# Decision Algorithms for Modelling, Optimal Control and Verification of Probabilistic Systems

Dissertation
zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften (Dr.-Ing.)
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

vorgelegt von

**Vahid Hashemi**

Saarbrücken, Germany

October 2017

| | |
|---|---|
| **Dean of Faculty** | Prof. Frank-Olaf Schreyer |
| **Date of Colloquium** | 21.12.2017 |
| **Chair of the Committee** | Prof. Dr. Christoph Weidenbach |
| **Reviewers** | Prof. Dr. Holger Hermanns<br>Prof. Dr. Stefan Kiefer<br>Prof. Dr. Benoît Delahaye |
| **Academic Assistant** | Dr. Daniel Stan |

# Abstract

Markov Decision Processes (MDPs) constitute a mathematical framework for modelling systems featuring both probabilistic and nondeterministic behaviour. They are widely used to solve sequential decision making problems and applied successfully in operations research, artificial intelligence, and stochastic control theory, and have been extended conservatively to the model of probabilistic automata in the context of concurrent probabilistic systems. However, when modeling a physical system they suffer from several limitations. One of the most important is the inherent loss of precision that is introduced by measurement errors and discretization artifacts which necessarily happen due to incomplete knowledge about the system behavior. As a result, the true probability distribution for transitions is in most cases an uncertain value, determined by either external parameters or confidence intervals. Interval Markov decision processes (IMDPs) generalize classical MDPs by having interval-valued transition probabilities. They provide a powerful modelling tool for probabilistic systems with an additional variation or uncertainty that reflects the absence of precise knowledge concerning transition probabilities.

In this dissertation, we focus on decision algorithms for modelling and performance evaluation of such probabilistic systems leveraging techniques from mathematical optimization. From a modelling viewpoint, we address probabilistic bisimulations to reduce the size of the system models while preserving the logical properties they satisfy. We also discuss the key ingredients to construct systems by composing them out of smaller components running in parallel. Furthermore, we introduce a novel stochastic model, Uncertain weighted Markov Decision Processes (UwMDPs), so as to capture quantities like preferences or priorities in a nondeterministic scenario with uncertainties. This model is close to the model of IMDPs but more convenient to work with in the context of bisimulation minimization. From a performance evaluation perspective, we consider the problem of multi-objective robust strategy synthesis for IMDPs, where the aim is to find a robust strategy that guarantees the satisfaction of multiple properties at the same time in face of the transition probability uncertainty. In this respect, we discuss the computational complexity of the problem and present a value iteration-based decision algorithm to approximate the Pareto set of achievable optimal points. Moreover, we consider the problem of computing maximal/minimal reward-bounded reachability probabilities on UwMDPs, for which we present an efficient algorithm running in pseudo-polynomial time. We demonstrate the practical effectiveness of our proposed approaches by applying them to a collection of real-world case studies using several prototypical tools.

# Zusammenfassung

Markov-Entscheidungsprozesse (MEPe) bilden den Rahmen für die Modellierung von Systemen, die sowohl stochastisches als auch nichtdeterministisches Verhalten beinhalten. Diese Modellklasse hat ein breites Anwendungsfeld in der Lösung sequentieller Entscheidungsprobleme und wird erfolgreich in der Operationsforschung, der künstlichen Intelligenz und in der stochastischen Kontrolltheorie eingesetzt. Im Bereich der nebenläufigen probabilistischen Systeme wurde sie konservativ zu probabilistischen Automaten erweitert. Verwendet man MEPe jedoch zur Modellierung physikalischer Systeme so zeigt es sich, dass sie an einer Reihe von Einschränkungen leiden. Eines der schwerwiegendsten Probleme ist, dass das tatsächliche Verhalten des betrachteten Systems zumeist nicht vollständig bekannt ist. Durch Messfehler und Diskretisierungsartefakte ist ein Verlust an Genauigkeit unvermeidbar. Die tatsächlichen Übergangswahrscheinlichkeitsverteilungen des Systems sind daher in den meisten Fällen nicht exakt bekannt, sondern hängen von äußeren Faktoren ab oder können nur durch Konfidenzintervalle erfasst werden. Intervall-Markov-Entscheidungsprozesse (IMEPe) verallgemeinern klassische MEPe dadurch, dass die möglichen Übergangswahrscheinlichkeitsverteilungen durch Intervalle ausgedrückt werden können. IMEPe sind daher ein mächtiges Modellierungswerkzeug für probabilistische Systeme mit unbestimmtem Verhalten, dass sich dadurch ergibt, dass das exakte Verhalten des realen Systems nicht bekannt ist.

In dieser Doktorarbeit konzentrieren wir uns auf Entscheidungsverfahren für die Modellierung und die Auswertung der Eigenschaften solcher probabilistischer Systeme indem wir Methoden der mathematischen Optimierung einsetzen. Im Bereich der Modellierung betrachten wir probabilistische Bisimulation um die Größe des Systemmodells zu reduzieren während wir gleichzeitig die logischen Eigenschaften erhalten. Wir betrachten außerdem die Schlüsseltechniken um Modelle aus kleineren Komponenten, die parallel ablaufen, kompositionell zu generieren. Weiterhin führen wir eine neue Art von stochastischen Modellen ein, sogenannte Unsichere Gewichtete Markov-Entscheidungsprozesse (UgMEPe), um Eigenschaften wie Implementierungsentscheidungen und Benutzerprioritäten in einem nichtdeterministischen Szenario ausdrücken zu können. Dieses Modell ähnelt IMEPe, ist aber besser für die Minimierung bezüglich Bisimulation geeignet. Im Bereich der Auswertung von Modelleigenschaften betrachten wir das Problem, Strategien zu generieren, die in der Lage sind den Nichtdeterminismus so aufzulösen, dass mehrere gewünschte Eigenschaften gleichzeitig erfüllt werden können, wobei jede mögliche Auswahl von Wahrscheinlichkeitsverteilungen aus den Übergangsintervallen zu respektieren ist. Wir betrachten die Komplexitätsklasse dieses Problems und diskutieren einen auf Werte-Iteration beruhenden Algorithmus um die Pareto-Menge der erreichbaren optimalen Punkte anzunähern. Weiterhin betrachten wir das Problem, minimale und maxi-

male Erreichbarkeitswahrscheinlichkeiten zu berechnen, wenn wir eine obere Grenze für die akkumulierten Pfadkosten einhalten müssen. Für dieses Problem diskutieren wir einen effizienten Algorithmus mit pseudopolynomieller Zeit. Wir zeigen die Effizienz unserer Ansätze in der Praxis, indem wir sie prototypisch implementieren und auf eine Reihe von realistischen Fallstudien anwenden.

# Acknowledgments

Throughout my amazing journey towards shaping this dissertation, I have been quite fortunate to have invaluable encouragement, support, advice, and friendship of many remarkable individuals who I wish to acknowledge.

First and foremost, I would like to thank my exceptional doctoral advisor Holger Hermanns for all his advice and endless support. During my doctoral program, he has provided me with plenty of freedom to pursue my own research directions. On top of all, he gave me a wide variety of opportunities to interact and collaborate with several people around the world.

I am deeply grateful to Benoît Delahaye and Stefan Kiefer for serving as reviewers on the thesis committee and also for their great feedback and comments. I would also like to thank Christoph Weidenbach my thesis committee chair and also Daniel Stan for acting as a part of the committee.

I am sincerely grateful to all my colleagues with whom I have collaborated on various projects during my doctoral research: Peter Buchholz, Khaled Elbassioni, Luis María Ferrer Fioriti, Daniel Gebler, Ernst Moritz Hahn, Hassan Hatefi, Jan Krcál, Morteza Lahijanian, Dimitri Scheftelowitsch, Lei Song, K. Subramani, Andrea Turrini and Piotr J. Wojciechowski. I would never have been able to accomplish this milestone without all of you. In particular, my deep appreciation goes to Andrea Turrini whose mathematical rigor and patience have always helped me to learn a lot from him. Thank you for all your generous support. Many thanks to my great office mate Ernst Moritz Hahn for all his support, careful comments on my manuscripts, and all fruitful scientific and political discussions. I thank Hassan Hatefi not only for all scientific discussions we had but also for sharing a lot of pleasant moments with me during these years. I am grateful to Luis María Ferrer Fioriti for all the long-lasting and deep discussions we had from time to time.

I would like to sincerely thank all former and present members of fabulous Dependable Systems and Software chair for their persistence support and a great working atmosphere. In particular, I would like to express my deep appreciation to Christa Schäfer for all her great help, dedication and correcting my German mistakes. My special thanks are furthermore extended to my colleagues Sebastian Biewer, Yuliya Butkova, Hernan Baro Graf, Pepijn Crouzen, Pedro R. D'Argenio, Christian Eisentraut, Gereon Fox, Felix Freiberger, Hubert Garavel, Alexander Graf-Brill, Arnd Hartmanns, Michaela Klauck, Martin Neuhässer, Gilles Nies, Florian Schießl, David Spieler and Daniel Stan. Outside the chair, I have had pleasure to keep in touch with inspiring colleagues including Alexander Andreychenko, Luca Bortolussi, Bettina Braitling, Carlos E. Budde, Dennis Guck, Abhineet Gupta, Kangli He, Joost-Pieter Katoen, Thilo Krüger, Charalampos Kyriakopoulos, Linar Mikeev, Raúl Monti, Silvia Pelozo, Ralf Wimmer and Verena Wolf. I am really thankful to

all of you.

I would like to extend my special thanks to the people outside of Germany whom I had the opportunity to collaborate or at least to communicate. In particular, many thanks to Khaled Elbassioni and Hans Raj Tiwary for all their fruitful discussion and also their constructive comments on my questions. I am sincerely grateful to Daniel Kuhn, Ardaki Nemirovski and James B. Orlin for all their patience and constructive suggestions to improve my approaches in robust optimization and network flows. I am truly grateful to K. Subramani and Piotr J. Wojciechowski for their pleasant collaboration. I want to thank Lijun Zhang for all his constant support and constructive discussion during these years. A very special thanks to Moshe Y. Vardi for his support which opened up my collaboration with his former postdoctoral researcher, Morteza Lahijanian.

I am grateful to all my friends who made my time in Saarbrücken more pleasant by sharing great moments.

I would like to thank my family especially my parents, my sister and my brothers who always believe in me, support and encourage me to follow my dreams.

And the last but not the least, my deepest and sincere appreciation goes to my wife, Nafiseh, for all her never-ending love, constant support, and for believing in me especially during the times when I could not believe in myself. Thank you for being my best friend and my main source of inspiration.

# Contents


**1 Introduction** **1**
- 1.1 Modelling Real World Systems . . . 1
- 1.2 System Specifications . . . 6
- 1.3 Formal Verification and Controller Synthesis of Probabilistic Systems . . . 7
- 1.4 Compositional Minimization of Probabilistic Systems . . . 10
- 1.5 Applications, Case Studies and Tool Development . . . 13
- 1.6 Main Contributions . . . 14
- 1.7 Dissertation Outline . . . 15
- 1.8 Origins of the Chapters and Credits . . . 17

**I Background** **19**

**2 Mathematical Background** **21**
- 2.1 Mathematical Preliminaries . . . 21
- 2.2 Measure and Probability . . . 24
- 2.3 Convex Polyhedra . . . 26
- 2.4 Systems of Linear Inequalities . . . 27
- 2.5 The NP-Completeness Theory . . . 28

**3 Basics of Mathematical Optimization** **31**
- 3.1 Linear Programming . . . 32
  - 3.1.1 Duality Theory . . . 33
  - 3.1.2 Solution Methods for LPs . . . 36
- 3.2 Network Flows . . . 37
- 3.3 Robust Optimization . . . 38
  - 3.3.1 Uncertain Linear Programming (ULPs) . . . 39
  - 3.3.2 Adjustable Robust Counterpart . . . 40
  - 3.3.3 Affinely Adjustable Robust Counterpart . . . 40

**4 An Overview of Probabilistic Systems** **41**
- 4.1 Markov Chains . . . 42
- 4.2 Markov Decision Processes . . . 43
- 4.3 Weighted Markov Decision Processes . . . 44
- 4.4 Probabilistic Automata . . . 45
  - 4.4.1 Parallel Composition and Hiding . . . 46

4.4.2 Weak Transitions . . . 47
4.5 Interval Markov Decision Processes . . . 50
4.6 Concluding Remarks . . . 53

**II Modelling and Performance Analysis of Probabilistic Systems 55**

**5 Efficiency of Deciding Probabilistic Automata Weak Bisimulation 57**
5.1 Weak Probabilistic Bisimulation . . . 58
5.2 Computing the Weak Bisimilarity for Minimizing Automata . . . 59
5.2.1 Deciding Weak Bisimilarity . . . 59
5.2.2 Minimization and Parallel Composition . . . 61
5.3 Weak Transition Construction as a Linear Programming Problem . . . 63
5.3.1 Network Construction . . . 63
5.3.2 LP Problem Construction . . . 64
5.3.3 Complexity Analysis of Deciding Weak Bisimulation . . . 70
5.4 Efficiency of Solving the LP Problem . . . 70
5.4.1 Efficient Solution: Theory . . . 71
5.4.2 Efficient Solution: Exploiting Structure . . . 78
5.4.3 Efficient Solution: Unsuitable Approaches . . . 80
5.5 Implementation of Minimization . . . 83
5.5.1 Implementation Details . . . 83
5.5.2 Case Studies . . . 85
5.5.3 Compositional Minimization . . . 89
5.6 Concluding Remarks . . . 91

**6 Compositional Minimization for Model Checking of Interval MDPs 93**
6.1 Probabilistic Computation Tree Logic (PCTL) . . . 94
6.2 Probabilistic Bisimulation for Model Checking IMDPs . . . 94
6.2.1 Complexity Analysis of Deciding $\sim_{(\forall)}$ for *IMDP*s . . . 97
6.2.2 Computational Tractability: An Approximation Algorithm . . . 104
6.3 Compositional Reasoning for Interval Markov Decision Processes . . . 115
6.3.1 Action Agnostic Probabilistic Automata . . . 115
6.3.2 *IMDP*s vs. *PA*s . . . 118
6.3.3 Compositional Reasoning for *IMDP*s . . . 120
6.3.4 Interleaved approach . . . 125
6.4 Case Studies . . . 127
6.5 Concluding Remarks . . . 128

**7 Compositional Minimization for Optimal Control of Interval MDPs 131**
7.1 Alternating Probabilistic Bisimulation Relations for *IMDP*s . . . 132
7.2 A PTIME Decision Algorithm for Bisimulation Minimization . . . 136
7.3 Compositional Reasoning . . . 140
7.4 Case Studies . . . 144
7.5 Concluding Remarks . . . 145

**8 Multi-objective Robust Controller Synthesis for Interval MDPs 147**

8.1 Multi-objective Robust Controller Synthesis for IMDPs . . . . . 147
8.1.1 Multi-objective Queries . . . . . 149
8.1.2 Robust Controller Synthesis . . . . . 152
8.1.3 Multi-objective Robust Controller Synthesis: Other Queries . . . 162
8.1.4 Generation of randomized controllers . . . . . 163
8.2 Case Studies . . . . . 167
8.3 Concluding Remarks . . . . . 171

**9 Bisimulation Minimization for Model Checking of UwMDPs 173**
9.1 Uncertain weighted Markov Decision Processes . . . . . 174
9.2 Bisimulation Minimization for UwMDPs . . . . . 176
9.2.1 Probabilistic Bisimulation . . . . . 176
9.2.2 Decision Algorithm . . . . . 177
9.3 Reward-Bounded Reachability Probability for *UwMDP*s . . . . . 179
9.4 Case Studies . . . . . 188
9.4.1 Autonomous Nondeterministic Tour Guides (ANTG) . . . . . 189
9.4.2 Randomized Consensus Protocol . . . . . 191
9.4.3 Randomized Dining Philosophers . . . . . 191
9.5 Concluding Remarks . . . . . 192

**III Conclusion 195**

**10 Conclusion 197**
10.1 Summary . . . . . 197
10.2 Future works . . . . . 200

**Bibliography 202**

CHAPTER 1

# Introduction

The indisputable role of computerized systems in our daily lives is ever accelerating. In many aspects of our everyday life, we rely on embedded software systems ranging from mobile phones to medical devices, automobiles and airplanes. Our reliance on embedded systems manifests the significance of their reliable performance. In particular, software quality assurance is of greatest importance when human lives or key investments are at stake or as failures may be economically or physically costly. For instance, the malfunction of some safety-critical systems in the past such as Therac-25 Radiation Overdosing (1985-87) or later Pentium FDIV Bug (1994) and Ariane 5 Crash (1996) had fatal consequences.

The ever increasing complexity of modern computerized systems solicits the need for development of sound mathematical formalisms, algorithmic techniques and tools for evaluation of qualitative and quantitative properties of such real world systems.

This is the concrete motivation for the present dissertation. We aim to advance quantitative evaluation of probabilistic systems despite their complexity from various aspects ranging from modelling to theory and tools. In particular, from modelling viewpoint, we introduce a novel stochastic model which encompasses essential features for designing complex systems and also is capable of capturing quantities like preferences or priorities in scenarios with uncertainties. From theoretical viewpoint, leveraging techniques from mathematical optimization and computational geometry, we propose sound and complete decision algorithms for problems arising in quantitative concurrency and performance evaluation of probabilistic systems. Finally, we indicate the applicability and efficiency of the developed algorithms by applying them on several case studies and successfully integrate them with a state-of-the-art probabilistic model checking tool.

## 1.1 Modelling Real World Systems

Real world systems are usually too complex to be analyzed in full detail. To reduce the complexity of such an analysis, a simplified but accurate enough model of the system has to be constructed and then verified with respect to a number of properties the system is expected to satisfy. Among others, probability, nondeterminism, and uncertainty are core aspects of a real world system that are worth considering in the model.

**Probability.** *Probability* for instance arises when a system, performing an action, is able to switch to more than one state and the likelihood of each of these states can be faithfully estimated. Probabilistic choices which are typically known as *quantified uncertainty* can model both specific system choices (such as flipping a coin, commonly used in randomized distributed algorithms) and general system properties (such as message loss probabilities when sending a message over a wireless medium). In particular, distributed algorithms like cryptographic protocols [Mao03] such as SSL are based on random choices to break symmetry or to deliberately insert uncertainty in order to achieve their goals. Each time a message is transmitted on the network, in fact, transmission protocols have to manage the potential of corruption of the messages as well as their loss, as the effect of the interference with other concurrent transmissions or of physical properties of the transmission medium. For instance, simultaneous transmissions on the same channel of a wireless network lead to the collision of the sent messages and their corruption.

**Nondeterminism.** *Nondeterminism* represents behaviors that we can not or we do not want to attach a precise (possibly probabilistic) outcome to. Contrary to the probabilistic approaches, nondeterminism is known as an unquantified uncertainty about the system behaviour which needs to be resolved amongst several alternative behaviours. For example, in a robot motion planning scenario, the robot transitions in its environment are nondeterministic: we do not know which action the robot is going to perform; nevertheless, we know the set of different actions the robot can select from. Nondeterminism can arise deliberately. For instance, it may appear as a reflection of behaviors we keep undetermined for system simplification or for allowing different implementations. Apart from that, it might reflect the concurrent execution of several components at unknown (relative) speeds or might take place in communication of open systems with other components in their environment.

**Parameter Uncertainty.** *Parameter uncertainty* relates to the fact that not all system parameters may be known exactly, including exact probability values. Parameter uncertainty almost often arises in partially observable or stochastic environments due to measurement or systemic errors. The presence of such an unavoidable uncertainty which reflects imperfect or unknown information about some aspects of the system demands careful handling in the process of decision making.

To study the properties of real-world applications, several models have been proposed in the literature: the basic model in the discrete time domain is that of discrete time Markov chains (*DTMC*s) or, briefly, Markov Chains (*MC*s) [Ste94, HJ94], where the time is discrete (i.e., the system performs one operation per clock tick) and probability determines the states to be reached in their entirety. The continuous-time counterpart is known as the continuous-time Markov chains (*CTMC*s) [ASSB00, BHHK03] model, where exponentially distributed sojourn times distributions control the evolution of the system.

*DTMC*s and *CTMC*s are purely probabilistic. They have been extended with nondeterminism to permit different operations or behaviors from a specific state. This extension results Markov decision processes (*MDP*s) [How60, Bel57, Put05, How07] and continuous-time Markov decision processes (*CTMDP*s) [How60, Ber05, BHKH05, Put05, WJ06], respectively. These models, despite being widely used to represent and study real systems, are

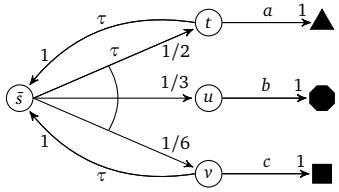



Figure 1.1: The *PA* $\mathfrak{P}$

not closed under composition, that is, there is no guarantee that complex systems can be obtained by composing smaller components while preserving the intended model class. This compositionality property is rather important as it is usually much easier to model and study (a set of) small systems and then combine them together rather than building a single large system. Moreover, real systems and applications involve several parties each one composed by modules working together in parallel. Two models have been proposed to achieve such compositional property: probabilistic automata (*PA*s) model [Seg95,Seg06] for discrete time systems and the interactive Markov chains (*IMC*s) [Her02] model for continuous-time systems.

Throughout this dissertation, we focus on *probabilistic automata* in the realm of probabilistic models featuring nondeterministic and probabilistic behaviours. Probabilistic automata extend classical concurrency models in a simple yet conservative fashion. In probabilistic automata, there is no global notion of time, and concurrent processes may perform probabilistic experiments inside a transition. This is represented by transitions of the form $s \xrightarrow{a} \mu$, where $s$ is a state, $a$ is an action label, and $\mu$ is a probability measure on states. Labelled transition systems are instances of this model family, obtained by restricting to Dirac measures (assigning full probability to single states). Moreover, foundational concepts and results of standard concurrency theory are retained in full and extend smoothly to the *PA* model. Since the *PA* model is akin to the *MDP* model, its fundamental beauty can be paired with powerful model checking techniques, as implemented for instance in the PRISM tool [KNP11].

**Example 1.1.** *An example of a probabilistic automaton is depicted in Figure 1.1. It includes seven states each of which represents a specific status of the system being modeled. The model consists of one internal action $\tau$ that cannot be observed from outside the system and three external actions $a$, $b$ and $c$ which are presumably shared with an external environment. Nondeterminism can, for instance, happen in state $t$ in which either an internal or external action can be performed.* ⧫

The recently introduced Markov automata (*MA*s) model [EHZ10b,DH12,Eis17] unifies and merges all such models in a single framework. This formalism is suitable for studying systems featuring continuous-time based behaviors as well as probabilistic and nondeterministic choices. Moreover, the Markov automata model provides the semantics to every generalized stochastic Petri net (GSPN) [EHKZ13], a popular modelling formalism for performance and dependability analysis.

Modelling formalisms like *MDP*s or *PA*s are used for representing systems that combine nondeterministic and probabilistic behaviour. They can be viewed as transition systems where in each step an outgoing transition of the current state is chosen *nondeterministically*, i.e., the transition action is performed and the successor state is chosen *randomly* according to a fixed probability distribution assigned to this transition. However, modelling

a physical or artificial system suffers from several limitations. An important limitation is the inherent loss of precision that is introduced by measurement errors, statistical estimates and discretization artifacts which necessarily happens due to incomplete knowledge about the system behavior. Thus, assigning fixed probability distributions to transitions is not realistic [JL91, KU02] in many modelling scenarios and in fact the true probability distribution to be associated with transitions is in most cases uncertain and is given by either external parameters or confidence intervals.

Several models have been proposed in the literature to study formally systems where a combination of probability, nondeterminism and uncertainty is considered. These modelling formalisms have been broadly investigated in the verification and optimal control community [NG05, BDL+17, DLP16].

Interval Markov chains [KU02, JL91] or abstract Markov chains [FLW06] extend standard *MC*s with interval uncertainties. They do not feature nondeterminism among transitions. Interval *MDP*s [NG05, WTM12, PLSVS13] (*IMDP*s) address this need and extend *MDP*s by bounding the probabilities of each successor state by an interval instead of a fixed number. Therefore, the real probability distribution for transitions is in most cases uncertain and is often given by confidence intervals. Interval *MDP*s which are also known as bounded-parameter *MDP*s (*BMDP*s) [GLD00] have been extended to the slightly more general classes of *MDP*s with incomplete or uncertain transition probabilities [SL73, WE94].

In *IMDP*s the transition probabilities are not fully specified and this uncertainty again needs to be resolved nondeterministically. In particular, each time a state is visited, an outgoing transition is chosen nondeterministically by a *scheduler* (also called controller or strategy) and then a transition probability distribution is chosen by a *nature* from a convex set of uncountably many choices. Then the successor state is chosen randomly, according to the selected probability transition.

The two sources of nondeterminism have *different* interpretation in different applications:

1. In verification of parallel systems with uncertain transition probabilities [PLSVS13] the transitions correspond to unpredictable interleaving of computation of the communicating agents. Hence, both the choice of transitions and their probability distributions is *adversarial*.
2. In control synthesis for systems with uncertain probabilities [WTM12, PSVS14] the transitions correspond to various control actions. We search for a choice of transitions that is *optimal* against an adversarial choice of probability distributions satisfying the interval bounds.
3. In parameter synthesis for parallel systems [HHZ11b] the transition probabilities are underspecified to allow freedom in implementation of such a model. We search for a choice of probability distributions that is optimal for adversarial choice of transitions (again stemming from the possible interleaving).

Furthermore, the choice of probability distributions satisfying the interval constraints can be either resolved statically [JL91], i.e. at the beginning once for all, or dynamically [Iye05,SVA06], i.e. independently for each computation step. Throughout this dissertation, we focus on the dynamic approach that is easier to work with algorithmically and can be seen as a relaxation of the static approach that is often intractable [BLW13, SVA06, CSH08, DLL+11].

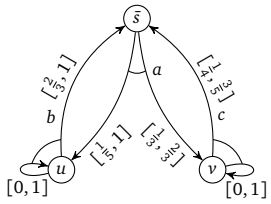



Figure 1.2: The *IMDP* $\mathcal{M}$

**Example 1.2.** *An instance of the IMDPs is depicted in Figure 1.2. When the system is at the initial state $\bar{s}$, a possible resolution of nondeterminisms can be as follows: the scheduler can select the action $a$ and afterwards, the nature may choose the probability $\frac{2}{5}$ to go to state $u$ and the probability $\frac{3}{5}$ to go to state $v$. Note that due to the dynamic way of resolving nondeterminisms, this choice of probability distribution can alter when we get back again to the state $\bar{s}$.* ♦

The uncertainty set in *IMDP*s model which features the set of all potential probability distributions is usually supposed to be closed intervals. Convex *MDP*s [NG05, WTM12, PLSVS13] allow more general sets of distributions to be associated with each transition, not only those described by intervals. In particular, convex *MDP*s increase the expressiveness of the *IMDP*s by generalizing the interval uncertainty with other nonlinear convex sets such as ellipsoidal or likelihood model of uncertainties. Convexity of the uncertainty set is usually considered in modelling real world systems to keep the computations tractable.

