Long-Run Averages for Concurrent Probabilistic Systems*." *QEST*. IEEE Computer Society, 2010, pages 27–36. DOI: 10.1109/QEST.2010.12 (cited on pages 155, 168).

[WD06] J. Wu and E. H. Durfee. "*Mixed-integer linear programming for transition-independent decentralized MDPs*." *AAMAS*. ACM, 2006, pages 1058–1060. DOI: 10.1145/1160633.1160822 (cited on page 277).

[WHGZ+16] Z. Wu, E. M. Hahn, A. Günay, L. Zhang, and Y. Liu. "*GPU-Accelerated Value Iteration for the Computation of Reachability Probabilities in MDPs*." *ECAI*. Volume 285. Frontiers in Artificial Intelligence and Applications. IOS Press, 2016, pages 1726–1727. DOI: 10.3233/978-1-61499-672-9-1726 (cited on page 146).

[Whi82] D. White. "*Multi-objective infinite-horizon discounted Markov decision processes*." *Journal of Mathematical Analysis and Applications* 89.2 (1982), pages 639–647. DOI: 10.1016/0022-247X(82)90122-6 (cited on page 12).

[WJ07] M. A. Wiering and E. D. de Jong. "*Computing Optimal Stationary Policies for Multi-Objective Markov Decision Processes*." *ADPRL*. 2007, pages 158–165. DOI: 10.1109/ADPRL.2007.368183 (cited on page 277).

[WJÁK+14] R. Wimmer, N. Jansen, E. Ábrahám, J.-P. Katoen, and B. Becker. "*Minimal counterexamples for linear-time probabilistic verification*." *Theor. Comput. Sci.* 549 (2014), pages 61–100. DOI: 10.1016/j.tcs.2014.06.020 (cited on pages 253, 256, 277).

[WJVÁ+15] R. Wimmer, N. Jansen, A. Vorpahl, E. Ábrahám, J.-P. Katoen, and B. Becker. "*High-level Counterexamples for Probabilistic Automata*." *Log. Methods Comput. Sci.* 11.1 (2015). DOI: 10.2168/LMCS-11(1:15)2015 (cited on page 277).

[WKHB08] R. Wimmer, A. Kortus, M. Herbstritt, and B. Becker. "*Probabilistic Model Checking and Reliability of Results*." *DDECS*. IEEE Computer Society, 2008, pages 207–212. DOI: 10.1109/DDECS.2008.4538787 (cited on pages 117, 146).

[WT98] K. Wakuta and K. Togawa. "*Solution procedures for multi-objective Markov decision processes*." *Optimization* 43.1 (1998), pages 29–46. DOI: 10.1080/02331939808844372 (cited on page 12).

[WWJB20] L. Winterer, R. Wimmer, N. Jansen, and B. Becker. "*Strengthening Deterministic Policies for POMDPs*." *NFM*. Volume 12229. LNCS. Springer, 2020, pages 115–132. DOI: 10.1007/978-3-030-55754-6_7 (cited on pages 231, 253, 277, 278).

[YLWA05] H. L. S. Younes, M. L. Littman, D. Weissman, and J. Asmuth. "*The First Probabilistic Track of the International Planning Competition*." *J. Artif. Intell. Res.* 24 (2005), pages 851–887. DOI: 10.1613/jair.1880 (cited on page 4).

[YLY98] S. Yu, Y. Lin, and P. Yan. "*Optimization Models for the First Arrival Target Distribution Function in Discrete Time*." *J. Math. Anal. Appl* 225 (Nov. 1998). DOI: 10.1006/jmaa.1998.6015 (cited on page 205).

[Zie95] G. M. Ziegler. "*Lectures on Polytopes*." Graduate Texts in Mathematics. Springer, 1995. DOI: 10.1007/978-1-4613-8431-1 (cited on pages 16, 24).

Co-authored Publications

We list all peer-reviewed publications. Section 1.6 considers publications whose result are (fully or partly) presented in this thesis together with an obligatory statement regarding author contributions.

Cited publications

[1] T. Quatmann. "*Multi-objective model checking of Markov Automata*." Master's thesis. RWTH Aachen University, 2016. Available at https://doi.org/10.5281/zenodo.5101292 (cited on pages 14, 99, 102, 103, 279).

[2] T. Quatmann, S. Junges, and J.-P. Katoen. "*Markov Automata with Multiple Objectives*." *CAV (1)*. Volume 10426. LNCS. Springer, 2017, pages 140–159. DOI: 10.1007/978-3-319-63387-9_7 (cited on pages 14, 45, 61, 154).

[3] A. Hartmanns, S. Junges, J.-P. Katoen, and T. Quatmann. "*Multi-cost Bounded Reachability in MDP*." *TACAS (2)*. Volume 10806. LNCS. Springer, 2018, pages 320–339. DOI: 10.1007/978-3-319-89963-3_19 (cited on pages 13, 14, 171).

[4] S. Junges, N. Jansen, R. Wimmer, T. Quatmann, L. Winterer, J.-P. Katoen, and B. Becker. "*Finite-State Controllers of POMDPs using Parameter Synthesis*." *UAI*. AUAI Press, 2018, pages 519–529. URL: http://auai.org/uai2018/proceedings/papers/195.pdf (cited on page 275).