*MDP*s with convex uncertainty sets have further been extended. Parametric *MDP*s [HHZ11b] to the contrary allow such dependencies as every probability is described as a rational function of a finite set of global parameters. On the other side, abstract probabilistic automata [DKL$^+$11a] constitutes an abstraction model for *PA*s in which uncertainty of the stochastic behavior is modeled by underspecified stochastic constraints.

Central to the content of this dissertation, we introduce *uncertain weighted Markov decision processes* (*UwMDP*s) as a novel stochastic modelling formalism to model systems featuring nondeterminism, probability and uncertainty. Different from *IMDP*s, each transition in an *UwMDP* is associated with a confidence interval of weights such that any integer value in this interval provides a feasible weight for that transition. Note, however, that our consideration of only integer weights is without loss of generality and the extension to rational weights is straightforward.

In the model of *UwMDP*, weights play a similar role as in GSPN [MCB84] or EMPA [BG96], namely, they will be used to induce a distribution over all transitions with the same label. For instance, consider an *UwMDP* which includes a single transition labelled by an action $a$ outgoing from a state $s$ and moves to state $s_1$ and $s_2$ with weight intervals $[1, 3]$ and $[5, 9]$, respectively. Then, choosing the concrete weights 2 in $[1, 3]$ and 7 in $[5, 9]$ induces a probability distribution over states $s_1$ and $s_2$: the system evolves to state $s_1$ with probability $\frac{2}{9}$ and to state $s_2$ with probability $\frac{7}{9}$.

Similar to *IMDP* models that include two sources of nondeterminisms which are resolved by a scheduler and nature, a notion of scheduler and nature is also needed to reason on *UwMDP*s. Hence, the *UwMDP*s can be observed as a game between a scheduler and a nature where each time a state is visited, an outgoing transition is nondeterministically chosen by scheduler and afterwards, nature selects a realization of weights from a set of feasible weights that is specified in terms of intervals. The selected weights by the nature then induce a probabilistic transition over successor states.

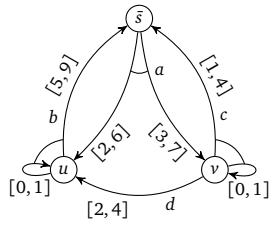



Figure 1.3: The *UwMDP* $\mathcal{W}$

**Example 1.3.** *Consider the UwMDP presented in the Figure 1.3. When the system is at state* $v$*, the scheduler may nondeterministically choose the probabilistic transition labelled by action* $c$ *and nature can select weights 1 and 3 leading to state* $v$ *and* $\bar{s}$*, respectively. This means that* $v$ *will evolve into* $v$ *and* $\bar{s}$ *with probabilities* $\frac{1}{4}$ *and* $\frac{3}{4}$*, respectively.* ♦

In the *UwMDP*s model, weights can be used to denote priorities or preferences, which are quantities used to generate probabilistic behaviors. For instance, in a motion planning scenario, a robot may choose to serve its clients stochastically relative to the preferences they expressed. In this respect, *UwMDP*s offer users more flexibility than *IMDP*s to model uncertainties.

## 1.2 System Specifications

Together with a probabilistic model which encodes precisely the behaviour of the system under consideration, we need a proper language to formally specify the system properties. A natural way to express such properties is the use of *temporal logics*. Temporal logics provide a logical framework to represent and reason about temporal behavior of a system [Lam83].

Temporal logics are usually classified as either *linear time* or *branching time* logics. Linear time logics consider properties of individual executions. To the contrary, branching time logics express properties of a computation tree. The most widely used logic to describe linear time properties in the model of probabilistic systems is *linear temporal logic* (LTL) [Pnu77]. For branching time properties, the most widely used logics is the Computation Tree Logic (CTL) [CE81]. Later, a probabilistic extension of both LTL and CTL logic presented in the form of PLTL [CY95] and PCTL [HJ90] which provides a tool for probabilistic quantification of described properties. The logics LTL and CTL can be unified to CTL* [EH86] in order to express linear time and branching time properties simultaneously. Similarly, the logic PCTL* [BDA95] also proposed as a probabilistic extension of CTL*. We refer the reader to [BMN00, Eme90] for a survey on temporal logics.

On the specification side, throughout this dissertation, we focus on Probabilistic Computation Tree Logic (PCTL) to express the properties of probabilistic systems. The PCTL logic permits us to express system properties such as "If a massage is sent in a wireless sensor network, the probability at which it is delivered within 5 seconds is at least 0.98%".

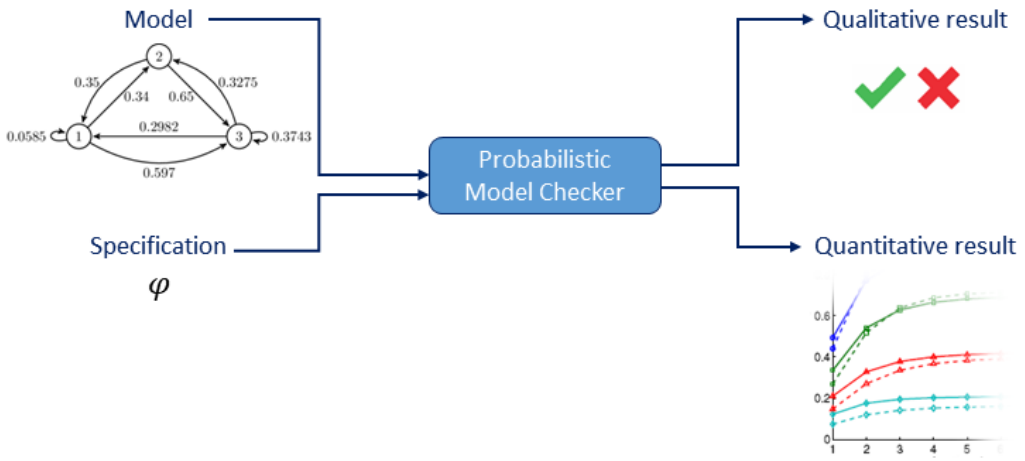



Figure 1.4: Probabilistic model checking in a nutshell

## 1.3 Formal Verification and Controller Synthesis of Probabilistic Systems

Ubiquitous computerized systems are prone to failure due to their complex nature. The risk of failure typically correlates positively with the complexity and expectations on the systems. Hence, powerful formal techniques are required in order to ensure the correctness of the systems being modeled against the requirements.

Formal verification is the act of mathematically reasoning about the correctness of computer systems. Soon after the emergence of formal verification in seminal works by Clarke and Emerson [CE81] and by Queille and Sifakis [QS82], it has played a major role in improving reliability of computer systems. *Probabilistic model checking* is one of the quantitative techniques offered by formal verification. In particular, for a given model of a probabilistic system which encodes all possible behaviour of the system under study at a certain level of abstraction, and a logical property that specifies the system requirement, probabilistic model checking automatically verifies if the model satisfies the property of interest. Therefore, probabilistic model checking provides a broad range of exhaustive and quantitative analyses of system properties. The probabilistic model checking routine is summarized in Figure 1.4. Over the past decades, much effort has been spent in the area of probabilistic model checking. Efficient algorithmic techniques and tools for model checking CTL formulas on *MDP*s have been proposed in [CE81]. On the other hand, model checking LTL logic for *MDP*s is shown to be polynomial in the size of the model and double exponential in the size of the LTL property [BK08]. As regards the probabilistic fragment of CTL, it is shown that model checking of PCTL properties for *MDP*s is in **P** [HJ94]. To the contrary, model checking probabilistic LTL (PLTL) properties of *MDP*s has been studied in [CY95] and shown to have the same complexity as for the model checking of LTL formulas which is 2**EXPTIME**-complete. For probabilistic systems under presence of uncertainty, the model checking of PLTL has been explored in [BLW13] for the interval *MC*s and proved

to be in **EXPTIME** and **PSPACE**-hard. Model checking PCTL properties on interval *MC*s has been addressed in [SVA06] and later investigated by Puggelli et al. [PLSVS13, Pug14] for *IMDP*s. In particular, in [PLSVS13] the authors presented a polynomial-time algorithm for model checking PCTL properties on *IMDP*s.

A closely related problem to the probabilistic model checking is the *controller synthesis* problem. The goal of controller synthesis is to construct a controller that limits the behavior of an existing system so as to make it satisfy a certain quantitative objective. This can be regarded as a game between controller and environment in which the controller has a winning strategy if the system property can be made to almost surely hold. The controller synthesis approach has successfully been applied to several application domains such as robots control [LWAB10] and power management for hardware [FKN$^+$11]. In the controller synthesis setting, Baier et al. [BGL$^+$04] explored the problem of controller synthesis for *MDP*s from PCTL and LTL properties. In particular, they showed that the problem of synthesizing Markovian deterministic strategies for PCTL properties is **NP**-complete. Nevertheless, synthesizing history-dependent strategies for LTL properties is elementary. In [KS05] it is shown that the problem of synthesizing Markovian randomized controllers for PCTL properties extended with long-run averages is decidable and the authors provide an algorithmic solution for that. On the other hand, Puggelli et al. studied the problem of synthesizing PCTL specifications for *MDP*s with convex uncertainties and proposed a sound and complete synthesis algorithm to solve it. Finally, the problem of synthesizing an optimal controller for Convex *MDP*s which satisfies a fragment of the PLTL logic is investigated in [WTM12]. satisfying a specification expressed in the fragment of the PLTL logic. Among other technical tools, all these approaches make use of (robust) dynamic programming relying on the fact that transition probability distributions are resolved dynamically. For the static resolution of distributions, adaptive discretization technique for PCTL parameter synthesis is given in [HHZ11a].

Thus far we have considered probabilistic model checking and controller synthesis with respect to a single property. However, in many application scenarios, we may need to care about several possibly conflicting quantitative properties and therefore, we need to know whether all properties can be satisfied simultaneously. More specifically, we will be interested in the trade-off or the Pareto curve for optimizing different properties. This setting allows us to formulate queries such as: *Can an attacker break into our server with a probability of at least 1%, and can he infect a communication subsystem with a probability of 2%, provided that some of the attack steps he needs to undertake have a chance of 10-20% to succeed?* An obvious generalization posing several such objectives asks whether there exists a model instance that fulfills all the given multi-objectives. Multi-objective approaches have received widespread attention in several areas such as operations research [DSH16, Ehr06], economics [Bra13] and stochastic control [MA04] and probabilistic verification [RRS15,CMH06,EKN$^+$12,KNPQ13,Mou04,OPW13,PWGH13,FKN$^+$11,FKP12]. In particular, in recent years there has been an increasing interest in multi-objective controller synthesis for probabilistic systems. Among other results, Chatterjee, Majumdar, and Henzinger in [CMH06] considered multi-objective model checking of *MDP*s with multiple discounted reward objectives and afterwards, Etessami et al. extended their approach for multi-objective probabilistic model checking of *MDP*s for probabilistic $\omega$-regular properties. In [Fur80, Whi82, Hen83, Gho90, WT98] multi-objective *MDP*s have also been investigated with regards to discounted reward or long-run average reward properties. The works in [FKN$^+$11, FKP12, KNPQ13] focused on multi-objective verification of *MDP*s with

respect to several other properties. In particular, in [FKN+11] the authors propose multi-objective verification techniques for quantitative properties of *PA*s (and also *MDP*s) including $\omega$-regular and expected total reward properties. Later on efficient algorithms relying on the generation of successive approximations of the Pareto curve for the multi-objective model checking of *MDP*s have been proposed in [FKP12]. In [CFK+13], these algorithms were extended to the more general models of 2-player stochastic games. Moreover, a rigorous complexity analysis of multi-objective model checking of *MDP*s with respect to several quantitative properties was undertaken in [RRS15].

Throughout this dissertation, we explore formal verification and controller synthesis of probabilistic systems in two orthogonal dimensions. We elaborate on our approaches as follows.

In the area of formal verification, we discuss extreme (maximal/minimal) reward-bounded reachability probabilities [AHK03] of *UwMDP*s. Thus each *UwMDP* is associated with a reward structure, which assigns to each weighted transition a reward. In this setting, we compute the extremal reward-bounded reachability probabilities under all resolutions of nondeterminism and uncertainty. In particular, we assume that the scheduler and nature which respectively resolve the nondeterminism and uncertainty are playing together against the model. On the one hand, this problem is an extension of the work in [AHK03] enriched with uncertainties, while on the other hand, it can also be seen as an extension of the work in [PLSVS13] to model with reward structures. Despite the fact that an *UwMDP* may represent an equivalent, but exponentially larger, model without uncertainties, we propose an algorithm to compute extreme reward-bounded reachability probabilities in pseudo polynomial time – polynomial with respect to the size of the given *UwMDP* and quadratic with respect to the reward bound. As a core part, our algorithm is based on a reduction to the similar problem on *IMDP*s which uses a construction that does not, surprisingly, involve an exponential blow up. Afterwards, we formally show that the extremal probabilities are achieved by deterministic reward-positional schedulers and natures. Finally, we present a polynomial time algorithm to compute these quantities. Along the line of [PLSVS13], extremal reachability probabilities without reward bounds can be computed efficiently for *UwMDP*s as well.

In the area of controller synthesis, we present a novel technique for robust controller synthesis of *IMDP*s with respect to multiple objectives. Our aim is to synthesize a robust controller that guarantees the satisfaction of the multi-objective property at the same time, despite the additional uncertainty over the transition probabilities in these models. Here, the goal is first to provide a complete trade-off analysis of several, possibly conflicting, quantitative properties and then to synthesize a controller that guarantees the user's desired behavior. Such properties, for instance, ask to "find a robot controller that maximizes $p_{\text{safe}}$, the probability of successfully completing a track by safely maneuvering between obstacles, while minimizing $t_{\text{travel}}$ the total expected travel time." This example has competing objectives: maximizing $p_{\text{safe}}$, which requires the robot to be conservative, and minimizing $t_{\text{travel}}$, which causes the robot to be reckless. In such contexts, the interest is in the *Pareto curve* of the possible solution points: the set of all pairs of $(p_{\text{safe}}, t_{\text{travel}})$ for which an increase in the value of $p_{\text{safe}}$ must induce an increase in the value of $t_{\text{travel}}$, and vice versa. Given a point on the curve, the computation of the corresponding controller is asked. Our approach views the uncertainty as making adversarial choices among the available transition probability distributions induced by the intervals, as the system evolves along state transitions. In this respect, we first show that this problem is **PSPACE**-hard. Then, we

provide a value iteration-based decision algorithm to approximate the Pareto set of achievable points. Our algorithm iterates over a weighted sum of objectives which are in turn optimized through a value-iteration procedure. It is worthwhile to note that the major difference between the provided value iteration algorithm and the standard value iteration is its need to optimize a mixture of unbounded and bounded properties. Moreover, the multi-objective robust controller synthesis for *IMDP*s cannot be solved on the *MDP*s generated from *IMDP*s by computing all feasible extreme transition probabilities and then applying the algorithm in [FKP12]. The latter solution approach could be valid if the two sources of nondeterminisms in *IMDP*s were resolved cooperatively. As for the sake of robust controller synthesis we need the competitive semantics, one can instead transform *IMDP*s to $2\frac{1}{2}$-player games [BKW14] and apply the algorithm in [CFK$^+$13]. Unfortunately, the transformation to (*MDP*s or) $2\frac{1}{2}$-player games induces an exponential blow up, adding an exponential factor to the worst case time complexity of the decision problem. Our proposed algorithm instead prevents this difficulty by solving the robust synthesis problem directly on the *IMDP* so that the core part, i.e., optimizing the weighted sum of objectives, can be solved with time complexity polynomial in the size of the given *IMDP* model.

## 1.4 Compositional Minimization of Probabilistic Systems

Given a real system, we can conceive several different probabilistic automata models to reflect its behavior: for example, we can use different names for the states, we can encode probabilistic choices as sequences of events or as single events, we can detail or abstract from particular details, and so on. It is clear that these choices affect the resulting model whose size may vary even if all these models represent the same real system. A possible way to abstract away from this modelling details is to use the so-called bisimulation relations that allow us to declare that two models are bisimilar or equivalent whenever they are related, respectively. Intuitively, a system $S_1$ is bisimilar to a system $S_2$ if $S_1$ is able to mimic whatever $S_2$ can do and vice versa. Hence, *bisimulation relations* provide a powerful tool to manifest whether two models describe essentially the same system.

The bisimilarity of two systems can be viewed in terms of a game played between a challenger and a defender. In each step of the possibly infinite bisimulation game, the challenger chooses one of the two automata, makes a step, and the defender then needs to match it with a step of the other automaton. Depending on how we want to treat internal computations, this leads to *strong* and *weak* bisimulations: the former requires that each single step of the challenger automaton is matched by an equally labelled single step of the defender automaton, the latter allows the matching up to internal computation steps. On the other hand, depending on how nondeterminism is resolved, probabilistic bisimulations can be varied by allowing the defender to match the challenger's step by a convex combination of enabled probabilistic transitions. This results in a spectrum of four bisimulations: strong [Seg95, Han91, Var85], strong probabilistic [Seg95], weak [PLS00, Seg95, EHZ10a, EHZ10b], and weak probabilistic [Seg95, EHZ10b, EHZ10a] bisimulations. For a recent survey on behavioral equivalences and preorders, we refer the interested reader to [GHT14].

Besides comparing automata, bisimulation relations allow us to reduce the size of an automaton without changing its properties (i.e., with respect to logic formulae satisfied by it). This is particularly useful to alleviate the state explosion problem notoriously encoun-

tered in model checking. If the bisimulation is a congruence with respect to the operators of a process calculus used to build up the automata out of smaller ones, this can give rise to a compositional strategy to associate a small automaton model to a large system without intermediate state space explosion. In several related settings, this strategy has been proven very effective [CGM+96, HK00, KKZJ07, BHH+09, CHLS09]; it can speed up the overall model analysis or turn a too large problem into a tractable one.

Both strong and weak bisimilarity are used in practice, with weaker relations leading to greater reduction. However, this approach has thus far not been explored in the context of *MDP*s or probabilistic automata. A striking reason is that until recently no effective decision algorithm was at hand for weak probabilistic bisimilarity on *PA*. A polynomial time decision algorithm has been proposed only recently [TH15], based on linear programming problems. In particular, the weak bisimilarity decision algorithm follows the standard partition refinement approach [KS90, PT87, PLS00, CS02], and thereby induces a polynomial number of linear programming problems that can be solved in polynomial time [Kar84,Kha79]. This algorithm can be embedded into a procedure to compress a given *PA* to its canonical minimal representative [EHS+13]. Since weak probabilistic bisimilarity is a congruence for parallel composition and hiding operators on *PA*s (we refer the interested reader to [Seg95, SL95] for more details), this paves the way for compositional strategies to associate a small *PA* model to a large system without intermediate state space explosion.

For probabilistic systems featuring uncertainty in the transition probability values, compositional specification theory has also been investigated in the literature. In particular, interval Markov chains [JL91] and abstract probabilistic automata [DKL+11a,DKL+11b] serve as specification theories for *MC*s and *PA*s featuring satisfaction relation, and various refinement relations. In order to be closed under parallel composition, abstract *PA* allow general polynomial constraints on probabilities instead of interval bounds. Since for interval *MC* it is not possible to explicitly construct parallel composition, the problem whether there is a common implementation of a set of interval Markov chains is addressed instead [DLL+11]. To the contrary, in the continuous-time setting, the authors in [KKN09] consider abstract interactive Markov chains which encompass interval *MC*s and modal transition systems in a unified framework and show that they are closed under parallel composition. The reason is that unlike probabilities, the rates do not need to sum up to 1.

In the area of compositional minimization of probabilistic systems, we advance the state-of-the-art in three dimensions detailed as follows.

First, we consider deciding *PA* weak bisimulation which is known to be polynomial [HT12]. In particular, we discuss the efficiency of solving the specific LP problems from both theoretical and practical viewpoints. We first consider the theoretical efficiency of solving the problem. We first look at rational *PA*s, i.e, *PA*s with only rational probability values, and study the complexity of the decision problem together with several optimizations. This entails reformulating the original LP problem [TH15] in order to simplify the construction of the dual LP problem [BT97] which is smaller in size than the original. By using a state-of-the-art preconditioned conjugate gradient (PCG) algorithm combined with a partial updating procedure [Ans99] we show that the dual LP problem can be solved efficiently. On the other hand, taking advantage of the small-sized dual LP problem, we give an upper bound on the complexity of checking the feasibility of the original LP problem. We also discuss how the efficiency of solving the decision problem can exploit the problem structure. In practice one would usually opt for the notoriously efficient simplex

method [Sha87] to solve the LP problems. However a small modification of the underlying network [TH15] enables us to adapt the corresponding LP problem into a variant of a minimum cost flow problem [AMO93] with flow proportional sets. This is a special class of linear programming problems where the underlying network structure can be exploited, in particular if it is sparse. Sparsity is indeed frequently observed in practical applications of probabilistic automata. We therefore compare the simplex method with a very efficient state-of-the-art network simplex algorithm [BF12] specialized for the minimum cost flow problem with additional side constraints. This is known to outperform the simplex method [MSJ11,HK95,Cal02] when the number of nodes is an order of magnitude larger than the number of side constraints. We furthermore discuss different implementations of the decision algorithm, focusing on effective minimization of *PA* with respect to weak probabilistic bisimilarity. One of the implementations exploits that the problem at hand can be encoded into SAT modulo linear arithmetic. We report on extensive empirical investigations in the context of concurrent probabilistic systems. It turns out that minimization can be applied effectively to standard *PA* benchmarks. Several techniques and heuristics are discussed to further reduce the actual execution time of the algorithm, by showing how an accurate management of transition computation and minimization helps in the reduction of large automata, in particular when they are the result of the composition of several automata. The problem of efficiently deciding bisimilarities for *PA*s and *MDP*s is of pivotal importance for compositional construction and minimization techniques for complex probabilistic models. Once in place, these techniques can be rolled out to operations research, automated planning, and decision support applications.

Next, we discuss different interpretations of uncertainty in *IMDP* models which are studied in the literature and that result in two different definitions of bisimulations: one for models where the two nondeterminisms are resolved in a cooperative way and another for models where it is resolved in a competitive way. Furthermore, we provide a thorough complexity analysis of deciding these bisimulations and show how to compute these bisimulations by algorithms based on comparing polytopes of probability distributions associated with each transition. As regards the cooperative semantics, we show that deciding probabilistic bisimulation is **coNP**-complete and present an algorithm for its computation which is fixed parameter tractable with respect to the maximal dimension of the polytopes (i.e. maximal number of different states that an uncertain transition can lead to). Furthermore, taking advantage of the results from robust optimization setting, especially theory of uncertain Linear Programming (LP) problems, we show that deciding bisimilarity of a pair of states can be encoded as adjustable robust counterpart of an uncertain LP. Accordingly, we show that using affine decision rules, probabilistic bisimulation relations can be approximated in polynomial time. As regards the competitive semantics, we define two alternating probabilistic bisimulations to compress the *IMDP* model size with respect to the controller synthesis and parameter synthesis semantics while preserving probabilistic CTL property satisfaction. In this respect, we first show that these two alternating bisimulations coincide and afterwards, show that the compressed models can be computed in polynomial time. We furthermore investigate how the parallel composition operator for *IMDP*s can be defined so as to arrive at a congruence closure. While nondeterminism is a genuine asset of *IMDP*s, a closure property can not be established for asynchronous parallelism with synchronisation. The possibility of establishing an asynchronous parallelism with synchronisation for *IMDP* models has been recently investigated in [HHS+16b]. However, the underlying construction is problematic since it does

not manage correctly the spurious distributions. More precisely, for a pair of *IMDP* components the equality of the emerged sets of spurious distributions as a parallelism result should be guaranteed in order to establish the congruence result. Alternatively, we show that for both cooperative and competitive semantics, *IMDP*s are closed under interleaving parallelism as well as under synchronous parallelism. This enables us to develop compositionality results with respect to bisimulation for these two facets of parallelism. We finally argue by several case studies that, if uncertainty stems from a small number of different phenomena such as node failure or loss of a message, the same shape of polytopes will repeat many times over the states space. We demonstrate that the redundancy in this case may result in a massive state space minimization.

Last, we show that *UwMDP*s are equipped with an efficient minimization theory based on probabilistic bisimulation. Such a bisimulation minimization approach can be applied to alleviate state space explosion problems. More concretely, we define probabilistic bisimulation relations for *UwMDP*s and show that they can be computed efficiently in polynomial time with respect to sizes of *UwMDP*s in contrast to the (cooperative) bisimulation for *IMDP*s which is computationally intractable. This observation indeed elucidates one of the substantial distinctions between *UwMDP*s and *IMDP*s. In particular, *UwMDP*s not only enables us more flexibility to model uncertainties but also provides us with a natural and tractable bisimulation minimization theory.

## 1.5 Applications, Case Studies and Tool Development

At this point, we briefly point out to the tool supports for the probabilistic model checking of probabilistic systems. As a matter of fact, apart from all aforementioned theoretical results several powerful model checkers have been developed and applied in practice successfully. Among others, tools such as PRISM [PRI], Modest [BDH+12], IscasMC [HLS+14] and quite recently JANI [BDH+17] and STORM [DJKV17] can be used to analyze a wide range of probabilistic systems and logic specifications. In parallel to that, tools such as PARAM [HHWZ10] and PROPhESY [DJJ+15] can be applied for parameter synthesis of parametric probabilistic systems.

Throughout this dissertation, we indicate the applicability and efficiency of the developed algorithms by applying them on several case studies: two developed merely for parametric probabilistic systems and the rest adapted from PRISM model checker [HKNP06]. In particular, our two developed case studies which are of high practical relevance motivate the use of parametric probabilistic systems as appropriate modelling formalisms to capture inherent uncertainties encoded in their scenarios. In the first case study, we consider *robot motion planning under uncertainty* in which designers are often interested in a plan that simultaneously satisfies multiple conflicting objectives, e.g., maximizing the chances of reaching the target while minimizing the energy consumption. To this aim, we analyze multi-objective robust controller synthesis of the model aiming at finding a robust controller which satisfies a given multi-objective property and also compute the Pareto curve for the property. Our second case study is inspired by *Autonomous Nondeterministic Tour Guides* (ANTG) in [CRI07], which models a complex museum with a variety of collections. It is worthwhile to note that the model introduced in [CRI07] is an *MDP* and we use both *IMDP* and *UwMDP* models to incorporate parameter uncertainties into the *MDP*. The *IMDP* model of ATNG case study is used for multi-objective robust con-

troller synthesis problem while its *UwMDP* model is used to compute maximal/minimal reward-bounded reachability probabilities. Apart from applications and developed case studies, a major part of our algorithms including the bisimulation minimization techniques for *IMDP*s under both cooperative and competitive semantics as well as multi-objective robust controller synthesis approaches have been integrated with IscasMC: a web-based probabilistic model checker [HLS+14].

## 1.6 Main Contributions

The current dissertation contributes to the field of probabilistic verification in various ways. In particular, the thesis focuses on complexity analysis and designing efficient decision algorithms for bisimulation minimization, formal verification and optimal control of probabilistic systems. While we initially work with probabilistic automata, we generalize these models further to more expressive model of *IMDP*s as well as a novel model of *UwMDP*s featuring parameter uncertainties. For all these models, we provide probabilistic bisimulation equivalences together with decision and minimization algorithms. In addition to bisimulation minimization techniques, we propose scalable algorithms to model check and to optimally control parametric models and apply them on a variety of case studies. Summarizing, our main contributions are classified as follows:

- We provide a thorough complexity analysis of deciding weak probabilistic bisimulation problem for probabilistic automata which in turn considers several practical algorithms and linear programming problem transformations that enable an efficient solution. We demonstrate that a small modification of the weak transition LP problem enables taking advantage of the underlying network structure to improve the practical efficiency of solving the problem. In addition, we discuss two different implementations of a probabilistic automata weak probabilistic bisimulation minimizer, one of them employing SAT modulo linear arithmetic as the solver technology. We have also investigated how compositional minimization techniques can be exploited for models consisting of several sub-automata running in parallel.
- We define the first probabilistic bisimulation for model checking of interval *MDP*s (that is also the first bisimulation for *MDP*s with uncertain transitions in general) where the two sources of nondeterminism are resolved in a *cooperative way*. Our notion of probabilistic bisimulation equivalence allows to reduce the size of such an interval *MDP* model while preserving PCTL properties it satisfies. We first show how to compute the probabilistic bisimulation equivalence by an algorithm whose core part is based on comparing polytopes of probability distributions associated with each transition. The algorithm is fixed parameter tractable with respect to the maximal dimension of the polytopes (i.e. maximal number of different states that an uncertain transition can lead to). Afterwards, we discuss computational complexity of the bisimulation minimization and show that the problem is **coNP**-complete. With the aim of designing an efficient approximation algorithm, we build a bridge between probabilistic verification and robust optimization and establish a novel modelling of the probabilistic bisimulation problem for interval *MDP*s as an instance of an uncertain LP problem. Furthermore, we show that, by using affine decision rules, the probabilistic bisimulation problem for *IMDP*s can be approximately decided in polynomial time.

- We define novel alternating probabilistic bisimulations for interval *MDP*s (which are in turn the first alternating probabilistic bisimulations for *MDP*s with uncertain transitions) where the two sources of nondeterminism are resolved in a *competitive way*. The alternating probabilistic bisimulations can be applied to reduce the size of interval *MDP*s while preserving the PCTL properties with respect to the controller synthesis or parameter synthesis semantics. We show that these alternating probabilistic bisimulations can be decided in polynomial time by formulating the core computational geometry problem as an LP problem.
- We discuss the key ingredients to build up the operations of parallel composition for composing interval *MDP* components at run-time. More precisely, we investigate how the parallel composition operator for interval *MDP*s can be defined so as to arrive at a congruence closure. As a result, we show that (alternating) probabilistic bisimulation for interval *MDP*s is a congruence with respect to two facets of parallelism, namely synchronous product and interleaving.
- We present a novel technique for multi-objective controller synthesis for *IMDP*s. Our aim is to synthesize a robust strategy that guarantees the satisfaction of a multi-objective property, despite the additional uncertainty over the transition probabilities in these models. Our approach views the uncertainty as making adversarial choices among the available transition probability distributions induced by the intervals, as the system evolves along state transitions. That is to say, we consider controller synthesis semantics in order to resolve the two sources of nondeterminisms in *IMDP*s. We first analyze the problem complexity, proving that it is **PSPACE**-hard then develop a value iteration-based decision algorithm to approximate the Pareto curve.
- We introduce *UwMDP*s as a novel stochastic model to capture quantities like preferences or priorities in a nondeterministic scenario with uncertainties. The model is very close to the model of interval *MDP*s but more convenient to model with when non-probability uncertainties like weights, preference, priority, etc. are involved. We consider the problem of computing maximal/minimal reward-bounded reachability probabilities on *UwMDP*s, for which we present an efficient algorithm running in pseudo polynomial time. In addition we show that, contrary to *IMDP*s models, *UwMDP*s are equipped with an efficient minimization theory based on probabilistic bisimulation. In particular, we define bisimulation relations for *UwMDP*s that can be decided in polynomial time in the size of the *UwMDP* models.
- We demonstrate the practical effectiveness of our proposed approaches by applying them on several case studies using a prototypical tool. A major part of our algorithms has been integrated with IscasMC which is a web-based probabilistic model checker [HLS+14]. The experimental evaluation of our proposed algorithms shows indeed their advantages in quantitative analysis of real world systems.

## 1.7 Dissertation Outline

The rest of the dissertation is organized as follows:

In **Chapter 2**, we provide the necessary mathematical background that will be used throughout the thesis. In particular, we summarize the definitions and main results from

measure and probability theory, convex polyhedra geometry as well as the theory of NP-completeness.

In **Chapter 3**, for a deeper understanding of the forthcoming chapters, we give an overview on the theory of linear programming, network flows and robust optimization. Most of the results in this section are based on the excellent textbooks ''Introduction to Linear Optimization'' by Dimitris Bertsimas and John N. Tsitsiklis [BT97], ''Network Flows: Theories, Algorithms and Applications'' [AMO93] and ''Robust Optimization'' [BTEGN09].

In **Chapter 4**, we formally introduce the probabilistic modelling formalisms and their semantics which form the basis of this dissertation. Furthermore, we introduce the notations and terminologies that are used in the later chapters.

In **Chapter 5**, we study efficiency analysis of deciding probabilistic automata weak bisimulation covering both theoretical and practical aspects. From the theoretical side, we establish an upper bound on the worst case complexity of the decision problem for general probabilistic automata. From the practical side, we present implementation considerations together with several cases studies substantiating the effectiveness of the minimization in particular for compositional analysis.

In **Chapter 6**, we address the probabilistic bisimulation for model checking PCTL properties of *IMDP*s and formally compute the computational complexity of the decision problem. Afterwards, we present algorithms to compute the probabilistic bisimulation equivalence. We next discuss compositionality methods for reasoning about *IMDP*s with respect to the cooperative semantics. Finally, we demonstrate the effectiveness of our approach on several case studies.

In **Chapter 7**, we give the definitions of alternating probabilistic bisimulation for *IMDP*s and discuss their properties. We then present a polynomial time decision algorithm to decide alternating probabilistic bisimulation for *IMDP*s followed by the compositional reasoning for these models with respect to the competitive semantics. Finally, we present effectiveness of our bisimulation minimization on several case studies.

In **Chapter 8**, we introduce multi-objective robust controller synthesis for *IMDP*s and discuss its computational time complexity. Afterwards, we present a value iteration-based algorithm to solve the problem and also discuss its extension to solve the other multi-objective queries. We finally illustrate the practical effectiveness of our proposed approaches by applying them on a couple of real world case studies.

In **Chapter 9**, we present an efficient probabilistic bisimulation minimization approach for our novel model of *UwMDP*s. We next define the notion of maximal reward-bounded reachability probability for *UwMDP*s and establish a fixed-point characterization for it. Afterwards, we present a pseudo polynomial algorithm to compute these probabilities and apply our solution approach on a variety of case studies.

In **Chapter 10**, we discuss some future research directions and conclude.

A sketch of the thesis structure is depicted in Figure 1.5 where the solid arrows describe dependencies between chapters and the dashed arrow indicates a relatively small dependency. The beginning of each chapter provides the motivation as well as the overview of

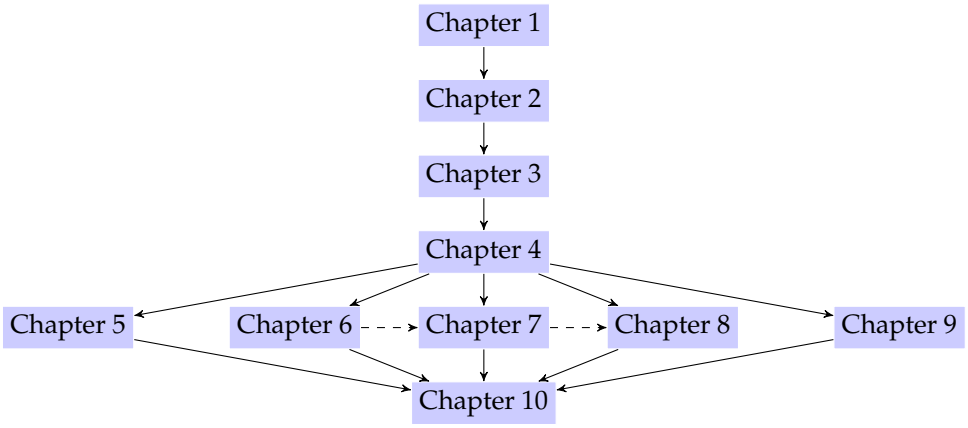



Figure 1.5: Chapters roadmap.

the content. Furthermore, we provide references to related publications at each technical chapter.

## 1.8 Origins of the Chapters and Credits

The results presented in Chapters 5 to 9 of this dissertation are based on extended version of the following publications (listed in reverse chronological order):

- [HTH+17] Hashemi, V., Turrini, A., Hahn, E.M., Hermanns, H., and Elbassioni, K., *Polynomial-Time Alternating Probabilistic Bisimulation for Interval MDPs*. In: Symposium on Dependable Software Engineering: Theories, Tools and Applications (SETTA) 2017. Lecture Notes in Computer Science, vol 10606, pp. 25-41. Springer International Publishing.
- [HHH+17b] Hahn, E.M., Hashemi, V., Hermanns, H., Lahijanian, M., and Turrini, A., *Multi-objective Robust Strategy Synthesis for Interval MDPs*. In: Quantitative Evaluation of Systems (QEST) 2017. Lecture Notes in Computer Science, vol 10503, pp. 207-223. Springer International Publishing.
- [FFHHT16] Fioriti, L.M.F., Hashemi, V., Hermanns, H. and Turrini, A., *Deciding probabilistic automata weak bisimulation: theory and practice*. In: Formal Aspects of Computing, 28(1), pp. 109-143 (2016).
- [HHHT16] Hahn, E.M., Hashemi, V., Hermanns, H. and Turrini, A., *Exploiting Robust Optimization for Interval Probabilistic Bisimulation*. In: Quantitative Evaluation of Systems (QEST) 2016. Lecture Notes in Computer Science, vol 9826, pp. 55-71. Springer International Publishing.
- [HHS+16b] Hashemi, V., Hermanns, H., Song, L., Subramani, K., Turrini, A. and Wojciechowski, P., *Compositional bisimulation minimization for interval Markov decision processes*. In: Language and Automata Theory and Applications (LATA) 2016. Lecture Notes in Computer Science, vol 9618, pp. 114-126. Springer International Publishing.

- [HHS16a] Hashemi, V., Hermanns, H. and Song, L., *Reward-bounded reachability probability for uncertain weighted MDPs*. In: Verification, Model Checking, and Abstract Interpretation (VMCAI) 2016. Lecture Notes in Computer Science, vol 9583, pp. 351-371. Springer Berlin Heidelberg.
- [HHK14] Hashemi, V., Hatefi H., and Krčál, J., *Probabilistic Bisimulations for PCTL Model Checking of Interval MDPs*. In: Synthesis of Continuous Parameters (SynCoP) 2014. EPTCS, vol 145, pp. 19-33. Electronic Proceedings in Theoretical Computer Science.
- [HHT13] Hashemi, V., Hermanns, H., and Turrini, A., *On the Efficiency of Deciding Probabilistic Automata Weak Bisimulation*. In: ECEASST, vol 66 (2013).

Further publications not included in this dissertation are

- [SBHH17] Scheftelowitsch, D., Buchholz, P., Hashemi, V., and Hermanns, H., *Multi-objective approaches to Markov decision processes with uncertain transition parameters*. In: Performance Evaluation Methodologies and Tools (ValueTools) 2017. To appear.
- [Has17] Hashemi, V., *Reformulation of the linear program for completely ergodic MDPs with average cost criteria*. In: Optimization Letters, 11(7): pages 1477-1487 (2017).
- [Has16] Hashemi, V., *Towards a combinatorial approach for undiscounted MDPs: student research abstract*. In: ACM Symposium on Applied Computing (SAC) 2016. ACM, pp. 1708-1709. ACM.
- [GHT14] Gebler, D., Hashemi, V. and Turrini, A., *Computing behavioral relations for probabilistic concurrent systems*. In: Stochastic Model Checking. Rigorous Dependability Analysis Using Model Checking Techniques for Stochastic Systems (ROCKS) 2014. Lecture Notes in Computer Science, vol 8453, pp. 117-155. Springer Berlin Heidelberg.

The compositional reasoning results in Chapter 6 are new and not published yet.

# Part I

# Background

CHAPTER 2

# Mathematical Background

This chapter establishes some of the mathematical notations and concepts that will be used throughout the dissertation. In particular, in Section 2.1 we start with essential mathematical concepts together with the notational conventions we adhere to in the remainder of this dissertation. Afterwards, we provide an abstract introduction to the essential concepts from measure and probability theory in Section 2.2. A compilation of important definitions and theorems of the theory of convex polyhedra as well as the systems of linear inequalities is provided in Sections 2.3 and 2.4, respectively. Finally, a brief description of the theory of **NP**-completeness which provides a tool to reason about the worst case time complexity of decision algorithms is discussed in Section 2.5.

## 2.1 Mathematical Preliminaries

Let us start with a recap of basic mathematical notations and notions that we will make use of in the rest of this thesis.

**Sets and Numbers** We denote the empty set by $\emptyset$. For a given set $X$, we denote its *power set* by $2^X$ which is the set of all of subsets of $X$ including $\emptyset$ and $X$ itself. Given two sets $X$ and $Y$, we denote by $X \uplus Y$ the *disjoint union* of $X$ and $Y$. We denote by $\mathbb{Z}$ the set of all integers and by $\mathbb{N}_0$ the set of non-negative integers. The set $\{1, 2, \dots\}$ of the natural numbers is denoted by $\mathbb{N}$. Similarly, we refer to the set of rational numbers, real numbers and non-negative real numbers as $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{R}_{\geq 0}$, respectively. The closed interval $[a, b]$ is defined as the set $\{ x \in \mathbb{R} \mid a \leq x \leq b \}$. We denote by $\mathbb{I}$ the set of closed sub-intervals of $[0, 1]$, i.e., $\mathbb{I} = \{ [a, b] \mid a \leq b; a, b \in [0, 1] \}$. For a given $[a, b] \in \mathbb{I}$, we denote by $\inf[a, b]$ the lower bound $a$ and by $\sup[a, b]$ the upper bound $b$. For any natural number $n \in \mathbb{N}$, we denote by $\mathbb{R}^n$ the set of all $n$-dimensional vectors of real numbers. We also denote by $\mathbf{1}_Y$ the indicator function of a given set $Y$, i.e, $\mathbf{1}_Y(y) = 1$ if $y \in Y$ and $\mathbf{1}_Y(y) = 0$ for $y \in X \setminus Y$, where $Y \subseteq X$ is an implicitly known set.

**Vectors and Norms** For a vector $\mathbf{x} \in \mathbb{R}^n$ we denote by $x_i$, its $i$-th component, and we call $\mathbf{x}$ a *weight vector* if $x_i \geq 0$ for all $i$ and $\sum_{i=1}^{n} x_i = 1$. Given $n \in \mathbb{N}$, we denote by $\mathbf{1} \in \mathbb{R}^n$ the

unit vector and by $\mathbf{e_i} \in \mathbb{R}^n$ the vector with a 1 in the $i$-th coordinate and 0's elsewhere. We denote by the superscript $^T$ the transpose of vectors or matrices. In the rest of this thesis, the comparison between vectors is element-wise and all vectors are column ones unless otherwise stated.

The Euclidean inner product $\mathbf{x} \cdot \mathbf{y}$ of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ is defined as $\sum_{i=1}^{n} x_i \cdot y_i$. For a set of vectors $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_t\} \subseteq \mathbb{R}^n$, we say that $\mathbf{s}$ is a *convex combination* of elements of $S$, if $\mathbf{s} = \sum_{i=1}^{t} w_i \cdot \mathbf{s}_i$ for some weight vector $\mathbf{w} \in \mathbb{R}^t$. We define the distance between vectors $\mathbf{x}$ and $\mathbf{y}$ in a Euclidean space $\mathbb{R}^n$ as $d(\mathbf{x}, \mathbf{y}) := \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$. Intuitively, the distance between two vectors indicates the length of the line segment connecting them.

**Functions and Relations** Let $X$ be a finite set and $f : X \to \mathbb{R}$ be a real-valued function. For $X' \subseteq X$, we denote by $f(X')$ the value $f(X') = \sum_{x \in X'} f(x)$. For a function $f : X \to \mathbb{R}_{\geq 0}$, we denote by $\mathrm{Supp}(f)$ the *support* set $\mathrm{Supp}(f) = \{ x \in X \mid f(x) > 0 \}$. Given a relation $\mathcal{R} \subseteq X \times Y$ and $x \in X$, we denote by $\mathcal{R}(x)$ the set of elements of $Y$ related to $x$, i.e., $\mathcal{R}(x) = \{ y \in Y \mid x \mathcal{R} y \}$ and we call $\mathcal{R}(x)$ the *relation set* of $x$.

Given an equivalence relation $\mathcal{R}$ on $X$, we denote by $X/\mathcal{R}$ the set of equivalence classes induced by $\mathcal{R}$ and, for $x \in X$, by $[x]_{\mathcal{R}}$ the class $\mathcal{C} \in X/\mathcal{R}$ such that $x \in \mathcal{C}$. In other words, $X/\mathcal{R} = \{[x]_{\mathcal{R}} \mid x \in X\}$ and $[x]_{\mathcal{R}} = \{ y \in X \mid y \mathcal{R} x \}$. Given a set $X$, we denote by $\mathcal{I}_X$ the identity equivalence relation $\mathcal{I}_X = \{(x, x) \mid x \in X\}$. We may drop the subscript $X$ from $\mathcal{I}_X$ when the set $X$ is clear from the context. Given two relations $\mathcal{R} \subseteq X \times Y$ and $\mathcal{S} \subseteq U \times V$, we denote by $\mathcal{R} \times \mathcal{S}$ the relation $\mathcal{R} \times \mathcal{S} = \{((x,u),(y,v)) \in (X \times Y) \times (U \times V) \mid (x,y) \in \mathcal{R}, (u,v) \in \mathcal{S}\}$. If $\mathcal{X}$ is an equivalence relation on $X$ and $\mathcal{Y}$ an equivalence relation on $Y$, then $\mathcal{X} \times \mathcal{Y}$ is an equivalence relation on $X \times Y$.

**Convex Sets and Functions** A set is called *convex* if it contains any point on the line segment between any two points in the set. Formally,

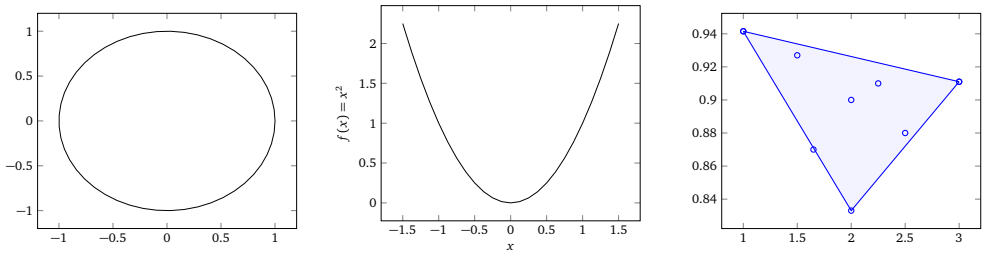
**Definition 2.1** (**Convex set** [BV04])**.** *A set $S \subseteq \mathbb{R}^n$ is convex if for all $\mathbf{s_1}, \mathbf{s_2} \in S$, and all $\alpha \in [0,1]$, it holds that $\alpha \mathbf{s_1} + (1-\alpha)\mathbf{s_2} \in S$.*

**Example 2.1.** *A half-space is defined as the set $H = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}^T \mathbf{x} \leq b\}$ where $\mathbf{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$. The boundary of a half-space is called a hyperplane. According to the Definition 2.1, half-spaces and hyperplanes are convex sets.* ♦

A real-valued function is convex if and only if any line segment which connects two points on the graph of the function lies above or on the graph. Formally,

**Definition 2.2** (**Convex function** [BV04])**.** *A real-valued function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if its domain $D$ is a convex set, and for all $\mathbf{x_1}, \mathbf{x_2} \in D$ and $\alpha \in [0,1]$, it holds that $f(\alpha \mathbf{x_1} + (1-\alpha)\mathbf{x_2}) \leq \alpha f(\mathbf{x_1}) + (1-\alpha) f(\mathbf{x_2})$.*

For a given set $S \subseteq \mathbb{R}^n$, we denote by $\mathrm{CH}(S)$ the convex hull of $S$ which is the set of all convex combinations of its points. Formally,




(a) (b) (c)

Figure 2.1: (a) Example of convex set. (b) Example of convex function. (c) Example of a convex hull.

**Definition 2.3** (**Convex hull** [BV04]). *Given a set $S = \{s_1, s_2, \ldots, s_k\} \subseteq \mathbb{R}^n$, its convex hull is defined as:*

$$\mathrm{CH}(S) = \left\{ \sum_{i=1}^{|S|} \alpha_i s_i \;\middle|\; (\forall i : \alpha_i \geq 0) \wedge \sum_{i=1}^{|S|} \alpha_i = 1 \right\}.$$

**Example 2.2.** *In Figure 2.1, a graphical representation of a convex set, a convex function and a convex hull is depicted.* ♦

One of the special properties of the convex sets which, as we will see later, plays an important role in our analysis is the notion of *extreme points* defined as follows:

**Definition 2.4** (**Extreme points** [BJS11]). *A point $\mathbf{s}$ in a convex set $S \subseteq \mathbb{R}^n$ is called an extreme point of $S$ if $\mathbf{s}$ cannot be represented as a strict convex combination of two distinct points in $S$. In other words, if $\mathbf{s} = \alpha\mathbf{s_1} + (1-\alpha)\mathbf{s_2}$ with $\alpha \in (0,1)$ and $\mathbf{s_1}, \mathbf{s_2} \in S$, then $\mathbf{s} = \mathbf{s_1} = \mathbf{s_2}$.*

*We denote by $\mathtt{Ext}(S)$ the set of extreme points of $S$.*

In the sequel we recall some other concepts associated to convex sets. For a given convex set $S$, we say that a point $\mathbf{s} \in S$ is on the boundary of $S$ denoted by $\mathbf{s} \in \partial S$ if, for every $\varepsilon > 0$ there is a point $\mathbf{y} \notin S$ such that the Euclidean distance between $\mathbf{s}$ and $\mathbf{y}$ is at most $\varepsilon$. Furthermore, we denote by $S{\downarrow}$ the *downward closure* of the convex hull of $S$ which is defined as $S{\downarrow} = \{\, \mathbf{y} \in \mathbb{R}^n \mid \mathbf{y} \leq \mathbf{z} \text{ for some convex combination } \mathbf{z} \text{ of } S \,\}$. We call a set $S$ *downward closed* if and only if $S{\downarrow} = S$.

As regards the downward-closed sets, we have the following geometric property:

**Theorem 2.1** ( [BV04, FKP12]). *Given a downward closed set $S \subseteq \mathbb{R}^n$, for any $\mathbf{z} \in \mathbb{R}^n$ such that $\mathbf{z} \notin S$, there is a weight vector $\mathbf{w} \in \mathbb{R}^n$ such that $\mathbf{w} \cdot \mathbf{z} > \mathbf{w} \cdot \mathbf{s}$ for all $\mathbf{s} \in S$. Moreover, for any $\mathbf{t} \in \mathbb{R}^n$ such that $\mathbf{t} \in \partial S$, there is a weight vector $\mathbf{w} \in \mathbb{R}^n$ such that $\mathbf{w} \cdot \mathbf{t} \geq \mathbf{w} \cdot \mathbf{s}$ for all $\mathbf{s} \in S$. We say that $\mathbf{w}$ separates $\mathbf{t}$ from $S{\downarrow}$.*

It is worthwhile to note that the above theorem is basically a direct result from the *separation theorems* for the convex sets discussed in [BV04].

Given a set $Y \subseteq \mathbb{R}^k$, we call a vector $\mathbf{y} \in Y$ *Pareto optimal* in $Y$ if there does not exist a vector $\mathbf{z} \in Y$ such that $\mathbf{y} \leq \mathbf{z}$ and $\mathbf{y} \neq \mathbf{z}$. We define the *Pareto set* or *Pareto curve* of $Y$ to be the set of all Pareto optimal vectors in $Y$, i.e., Pareto set $\mathscr{Y} = \{\mathbf{y} \in Y \mid \mathbf{y} \text{ is Pareto optimal}\}$.

**Satisfiability** Given a set $V$ of variables taking values in $\{\mathbf{true}, \mathbf{false}\}$, a *literal* $l$ is either a variable $v$ or the negation of a variable $\neg v$, where $v \in V$. A *clause* $Cl$ is a disjunction of literals. A formula $\varphi$ is written in conjunctive normal form with three variables per clause (*3-CNF*) if $\varphi = \bigwedge_{i=1}^{n} Cl_i$ where each clause $Cl_i$ is a disjunction of three literals. A formula $\varphi$ is *satisfiable* if there exists a logical value assignment for the variables that makes the formula true. Given a formula $\varphi$, we denote by $Var(\varphi)$ the set of variables occurring in $\varphi$, by $Lit(\varphi)$ the set of literals occurring in $\varphi$, by $Cl(\varphi)$ the set of clauses of $\varphi$, and, given a literal $l$, we denote by $Cl(\varphi, l)$ the set of clauses of $\varphi$ where $l$ occurs.

## 2.2 Measure and Probability

Modelling uncertainty in many complex real world systems is of great importance. Probability theory is an extremely useful tool to provide an understating for such uncertainty and to manage it. In this section, we present some of the essential notions from measure-theoretic probability that will be used later in the quantitative evaluation of probabilistic systems. For a more complete exposition the reader could consult with the standard texts on probability and measure theory such as [ADD00].