[5] T. Quatmann and J.-P. Katoen. "*Sound Value Iteration*." *CAV (1)*. Volume 10981. LNCS. Springer, 2018, pages 643–661. DOI: 10.1007/978-3-319-96145-3_37 (cited on pages 13, 99, 279).

[6] E. M. Hahn, A. Hartmanns, C. Hensel, M. Klauck, J. Klein, J. Křetínský, D. Parker, T. Quatmann, E. Ruijters, and M. Steinmetz. "*The 2019 Comparison of Tools for the Analysis of Quantitative Formal Models - (QComp 2019 Competition Report)*." *TACAS (3)*. Volume 11429. LNCS. Springer, 2019, pages 69–92. DOI: 10.1007/978-3-030-17502-3_5 (cited on pages 279, 280).

[7] A. Hartmanns, M. Klauck, D. Parker, T. Quatmann, and E. Ruijters. "*The Quantitative Verification Benchmark Set*." *TACAS (1)*. Volume 11427. LNCS. Springer, 2019, pages 344–350. DOI: 10.1007/978-3-030-17462-0_20 (cited on pages 284, 288).

[8] C. E. Budde, A. Hartmanns, M. Klauck, J. Křetínský, D. Parker, T. Quatmann, A. Turrini, and Z. Zhang. "*On Correctness, Precision, and Performance in Quantitative Verification - QComp 2020 Competition Report*." *ISoLA (4)*. Volume 12479. LNCS. Springer, 2020, pages 216–241. DOI: 10.1007/978-3-030-83723-5_15 (cited on pages 116, 280).

[9] F. Delgrange, J.-P. Katoen, T. Quatmann, and M. Randour. "*Simple Strategies in Multi-Objective MDPs.*" *TACAS (1).* Volume 12078. LNCS. Springer, 2020, pages 346–364. DOI: 10.1007/978-3-030-45190-5_19 (cited on pages 14, 231, 258, 279).

[10] A. Hartmanns, S. Junges, J.-P. Katoen, and T. Quatmann. "*Multi-cost Bounded Tradeoff Analysis in MDP.*" *J. Autom. Reason.* 64.7 (2020), pages 1483–1522. DOI: 10.1007/s10817-020-09574-9 (cited on pages 13, 171, 207, 279, 288).

[11] T. Quatmann and J.-P. Katoen. "*Multi-objective Optimization of Long-run Average and Total Rewards.*" *TACAS (1).* Volume 12651. LNCS. Springer, 2021, pages 230–249. DOI: 10.1007/978-3-030-72016-2_13 (cited on pages 14, 15, 63, 153, 279, 288).

[12] C. Hensel, S. Junges, J.-P. Katoen, T. Quatmann, and M. Volk. "*The probabilistic model checker Storm.*" *Int. J. Softw. Tools Technol. Transf.* 24.4 (2022), pages 589–610. DOI: 10.1007/s10009-021-00633-z (cited on pages 4, 146, 150, 279, 280).

[13] T. Quatmann, S. Junges, and J.-P. Katoen. "*Markov automata with multiple objectives.*" *Formal Methods Syst. Des.* 60.1 (2022), pages 33–86. DOI: 10.1007/s10703-021-00364-6 (cited on pages 14, 63, 99, 102, 103, 231, 281, 288).

[14] A. Hartmanns, S. Junges, T. Quatmann, and M. Weininger. "*A Practitioner's Guide to MDP Model Checking Algorithms.*" *TACAS (1).* Volume 13993. LNCS. Springer, 2023, pages 469–488. DOI: 10.1007/978-3-031-30823-9_24 (cited on page 279).

Further publications

T. Quatmann, N. Jansen, C. Dehnert, R. Wimmer, E. Ábrahám, J.-P. Katoen, and B. Becker. "*Counterexamples for Expected Rewards.*" *FM.* Volume 9109. LNCS. Springer, 2015, pages 435–452. DOI: 10.1007/978-3-319-19249-9_27.

T. Quatmann, C. Dehnert, N. Jansen, S. Junges, and J.-P. Katoen. "*Parameter Synthesis for Markov Models: Faster Than Ever.*" *ATVA.* Volume 9938. LNCS. 2016, pages 50–67. DOI: 10.1007/978-3-319-46520-3_4.

N. Jansen, S. Junges, J. Katoen, T. Quatmann, B. Becker, R. Wimmer, and L. Winterer. "*Correct-by-construction policies for POMDPs.*" *SNR.* ACM, 2019, pages 6–8. DOI: 10.1145/3313149.3313366.

A. Bork, S. Junges, J.-P. Katoen, and T. Quatmann. "*Verification of Indefinite-Horizon POMDPs.*" *ATVA.* Volume 12302. LNCS. Springer, 2020, pages 288–304. DOI: 10.1007/978-3-030-59152-6_16.

A. Bork, J. Katoen, and T. Quatmann. "*Under-Approximating Expected Total Rewards in POMDPs*." *TACAS (2)*. Volume 13244. LNCS. Springer, 2022, pages 22–40. DOI: 10.1007/978-3-030-99527-0_2.

Index