We start with the definition of $\sigma$-field:

**Definition 2.5** (**$\sigma$-field**). *Let $X$ be some set. A* $\sigma$-field *over $X$ is a set $\mathscr{F} \subseteq 2^X$ if it satisfies the following properties:*

*(a)* $X \in \mathscr{F}$,

*(b)* $A \in \mathscr{F} \Longrightarrow A^c \in \mathscr{F}$,

*(c)* $A_1, A_2, A_3, \ldots \in \mathscr{F} \Longrightarrow \cup_{i=1}^{\infty} A_i \in \mathscr{F}$.

A *measurable space* is a pair $(X, \mathscr{F})$ where $X$ is a set, also called the *sample space*, and $\mathscr{F}$ is a $\sigma$-field over $X$. A measurable space $(X, \mathscr{F})$ is called *discrete* if $\mathscr{F} = 2^X$.

**Example 2.3.** *Let $X$ be a set. According to Definition 2.5, the smallest $\sigma$-field of subsets of $X$ is the set $\mathscr{F} = \{\emptyset, X\}$ and the largest $\sigma$-field is the set $\mathscr{F} = 2^X$.* ♦

The definition of $\sigma$-field provides a persuasive link between measure and probability theory. As a matter of fact, in probability theory the set $X$ is called the *sample space* which indicates the set of all possible outcomes or results of a random experiment. Hence, probability theory is concerned with measuring the probability of events where an even is a subset of $X$ which belongs to $X$'s associated $\sigma$-field. Therefore, measurement of an event $A$ in $\mathscr{F}$ provides the probability of occurrence of the event $A$.

**Definition 2.6** (**Measure, probability and sub-probability measure**). *A* measure *over a measurable space $(X, \mathscr{F})$ is a function $\rho : \mathscr{F} \to \mathbb{R}_{\geq 0}$ such that, for each countable collection $\{X_i\}_{i \in I}$ of pairwise disjoint elements of $\mathscr{F}$, $\rho(\cup_{i \in I} X_i) = \sum_{i \in I} \rho(X_i)$.*

*A* probability measure *over a measurable space* $(X, \mathscr{F})$ *is a measure* $\rho$ *over* $(X, \mathscr{F})$ *such that* $\rho(X) = 1$. *A* sub-probability measure *over* $(X, \mathscr{F})$ *is a measure over* $(X, \mathscr{F})$ *such that* $\rho(X) \leq 1$.

A measure over a discrete measurable space $(X, 2^X)$ is called a *discrete measure* over $X$. The *support* of a measure $\rho$ over $(X, \mathscr{F})$, denoted by $\mathrm{Supp}(\rho)$, is the set $\{ x \in X \mid \rho(x) > 0 \}$. To simplify the notation, we may write $\rho(x)$ instead of $\rho(\{x\})$, for $x \in X$. We often write $\{x : \rho(x) \mid x \in \mathrm{Supp}(\rho)\}$ alternatively for a distribution $\rho$. For instance, $\{x_1 : 0.4, x_2 : 0.6\}$ denotes a distribution $\rho$ such that $\rho(x_1) = 0.4$ and $\rho(x_2) = 0.6$.

We continue by introducing some terminology. Given a set $X$, we denote by $\mathrm{Disc}(X)$ the set of discrete probability measures over $X$, and by $\mathrm{SubDisc}(X)$ the set of discrete sub-probability measures over $X$. For a given discrete sub-probability measure $\rho$ of $\mathrm{SubDisc}(X)$, we denote by $\rho(\bot)$ the value $1 - \rho(X)$. For a discrete sub-probability measure $\rho$, we also write $\rho = \{ (x, p_x) \mid x \in X \}$ where $p_x$ is the measure $\rho(x)$ of $x$. We call a discrete (sub-)probability measure $\rho \in \mathrm{SubDisc}(X)$ a *uniform* measure on a set $\emptyset \neq Y \subseteq X$, denoted by $\upsilon_Y$, if $\upsilon_Y(y) = \frac{1}{|Y|}$ for each $y \in Y$.

We call a discrete (sub-)probability measure a *Dirac* measure if it assigns measure 1 to exactly one object $x \in X$ (denote this measure by $\delta_x$), that is, $\delta_x(y) = 1$ if $y = x$, 0 otherwise. For a Dirac measure $\rho$, $\mathrm{Supp}(\rho) = \{x\}$ with $x \in X$. We also call Dirac a discrete sub-probability measure that assigns measure 0 to all objects, and we denote it by $\delta_\bot$. Given $\rho \in \mathrm{SubDisc}(X)$, we denote by $\rho \backslash z$ the *z-conditional* sub-probability measure such that $\rho \backslash z(x) = 0$ if $x = z$ and $\rho \backslash z(x) = \frac{\rho(x)}{\rho(X \setminus \{z\})}$ otherwise, provided that $\rho(X \setminus \{z\}) \neq 0$. Given $\rho_x \in \mathrm{SubDisc}(X)$ and $\rho_y \in \mathrm{SubDisc}(Y)$, we denote by $\rho_x \times \rho_y$ the sub-probability measure over $X \times Y$ defined by $\rho_x \times \rho_y(u, v) = \rho_x(u) \cdot \rho_y(v)$ for each $(u, v) \in X \times Y$. Given a finite set $I$ of indexes, a family $\{p_i \in \mathbb{R}_{>0}\}_{i \in I}$ such that $\sum_{i \in I} p_i = 1$, and a family $\{\rho_i \in \mathrm{SubDisc}(X)\}_{i \in I}$, we say that $\rho$ is the *convex combination* of $\{\rho_i\}_{i \in I}$ according to $\{p_i\}_{i \in I}$, denoted by $\sum_{i \in I} p_i \cdot \rho_i$, if, for each $x \in X$, $\rho(x) = \sum_{i \in I} p_i \cdot \rho_i(x)$.

With these definitions, we are ready to lift relations between sets to relations between distributions. The lifted relations which play a crucial role in computing behavioral relations for probabilistic systems are formally defined as follows:

**Definition 2.7** (**Lifted relations** [JL91])**.** *The lifting of a relation* $\mathcal{R} \subseteq X \times Y$ *to a relation* $\mathcal{L}(\mathcal{R}) \subseteq \mathrm{Disc}(X) \times \mathrm{Disc}(Y)$ *is defined as follows: for* $\rho_X \in \mathrm{Disc}(X)$ *and* $\rho_Y \in \mathrm{Disc}(Y)$, $\rho_X \; \mathcal{L}(\mathcal{R}) \; \rho_Y$ *holds if there exists a* weighting function $\omega : X \times Y \to [0, 1]$ *such that*

- $\omega(x, y) > 0$ *implies* $x \; \mathcal{R} \; y$,
- $\sum_{y \in Y} \omega(x, y) = \rho_X(x)$, *and*
- $\sum_{x \in X} \omega(x, y) = \rho_Y(y)$.

*When* $\mathcal{R}$ *is an equivalence relation on a set* $X$, $\rho_1 \; \mathcal{L}(\mathcal{R}) \; \rho_2$ *holds if, for each* $\mathcal{C} \in X/\mathcal{R}$, $\rho_1(\mathcal{C}) = \rho_2(\mathcal{C})$.

When particularly $\mathcal{R} = \mathcal{I}$, $\rho_1 \; \mathcal{L}(\mathcal{I}) \; \rho_2$ holds if and only if $\rho_1 = \rho_2$. This property can be generalized to: if $\mathcal{R} \cap \mathrm{Supp}(\rho_1) \times \mathrm{Supp}(\rho_2) \subseteq \mathcal{I}$, then $\rho_1 \; \mathcal{L}(\mathcal{R}) \; \rho_2$ holds if and only if $\rho_1 = \rho_2$. By abuse of notation, we extend $\mathcal{L}(\mathcal{R})$ to distributions over $X/\mathcal{R}$, i.e., for $\rho_1, \rho_2 \in \mathrm{Disc}(X/\mathcal{R})$, we write $\rho_1 \; \mathcal{L}(\mathcal{R}) \; \rho_2$ if for each $\mathcal{C} \in X/\mathcal{R}$, it holds that $\rho_1(\mathcal{C}) = \rho_2(\mathcal{C})$.

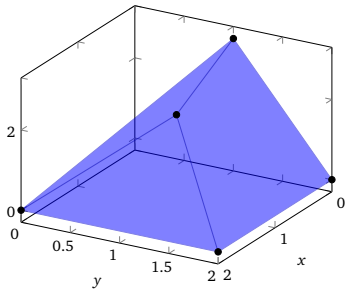



Figure 2.2: An example of a polytope.

## 2.3 Convex Polyhedra

Convex polyhedral are useful data structures to describe the feasible region of states transition probability in probabilistic systems in which the probability distributions are not known completely. They also form the cornerstone of various systems analysis such as [Fre05, Hag14, HHK14]. In this section, we present some important definitions and results from the theory of convex polyhedral. We refer the reader to [Zie95] for a more complete treatment of this topic.

A polyhedron is specified by the solution set of a finite number of linear inequalities with real coefficients. A bounded polyhedron is called *polytope*. There are basically two different representations of convex polyhedral introduced in the literature: the $\mathcal{H}$-*representation*, where the polyhedron is presented as intersection of finitely many half-spaces and the $\mathcal{V}$-*representation*, where it is presented as convex hull of finitely many points. Formally,

**Definition 2.8** (**$\mathcal{H}$- and $\mathcal{V}$-polytopes**). *An $\mathcal{H}$-polyhedron in Euclidean space $\mathbb{R}^n$ is any subset $P = \cap_{i=1}^{n} H_i$ of $\mathbb{R}^n$ defined as the intersection of a finite number of closed half-spaces $H_i$; a bounded polyhedron is called an $\mathcal{H}$-polytope. A $\mathcal{V}$-polytope is the convex hull $P = \mathrm{CH}(S)$ of a finite set of points $S \subseteq \mathbb{R}^n$.*

It is not difficult to see that all $\mathcal{V}$-polytopes are bounded. The main theorem of polytope theory formalizes the equivalence of $\mathcal{H}$-polytopes and $\mathcal{V}$-polytopes. More precisely,

**Theorem 2.2** (**Equivalence of $\mathcal{H}$-polytopes and $\mathcal{V}$-polytopes** [Zie95, GOR00]). *Every $\mathcal{H}$-polytope can be obtained as the convex hull of its finitely many extreme points; thus every $\mathcal{H}$-polytope is a $\mathcal{V}$-polytope. Conversely, every $\mathcal{V}$-polytope has a description by a finite system of inequalities; thus every $\mathcal{V}$-polytope is an $\mathcal{H}$-polytope.*

**Example 2.4.** *An example of an $\mathcal{H}$-polytope $P$ is depicted in Figure 2.2. It can be alternatively seen as a $\mathcal{V}$-polytope which is the convex hull of its extreme points, i.e., $P = \mathrm{CH}(\{(0,2,0),(2,2,0),(0,1,3),(2,0,0),(1,1,2)\})$.* ♦

**Remark 2.1.** *It is worthwhile to note that the translations between the $\mathcal{H}$-representation and $\mathcal{V}$-representation of polytopes in an $n$-dimensional Euclidean space can be exponential in the space dimension $n$.*

One of the polyhedral operations which forms a basis in probabilistic systems analysis is the projection problem. In principal, the projection problem can have different variants depending on the representation of the input polytope and the favorable representation of the projected polytope [Tiw08]. In our setting, if $P$ is a polytope in $\mathbb{R}^n$ then for each $i \in \{1,\ldots,n\}$, we define the projection $\text{proj}_{\mathbf{e}_i} P$ of $P$ as the interval $[\min_i P, \max_i P]$ where $\min_i P = \min\{ x_i \mid (x_1,\ldots,x_i,\ldots,x_n) \in P \}$ and $\max_i P = \max\{ x_i \mid (x_1,\ldots,x_i,\ldots,x_n) \in P \}$. We also define

$$\Delta_n = \{ (x_1,\ldots,x_n) \in \mathbb{R}^n_{\geq 0} \mid \sum_{i=1}^{n} x_i = 1 \}$$

as the standard simplex in $\mathbb{R}^n_{\geq 0}$.

## 2.4 Systems of Linear Inequalities

As we have discussed in Section 2.3, a convex polyhedron captures the set of feasible solutions to a system of linear inequalities. Apart from convexity, many properties of polyhedra and, as we will see in the next chapter linear programming, rely on characterizing if a system of linear inequalities has a solution. In this section we discuss the well-known *Fourier-Motzkin (FM) elimination method* to solve such systems and afterwards present the Farkas's lemma.

The FM elimination method consists of successive elimination of variables from a system of linear inequalities. More precisely, the main idea of the FM elimination is to partition all inequalities relevant to $y$ into two sets: $\{\sum_{1\leq j\leq n} e_{ij}x_j \leq y\}_{1\leq i\leq m_1}$ and $\{\sum_{1\leq j\leq n} e_{ij}x_j \geq y\}_{m_1<i\leq m}$, where $e_{ij}$ $(1 \leq i \leq m,\ 1 \leq j \leq n)$ are coefficients and $y$ is the variable to be eliminated. The resulting set of inequalities will contain those in form of $\sum_{1\leq j\leq n} e_{ij}x_j \leq \sum_{1\leq j\leq n} e_{i'j}x_j$ for each $1 \leq i \leq m_1$ and $m_1 < i' \leq m$, which defines a projection of the original $\mathcal{H}$-polytope on variables $\{x_j\}_{1\leq j\leq n} \cup \{y\}$ to an $\mathcal{H}$-polytope on $\{x_j\}_{1\leq j\leq n}$.

For a given system of linear inequalities, this successive elimination procedure yields a system of linear inequalities over constants whose satisfiability can be instantly verified. If the system is satisfiable then a solution to the original linear system can be found from the successively generated systems of linear inequalities using a backward substitution. The FM method can also be applied in order to find the projection of a polyhedron into a subspace.

As regards time complexity, the FM elimination causes an exponential blow-up and results in $4(\frac{m}{4})^{2^d}$ inequalities in the worst case, where $m$ is the number of inequalities in the original representation and $d$ the number of variables having been eliminated [Sch98].

The FM elimination method provides an arithmetic proof for a well-known theoretical result, i.e., Farkas's lemma. Farkas's lemma (aka Farkas' alternative theorem) provides necessary and sufficient conditions for linear systems of inequalities to have a solution. Formally,

**Lemma 2.1.** *(**Farkas's Lemma** [Sch98]) Let $A$ be a real $m \times n$ matrix and $\mathbf{b}$ an $m$-dimensional real vector. Then, exactly one of the following two statements is true:*

- *There exists an $\mathbf{x} \in \mathbb{R}^n$ such that $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq 0$.*
- *There exists a $\mathbf{y} \in \mathbb{R}^m$ such that $\mathbf{y}^T A \geq 0$ and $\mathbf{y}^T \mathbf{b} < 0$.*

There are several slightly different (but equivalent) variants of Farkas's lemma in the literature. For more details, we refer the reader to [Sch98, BJS11].

## 2.5 The NP-Completeness Theory

When we come across a computational problem, care should be taken with respect to the amount of resources such as time, space and communication required to solve the problem. The theory of computation is concerned with the classification of computational problems according to the amount of resources their solutions need. Additionally, it also tries to understand the relation between those classes in a more general sense. This section reviews briefly the main concepts of complexity theory we need throughout this thesis. We refer the reader to textbooks on complexity theory such as [AB09, Tre02] for a complete treatment of the topic.

From computational complexity viewpoint, the computational problems are distinguished in two different categories, the *tractable* problems that can be solved in polynomial time, and the *intractable* problems for which there is no efficient algorithm to solve them.

A computational problem that has a yes or no answer is called a *decision problem*. We shall denote by **P** the class of all decision problems which can be solved in polynomial time. We shall also denote by **NP** the class of all decision problems for which a candidate solution can be verified in polynomial time. The class **NP** stands for *non-deterministic polynomial time* as every decision problem in this class can be solved in polynomial time on a non-deterministic Turing machine. Obviously, $\mathbf{P} \subseteq \mathbf{NP}$. However, it remains an open problem to determine if $\mathbf{P} = \mathbf{NP}$. A decision problem $A$ is *polynomially reducible* to a decision problem $B$ if problem $A$ can be solved in polynomial time given a polynomial time algorithm for problem $B$ [Pap03]. A problem is **NP**-hard if all the problems in **NP** are polynomially reducible to it. Furthermore, a problem is **NP**-complete if it is an **NP**-hard problem and also belongs to the class **NP**. Therefore, **NP**-complete problems are indeed the hardest problems in the class **NP**.

**Example 2.5.** *A Boolean formula is in 3SAT if it in 3CNF form and is also satisfiable. 3SAT is* ***NP****-complete [GJ90].* ⧫

The complements of the problems which belong to the class **NP** are classified as the class **coNP**. In other words, the class **coNP** is the class of all decision problems whose *no* answers can be verified in polynomial time. Similarly to the **NP**-complete, a decision problem is **coNP**-complete if it is in **coNP** and if every problem in **coNP** is polynomial-time reducible to it. In fact, **coNP**-complete problems are the hardest problems in **coNP**.

**Example 2.6.** *A Boolean formula is in* tautology *if it is satisfied by every assignment. Tautology is* ***coNP****-complete [GJ90].* ⧫

The theory of **NP**-completeness is of great importance in theoretical computer science. In particular, if a computational problem falls into this class, then it does likely not admit an efficient (or polynomial time) algorithm. Therefore, a solution approach should primarily search for efficient approximations or heuristics algorithms [Vaz04].

The complexity class moved beyond **NP**-complete is the class **EXPTIME** which is defined as the set of decision problems that can be decided in exponential time $\mathcal{O}(2^{p(n)})$ where

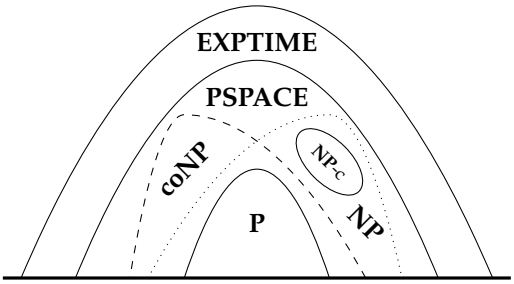



Figure 2.3: Comparison of the complexity classes.

$p(n)$ is a polynomial function of $n$. We also define **EXPTIME**-complete using the similar idea as the class of all problems that are in **EXPTIME** and all other problems in **EXPTIME** can be reduced to them in polynomial time. It is shown that problems in the class of **EXPTIME**-complete cannot be solved in polynomial time, using the time hierarchy theorem [HS65].

**Example 2.7.** *Computing a perfect strategy for an $n\times$ $n$ chess game is **EXPTIME**-complete [FL81].* ⧫

So far, we have argued the complexity of solving the decision problems based on the resource *time*. An alternative approach is to discuss the complexity of solving the problems within a given amount of *space* or *memory*. In particular, the class of **PSPACE** contains the set of all problems that can be solved in polynomial space. Accordingly, a problem is **PSPACE**-complete if it is in **PSPACE** and if every other problem in **PSPACE** can be transformed to it in polynomial time.

**Example 2.8.** *An instance of Quantified Boolean formula (QBF) is given as $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n \varphi$ where $Q$ is $\forall$ if $n$ is even and $\exists$ if $n$ is odd and $\varphi$ is a Boolean expression. The satisfiability problem for QBF is **PSPACE**-complete [SM73].* ⧫

We provide a representation of the relation between complexity classes discussed in this section in Figure 2.3.

CHAPTER 3

# Basics of Mathematical Optimization

The focus of this dissertation is on the analysis of discrete-time probabilistic systems where the time is discrete, i.e., the system execution is modelled as a sequence of discrete time steps. We consider systems where the transition probabilities are either fixed or not known with precision but reside in a so-called uncertainty set. In the later chapters, we reason about efficiency of deciding if two probabilistic systems are similar or equivalent or we analyze quantitative properties of systems such as the maximum probability of reaching a target set within a certain time bound.

In this context, there is a need for efficient and scalable algorithms to decide quantitative properties of interest. Examples include the efficiency of deciding if two probabilistic systems have the same behaviour or of verifying whether the system satisfies a set of properties. These decision problems are even harder to analyze when the system is uncertain. In such cases, the analysis shall be robust against all possible adversarial resolutions of the uncertainty in the state-transition probabilities.

However, we pay for this greater efficiency by more complex analytic tools: in our setting, we resort to modern mathematical optimization techniques to model and solve complex optimization problems arising in modelling and analyzing of probabilistic systems. Therefore, this chapter provides an overview of the optimization techniques which are used throughout the dissertation.

In Section 3.1, we provide an overview on the theory of linear programming including formulation, duality theory and a brief survey of the algorithms. Most of the results in this section are taken from the textbooks "Introduction to Linear Optimization" [BT97] and "Linear and Nonlinear Programming" [LY84]. A brief introduction on network flows is provided in Section 3.2. In particular, we discuss maximum flow problem as one of the fundamental problems in combinatorial optimization which will be significantly used later in our analysis. The results in this section are based on the textbook "Network Flows: Theory, Algorithms, and Applications" [AMO93]. Finally, Section 3.3 explains basics of robust optimization. In this section, we focus on uncertain linear programs and shortly describe their corresponding robust counterparts. The materials in this section are based on the textbook "Robust Optimization" [BTEGN09] and also [BTN99, BTGGN04].

## 3.1 Linear Programming

Linear programming (LP) is a powerful and robust algorithmic tool which deals with finding the best possible solution in a mathematical model representing a linear relationship among resource requirements. Many real-world problems which aim to find the best solution in assigning limited resources such as energy, space and time to achieve maximum profit or minimum cost can be formulated in this framework.

Formally, an LP problem refers to the problem of optimizing a linear *objective function* of several variables subject to a given set of alternatives specified by linear equality or inequality constraints. Every LP problem can be formulated in the following *canonical* form:

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to:} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned}$$

In the above formulation, a linear objective function (or cost function) $\mathbf{c}^T\mathbf{x}$ for $\mathbf{c} \in \mathbb{R}^n$ is maximized subject to linear inequalities $A\mathbf{x} \leq \mathbf{b}$ for $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$. We often call the inequality $\mathbf{x} \geq 0$ as *sign* constraint. The inequality $A\mathbf{x} \leq \mathbf{b}$ is componentwise; if $a_i$ is the $i$-th row of $A$ and $b_i$ is the $i$-th element of the vector $\mathbf{b}$, then the inequality is satisfied if $a_i\mathbf{x} \leq b_i$ for $i \in \{1, \ldots, n\}$. A constraint of the form $a_i\mathbf{x} \leq b_i$, $a_i\mathbf{x} = b_i$ or $a_i\mathbf{x} \geq b_i$ in an LP is said to be *tight* (or active) for a certain point $\mathbf{y}$, if $a_i\mathbf{y} = b_i$. Any decision vector $\mathbf{x}$ for which the inequality $A\mathbf{x} \leq \mathbf{b}$ holds true is called *feasible* (or a feasible solution). The set $P = \{\mathbf{x} \in \mathbb{R}^n_{\geq 0} : A\mathbf{x} \leq \mathbf{b}\}$ of all points satisfying the set of constraints is called the feasible region or feasible set. An LP problem is said to be feasible if the feasible set is not empty; otherwise it is said to be infeasible. For any feasible solution $\mathbf{x}$, $\mathbf{c}^T\mathbf{x}$ is the *value* of $\mathbf{x}$. An optimal solution $\mathbf{x}^*$ of the LP is a feasible solution $\mathbf{x}$ which attains the maximum value (if it exists) among all feasible solutions and the optimal value of the linear program is $\mathbf{c}^T\mathbf{x}^*$ for optimal $\mathbf{x}^*$. On the other hand, an LP is unbounded above (or its optimal value is $+\infty$) if for every real number $K$ we can find a feasible solution $\mathbf{x}$ whose value is greater than $K$. We sometimes will abuse terminology and say that the problem is unbounded. We finally note that an LP is said to be in *standard* form if the linear inequalities $A\mathbf{x} \leq \mathbf{b}$ are of the form $A\mathbf{x} = \mathbf{b}$. The standard form of an LP is not restrictive and basically every general LP problem can be transformed into an equivalent problem in standard form.

**Example 3.1.** *Consider the following linear program:*

$$\begin{aligned} \max \quad & -2x_1 + 7x_2 \\ \textit{subject to:} \quad & -3x_1 + x_2 \leq 1 \\ & -4x_1 + 9x_2 \leq 3 \\ & 6x_1 - 11x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned}$$

*In a succinct reformulation, the LP is* $\max\{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0\}$ *where*

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, A = \begin{pmatrix} -3 & 1 \\ -4 & 9 \\ 6 & -11 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix}, \textit{ and } \mathbf{c} = \begin{pmatrix} -2 \\ 7 \end{pmatrix}$$

*The unique optimal solution of this LP is* $\mathbf{x}^* = (\frac{21}{5}, \frac{11}{5})$ *which has value* $-2 \cdot \frac{21}{5} + 7 \cdot \frac{11}{5} = 7$. ♦

The feasible set of any LP can be described by inequality constraints of the form $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$; therefore, it is a polyhedron. On the other hand, if the problem has at least one optimal solution, then an optimal solution can be found among the corners of the feasible set.

**Definition 3.1** (**Basic feasible solutions**). *Consider a polyhedron $P$ defined by linear equality and inequality constraints, and let $\mathbf{x}$ be an element of $\mathbb{R}^n$.*

*(a) The vector $\mathbf{x}$ is a* basic solution *if:*

*i. All the equality constraints defining $P$ are active at $\mathbf{x}$;*

*ii. Of all the constraints that are active at that vector, at least $n$ of them must be linearly independent.*

*(b) If $\mathbf{x}$ is a basic solution that satisfies all of the constraints, we say that it is a basic feasible solution.*

In the sequel, we provide an algebraic definition of a corner point as a basic feasible solution. Formally,

**Theorem 3.1.** *Let $P$ be a nonempty polyhedron and let $\mathbf{x} \in P$ Then, the following are equivalent:*

*(a) $\mathbf{x}$ is a vertex;*

*(b) $\mathbf{x}$ is an extreme point;*

*(c) $\mathbf{x}$ is a basic feasible solution.*

As a result of Theorem 3.1, for an LP the vertices (i.e., extreme points) of its feasible region are precisely its basic feasible solutions. The observations so far provides us a way to characterize the optimal solution of an LP. Formally,

**Theorem 3.2.** *Consider the linear programming problem of maximizing $\mathbf{c}^T\mathbf{x}$ over a polyhedron $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} \leq \mathbf{b}\}$. Suppose that $P$ has at least one extreme point. Then, either the optimal value is equal to $+\infty$, or there exists an extreme point which is optimal.*

### 3.1.1 Duality Theory

The concept of duality plays an important rule in the theory of LP. Duality acts as a unifying theory which associates to every LP problem another LP problem that can be derived from it. The original LP problem is called *primal* while the derived LP is called *dual*. The dual form of a primal LP is informally constructed by swapping the role of variables and constraints between primal and dual problem: for each constraint in the primal LP problem (other than the sign constraints) , we introduce a variable in the dual problem; for each variable in the primal, we introduce a constraint in the dual.

A one-to-one correspondence between the constraint type and the variable type for both the primal and the dual problems is summarized in Table 3.1.

| PRIMAL | Minimize | Maximize | DUAL |
|---|---|---|---|
| Constraints | $\geq b_i$ | $\geq 0$ | Variables |
| | $\leq b_i$ | $\leq 0$ | |
| | $= b_i$ | free | |
| Variables | $\geq 0$ | $\leq c_j$ | Constraints |
| | $\leq 0$ | $\geq c_j$ | |
| | free | $= c_j$ | |

Table 3.1: Relation between primal and dual variables and constraints.

**Example 3.2.** *Consider the following primal LP:*

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \textit{subject to:} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned}$$

*After applying the rules in Table 3.1, the dual LP is derived as:*

$$\begin{aligned} \min \quad & \mathbf{b}^T\mathbf{y} \\ \textit{subject to:} \quad & A^T\mathbf{y} \geq \mathbf{c} \\ & \mathbf{y} \geq 0 \end{aligned}$$

*where* $A \in \mathbb{R}^{m\times n}$ *and* $\mathbf{y} \in \mathbb{R}^m$. ♦

From the definition of dual, it immediately follows that the dual of the dual is the primal itself. Additionally, there are two fundamental principals in the duality theory which provide an intuition behind the relation between primal and dual problems. The first principle provides a relation between the optimal solution of primal and dual LPs and is known as *weak duality* theorem. Formally,

**Theorem 3.3 (Weak duality theorem).** *Given a pair of primal and dual LPs,*

$$(\mathscr{P}) \quad \begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \textit{subject to:} \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned} \qquad (\mathscr{D}) \quad \begin{aligned} \min \quad & \mathbf{b}^T\mathbf{y} \\ \textit{subject to:} \quad & A^T\mathbf{y} \geq \mathbf{c} \\ & \mathbf{y} \geq 0 \end{aligned}$$

*If* $\mathbf{x}$ *is a feasible solution for* $\mathscr{P}$ *and* $\mathbf{y}$ *is a feasible solution for* $\mathscr{D}$*, then* $\mathbf{c}^T\mathbf{x} \leq \mathbf{b}^T\mathbf{y}$.

An immediate implication of the weak duality theorem is the following.

**Corollary 3.1.**

*(i) (Certificate of Optimality) If* $\mathbf{x}$ *and* $\mathbf{y}$ *are feasible solutions of the primal and dual and* $\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{y}$*, then* $\mathbf{x}$ *and* $\mathbf{y}$ *must be optimal solutions to the primal and dual.*

*(ii) (Infiniteness and Feasibility in Duality) If the optimal value in the primal is* $+\infty$*, then the dual must be infeasible. If the optimal value in the dual is* $-\infty$*, then the primal must be infeasible.*

The next theorem provides a stronger result than the weak duality and is basically the central result on LP duality.

**Theorem 3.4** (**Strong duality theorem**). *If the primal problem has an optimal solution, so does the dual, and the respective optimal values are equal.*

The weak duality and strong duality theorems can be also characterized by a notion of *duality gap*. Formally,

**Definition 3.2** (**Duality gap**). *Given a pair of primal and dual LPs,*

$$
\begin{array}{rlll}
 & \max & \mathbf{c}^T\mathbf{x} & \\
(\mathscr{P}) & \textit{subject to:} & A\mathbf{x} \leq \mathbf{b} & \\
 & & \mathbf{x} \geq 0 &
\end{array}
\qquad
\begin{array}{rlll}
 & \min & \mathbf{b}^T\mathbf{y} & \\
(\mathscr{D}) & \textit{subject to:} & A^T\mathbf{y} \geq \mathbf{c} & \\
 & & \mathbf{y} \geq 0 &
\end{array}
$$

*if $p^*$ is the primal optimal value and $d^*$ is the dual optimal value, then the duality gap is equal to $d^* - p^*$.*

It is immediate to see that as a result of the weak duality theorem, the duality gap is always nonnegative. Furthermore, strong duality holds if and only if the duality gap is equal to zero.

For a given pair of primal and dual LPs, it is useful to know *optimality conditions* that must be satisfied by optimal solutions of primal and dual problems. These conditions which are known as *complementary slackness* conditions provide a way to compute the dual optimal solution from the primal optimal solution. Formally,

**Theorem 3.5** (**Complementary slackness conditions**). *Let $\mathbf{x}$ be a feasible solution to the primal $(\mathscr{P})$ and $\mathbf{y}$ be a feasible solution to the dual $(\mathscr{D})$ where*

$$
\begin{array}{rlll}
 & \max & \mathbf{c}^T\mathbf{x} & \\
(\mathscr{P}) & \textit{subject to:} & A\mathbf{x} \leq \mathbf{b} & \\
 & & \mathbf{x} \geq 0 &
\end{array}
\qquad
\begin{array}{rlll}
 & \min & \mathbf{b}^T\mathbf{y} & \\
(\mathscr{D}) & \textit{subject to:} & A^T\mathbf{y} \geq \mathbf{c} & \\
 & & \mathbf{y} \geq 0. &
\end{array}
$$

*Then $\mathbf{x}, \mathbf{y}$ are optimal solutions for their respective problems if and only if:*

- $(b_i - \sum_{j=1}^{n} a_{ij}x_j)y_i = 0 \quad$ *for* $i = 1, \ldots, m$
- $(\sum_{i=1}^{m} a_{ji}y_i - c_j)x_j = 0 \quad$ *for* $j = 1, \ldots, n$

**Example 3.3.** *Consider the following primal LP:*

$$
\begin{array}{rll}
 & \max & 2x_1 + 5x_2 \\
(\mathscr{P}) & \textit{subject to:} & x_1 + 4x_2 \leq 3 \\
 & & 4x_1 + 7x_2 \leq 11 \\
 & & x_1, x_2 \geq 0
\end{array}
$$

*The optimal solution of this LP is $\mathbf{x}^* = (\frac{23}{9}, \frac{1}{9})$ which implies the optimal value $v(\mathscr{P}) = \frac{17}{3}$. The*

*dual problem associated with the primal is:*

$$(\mathscr{D}) \quad \begin{array}{rl} \min & 3y_1 + 11y_2 \\ \textit{subject to:} & y_1 + 4y_2 \geq 2 \\ & 4y_1 + 7y_2 \geq 5 \\ & y_1, y_2 \geq 0 \end{array}$$

*The complementary slackness conditions are derived as follows:*

$$\begin{array}{c} y_1(3 - x_1 - 4x_2) = 0 \\ y_2(11 - 4x_1 - 7x_2) = 0 \\ x_1(y_1 + 4y_2 - 2) = 0 \\ x_2(4y_1 + 7y_2 - 5) = 0. \end{array}$$

*From the optimal solution of the primal we know that* $x_1 = \frac{23}{9}$ *and* $x_2 = \frac{1}{9}$*. If we replace these values of* $x_1$ *and* $x_2$ *in the third and fourth equations of the above linear system, we end up with a linear system whose solution is the dual optimal solution* $\mathbf{y}^* = (\frac{2}{3}, \frac{1}{3})$ *with objective value* $v(\mathscr{D}) = \frac{17}{3}$*. As expected, the primal and dual optimal values are the same and therefore, strong duality theorem is satisfied.* ♦

### 3.1.2 Solution Methods for LPs

Several approaches have been developed to solve LP problems. In this section, we very briefly overview some of these methods. We refer the reader to [LY84] and references therein for a detailed treatment.

*Fourier-Motzkin elimination*, also known as FME method is one of the earliest approaches to solve a linear program. The method is based essentially on the basis of dimensionality reduction: in each iteration the dimension of an LP reduces by one without changing feasibility. In fact, the FME method solves an LP by transforming it to successive feasibility checking. The worst case computational complexity of the method is exponential and it gave way to the *simplex method*.

The simplex method provides an algorithmic procedure to solve LPs by testing adjacent vertices of the feasible set. More precisely, it finds a vertex of the feasible set (polyhedron) of an LP and successively moves to a new neighboring vertex which possibly improves the objective function. Although the simplex method is very efficient in practice, its worst case computational complexity is exponential [KM72].

The first worst-case polynomial time algorithm for solving LPs is the Khachiyan's *ellipsoid method* [Kha79] and it can also be applied to solve convex optimization problems, a special class of optimization problems which study the problem of minimizing convex functions over convex sets. Even though the ellipsoid method has a polynomial time complexity, its performance is poor in practice and it suffers from numerical instability and slow convergence.

Later on, Karmarkar's interior-point method [Kar84] was introduced with a polynomial time worst case complexity and behaves faster in practice than the ellipsoid method. The algorithm starts with finding a point inside the feasible set and then moves through the interior of the feasible set while improving the approximations of the optimal solution. Karmarkar's interior-point algorithm was proved also to be faster than Khachiyan's ellipsoid method in the worst case.

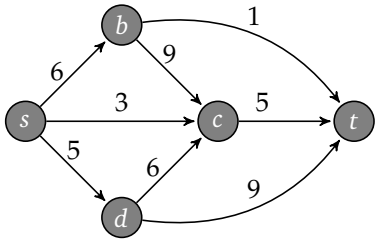



Figure 3.1: A flow network with source node $s$ and sink node $t$. The numbers next to the edges are the capacities.

## 3.2 Network Flows

The efficiency of algorithms to solve linear programming problems depends significantly on the inherent special structures of the problems such as sparsity. Sometimes linear programming problems modelling real world applications have a combinatorial structure meaning that they are formulated over networks or mastoids [Law76]. In such cases the underlying combinatorial structures can be exploited in order to speed up the overall efficiency of decision algorithms.

Combinatorial optimization deals with optimizing an objective function over a finite set of alternatives which are in turn described through mathematical structures. Network flows is a subclass of combinatorial optimization with non-trivial applications to network reliability, distributed computing, airline scheduling as well as other domains. The network flow problems can be treated in a purely combinatorial term, i.e., they can be solved using decision algorithms performing operations directly on the network itself [BJS11] and therefore, most of them admit more efficient algorithms [GTT89].

A flow network is a directed graph $G = (V, E)$ defined by a set $V$ of $|V|$ nodes and a set $E$ of $|E|$ arcs. Two nodes of the graph are distinguished as the *source* node $s$ and the *sink* node $t$. The source node $s$ has no entering arcs and the sink node has no leaving arcs. The *capacity* of each arc $(u, v) \in E$ is a mapping $c : E \to \mathbb{R}_{\geq 0}$, denoted by $c(u, v)$ or $c_{uv}$.

**Definition 3.3 (Flow).** *Given a flow network $G = (V, E)$, a* flow *of the network is a mapping $f : E \to \mathbb{R}_{\geq 0}$, denoted by $f_{uv}$ or $f(u, v)$, satisfying the following constraints:*

- Capacity constraint: *for all $(u, v) \in E$, $f_{uv} \leq c_{uv}$.*
- Flow conservation: *for all $u \in V \setminus \{s, t\}$, $\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}$.*

*The* value of the flow $f$ *is $\sum_{u \in V} f_{su}$ which is the amount of flow leaving the source node $s$.*

One of the classical problems in network flows is the *maximum flow problem* which asks to compute the flow of maximum possible value in a flow network. As we shall see later, this problem plays a significant role in our analysis to compute or improve the worst case time complexity of decision algorithms for probabilistic systems.

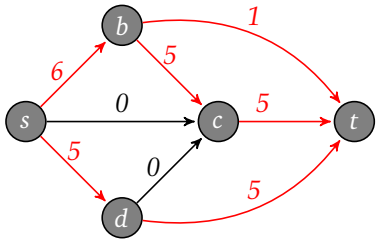
The maximum flow problem can be formulated as an LP as follows:

**Definition 3.4 (The maximum flow problem).** *Given a flow network $G = (V, E)$ with arc capacities $c : E \to \mathbb{R}_{\geq 0}$, source node $s$ and sink node $t$, the maximum flow problem is modeled as*

*the following linear program:*

$$\begin{array}{rll} \max & \sum_{v:(s,v)\in E} f(s,v) & \\ \textit{subject to:} & \sum_{u:(u,v)\in E} f_{u,v} - \sum_{w:(v,w)\in E} f_{v,w} = 0 & \forall v \in V \setminus \{s,t\} \\ & f_{u,v} \leq c_{u,v} & \forall (u,v) \in E \\ & f_{u,v} \geq 0 & \forall (u,v) \in E \end{array}$$

**Example 3.4.** *Consider again the flow network depicted in Figure 3.1. After solving the LP for the maximum flow problem, the value of the maximum flow is 11 and the optimal flow is depicted as follows:*




♦

As it is clear from the example, the value of maximum flow for the network in Figure 3.1 which consists of only integer arc capacities is an integer value. This observation is indeed correct in general. More precisely, as a result of the *flow integrality theorem* [AMO93], if the arc capacities of a flow network are integers, then there is always a maximum flow that is integer-valued. Moreover, from a computational point of view, since the maximum flow problem is formulated as an LP, it can be solved through any standard LP solver such as the simplex algorithm. However, as discussed earlier, the inherent combinatorial structure of the problem enables much faster combinatorial algorithms. Apart from theoretical efficiency, these combinatorial algorithms usually run many times faster than the general-purpose LP solvers [AMO93, Chi06].

## 3.3 Robust Optimization

Robust optimization is a new approach in mathematical optimization that is concerned with optimization problems in which a certain level of robustness is desirable against *uncertainty* [BTEGN09, BTGGN04]. This approach has been shown to be very useful in real-world applications that are entirely or to a certain extent affected by uncertainty [BTN99, BBC11, Lof12]. Moreover, this modelling methodology is integrated with computational tools to treat optimization problems with uncertain data that is only known to be included in some uncertainty set. For instance, several tools such as PICOS [PIC], ROPI [Goe14] and YALMIP [Lof04] have been developed and applied successfully in practice.

In this section, we introduce the concept of uncertain linear programming problems and afterwards, we provide an overview of the essential background required for the rest

of the dissertation. We refer the reader to [BTEGN09, BBC11] for the comprehensive references on robust optimization.

### 3.3.1 Uncertain Linear Programming (ULPs)

As we discussed earlier in Section 3.1, linear programming problems can be described in the general succinct form as:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \left\{ \mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b} \right\}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the vector of *decision variables*, $\mathbf{c} \in \mathbb{R}^n$ is the vector of *coefficients*, $A \in \mathbb{R}^{m\times n}$ is the constant *coefficient matrix* and $\mathbf{b} \in \mathbb{R}^m$ is the *right hand side* vector.

The data of an LP problem, i.e., the collection of tuples $[\mathbf{c}, A, \mathbf{b}]$, are often not known precisely when the LP encodes a real-world problem. This issue reveals the need for an approach to produce LP solutions which are immune against uncertainty.

**Definition 3.5** (**Uncertain linear programs** [BTEGN09, BTGGN04]). *An Uncertain Linear Program (ULP) is a family*

$$\left\{ \min_{\mathbf{x}\in\mathbb{R}^n} \{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\} \right\}_{[\mathbf{c},A,\mathbf{b}]\in\mathscr{Z}} \tag{3.1}$$

*of LP problems* $\min_{\mathbf{x}\in\mathbb{R}^n}\{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\}$ *with the same structure (i.e., same number of constraints and variables) in which the data ranges over a given nonempty* ***compact*** *uncertainty set* $\mathscr{Z} \subset \mathbb{R}^n \times \mathbb{R}^{m\times n} \times \mathbb{R}^m$.

To simplify the notation, we may write $\left\{ \min\{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\} \right\}_{\mathscr{Z}}$. In contrast to an usual single LP problem, it is not possible to associate the notions of feasibility/optimal solutions and optimal objective value with a collection of optimization problems like ULPs. In the setting of ULPs, the feasible solutions are solutions which are *robust feasible*. Roughly speaking, feasible solutions are those which satisfy the set of constraints whatever the realization of uncertain data is. More precisely,

**Definition 3.6** (**Robust feasible/value of an ULP** [BTEGN09, BTN99]). *A vector* $\mathbf{x} \in \mathbb{R}^n$ *is* robust feasible *to an ULP with uncertainty set* $\mathscr{Z}$ *if for each* $[\mathbf{c}, A, \mathbf{b}] \in \mathscr{Z}$, $A\mathbf{x} \leq \mathbf{b}$. *Given a robust feasible solution* $\mathbf{x}$, *the* robust value $\hat{z}(\mathbf{x})$ *of the objective function is* $\hat{z}(\mathbf{x}) := \sup_{[\mathbf{c},A,\mathbf{b}]\in\mathscr{Z}} \mathbf{c}^T\mathbf{x}$.

After carefully defining the robust feasible/optimal solutions as well as their robust objective value, we can describe the central concept in robust optimization setting that is the *robust counterpart* (RC) of an uncertain LP problem. Formally,

**Definition 3.7** (**Robust counterpart of an ULP** [BTN99, BTGGN04]). *Given an ULP problem* $\left\{\min\{\mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b}\}\right\}_{\mathscr{Z}}$, *the* Robust Counterpart *(RC) of ULP is the optimization problem*

$$\min_{\mathbf{x}\in\mathbb{R}^n} \left\{ \hat{z}(\mathbf{x}) = \sup_{[\mathbf{c},A,\mathbf{b}]\in\mathscr{Z}} \mathbf{c}^T\mathbf{x} : A\mathbf{x} \leq \mathbf{b} \quad \forall [\mathbf{c}, A, \mathbf{b}] \in \mathscr{Z} \right\}$$

*that seeks for the best possible value of the objective function among all possible robust feasible solutions to the ULP. Furthermore, the optimal solution/value to the robust counterpart is called the robust optimal solution/value to the ULP.*

In the RC approach, all the variables are "*here and now decisions*" which means that they must be decided before the actual realization of uncertain data is known. However, in some cases, some part of the variables are "*wait and see decisions*", i.e., they tune themselves to the varying data. We follow the terminology in [BTGGN04] and call the variables that may depend on the realizations of the uncertain data as *adjustable* and the other variables as *non-adjustable*. Therefore, we can split the vector $\mathbf{x}$ in the LP representations (3.1) from Definition 3.5 as $\mathbf{x} = (\mathbf{u}, \mathbf{v})^T$ where the sub-vectors $\mathbf{u}$ and $\mathbf{v}$ indicate the non-adjustable and the adjustable variables, respectively.

### 3.3.2 Adjustable Robust Counterpart

Splitting the decision variable $\mathbf{x}$ to the adjustable and non-adjustable variables allows us to rewrite the uncertain LP (3.1) as the following equivalent form:

$$\left\{ \min_{\mathbf{u},\mathbf{v}} \{\mathbf{c}^T\mathbf{u} : U\mathbf{u} + \mathbf{V}\mathbf{v} \leq \mathbf{b}\} \right\}_{[\mathbf{c},U,V,\mathbf{b}]\in\mathscr{Z}} \tag{3.2}$$

The above presentation of the uncertain linear programming problems is normalized in such a way that the objective function is independent of adjustable variables. Moreover, the matrix $V$ is called *recourse matrix* [DM61] and when it is not uncertain, we call the uncertain LP (3.2) a *fixed recourse* one. We can now define the RC and the *Adjustable Robust Counterpart (ARC)* as follows:

$$\text{RC: } \min_{\mathbf{u}} \{\mathbf{c}^T\mathbf{u} : \exists \mathbf{v} : \forall [U,V,\mathbf{b}] \in \mathscr{Z} : U\mathbf{u} + V\mathbf{v} \leq \mathbf{b}\}; \tag{3.3}$$

$$\text{ARC: } \min_{\mathbf{u}} \{\mathbf{c}^T\mathbf{u} : \forall [U,V,\mathbf{b}] \in \mathscr{Z} : \exists \mathbf{v} : U\mathbf{u} + V\mathbf{v} \leq \mathbf{b}\}. \tag{3.4}$$

It is not difficult to see that ARC is less conservative than RC allowing for better optimal values while still having all realizations of the constraints satisfied. The distinction between RC and ARC can be very significant (see, e.g., [BTGGN04, BTEGN09]).

### 3.3.3 Affinely Adjustable Robust Counterpart

The RC of an uncertain LP is a computationally tractable problem in general [BTN99]. On the contrary, this is not the case with ARC and they are in general computationally intractable. This fact stimulates a very good reason to introduce the notion of *Affinely Adjustable Robust Counterpart (AARC)* of an uncertain LP in which we make a simplification on how the adjustable variables can tune themselves upon the uncertain data. By posing $\mathbf{v} = \mathbf{w} + W\xi$, we consider an affine dependency between adjustable variables and uncertain parameter. Therefore, the AARC of the uncertain LP (3.2) reads as:

$$\begin{aligned} &\min_{\mathbf{u},\mathbf{w},W} \left\{ \mathbf{c}^T\mathbf{u} : U\mathbf{u} + V(\mathbf{w} + W\xi) \leq \mathbf{b}, \forall (\xi \equiv [U,V,\mathbf{b}] \in \mathscr{Z}) \right\} \\ \equiv &\min_{\mathbf{u}} \left\{ \mathbf{c}^T\mathbf{u} : \forall (\xi \equiv [U,V,\mathbf{b}] \in \mathscr{Z}) : \exists (\mathbf{w}, W) : U\mathbf{u} + V(\mathbf{w} + W\xi) \leq \mathbf{b} \right\}. \end{aligned} \tag{3.5}$$

The seemingly more simple AARC than general ARC of an ULP is promising in terms of computational tractability where the ARC is intractable. In [BTGGN04], the authors provide a thorough analysis of AARCs of ULPs and propose conditions under which this class of problems admit efficient solutions.

CHAPTER 4

# An Overview of Probabilistic Systems

This chapter introduces the formal models that we will use throughout the thesis. These models which are mathematical formalisms to describe the system behaviour in discrete time, encode nondeterminism and probability as two core features of real systems. Nondeterminism represents a non-quantified choice among two or more alternative behaviours and can be used to model the impact of an unknown environment on the system or indicate lack of knowledge of the system designer about a specific attribute. In contrast to nondeterminism, probability is a quantified property assigning a precise probability value to the alternative system choices.

In this overview chapter, we start in Section 4.1 with a concise introduction of Markov chains as the basic models to describe systems with probabilistic behaviour. In Section 4.2, we extend Markov chains with nondeterminism and thereby attain Markov decision processes. In order to capture quantities like preferences or priorities in nondeterministic scenarios, we introduce weighted Markov decision processes in Section 4.3 and discuss how these models relate to Markov decision processes.

Probabilistic automata constitute a mathematical framework for the modelling and analysis of concurrent probabilistic systems. These models which subsume Markov chains and Markov decision processes are discussed in Section 4.4. We review the semantics of the probabilistic automata framework and define the parallel composition operator for this model. Afterwards, we discuss in detail the notion of weak transitions in probabilistic automata.

Apart from nondeterminism and probability, *uncertainty* is another key feature that can influence the design and operation of complex probabilistic systems. Uncertainty relates to the fact that not all system parameters may be known exactly, including exact probability values. In Section 4.5, we introduce interval Markov decision processes as a modelling formalism to incorporate uncertainty in discrete-time probabilistic systems. These models are at the core of our studies in the forthcoming chapters. Hence, we provide a more detailed discussion on their semantics. Throughout this thesis, we consider only finite models, i.e., systems such that states, actions, and transition relations are finite.

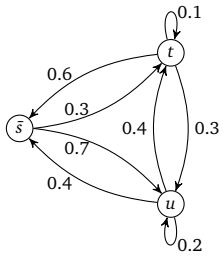



Figure 4.1: An example of *MC*s: the *MC* $\mathscr{D}$

## 4.1 Markov Chains

Markov chains are a well-known subclass of stochastic processes to model systems which indicate probabilistic behaviour. Markov chain models that are supported by an elegant and relatively easy theory are central to model a wide spectrum of stochastic phenomena. Markov chains are distinguished from other types of stochastic processes by the *Markov property* which is also known as the *memoryless* property. This property informally states that the future behaviour of a Markov chain is independent of the past given its current state. A Markov chain with a finite state space can be formally defined as follows:

**Definition 4.1** (**Markov chain** [Ste94, HJ94])**.** *A* Markov Chain *(MC) is a tuple* $\mathscr{D} = (S, \bar{s}, \mathtt{AP}, \mathbf{P}, L)$ *where* $S$ *is a finite set of* states, $\bar{s}$ *is the* start state, $\mathtt{AP}$ *is a finite set of atomic propositions,* $\mathbf{P} \colon S \times S \to [0,1]$ *is a* transition probability matrix *such that* $\sum_{s' \in S} \mathbf{P}(s,s') = 1$ *for all* $s \in S$*, and* $L \colon S \to 2^{\mathtt{AP}}$ *is a* labeling function.

**Example 4.1.** *An example of a MC is shown in Figure 4.1. In the graph representation, states are depicted as circles and transitions as arrows. Every transition is labelled with its associated probability value. The state space is* $S = \{\bar{s}, t, u\}$ *and the start state is* $\bar{s}$*. Given set of atomic propositions* $\mathtt{AP} = \{\mathtt{target}\}$*, let* $L(\bar{s}) = \emptyset$*,* $L(t) = \emptyset$ *and* $L(u) = \{\mathtt{target}\}$*. Moreover, the transition probability matrix is given as follows:*

$$\mathbf{P} = \begin{bmatrix} 0 & 0.3 & 0.7 \\ 0.6 & 0.1 & 0.3 \\ 0.4 & 0.4 & 0.2 \end{bmatrix}.$$

♦

A Markov chain model can be unfolded into a set of paths. A path in a Markov chain $\mathscr{D}$ is a finite or infinite sequence of states $\xi = s_0 s_1 s_2 \ldots$ where $s_i \in S$ and $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i \geq 0$. We denote the sets of all finite and infinite paths in $\mathscr{D}$ by $Paths_{\mathscr{D}}^{fin}$ and $Paths_{\mathscr{D}}^{inf}$, respectively. The last state of path $\xi$ is denoted by $last(\xi)$. Moreover, let $Paths_{\mathscr{D}}^{\xi} = \{\xi' \in Paths_{\mathscr{D}}^{inf} \mid \xi \text{ is a prefix of } \xi'\}$ denote the set of infinite paths with the prefix $\xi \in Paths_{\mathscr{D}}^{fin}$ which is also known as the *cylinder* set of $\xi$.

The probability matrix $\mathbf{P}$ induces a probability measure over the set of infinite paths $Paths_{\mathscr{D}}^{inf}$ applying the cylinder construction [KSK66] as follows. The probability $\mathbf{Pr}_{\mathscr{D}}$ of a state $s'$ is defined to be $\mathbf{Pr}_{\mathscr{D}}[Paths_{\mathscr{D}}^{s'}]$ which equals 1 if $s' = \bar{s}$ and 0, otherwise; and the

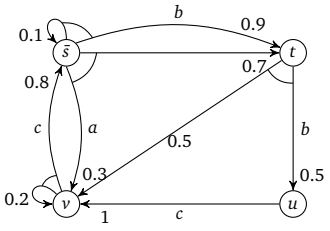



Figure 4.2: An example of *MDP*s: the *MDP* $\mathscr{M}$

probability $\mathbf{Pr}_{\mathscr{D}}[Paths_{\mathscr{D}}^{\xi s'}]$ of traversing a finite path $\xi s'$ equals $\mathbf{Pr}_{\mathscr{D}}[Paths_{\mathscr{D}}^{\xi}] \cdot \mathbf{P}(last(\xi), s')$. This extends to a unique probability measure on the set of infinite paths $Paths_{\mathscr{D}}^{inf}$ [KSK66].

## 4.2 Markov Decision Processes

Markov chain models are suitable for systems exhibiting only probabilistic behaviours, that is, they are not able to represent systems where different transition options can be selected in the states. For instance, a system may in a particular state react differently to different stimuli. This can be modeled by performing different transitions leading to different distributions over the states of the system. We call this capacity *nondeterminism* that is encoded, together with probability, by Markov decision processes.

**Definition 4.2** (**Markov decision process**). *A* Markov Decision Process *(MDP)* $\mathscr{M}$ *is a tuple* $\mathscr{M} = (S, \bar{s}, \mathscr{A}, \mathtt{AP}, L, T)$, *where* $S$ *is a finite set of* states, $\bar{s} \in S$ *is the* initial state, $\mathscr{A}$ *is a finite set of* actions, $\mathtt{AP}$ *is a finite set of* atomic propositions, $L : S \to 2^{\mathtt{AP}}$ *is a* labeling function, *and* $T : S \times \mathscr{A} \to \mathrm{Disc}(S)$ *is a* transition probability function.

Given an *MDP* $\mathscr{M}$, we denote by $\mathscr{A}(s)$ the set of actions that are enabled from state $s$. Markov decision processes [How71, Ber95, Put05] (*MDP*s) are powerful models for systems involving both decision-making and probabilistic dynamics [Put05]. *MDP*s are used extensively to model long-term sequential decision-making in stochastic situations. They have been applied broadly in quantitative model checking [FKNP11], operations research [Ye11], machine learning [Mar15] and artificial intelligence [Kol12]. Model checkers for *MDP*s, such as PRISM [KNP11], Modest [BDH+12], and IscasMC [HLS+14] have been developed and applied in practice successfully. These tools mostly rely on Bellmann's value iteration [Bel57] and Howard's policy iteration [How71] algorithms to solve *MDP*s. In *MDP* models, each state is equipped with a finite set of probability distributions each of which is uniquely identified by an action. Thus, the set of actions in each state indicates the available nondeterministic choices in that state.

The class of *MDP*s subsumes the class of *MC*s. To see this, note that every *MC* can be thought of as an *MDP* with $\mathscr{A}$ being a singleton set in Definition 4.2. Conversely, every *MDP* whose action set $\mathscr{A}$ is a singleton semantically equals to a *MC*. Therefore, such *MDP* model does not include nondeterministic choices in each of its states.

**Example 4.2.** *Figure 4.2 depicts an example of MDP with initial state* $\bar{s}$. *The set of states in* $S = \{\bar{s}, t, u, v\}$ *and the set of actions* $\mathscr{A} = \{a, b, c\}$. *The labeling of states are given as follows:* $L(\bar{s}) = \{\mathtt{init}\}$, $L(t) = \emptyset$, $L(u) = \emptyset$, $L(v) = \{\mathtt{success}\}$. *At the initial state, a nondeterministic*

*choice happens between actions a and b at the initial state $\bar{s}$. For the rest of the states, there is only one single action.* ♦

An infinite path through an *MDP* $\mathscr{M}$ is defined as a sequence $\xi = s_1 a_1 \ldots$ where $s_i \in S$, $a_i \in \mathscr{A}(s_i)$ and $T(s_i, a_i)(s_{i+1}) > 0$ for each $i \geq 1$. A finite path is specified as a prefix of an infinite path terminating in a state. We denote the sets of all finite and infinite paths in $\mathscr{M}$ by $Paths^{fin}_{\mathscr{M}}$ and $Paths^{inf}_{\mathscr{M}}$, respectively. Moreover, let $Paths^{\xi}_{\mathscr{M}} = \{ \xi' \in Paths^{inf}_{\mathscr{M}} \mid \xi \text{ is a prefix of } \xi' \}$ denote the set of infinite paths with the prefix $\xi \in Paths^{fin}_{\mathscr{M}}$ which is also known as the *cylinder* set of $\xi$.

In order to formally analyze *MDP*s, we require a probability space over infinite paths. Nevertheless, the construction of a probability space is feasible once all nondeterministic choices of the *MDP* are resolved. The resolution of nondeterminism is done by a *scheduler* (also known as *strategy* or *controller*) which chooses an action in each state of the *MDP* $\mathscr{M}$ based on the past history of its execution. Formally,

**Definition 4.3** (**Scheduler for Markov decision processes**). *A scheduler for an MDP $\mathscr{M} = (S, \bar{s}, \mathscr{A}, \mathtt{AP}, L, T)$ is a function $\sigma : Paths^{fin}_{\mathscr{M}} \to \mathrm{Disc}(\mathscr{A})$ such that $\sigma(\xi)(a) > 0$ only if $a \in \mathscr{A}(last(\xi))$. A scheduler $\sigma$ is memoryless if $\sigma(\xi)$ depends only on $last(\xi)$. Furthermore, a scheduler $\sigma$ is called deterministic if the distribution $\sigma(\xi)$ is Dirac, i.e., it always chooses a single action with probability 1.*

A strategy $\sigma$ induces a probability measure over infinite paths as follows. The probability $\mathbf{Pr}^{\sigma}_{\mathscr{M}}$ of a state $s'$ is defined to be $\mathbf{Pr}^{\sigma}_{\mathscr{M}}[Paths^{s'}_{\mathscr{M}}] = \delta_{\bar{s}}(s')$ and the probability $\mathbf{Pr}^{\sigma}_{\mathscr{M}}[Paths^{\xi a s'}_{\mathscr{M}}]$ of traversing a finite path $\xi a s'$ is defined to be $\mathbf{Pr}^{\sigma}_{\mathscr{M}}[Paths^{\xi a s'}_{\mathscr{M}}] = \mathbf{Pr}^{\sigma}_{\mathscr{M}}[Paths^{\xi}_{\mathscr{M}}] \cdot \sigma(\xi)(a).T(last(\xi), a)(s')$. Then, $\mathbf{Pr}^{\sigma}_{\mathscr{M}}$ extends uniquely to the $\sigma$-field generated by cylinder sets.

## 4.3 Weighted Markov Decision Processes

*Weighted Markov Decision Processes* are modelling formalisms to capture quantities like preferences or priorities in a nondeterministic scenario. In these models, weights can be used to denote priorities or preferences, which are quantities used to generate probabilistic behaviours. Weights in the weighted Markov decision processes play a similar role as in GSPN [MCB84] or EMPA [BG96], namely, they are used to induce a distribution over all transitions. For instance, if from a state $s$, there are transitions leading to $s_1$ and $s_2$ with weights 2 and 3, respectively, then this means that $s$ will evolve into $s_1$ and $s_2$ with probabilities $\frac{2}{5}$ and $\frac{3}{5}$, respectively. Below, we introduce the definition of weighted Markov decision processes. Formally,

**Definition 4.4** (**Weighted Markov decision process**). *A* weighted Markov Decision Process *(wMDP) is a tuple $\mathscr{W} = (S, \bar{s}, \mathscr{A}, AP, L, W)$, where $S$ is a finite set of states, $\bar{s} \in S$ is the initial state, $\mathscr{A}$ is a finite set of actions, $AP$ is a finite set of atomic propositions, $L : S \mapsto 2^{AP}$ is a labelling function and $W : S \times \mathscr{A} \times S \mapsto \mathbb{N}_0$ defines a transition relation.*

We write $s \xrightarrow{a,w} \mu$ if and only if $w = \sum_{t \in S} W(s, a, t) > 0$ and $\mu(t) = \frac{W(s,a,t)}{w}$. Let $s \xrightarrow{a,w}_{\mathrm{c}} \mu$, called combined transitions [SL95], if and only if there exists $\{\mu_i\}_{i \in I}$ and $\{p_i \in [0,1]\}_{i \in I}$

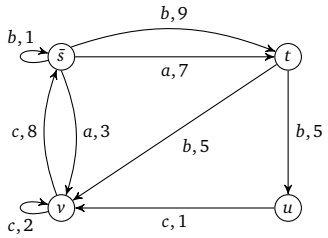



Figure 4.3: An example of *wMDP*s: the *wMDP* $\mathscr{W}$

such that $\sum_{i\in I} p_i \cdot \mu_i = \mu$ where $\sum_{i\in I} p_i = 1$ and $s \xrightarrow{a,w} \mu_i$ for each $i \in I$. In the following, we formalize the relation between *MDP*s and *wMDP*s with respect to the model transformation.

**Remark 4.1.** *As discussed earlier, in wMDPs weights are used to generate probabilistic behaviours. Hence, by normalizing the weights assigned to every transition in the wMDPs, an MDP is generated. Conversely, given any MDP with rational transition probabilities the corresponding wMDP is generated as follows: for every transition of the MDP, we multiply each transition probability value by the least common multiplier of denominators of all transition probability values. Hence, the models MDPs with rational transition probabilities and wMDPs can be transformed to each other with a time complexity polynomial in the number of states and transitions. For the general case, i.e., when some of transition probabilities in the MDP model are irrational, we have wMDP* $\subseteq$ *MDP.*

**Example 4.3.** *Figure 4.3 depicts an example of wMDP which also corresponds to the MDP in Figure 4.2.* ♦

## 4.4 Probabilistic Automata

*Probabilistic automata* (*PA*) constitute a mathematical framework for the specification of probabilistic concurrent systems [Seg95, CSV07]. Probabilistic automata extend classical concurrency models in a simple yet conservative fashion. In probabilistic automata, concurrent processes may perform probabilistic experiments inside a transition. This is represented by transitions of the form $s \xrightarrow{a} \mu$, where $s$ is a state, $a$ is an action label, and $\mu$ is a probability measure on states. We now recall the main parts of the probabilistic automata framework [Seg95] we use in this thesis, following the notation of [Seg06]. Note that the probabilistic automata we use here correspond to the *simple* probabilistic automata of [Seg95].

**Definition 4.5 (Probabilistic automata).** *A* probabilistic automaton *(PA) is a tuple* $\mathfrak{P} = (S, \bar{s}, \mathscr{A}, T)$*, where* $S$ *is a set of* states*,* $\bar{s} \in S$ *is the* start *state,* $\mathscr{A}$ *is the set of* actions*, and* $T \subseteq S \times \mathscr{A} \times \mathrm{Disc}(S)$ *is a* probabilistic transition relation*. We denote by* $[\mathfrak{P}]$*, the class of all finite-state finite-transition PAs.*

The start state is also called the *initial* state. The set $\mathscr{A}$ is divided in two disjoint sets $H$ and $E$ of internal (hidden) and external actions, respectively; we let $s$, $t$, $u$, $v$, and their variants with indices range over $S$; $a$, $b$ range over external actions; and $\tau$ range over

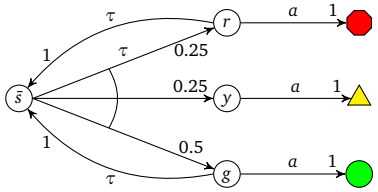



Figure 4.4: An example of *PA*s: the *PA* $\mathfrak{E}$

internal actions. We denote the generic elements of a probabilistic automaton $\mathfrak{P}$ by $S$, $\bar{s}$, $\mathscr{A}$, $H$, $E$, $T$, and we propagate primes and indices when necessary. Thus, for example, the probabilistic automaton $\mathfrak{P}'_i$ has states $S'_i$, start state $\bar{s}'_i$, actions $\mathscr{A}'_i$, internal actions $H'_i$, external actions $E'_i$, and transition relation $T'_i$.

A transition $tr = (s, a, \mu) \in T$, also denoted by $s \xrightarrow{a} \mu$, is said to *leave* from state $s$, to be *labelled* by $a$, and to *lead* to the measure $\mu$. We denote by $src(tr)$ the *source* state $s$, by $act(tr)$ the *action* $a$, and by $trg(tr)$ the *target* measure $\mu$, also denoted by $\mu_{tr}$. We also say that $s$ enables the action $a$, that the action $a$ is enabled from $s$, and that $(s, a, \mu)$ is enabled from $s$. We call a transition $s \xrightarrow{a} \mu$ *internal* or *external* whenever $a \in H$ or $a \in E$, respectively. Finally, we let $T(a) = \{ tr \in T \mid act(tr) = a \}$ be the set of transitions with label $a$.

We say that a state $s$ is a *deadlock* state if it enables no transitions, i.e., $\{ tr \in T \mid src(tr) = s \} = \emptyset$.

Given a *PA* $\mathfrak{P}$, we denote by $|\mathfrak{P}| = \max\{|S|, |T|\}$ the *size* of $\mathfrak{P}$. Throughout this thesis, we assume that $\mathfrak{P}$ is finite, that is, both $S$ and $T$ are finite sets; moreover, we assume that each state of $\mathfrak{P}$ can be reached from $\bar{s}$.

**Example 4.4.** *An example of PA is shown in Figure 4.4: the set of states is* $S = \{\bar{s}, r, y, g, \text{●}, \text{▲}, \text{●}\}$, *the start state is* $\bar{s}$, *the set of actions* $\mathscr{A}$ *is the union of the set of external actions* $E = \{a\}$ *and of the set of internal actions* $H = \{\tau\}$, *and the transition relation* $T$ *contains the following transitions:* $\bar{s} \xrightarrow{\tau} \rho$ *with* $\rho = \{(r, 0.25), (y, 0.25), (g, 0.5)\}$, $r \xrightarrow{a} \delta_{\text{●}}$, $y \xrightarrow{a} \delta_{\text{▲}}$, $g \xrightarrow{a} \delta_{\text{●}}$, $r \xrightarrow{\tau} \delta_{\bar{s}}$, *and* $g \xrightarrow{\tau} \delta_{\bar{s}}$. ●, ▲, *and* ● *are deadlock states and the size of* $\mathfrak{E}$ *is* $|\mathfrak{E}| = 7$. ♦

**Remark 4.2.** *The main difference between the definition of probabilistic automata and Markov decision processes is the probabilistic transition relation: for the former model, a state can enable multiple transitions with the same label, while for the latter model each transition from the same state has to have a different label. A second difference is that the PA framework distinguishes between internal and external actions, the latter used by a PA for the synchronization when composed in parallel with other PAs. The MDP model, on the contrary, does not make such a distinction and treats each action in the same manner.*

### 4.4.1 Parallel Composition and Hiding

Real world applications and protocols typically involve several parts each one composed by modules working together in parallel. Probabilistic automata model has been proposed to achieve such compositional property. The following definition of parallel composition is an equivalent rewriting of the definition provided in [Seg06].

**Definition 4.6** (**Probabilistic automata parallel composition**). *Given two PAs $\mathfrak{P}_1$ and $\mathfrak{P}_2$, we say that $\mathfrak{P}_1$ and $\mathfrak{P}_2$ are* compatible *if $\mathscr{A}_1 \cap H_2 = \emptyset = H_1 \cap \mathscr{A}_2$.*
*Given two compatible PAs $\mathfrak{P}_1$ and $\mathfrak{P}_2$, the* parallel composition *of $\mathfrak{P}_1$ and $\mathfrak{P}_2$, denoted by $\mathfrak{P}_1 \parallel \mathfrak{P}_2$, is the probabilistic automaton $\mathfrak{P} = (S, \bar{s}, \mathscr{A}, T)$ where*

- $S = S_1 \times S_2$,
- $\bar{s} = (\bar{s}_1, \bar{s}_2)$,
- $\mathscr{A} = E \cup H$ *where* $E = E_1 \cup E_2$ *and* $H = H_1 \cup H_2$, *and*
- $((s_1, s_2), a, \mu_1 \times \mu_2) \in T$ *if and only if*
  - *whenever* $a \in \mathscr{A}_1 \cap \mathscr{A}_2$, $(s_1, a, \mu_1) \in T_1$ *and* $(s_2, a, \mu_2) \in T_2$,
  - *whenever* $a \in \mathscr{A}_1 \setminus \mathscr{A}_2$, $(s_1, a, \mu_1) \in T_1$ *and* $\mu_2 = \delta_{s_2}$, *and*
  - *whenever* $a \in \mathscr{A}_2 \setminus \mathscr{A}_1$, $(s_2, a, \mu_2) \in T_2$ *and* $\mu_1 = \delta_{s_1}$.

*For $a \in \mathscr{A}_1 \setminus \mathscr{A}_2$, we denote by $(s_2, a, \delta_{s_2})$ the* apparent *internal transition corresponding to not performing any transition from $s_2$ in the combined transition, and similarly for $a \in \mathscr{A}_2 \setminus \mathscr{A}_1$.*

For two compatible *PA*s $\mathfrak{P}_1$ and $\mathfrak{P}_2$ and their parallel composition $\mathfrak{P}_1 \parallel \mathfrak{P}_2$, we refer to $\mathfrak{P}_1$ and $\mathfrak{P}_2$ as the component automata and to $\mathfrak{P}_1 \parallel \mathfrak{P}_2$ as the composed automaton.

**Definition 4.7** (**Hiding in probabilistic automata**). *Given a PA $\mathfrak{P}$ and a set $A$ of actions, the* hiding *of $A$ in $\mathfrak{P}$, denoted by* $\text{Hide}_A(\mathfrak{P})$, *is the automaton $\mathfrak{P}'$ that is the same as $\mathfrak{P}$ except for $E' = E \setminus A$ and $H' = H \cup A$.*

**Remark 4.3.** *In the above definition of parallel composition between PAs, we require that they are compatible, i.e., the internal actions of one automaton can not be actions of the other automaton. This requirement seems to be never fulfilled when we consider the internal action $\tau$.*

*In the Process Algebra world, usually $\tau$ is the only internal action available, and it is used by every process to denote an internal transition. In the Probabilistic Automata framework, $\tau$ is used as a symbol for referring to internal actions, but usually it is not an actual action of the automaton. This means that, for two automata $\mathfrak{P}_1$ and $\mathfrak{P}_2$, when we write $(s_1, \tau, \mu_1) \in T_1$ and $(s_2, \tau, \mu_2) \in T_2$, we are not requiring that the label is the same for both transitions, but we are just referring to $(s_1, a_1, \mu_1) \in T_1$ and $(s_2, a_2, \mu_2) \in T_2$ for some $a_i \in H_i$, $i \in \{1, 2\}$.*

*The role of $\tau$ as symbol for internal actions and not as actual action becomes clear from the definition of the hiding operator: for a given set $A$ of actions to be hidden, instead of replacing each action in $A$ with $\tau$ as happens in process algebra world, we simply move the actions in $A$ from $E$ to $H$; the actual actions remain unchanged.*

*Note that it is rather easy to transform two automata $\mathfrak{P}_1$ and $\mathfrak{P}_2$ that are not compatible into compatible ones, by means of the* action renaming *operator [Seg95] that allows us to rename actions under the assumption that external actions remain external and internal actions remain internal. So, we can just rename the internal actions of both $\mathfrak{P}_1$ and $\mathfrak{P}_2$ with fresh (internal) actions and the resulting automata are then compatible.*

### 4.4.2 Weak Transitions

In the setting of labelled transition systems, weak transitions are used to abstract from internal computations [Mil89]. Intuitively, an internal weak transition is formed by an

arbitrarily long sequence of internal transitions, and an external weak transition is formed by an external transition preceded and followed by arbitrarily long sequences of internal transitions. Note that the empty sequence is a valid arbitrary long sequence of internal transitions. To lift this idea to the setting of probabilistic automata is a little intricate owed to the fact that transitions branch into probability measures, and one thus has to work with tree-like objects instead of sequences, as detailed in the sequel.

An *execution fragment* of a *PA* $\mathfrak{P}$ is a finite or infinite sequence of alternating states and actions $\alpha = s_0 a_1 s_1 a_2 s_2 \dots$ starting from a state $s_0$, also denoted by $\mathit{first}(\alpha)$, and, if the sequence is finite, ending with a state denoted by $\mathit{last}(\alpha)$, such that for each $i > 0$ there exists a transition $(s_{i-1}, a_i, \mu_i) \in T$ such that $\mu_i(s_i) > 0$. The *length* of $\alpha$, denoted by $|\alpha|$, is the number of occurrences of actions in $\alpha$. If $\alpha$ is infinite, then $|\alpha| = \infty$. We denote by $\mathit{state}(\alpha, i)$ the state $s_i$ and by $\mathit{action}(\alpha, j)$ the action $a_j$, provided that $0 \leq i \leq |\alpha|$ and $0 < j \leq |\alpha|$. Denote by $\mathit{frags}(\mathfrak{P})$ the set of execution fragments of $\mathfrak{P}$ and by $\mathit{frags}^*(\mathfrak{P})$ the set of finite execution fragments of $\mathfrak{P}$. An execution fragment $\alpha$ is a *prefix* of an execution fragment $\alpha'$, denoted by $\alpha \leqslant \alpha'$, if the sequence $\alpha$ is a prefix of the sequence $\alpha'$. The *trace* of $\alpha$, denoted by $\mathit{trace}(\alpha)$, is the sub-sequence of external actions of $\alpha$; we denote by $\varepsilon$ the empty trace and we extend $\mathit{trace}(\cdot)$ to actions by defining $\mathit{trace}(a) = a$ if $a \in E$ and $\mathit{trace}(a) = \varepsilon$ if $a \in H$.

**Definition 4.8 (*PA* scheduler).** *A* scheduler *for a PA* $\mathfrak{P}$ *is a function* $\sigma : \mathit{frags}^*(\mathfrak{P}) \to \mathrm{SubDisc}(T)$ *such that for each* $\alpha \in \mathit{frags}^*(\mathfrak{P})$, $\sigma(\alpha) \in \mathrm{SubDisc}(\{\, tr \in T \mid \mathit{src}(tr) = \mathit{last}(\alpha) \,\})$ *or, equivalently,* $\mathrm{Supp}(\sigma(\alpha)) \subseteq \{\, tr \in T \mid \mathit{src}(tr) = \mathit{last}(\alpha) \,\}$.

Given a scheduler $\sigma$ and a finite execution fragment $\alpha$, the measure $\sigma(\alpha)$ describes how transitions are chosen to move on from $\mathit{last}(\alpha)$. We call a scheduler *determinate* [CS02] if, for each $\alpha, \alpha' \in \mathit{frags}^*(\mathfrak{P})$ such that $\mathit{trace}(\alpha) = \mathit{trace}(\alpha')$ and $\mathit{last}(\alpha) = \mathit{last}(\alpha')$, then $\sigma(\alpha) = \sigma(\alpha')$. Essentially, a determinate scheduler bases its choice only on the current state and on the past external actions. In other words, a determinate scheduler acts as a history-independent scheduler between one external action and the following external action (or the choice of stopping).

A scheduler $\sigma$ and a state $s$ induce a probability measure $\mu_{\sigma,s}$ over execution fragments as follows. The basic measurable events are the cones of finite execution fragments, where the cone of $\alpha$, denoted by $C_\alpha$, is the set $C_\alpha = \{\, \alpha' \in \mathit{frags}(\mathfrak{P}) \mid \alpha \leqslant \alpha' \,\}$. The probability $\mu_{\sigma,s}$ of a cone $C_\alpha$ is defined recursively as follows:

$$\mu_{\sigma,s}(C_\alpha) = \begin{cases} 1 & \text{if } \alpha = s, \\ 0 & \text{if } \alpha = t \text{ for a state } t \neq s, \\ \mu_{\sigma,s}(C_{\alpha'}) \cdot \sum_{tr \in T(a)} \sigma(\alpha')(tr) \cdot \mu_{tr}(t) & \text{if } \alpha = \alpha' a t. \end{cases}$$

Standard measure theoretical arguments ensure that $\mu_{\sigma,s}$ extends uniquely to the $\sigma$-field generated by cones. We call the resulting measure $\mu_{\sigma,s}$ a *probabilistic execution fragment* of $\mathfrak{P}$ and we say that it is generated by $\sigma$ from $s$. Given a finite execution fragment $\alpha$, we define $\mu_{\sigma,s}(\alpha)$ as $\mu_{\sigma,s}(\alpha) = \mu_{\sigma,s}(C_\alpha) \cdot \sigma(\alpha)(\bot)$, where $\sigma(\alpha)(\bot)$ is the probability of choosing no transitions after $\alpha$ has occurred.

**Definition 4.9 (Weak combined transition).** *Given a PA* $\mathfrak{P}$*, we say that there is a* weak combined transition *from* $s \in S$ *to* $\mu \in \mathrm{Disc}(S)$ *labelled by* $a \in \mathscr{A}$*, denoted by* $s \overset{a}{\Longrightarrow}_{\mathrm{c}} \mu$*, if*

*there exists a scheduler $\sigma$ such that the following holds for the induced probabilistic execution fragment $\mu_{\sigma,s}$:*

1. $\mu_{\sigma,s}(frags^*(\mathfrak{P})) = 1$;
2. *for each* $\alpha \in frags^*(\mathfrak{P})$, *if* $\mu_{\sigma,s}(\alpha) > 0$ *then* $trace(\alpha) = trace(a)$;
3. *for each state* $t$, $\mu_{\sigma,s}(\{\, \alpha \in frags^*(\mathfrak{P}) \mid last(\alpha) = t \,\}) = \mu(t)$.

In this case, we say that the weak combined transition $s \overset{a}{\Longrightarrow}_c \mu$ is induced by $\sigma$, that $s \overset{a}{\Longrightarrow}_c \mu$ exists in $\mathfrak{P}$, and that $\mathfrak{P}$ enables $s \overset{a}{\Longrightarrow}_c \mu$.

Albeit the definition of weak combined transitions is admittedly intricate, it is just the obvious extension of weak transitions on labelled transition systems to the setting with probabilities. We refer to Segala [Seg06] for more details on weak combined transitions.

**Example 4.5.** *Consider the automaton $\mathfrak{E}$ depicted in Figure 4.4 and let $\rho$ be $\rho = \{(r, 0.3), (y, 0.1), (g, 0.6)\}$; $\mathfrak{E}$ enables the weak combined transition $\bar{s} \overset{a}{\Longrightarrow}_c \mu$ where $\mu = \{(\bigcirc, \frac{9}{50}), (\triangle, \frac{8}{50}), (\bigcirc, \frac{33}{50})\}$ via the scheduler $\sigma$ defined as follows:*

$$\sigma(\alpha) = \begin{cases} \delta_{\bar{s} \xrightarrow{\tau} \rho} & \text{if } last(\alpha) = \bar{s}, \\ \delta_{r \xrightarrow{\tau} \delta_{\bar{s}}} & \text{if } \alpha = \bar{s}\tau r, \\ \delta_{r \xrightarrow{a} \delta_{\bigcirc}} & \text{if } \alpha \neq \bar{s}\tau r \text{ and } last(\alpha) = r, \\ \delta_{y \xrightarrow{a} \delta_{\triangle}} & \text{if } last(\alpha) = y, \\ \{(g \xrightarrow{\tau} \delta_{\bar{s}}, 0.5), (g \xrightarrow{a} \delta_{\bigcirc}, 0.5)\} & \text{if } \alpha = \bar{s}\tau g, \\ \delta_{g \xrightarrow{a} \delta_{\bigcirc}} & \text{if } \alpha \neq \bar{s}\tau g \text{ and } last(\alpha) = g, \\ \delta_{\perp} & \text{otherwise.} \end{cases}$$

*We now verify the three properties that $\mu_{\sigma,\bar{s}}$ has to satisfy in order to justify $\bar{s} \overset{a}{\Longrightarrow}_c \mu$: we start from the third property, since the first two can be derived from it. Consider the state $\bigcirc$: it is reached with probability*

$$\begin{aligned} &\mu_{\sigma,\bar{s}}(\{\, \alpha \in frags^*(\mathfrak{E}) \mid last(\alpha) = \bigcirc \,\}) \\ &= \mu_{\sigma,\bar{s}}(\{\bar{s}\tau r\tau\bar{s}\tau ra\bigcirc\}) + \mu_{\sigma,\bar{s}}(\{\bar{s}\tau g\tau\bar{s}\tau ra\bigcirc\}) \\ &\quad + \mu_{\sigma,\bar{s}}(\{\, \alpha \in frags^*(\mathfrak{E}) \mid last(\alpha) = \bigcirc \,\} \setminus \{\bar{s}\tau r\tau\bar{s}\tau ra\bigcirc, \bar{s}\tau g\tau\bar{s}\tau ra\bigcirc\}) \\ &= \mu_{\sigma,\bar{s}}(\{\bar{s}\tau r\tau\bar{s}\tau ra\bigcirc\}) + \mu_{\sigma,\bar{s}}(\{\bar{s}\tau g\tau\bar{s}\tau ra\bigcirc\}) + 0 \\ &= \left(\left(\left(\left((1) \cdot 1 \cdot \frac{3}{10}\right) \cdot 1 \cdot 1\right) \cdot 1 \cdot \frac{3}{10}\right) \cdot 1 \cdot 1\right) \cdot 1 + \left(\left(\left(\left((1) \cdot 1 \cdot \frac{6}{10}\right) \cdot \frac{1}{2} \cdot 1\right) \cdot 1 \cdot \frac{3}{10}\right) \cdot 1 \cdot 1\right) \cdot 1 \\ &= \frac{9}{100} + \frac{9}{100} = \frac{9}{50} = \mu(\bigcirc), \end{aligned}$$

*as required. The fact that*

$$\mu_{\sigma,\bar{s}}(\{\, \alpha \in frags^*(\mathfrak{E}) \mid last(\alpha) = \bigcirc \,\} \setminus \{\bar{s}\tau r\tau\bar{s}\tau ra\bigcirc, \bar{s}\tau g\tau\bar{s}\tau ra\bigcirc\}) = 0$$

*is justified as follows: let $\alpha \in \{\beta \in frags^*(\mathfrak{E}) \mid last(\beta) = \bigcirc\}$ such that $\alpha \notin \{\bar{s}\tau r\tau\bar{s}\tau ra\bigcirc, \bar{s}\tau g\tau\bar{s}\tau ra\bigcirc\}$; if $first(\alpha) = t \neq \bar{s}$, then by the recursive definition of $\mu_{\sigma,\bar{s}}(C_\alpha)$ we*

*have that the base case is* $\mu_{\sigma,\bar{s}}(C_t) = 0$, *hence* $\mu_{\sigma,\bar{s}}(C_\alpha) = 0$ *as well. Suppose that* $first(\alpha) = \bar{s}$ *and consider the case* $\alpha = \bar{s}\tau r a$●:

$$\begin{aligned}
\mu_{\sigma,\bar{s}}(C_\alpha) &= \mu_{\sigma,\bar{s}}(C_{\bar{s}\tau r a●}) \\
&= \mu_{\sigma,\bar{s}}(C_{\bar{s}\tau r}) \cdot \sum_{tr \in T(a)} \sigma(\bar{s}\tau r)(tr) \cdot \mu_{tr}(●) \\
&= \mu_{\sigma,\bar{s}}(C_{\bar{s}\tau r}) \cdot (\sigma(\bar{s}\tau r)(r \xrightarrow{a} \delta_●) \cdot \delta_●(●) \\
&\qquad + \sigma(\bar{s}\tau r)(y \xrightarrow{a} \delta_▲) \cdot \delta_▲(●) \\
&\qquad + \sigma(\bar{s}\tau r)(g \xrightarrow{a} \delta_●) \cdot \delta_●(●)) \\
&= \mu_{\sigma,\bar{s}}(C_{\bar{s}\tau r}) \cdot (0 \cdot 1 + 0 \cdot 0 + 0 \cdot 0) = 0.
\end{aligned}$$

*Finally, the remaining finite execution fragments are such that* $\alpha \in C_{\bar{s}\tau r\tau\bar{s}\tau r\tau\bar{s}} \cup C_{\bar{s}\tau r\tau\bar{s}\tau g\tau\bar{s}} \cup C_{\bar{s}\tau g\tau\bar{s}\tau r\tau\bar{s}} \cup C_{\bar{s}\tau g\tau\bar{s}\tau g\tau\bar{s}}$. *Consider the case* $\alpha \in C_{\bar{s}\tau r\tau\bar{s}\tau r\tau\bar{s}}$: *by the recursive definition of* $\mu_{\sigma,\bar{s}}(C_\alpha)$ *we have that* $\mu_{\sigma,\bar{s}}(C_\alpha) = \mu_{\sigma,\bar{s}}(C_{\bar{s}\tau r\tau\bar{s}\tau r\tau\bar{s}}) \cdot p$ *for some value* $p \in \mathbb{R}_{\geq 0}$; *now, consider* $\mu_{\sigma,\bar{s}}(C_{\bar{s}\tau r\tau\bar{s}\tau r\tau\bar{s}})$:

$$\begin{aligned}
\mu_{\sigma,\bar{s}}(C_{\bar{s}\tau r\tau\bar{s}\tau r\tau\bar{s}}) &= \mu_{\sigma,\bar{s}}(C_{\bar{s}\tau r\tau\bar{s}\tau r}) \cdot \sum_{tr \in T(\tau)} \sigma(\bar{s}\tau r\tau\bar{s}\tau r)(tr) \cdot \mu_{tr}(\bar{s}) \\
&= \mu_{\sigma,\bar{s}}(C_{\bar{s}\tau r\tau\bar{s}\tau r}) \cdot (\sigma(\bar{s}\tau r\tau\bar{s}\tau r)(r \xrightarrow{\tau} \delta_{\bar{s}}) \cdot \delta_{\bar{s}}(\bar{s}) \\
&\qquad + \sigma(\bar{s}\tau r\tau\bar{s}\tau r)(\bar{s} \xrightarrow{\tau} \rho) \cdot \rho(\bar{s}) \\
&\qquad + \sigma(\bar{s}\tau r\tau\bar{s}\tau r)(g \xrightarrow{\tau} \delta_{\bar{s}}) \cdot \delta_{\bar{s}}(\bar{s})) \\
&= \mu_{\sigma,\bar{s}}(C_{\bar{s}\tau r\tau\bar{s}\tau r}) \cdot (0 \cdot 1 + 0 \cdot 0 + 0 \cdot 1) = 0,
\end{aligned}$$

*and similarly for the remaining cases* $\alpha \in C_{\bar{s}\tau r\tau\bar{s}\tau g\tau\bar{s}}$, $\alpha \in C_{\bar{s}\tau g\tau\bar{s}\tau r\tau\bar{s}}$, *and* $\alpha \in C_{\bar{s}\tau g\tau\bar{s}\tau g\tau\bar{s}}$. *This completes the justification of*

$$\mu_{\sigma,\bar{s}}(\{\, \alpha \in frags^*(\mathfrak{E}) \mid last(\alpha) = ●\,\} \setminus \{\bar{s}\tau r\tau\bar{s}\tau r a●, \bar{s}\tau g\tau\bar{s}\tau r a●\}) = 0.$$

*A similar analysis shows that* $\mu_{\sigma,\bar{s}}(\{\, \alpha \in frags^*(\mathfrak{E}) \mid last(\alpha) = ▲\,\}) = \mu(▲)$ *and* $\mu_{\sigma,\bar{s}}(\{\, \alpha \in frags^*(\mathfrak{E}) \mid last(\alpha) = ●\,\}) = \mu(●)$; *for each remaining state* $s \in \{\bar{s}, r, y, g\}$, *it is easy to verify that* $\mu_{\sigma,\bar{s}}(\{\, \alpha \in frags^*(\mathfrak{E}) \mid last(\alpha) = s\,\}) = 0$. *Regarding the first two properties of the definition of weak combined transition, we have that* $\mu_{\sigma,\bar{s}}(frags^*(\mathfrak{E})) = 1$ *follows directly from the third condition, as well as the second property by considering the trace of the finite execution fragments occurring with non-zero probability.* ♦

## 4.5 Interval Markov Decision Processes

Real-world systems operate in the presence of uncertainty. There are various types of uncertainty that can influence the modelling of complex (physical or artificial) systems. One of the most important types of uncertainty is the inherent imprecision that is introduced by measurement errors and discretization artifacts which necessarily happen due to incomplete knowledge about the system behaviour.

A severe limitation of Markov Decision Processes (*MDP*s) is that the probability values used in the transitions are specific, fixed values, which can have a considerable impact on the outcome of the model checking [KU02, NG05, Kat16]. In fact, due to measurement

uncertainties or modeling errors, it is possible to obtain two MDP models of an underlying (physical) system with slightly different values on the transition probabilities such that they give very different outcomes, even though the two MPDs model the same system. A favorable method for avoiding this issue is to include such uncertainties and errors in the model itself.

We now define *Interval Markov Decision Processes* (*IMDP*s) as an extension of *MDP*s, which allows for the inclusion of transition probability uncertainties as *intervals*. That is, the probabilities assigned by transition probability distributions to states are not fixed numbers; rather, they are known to lie within a given interval. *IMDP*s belong to the family of uncertain *MDP*s and allow to describe a set of *MDP*s with identical (graph) structures that differ in distributions associated with transitions. Formally,

**Definition 4.10 (Interval Markov decision processes).** *An* Interval Markov Decision Process *(IMDP)* $\mathcal{M}$ *is a tuple* $(S, \bar{s}, \mathscr{A}, \mathtt{AP}, L, I)$*, where* $S$ *is a finite set of* states, $\bar{s} \in S$ *is the initial state,* $\mathscr{A}$ *is a finite set of* actions, $\mathtt{AP}$ *is a finite set of* atomic propositions, $L : S \to 2^{\mathtt{AP}}$ *is a* labelling function*, and* $I \colon S \times \mathscr{A} \times S \to \mathbb{I} \cup \{[0,0]\}$ *is a total* interval transition probability function *with* $\mathbb{I} = \{[l,u] \subseteq \mathbb{R} \mid 0 < l \leq u \leq 1\}$*. We denote by* $[\mathcal{M}]$*, the class of all finite-state finite-transition IMDPs.*

We denote the set of available actions at state $s \in S$ by $\mathscr{A}(s)$. Furthermore, for each state $s$ and action $a \in \mathscr{A}(s)$, we write $s \xrightarrow{a} \mathfrak{h}_s^a$ if $\mathfrak{h}_s^a \in \mathrm{Disc}(S)$ is a *feasible distribution*, i.e. for each state $s' \in S$ we have $\mathfrak{h}_{ss'}^a = \mathfrak{h}_s^a(s') \in I(s,a,s')$. By $\mathscr{H}_s^a$, we denote the set of feasible distributions for state $s$ and action $a$. We require that the set $\mathscr{H}_s^a = \{\mathfrak{h}_s^a \mid s \xrightarrow{a} \mathfrak{h}_s^a\}$ is non-empty for each state $s$ and action $a \in \mathscr{A}(s)$. Hence, the set of actions that are enabled from $s$ can also be described as $\mathscr{A}(s) = \{a \in \mathscr{A} \mid \mathscr{H}_s^a \neq \emptyset\}$.

We extend $I$ to sets of states as follows: given $S' \subseteq S$, we let

$$I(s,a,S') = \left[\min\left\{1, \sum_{s' \in S'} \inf I(s,a,s')\right\}, \min\left\{1, \sum_{s' \in S'} \sup I(s,a,s')\right\}\right].$$

**Remark 4.4.** *The size of a given* $\mathcal{M}$ *is determined as follows. Let* $|S|$ *denote the number of states in* $\mathcal{M}$*. Then each state has* $\mathcal{O}(|\mathscr{A}|)$ *actions and at most* $\mathcal{O}(|\mathscr{A}| \cdot |S|)$ *transitions, each of which is associated with a probability interval. Therefore, the overall size of* $\mathcal{M}$ *i.e.,* $|\mathcal{M}|$ *is in* $\mathcal{O}(|S|^2 \, |\mathscr{A}|)$*.*

The formal semantics of an *IMDP* is as follows. A *path* in $\mathcal{M}$ is a finite or infinite sequence of states in the form $\xi = s_1 \mathfrak{h}_{s_1 s_2}^{a_1} s_2 \cdots$, where $s_1 = \bar{s}$ and for each $i \geq 1$, $s_i \in S$, $a_i \in \mathscr{A}(s_i)$, the transition probability $\mathfrak{h}_{s_i s_{i+1}}^{a_i} > 0$. Path $\xi$ can be finite or infinite. The sets of all finite and infinite paths in $\mathcal{M}$ are denoted by $Paths_{\mathcal{M}}^{fin}$ and $Paths_{\mathcal{M}}^{inf}$, respectively. The i-th state and action along the path $\xi$ are denoted by $\xi[i]$ and $\xi(i)$, respectively. For a finite path $\xi \in Paths_{\mathcal{M}}^{fin}$, let $last(\xi)$ indicate its last state. Moreover, let $Paths_{\mathcal{M}}^{\xi} = \{\xi' \in Paths_{\mathcal{M}}^{inf} \mid \xi \text{ is a prefix of } \xi'\}$ denote the set of infinite paths with the prefix $\xi \in Paths_{\mathcal{M}}^{fin}$ which is also known as the *cylinder* set of $\xi$.

In order to resolve nondeterministic transitions, schedulers and natures need to be defined for *IMDP*s. Intuitively, a scheduler is referred to every possible resolution of nondeterminism while a nature is referred to every resolution of uncertainty. Formally,

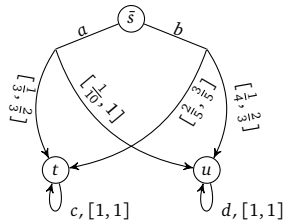



Figure 4.5: An example of *IMDP*s: the *IMDP* $\mathcal{M}$

**Definition 4.11** (**Scheduler and nature in *IMDP*s**). *Given an IMDP* $\mathcal{M}$*, a* scheduler *is a function* $\sigma : Paths^{fin}_{\mathcal{M}} \to \mathrm{Disc}(\mathscr{A})$ *that to each finite path* $\xi$ *assigns a distribution over the set of actions enabled by the last state of* $\xi$*, that is,* $\sigma(\xi) \in \mathrm{Disc}(\mathscr{A}(last(\xi))$*. A* nature *is a function* $\pi : Paths^{fin}_{\mathcal{M}} \times \mathscr{A} \to \mathrm{Disc}(S)$ *that to each finite path* $\xi$ *and action* $a \in \mathscr{A}(last(\xi))$ *assigns a feasible distribution, i.e. an element of* $\mathscr{H}^a_s$ *where* $s = last(\xi)$*. The sets of all schedulers and all natures of* $\mathcal{M}$ *are denoted by* $\Sigma$ *and* $\Pi$*, respectively.*

A scheduler $\sigma$ is said to be *deterministic* (**D**) if $\sigma(\xi) = \delta_a$ for all finite paths $\xi$ and some $a \in \mathscr{A}(last(\xi))$. Similarly, a nature is said to be *deterministic* if $\pi(\xi, a) = \delta_{\mathfrak{h}^a_{last(\xi)}}$ for all finite paths $\xi$, for all $a \in \mathscr{A}(last(\xi))$, and some $\mathfrak{h}^a_{last(\xi)} \in \mathscr{H}^a_{last(\xi)}$. Furthermore, a scheduler $\sigma$ (nature $\pi$) is *Markovian* (**M**) if it depends only on $last(\xi)$. Given a finite path $\xi$ of an *IMDP*, a scheduler $\sigma$, and a nature $\pi$, the system evolution proceeds as follows. First, an action $a \in \mathscr{A}(s_i)$, where $s_i = last(\xi)$, is chosen nondeterministically by $\sigma$. Then, $\pi$ resolves the uncertainties and chooses nondeterministically one feasible distribution $\mathfrak{h}^a_{s_i} \in \mathscr{H}^a_{s_i}$. Finally, the next state $s_{i+1}$ is chosen randomly according to the distribution $\mathfrak{h}^a_{s_i}$, and path $\xi$ is appended by $s_{i+1}$.

For a scheduler $\sigma$ and a nature $\pi$, let $\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}$ denote the unique probability measure over $(Paths^{inf}_{\mathcal{M}}, \mathscr{B})$ such that the probability $\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}[Paths^{s'}_{\mathcal{M}}]$ of starting in $s'$ equals 1 if $s' = \bar{s}$ and 0, otherwise; and the probability $\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}[Paths^{\xi \mathfrak{h}^a_{last(\xi)s'} s'}_{\mathcal{M}}]$ of traversing a finite path $\xi \mathfrak{h}^a_{last(\xi)s'} s'$ equals $\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}[Paths^{\xi \mathfrak{h}^a_{last(\xi)s'} s'}_{\mathcal{M}}] = \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}[Paths^{\xi}_{\mathcal{M}}] \cdot \sigma(\xi)(a) \cdot \pi(\xi, a)(s')$. Here, $\mathscr{B}$ is the standard $\sigma$-algebra over $Paths^{inf}_{\mathcal{M}}$ generated from the set of all cylinder sets $\{Paths^{\xi}_{\mathcal{M}} \mid \xi \in Paths^{fin}_{\mathcal{M}}\}$. The unique probability measure is obtained by the application of the extension theorem (see, e.g. [Bil79]).

It is worthwhile to note that the scheduler does not choose an action but a *distribution* over actions. Such a randomization typically simplifies and speeds up the procedure of solving difficult problems. For instance, it is well-known that randomization is useful in the context of bisimulations as it allows to define coarser equivalence relations [Seg95]. To the contrary, nature is not allowed to randomize over the set of feasible distributions $\mathscr{H}^a_s$. This is in fact not necessary, since the set $\mathscr{H}^a_s$ is closed under convex combinations.

In order to avoid ambiguity, we sometimes describe the *IMDP* $\mathcal{M}$ as a tuple $(S_{\mathcal{M}}, \bar{s}_{\mathcal{M}}, \mathscr{A}_{\mathcal{M}}, \mathtt{AP}_{\mathcal{M}}, L_{\mathcal{M}}, I_{\mathcal{M}})$ by adding the *IMDP* model symbol as a subindex to its generic elements.

**Example 4.6.** *An instance of IMDPs is depicted in Figure 4.5. The set of states is* $S = \{\bar{s}, t, u\}$ *with*

*$\bar{s}$ being the initial one. Furthermore, let* $\mathtt{AP} = \{\mathfrak{u}, \mathfrak{v}\}$ *and* $L(\bar{s}) = \{\mathfrak{v}\}$, $L(t) = \{\mathfrak{u}\}$ *and* $L(u) = \{\mathfrak{u}, \mathfrak{v}\}$. *The transition probability intervals are* $I(\bar{s}, a, t) = [\frac{1}{3}, \frac{2}{3}]$, $I(\bar{s}, a, u) = [\frac{1}{10}, 1]$, $I(\bar{s}, b, t) = [\frac{2}{5}, \frac{3}{5}]$, $I(\bar{s}, b, u) = [\frac{1}{4}, \frac{2}{3}]$, $I(t, c, t) = [1, 1]$ *and* $I(u, d, u) = [1, 1]$. ♦

## 4.6 Concluding Remarks

In this chapter, we have discussed the preliminaries of probabilistic systems that are essential for the remainder of the thesis. More details about *MC*s can be found in the textbooks [KS76, Kul96, KNP07]. *MDP*s are discussed in a greater detail in [How71, Ber95, Put05, FKNP11] and the textbook [Put05]. A complete treatment of *PA*s, along with their fundamental properties can be found in [Seg95]. Additional information about parallel composition operator for probabilistic systems can be found in [SDV04].

As regards the parametric probabilistic systems, *IMDP* models have been introduced and analyzed with respect to several quantitative properties in [WK08, NG05, WTM12, PLSVS13]. *IMDP*s extend classical *MDP*s where uncertainty is represented by intervals of probability values. Compositional minimization of probabilistic systems is well understood in the literature; however, it has received less attention for parametric probabilistic systems. This thesis extends the notion of parallel composition for probabilistic systems with parameter uncertainty. These contributions are described in detail in the following chapters.

# Part II

# Modelling and Performance Analysis of Probabilistic Systems

CHAPTER 5

# Efficiency of Deciding Probabilistic Automata Weak Bisimulation

In this chapter, we address efficiency analysis of deciding weak probabilistic bisimulation for probabilistic automata which is known to be in class **P** [TH15] based on a reduction to a linear programming problem. We analyze the efficiency of solving the decision problem in two dimensions. In the first dimension, we consider the theoretical efficiency of solving the problem. In particular, we study the complexity of the decision problem together with several optimizations and give an upper bound on the complexity of checking the feasibility of the original LP problem [TH15].

In the second dimension, we discuss the practical efficiency of solving the decision problem and show how it can exploit the problem structure. We also present an implementation which can use either linear programming solvers or an SMT solver and can be used to minimize probabilistic automata under weak probabilistic bisimulation. Such a minimization has clear implications for reducing the state space explosion problem when model checking such automata. To further mitigate this problem we investigate how to use this approach in a compositional manner when systems are expressed as the parallel composition of a number of sub-automata. The implementation is tested on a number of case studies both to analyze different optimizations and the advantages of using a compositional approach.

The material presented in this chapter is an extended version of the results reported in [HHT13, FFHHT16].

**Organization of the chapter.** We start in Section 5.1 and introduce weak probabilistic bisimulation for probabilistic automata. In Section 5.2, we show how to compute the weak probabilistic bisimulation and how to minimize an automaton. We devote Section 5.3 to the LP problem construction and in Section 5.4 we focus on the efficiency of solving the LP problem. Section 5.5 presents implementation considerations together with several cases studies showing the effectiveness of the minimisation in particular for compositional analysis.

## 5.1 Weak Probabilistic Bisimulation

Bisimulation relations constitute a powerful tool that allows us to verify whether two models describe essentially the same real system. Moreover, they allow us to compute the minimal automaton that is bisimilar to the given one [EHS+13]. We now recall the definition of weak probabilistic bisimulation [Seg95, Seg06], that is the relation that allows us to abstract away from internal computations while solving nondeterministic choices via convex combinations of the available transitions.

**Definition 5.1** (**Weak probabilistic bisimulation**). *Given a PA $\mathfrak{P}$, an equivalence relation $\mathcal{R}$ on $S$ is a* weak probabilistic bisimulation *if, for each pair of states $s, t \in S$ such that $s \mathcal{R} t$, if $s \xrightarrow{a} \mu_s$ for some probability measure $\mu_s$, then there exists a probability measure $\mu_t$ such that $t \stackrel{a}{\Longrightarrow}_c \mu_t$ and $\mu_s \mathcal{L}(\mathcal{R}) \mu_t$.*

In the following, we may refer to the condition "there exists $\mu_t$ such that $t \stackrel{a}{\Longrightarrow}_c \mu_t$ and $\mu_s \mathcal{L}(\mathcal{R}) \mu_t$" as the *step condition* of the bisimulation. Specially, when the bisimulation is seen as a two-player game between the two automata, the *step condition* is the condition on the weak transition (or weak step) performed by the *defender* state $t$ while matching the transition (or step) performed by the *challenger* state $s$.

To check whether two *PA*s $\mathfrak{P}_1$ and $\mathfrak{P}_2$ are weak probabilistic bisimilar, we can either adapt the above definition to work with pairs of automata, or we can just consider the *PA* $\mathfrak{P} = \mathfrak{P}_1 \uplus \mathfrak{P}_2$ such that $S = S_1 \uplus S_2$, $\bar{s} = \bar{s}_1$, $H = H_1 \cup H_2$, $E = E_1 \cup E_2$, $T = T_1 \uplus T_2$. Note that the choice $\bar{s} = \bar{s}_1$ is arbitrary, since it does not affect the weak probabilistic bisimulation; similarly, we can ignore the requirement $E \cap H = \emptyset$ since actions are taken into account by the step condition: if the same action is external for $\mathfrak{P}_1$ and internal for $\mathfrak{P}_2$, then $\mathfrak{P}_1$ and $\mathfrak{P}_2$ are not bisimilar since the external transition proposed by $\mathfrak{P}_1$ can not be matched by $\mathfrak{P}_2$. Deciding whether two automata are bisimilar then reduces to computing the bisimulation $\mathcal{R}$ on $\mathfrak{P}$ and to checking whether their start states are related by $\mathcal{R}$, i.e., whether $\bar{s}_1 \mathcal{R} \bar{s}_2$.

**Definition 5.2** (***PA*s weak probabilistic bisimulation**). *Given two PAs $\mathfrak{P}_1$ and $\mathfrak{P}_2$, we say that $\mathfrak{P}_1$ and $\mathfrak{P}_2$ are* weakly probabilistic bisimilar *if there exists a weak probabilistic bisimulation $\mathcal{R}$ on $S_1 \uplus S_2$ such that $\bar{s}_1 \mathcal{R} \bar{s}_2$. We denote the coarsest weak probabilistic bisimulation by $\approx$, and call it weak probabilistic bisimilarity.*

Weak probabilistic bisimilarity is an equivalence relation preserved by standard process algebraic composition operators on *PA* [PS04], such as parallel composition, action hiding, action renaming, and action prefixing. As we will see in the next section, the complexity of deciding $\mathfrak{P}_1 \approx \mathfrak{P}_2$ strictly depends on finding the matching weak combined transition $t \stackrel{a}{\Longrightarrow}_c \mu_t$ for which determinate schedulers suffice (cf. [CS02, Proposition 3]): in Section 5.3 we will show how to find them in polynomial time.

**Remark 5.1.** *In this chapter we do not consider the weak bisimulation relation obtained by restricting to weak transitions $t \stackrel{a}{\Longrightarrow} \mu_t$ induced by a deterministic (or Dirac) scheduler, i.e., by a scheduler $\sigma$ such that for each finite execution fragment $\alpha$, either $\sigma(\alpha) = \delta_{tr}$ for some $tr \in T$, or $\sigma(\alpha) = \delta_{\perp}$. In fact, as shown in [Den05], the resulting bisimulation is not transitive and this makes the usual compositional minimization approach much more difficult to use. In such an approach a given automaton $\mathfrak{P}_0$ is decomposed into multiple sub-automata running in parallel, i.e.,*

$\mathfrak{P}_0 = \mathfrak{B}_1 \parallel \mathfrak{B}_2 \parallel \ldots \parallel \mathfrak{B}_n$*; then one component* $\mathfrak{B}_i$ *at a time is replaced by another component* $\mathfrak{B}'_i$ *that is bisimilar to but smaller than* $\mathfrak{B}_i$*. This gives rise to a sequence of automata* $\mathfrak{P}_0, \mathfrak{P}_1, \ldots, \mathfrak{P}_n$ *such that for each* $0 \leq i < n$, $\mathfrak{P}_i$ *and* $\mathfrak{P}_{i+1}$ *are bisimilar. If the bisimulation relation is not transitive, then we can not derive that* $\mathfrak{P}_0$ *and* $\mathfrak{P}_n$ *are bisimilar. Instead, we have to provide a relation witnessing the bisimilarity of* $\mathfrak{P}_0$ *and* $\mathfrak{P}_n$*. Moreover, the construction we present in Section 5.3 to efficiently find a weak combined transition is not easily extendable to weak (non-combined) transitions; see Remark 5.3 for a more detailed explanation.*

Since in this chapter we consider only weak combined transitions and weak probabilistic bisimulation and bisimilarity, from now on we omit the adjectives "combined" and "probabilistic", respectively.

## 5.2 Computing the Weak Bisimilarity for Minimizing Automata

In this section, we recast the decision procedure of [CS02] that decides whether two probabilistic automata $\mathfrak{P}_1$ and $\mathfrak{P}_2$ are weak bisimilar by following the standard partition refinement approach [KS90, PT87, PLS00].

### 5.2.1 Deciding Weak Bisimilarity

We now study in detail the decision procedure for the weak bisimulation and then we analyse the complexity of the algorithm.

#### 5.2.1.1 Weak Bisimilarity Decision Algorithm

The decision algorithm for the weak bisimulation is sketched in Figure 5.1; the procedure Quotient iteratively constructs the set $S/\approx$, the set of equivalence classes of states $S$ under $\approx$, starting with the partitioning $\mathcal{R} = \{S\}$ and refining it until $\mathcal{R}$ satisfies the definition of weak bisimulation and thus the resulting partitioning is the coarsest one, i.e., we compute the weak bisimilarity. In the following, we treat $\mathcal{R}$ both as a set of partitions and as an equivalence relation without further mention.

The partitioning is refined by procedure Refine into a finer partitioning as long as there is a partition containing two states that violate the bisimulation condition, which is checked for in procedure FindSplit. Procedure Refine splits the partition $[s]_{\mathcal{R}}$ into two new partitions $\mathcal{C}_s$ and $\mathcal{C}_{\neg s}$ according to the discriminating information $(s, a, \mu_s)$ identified by FindSplit before. More precisely, $\mathcal{C}_s$ contains all states belonging to $[s]_{\mathcal{R}}$ that are able to match $(s, a, \mu_s)$, while $\mathcal{C}_{\neg s}$ contains the remaining states in $[s]_{\mathcal{R}}$ that fail to match $(s, a, \mu_s)$. It is clear that at the termination of the **for** loop at line 2 of Refine, both $\mathcal{C}_s$ and $\mathcal{C}_{\neg s}$ are not empty: $\mathcal{C}_s$ obviously contains the state $s$ while $\mathcal{C}_{\neg s}$ contains for sure the state $t$ that caused FindSplit to return $(s, a, \mu_s)$ at line 4. So far, the procedure essentially agrees with the *DecideBisim*$(\mathfrak{P}_1, \mathfrak{P}_2)$ procedure of [CS02].

The real difference between the decision procedure we provide here and the one presented in [CS02] however appears inside the procedure FindSplit, where we check directly the step condition by looking for a weak transition $t \overset{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ such that $\mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t$, instead

**Algorithm 1:** Quotient($\mathfrak{P}$)

**Input:** Probabilistic automaton $\mathfrak{P}$
**Output:** Weak bisimulation $\mathcal{R}$

1 **begin**
2 $\quad\mathcal{R} = \{S\}$;
3 $\quad(s, a, \mu_s) = \text{FindSplit}(\mathcal{R})$;
4 $\quad$**while** $s \neq \bot$ **do**
5 $\quad\quad\mathcal{R} = \text{Refine}(\mathcal{R}, (s, a, \mu_s))$;
6 $\quad\quad(s, a, \mu_s) = \text{FindSplit}(\mathcal{R})$;
7 $\quad$**return** $\mathcal{R}$;

**Procedure 2:** FindSplit($\mathcal{R}$)

**Input:** An equivalence relation $\mathcal{R}$
**Output:** A splitter for the relation $\mathcal{R}$

1 **begin**
2 $\quad$**forall** $s \in S$ **do**
3 $\quad\quad$**forall** $(s, a, \mu_s) \in T$ **do**
4 $\quad\quad\quad$**forall** $t \in [s]_{\mathcal{R}}$ **do**
5 $\quad\quad\quad\quad$**if** *there does not exist* $t \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ *such that* $\mu_s \, \mathcal{L}(\mathcal{R}) \, \mu_t$ **then**
6 $\quad\quad\quad\quad\quad$**return** $(s, a, \mu_s)$;
7 $\quad$**return** $(\bot, \tau, \delta_{\bar{s}})$;

**Procedure 3:** Refine($\mathcal{R}, (s, a, \mu_s)$)

**Input:** An equivalence relation $\mathcal{R}$ and a splitter $(s, a, \mu_s)$
**Output:** Refinement of $\mathcal{R}$

1 **begin**
2 $\quad\mathcal{C}_s = \mathcal{C}_{\neg s} = \emptyset$;
3 $\quad$**forall** $t \in [s]_{\mathcal{R}}$ **do**
4 $\quad\quad$**if** *there exists* $t \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ *such that* $\mu_s \, \mathcal{L}(\mathcal{R}) \, \mu_t$ **then**
5 $\quad\quad\quad\mathcal{C}_s = \mathcal{C}_s \cup \{t\}$
6 $\quad\quad$**else**
7 $\quad\quad\quad\mathcal{C}_{\neg s} = \mathcal{C}_{\neg s} \cup \{t\}$
8 $\quad$**return** $\mathcal{R} \setminus \{[s]_{\mathcal{R}}\} \cup \{\mathcal{C}_s, \mathcal{C}_{\neg s}\}$;

Figure 5.1: The decision algorithm for the weak bisimilarity

of computing the information associated by $a$ to $s$ and $t$, i.e., the set with respect to $\mathcal{R}$ of the probability measures reached from $s$ (and $t$) via a weak transition labelled by $a$.

**Remark 5.2.** *In the context of model checking, the definition of bisimulation usually requires that two related states are labelled with identical sets of atomic propositions. The decision procedure presented in Figure 5.1 can be easily adapted to such a definition by modifying line 1 of* Quotient *as follows: the initial partitioning $\mathcal{R}$ is such that for each class $\mathcal{C}$ of $\mathcal{R}$, $s, s' \in \mathcal{C}$ if and only if $s$ and $s'$ are labelled with identical sets of atomic propositions.*

### 5.2.1.2 Complexity of the Decision Algorithm

Assume we are given the *PA* $\mathfrak{P}$; let $N = |\mathfrak{P}|$. The **for** loop at line 2 of the procedure FindSplit cycles at most $N$ times. Now, consider the **for** loop at line 3: since $T = \bigcup_{s \in S} \{ tr \in T \mid src(tr) = s \}$ and $\{ tr \in T \mid src(tr) = s \} \cap \{ tr \in T \mid src(tr) = t \} = \emptyset$ for each $s, t \in S$ with $s \neq t$, it follows that the two **for** loops together cycle at most $N$ times. In the worst case (that occurs when

$[s]_{\mathcal{R}} = S$ and each state $t$ satisfies the step condition), the **for** loop at line 4 cycles at most $N$ times as well. This means that the existential check of $t \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ such that $\mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t$ at line 5 is performed at most $N^2$ times. Let $W(N)$ be the complexity of such check which will be discussed later; it is immediate to see that FindSplit $\in \mathcal{O}(N^2 \cdot W(N))$.

The **for** loop in procedure Refine can be performed at most $N$ times; this happens when $[s]_{\mathcal{R}} = S$. In each loop, an instance of the existential check of $t \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ such that $\mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t$ has to be computed, with complexity $W(N)$; the resulting complexity of Refine is therefore $\mathcal{O}(N \cdot W(N))$.

The **while** loop in the procedure Quotient can be performed at most $N$ times; this happens when in each loop the procedure FindSplit returns $(s, a, \mu_s)$ where $s \neq \bot$, that is, not every pair of states in $[s]_{\mathcal{R}}$ satisfies the step condition. Since in each loop the procedure Refine replaces such class $[s]_{\mathcal{R}}$ with two non-empty classes $\mathcal{C}_s$ and $\mathcal{C}_{\neg s}$, after at most $N$ loops every class contains a single state and the procedure FindSplit returns $(\bot, \tau, \delta_{\bar{s}})$ since each transition $s \xrightarrow{a} \mu_s$ is obviously matched by $s$ itself. Since Refine has complexity $\mathcal{O}(N \cdot W(N))$ and FindSplit $\mathcal{O}(N^2 \cdot W(N))$, it follows that the overall complexity of Quotient is $\mathcal{O}(N \cdot (N^2 \cdot W(N) + N \cdot W(N))) = \mathcal{O}(N^3 \cdot W(N))$.

**Proposition 5.1.** *Given two PAs $\mathfrak{P}_1$ and $\mathfrak{P}_2$, let $S = S_1 \uplus S_2$ and $N = |\mathfrak{P}_1| + |\mathfrak{P}_2|$; given a state $t \in S$, an action $a \in \mathscr{A}$, the probability measures $\mu_s, \mu_t \in \mathrm{Disc}(S)$, and an equivalence relation $\mathcal{R}$ on $S$, let $W(N)$ be the complexity of checking the existence of $t \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ such that $\mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t$. Checking $\mathfrak{P}_1 \approx \mathfrak{P}_2$ has complexity $\mathcal{O}(N^3 \cdot W(N))$.*

*Proof.* Immediate by the previous analysis. ∎

### 5.2.2 Minimization and Parallel Composition

In this section, we explain in detail the practical steps that lead from a *PA* $\mathfrak{P}$ to the minimal automaton $\mathfrak{M}$ that is weak bisimilar to $\mathfrak{P}$, as formalized in [EHS+13, HK00, CGM+96]: the first step extracts the reachable fragment $\mathfrak{P}_{\circlearrowleft}$ of $\mathfrak{P}$, i.e., the states and the corresponding transitions that can be reached with non-zero probability from the start state. [1] The second step generates the quotient automaton by computing the weak bisimilarity $\approx$. Once $\approx$ is at hand, the quotient automaton $[\mathfrak{P}_{\circlearrowleft}]_{\approx}$ is extracted in a third step: it has as set of states the set of equivalence classes of $\approx$ and as the start state the class of $\bar{s}$; the sets of internal and external actions are the same as in $\mathfrak{P}_{\circlearrowleft}$ while the transition relation contains only the transitions $[s]_{\approx} \xrightarrow{a} \rho$ such that there exists $s \xrightarrow{a} \mu \in T_{\circlearrowleft}$ where $\rho(\mathcal{C}) = \sum_{t \in \mathcal{C}} \mu(t)$ for each $\mathcal{C} \in [S_{\circlearrowleft}]_{\approx}$. The fourth step of the minimization procedure removes from $[\mathfrak{P}_{\circlearrowleft}]_{\approx}$ the transitions that are redundant, i.e., the transitions that can be removed from the automaton since they can be weakly matched by the remaining transitions; the fifth and final step normalizes the internal transitions, i.e., each transition $s \xrightarrow{\tau} \mu$ is replaced by $s \xrightarrow{\tau} \mu \backslash s$. Note that the fourth step ensures that there are no transitions $s \xrightarrow{\tau} \delta_s$ since they are trivially redundant.

The correctness of the above construction is justified by the following properties of weak probabilistic bisimulation: let $A$ be a set of actions and $\mathfrak{P}$, $\mathfrak{P}'$, $\mathfrak{P}''$, and $\mathfrak{P}_e$ be four *PA*s such that $\mathfrak{P}_e$ is compatible with both $\mathfrak{P}$ and $\mathfrak{P}'$. Then the following holds:

- $\approx$ is transitive [Seg95]: if $\mathfrak{P} \approx \mathfrak{P}'$ and $\mathfrak{P}' \approx \mathfrak{P}''$, then $\mathfrak{P} \approx \mathfrak{P}''$;

[1] Note that by our assumptions on the automata we do not need this initial step.

- $\approx$ is preserved by parallel composition [Seg95]: if $\mathfrak{P} \approx \mathfrak{P}'$, then $\mathfrak{P} \parallel \mathfrak{P}_e \approx \mathfrak{P}' \parallel \mathfrak{P}_e$;
- $\approx$ is preserved by the hiding operator: if $\mathfrak{P} \approx \mathfrak{P}'$, then $\text{Hide}_A(\mathfrak{P}) \approx \text{Hide}_A(\mathfrak{P}')$;
- $\mathfrak{P} \approx \mathfrak{P}_{\circlearrowleft}$ [EHS+13];
- $\mathfrak{P} \approx [\mathfrak{P}]_{\approx}$ [EHS+13];
- removing redundant transitions preserves weak bisimilarity [EHS+13]; and
- normalizing internal transitions preserves weak bisimilarity [EHS+13].

The main computational bottleneck of this overall minimization procedure applied to an automaton $\mathfrak{P}$ is the second step, the weak bisimulation computation, that we have already seen by Proposition 5.1 to be $\mathcal{O}(N^3 \cdot W(N))$.

Therefore, this bottleneck has to be carefully considered, with respect to the size of the models to be processed by it: when we want to minimize a large automaton that is the result of the parallel composition of several smaller automata, according to the definition of parallel composition, the resulting state space is the Cartesian product of the single state spaces. This means that the state space of the composed automaton grows exponentially in the number of components, in particular when they are different instances of the same system, leading quickly to prohibitively large automata. However, it is quite common to generate in this way states and transitions that are actually useless since they are not reachable from the start state of the composed automaton, in particular when the resulting transition has as label an internal action. For instance, suppose that we have a transition $s_1 \xrightarrow{\tau} \mu_1 \in T_1$. According to the definition of parallel composition, for each $s_2 \in S_2$ we have to generate the transition $(s_1, s_2) \xrightarrow{\tau} \mu_1 \times \delta_{s_2}$, even when $(s_1, s_2)$ can not be reached from $(\bar{s}_1, \bar{s}_2)$. To alleviate the fast growth of the parallel composition it is advisable to generate only the reachable fragment or adopt more advanced techniques [GSL96, KM00].

Furthermore, consider the two *PA*s $\mathfrak{P}_1$ and $\mathfrak{P}_2$ such that their only transitions are $\{s \xrightarrow{\tau} \delta_t, t \xrightarrow{a} \delta_t\}$ and $\{x \xrightarrow{a} \delta_y, y \xrightarrow{a} \delta_y\}$, respectively: it is immediate to see that both automata are weak bisimilar to $\mathfrak{P}_3$ whose only transition is $v \xrightarrow{a} \delta_v$ and that $\mathfrak{P}_1 \parallel \mathfrak{P}_2$ is weak bisimilar to $\mathfrak{P}_3 \parallel \mathfrak{P}_3$ whose only transition is $(v, v) \xrightarrow{a} \delta_{(v,v)}$. Such weak bisimilarity between $\mathfrak{P}_1 \parallel \mathfrak{P}_2$ and $\mathfrak{P}_3 \parallel \mathfrak{P}_3$ is not fortuitous but derives from the fact that the weak bisimulation is preserved by the parallel composition. In fact, for any pair of compatible *PA*s $\mathfrak{P}_1$ and $\mathfrak{P}_2$, we have that $\mathfrak{P}_1 \parallel \mathfrak{P}_2 \approx [\mathfrak{P}_1]_{\approx} \parallel \mathfrak{P}_2 \approx [\mathfrak{P}_1]_{\approx} \parallel [\mathfrak{P}_2]_{\approx}$. The first bisimulation is justified by taking $\mathfrak{P}_2$ as context and the fact that $\mathfrak{P}_1 \approx [\mathfrak{P}_1]_{\approx}$, and similarly for the second bisimulation. The compatibility of the pair of automata we compose is ensured by the fact that an automaton and its quotient have the same sets of actions. In general $[\mathfrak{P}_1]_{\approx} \parallel [\mathfrak{P}_2]_{\approx}$ is not the minimal automaton that is weak bisimilar to $\mathfrak{P}_1 \parallel \mathfrak{P}_2$: in fact, the presence of internal transitions may lead to symmetric constructions that are identified and collapsed by computing the weak bisimulation. For instance, suppose that we have the states $s_1$ and $s_2$ enabling the transitions $s_1 \xrightarrow{\tau} \delta_{s_1'}$, $s_1 \xrightarrow{a} \mu_1$, and $s_1' \xrightarrow{b} \mu_1'$ and $s_2 \xrightarrow{\tau} \delta_{s_2'}$, $s_2 \xrightarrow{a} \mu_2$, and $s_2' \xrightarrow{b} \mu_2'$, respectively. In the parallel composition we obtain the four internal transitions $(s_1, s_2) \xrightarrow{\tau} \delta_{(s_1', s_2)}$, $(s_1, s_2) \xrightarrow{\tau} \delta_{(s_1, s_2')}$, $(s_1', s_2) \xrightarrow{\tau} \delta_{(s_1', s_2')}$, and $(s_1, s_2') \xrightarrow{\tau} \delta_{(s_1', s_2')}$ and the two external transitions $(s_1, s_2) \xrightarrow{a} \mu_1 \times \mu_2$ and $(s_1', s_2') \xrightarrow{b} \mu_1' \times \mu_2'$. It is clear that the states $(s_1', s_2)$, $(s_1, s_2')$, and $(s_1', s_2')$ are weak bisimilar, so they can be collapsed. Applying the hiding operator after a parallel composition increases this effect considerably.

## 5.3 Weak Transition Construction as a Linear Programming Problem

As discussed in the previous section, the main source of the worst case behaviour of the decision algorithms [TH15, CS02] for *PA* weak probabilistic bisimulation is the recurring need to check for the existence of a weak transition. This is solved with an exponential algorithm in [CS02] and a polynomial algorithm in [TH15]. The latter approach takes inspiration from network flow problems: a weak transition $t \overset{a}{\Longrightarrow}_c \mu_t$ of a *PA* $\mathfrak{P}$ is described as an enriched flow problem in which the initial probability mass $\delta_t$ splits along internal transitions, and precisely one external transition with label $a \neq \tau$ for every stream, in order to reach $\mu_t$. The enriched flow problem is then translated into a Linear Programming (LP) problem extended with *balancing constraints* that encode the need to respect transition probability measure.

### 5.3.1 Network Construction

To describe the structure of the enriched LP problem, we first recall the definition of the network graph corresponding to a weak transition.

**Definition 5.3** (**Weak transition network graph** (cf. [TH15, Sect. 5.2])). *Given a PA $\mathfrak{P}$, a state $t$, an action $a$, a probability measure $\mu$, and an equivalence relation $\mathcal{R}$ on $S$, the network graph $G(t, a, \mu, \mathcal{R}) = (V, E)$ relative to the weak transition $t \overset{a}{\Longrightarrow}_c \mu_t$ is defined as follows. Given $v \in S$, $a \in E$, and $tr \in T$, let $v_a$, $v^{tr}$, and $v_a^{tr}$ be three copies of $v$. For $a \in E$, the set $V$ of vertices is*

$$V = \{\triangle, \blacktriangledown\} \cup S \cup S^{tr} \cup S_a \cup S_a^{tr} \cup S/\mathcal{R}$$

*where*

$$\begin{aligned} S^{tr} &= \{\, v^{tr} \mid tr = v \xrightarrow{b} \rho \in T, b \in \{a\} \cup H \,\}, \\ S_a &= \{\, v_a \mid v \in S \,\}, \text{ and} \\ S_a^{tr} &= \{\, v_a^{tr} \mid v^{tr} \in S^{tr} \,\} \end{aligned}$$

*and the set $E$ of arcs is*

$$E = \{(\triangle, t)\} \cup L_1 \cup L_a \cup L_2 \cup L_{\mathcal{R}}^a$$

*where*

$$\begin{aligned} L_1 &= \{\, (v, v^{tr}), (v^{tr}, v') \mid tr = v \xrightarrow{\tau} \rho \in T, v' \in \mathrm{Supp}(\rho) \,\}, \\ L_a &= \{\, (v, v_a^{tr}), (v_a^{tr}, v_a') \mid tr = v \xrightarrow{a} \rho \in T, v' \in \mathrm{Supp}(\rho) \,\}, \\ L_2 &= \{\, (v_a, v_a^{tr}), (v_a^{tr}, v_a') \mid tr = v \xrightarrow{\tau} \rho \in T, v' \in \mathrm{Supp}(\rho) \,\}, \text{ and} \\ L_{\mathcal{R}}^a &= \{\, (v_a, \mathcal{C}), (\mathcal{C}, \blacktriangledown) \mid \mathcal{C} \in S/\mathcal{R}, v \in \mathcal{C} \,\}. \end{aligned}$$

*For $a \in H$ the definition is similar:*

$$V = \{\triangle, \blacktriangledown\} \cup S \cup S^{tr} \cup S_{\mathcal{R}} \cup S/\mathcal{R}$$

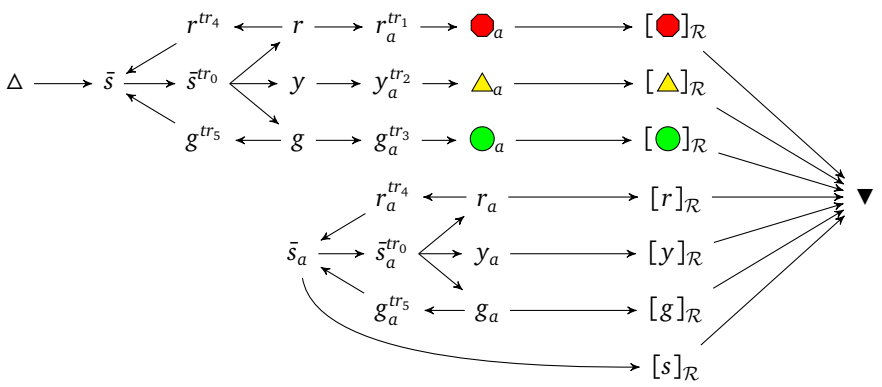



Figure 5.2: The network $G(\bar{s}, a, \mu, \mathcal{R})$ of Example 5.1

*and*

$$E = \{(\Delta, t)\} \cup L_1 \cup L_\perp \cup L_{\mathcal{R}},$$

*where* $L_{\mathcal{R}} = \{(v, \mathcal{C}), (\mathcal{C}, \blacktriangledown) \mid \mathcal{C} \in S/\mathcal{R}, v \in \mathcal{C}\}$.

We refer to the elements of $S \cup S_a$ as state nodes, of $\mathcal{T} = S^{tr} \cup S_a^{tr}$ as transition nodes, and of $S/\mathcal{R}$ as class nodes.

**Example 5.1.** *Consider again the PA* $\mathfrak{E}$ *in Figure 4.4 and suppose that we want to check whether there exists a weak transition* $\bar{s} \stackrel{a}{\Longrightarrow}_c \rho$ *such that* $\rho \; \mathcal{L}(\mathcal{R}) \; \mu$ *where* $\mu = \{(\bigcirc, \frac{9}{50}), (\triangle, \frac{8}{50}), (\bigcirc, \frac{33}{50})\}$ *and* $\mathcal{R} = \mathcal{I}$. *Note that this implies that* $\rho = \mu$. *Denote as usual the transitions of* $\mathfrak{E}$ *as follows:* $tr_0 = \bar{s} \stackrel{\tau}{\longrightarrow} \{(r, 0.3), (y, 0.1), (g, 0.6)\}$, $tr_1 = r \stackrel{a}{\longrightarrow} \delta_{\bigcirc}$, $tr_2 = y \stackrel{a}{\longrightarrow} \delta_{\triangle}$, $tr_3 = g \stackrel{a}{\longrightarrow} \delta_{\bigcirc}$, $tr_4 = r \stackrel{\tau}{\longrightarrow} \delta_{\bar{s}}$, *and* $tr_5 = g \stackrel{\tau}{\longrightarrow} \delta_{\bar{s}}$. *The network* $G(\bar{s}, a, \mu, \mathcal{R})$ *is shown in Figure 5.2, where we omit the state vertices* $\bigcirc$, $\triangle$, *and* $\bigcirc$ *as well as the transition vertices* $r^{tr_1}$, $y^{tr_2}$, *and* $g^{tr_3}$ *since they are not involved in any arc of the network.*

*It is worthwhile to note that for* $a \in E$, *each path in the network graph from* $\Delta$ *to* $\blacktriangledown$ *has to pass through a transition vertex* $v_a^{tr}$ *where* $act(tr) = a$, *i.e.,* $r_a^{tr_1}$, $y_a^{tr_2}$, *or* $g_a^{tr_3}$. *This construction ensures that the external action is performed with probability* 1. ⧫

### 5.3.2 LP Problem Construction

As pointed out in [TH15], the fact that the network admits a flow that respects the probability measure $\mu_t$ does by itself not imply the existence of a corresponding weak transition, because the flow may not respect probability ratios. To account for the latter, the network is converted into a linear programming problem for which the feasibility is shown to be equivalent to the existence of the desired weak transition. The idea is to convert the flow network into the canonical LP problem and then add the balancing constraints that force the "flow" to split according to transition probability measures.

**Definition 5.4** (**LP of the network graph** (cf. [TH15, Definition 7])). *Given a PA $\mathfrak{P}$, a state $t \in S$, an action $a \in \mathscr{A}$, a probability measure $\mu \in \mathrm{Disc}(S)$, and a binary relation $\mathcal{R}$ on $S$, for $a \in E$ we define the LP problem $LP(t, a, \mu, \mathcal{R})$ associated to the network graph $(V, E) = G(t, a, \mu, \mathcal{R})$ as follows.*

$$
\begin{array}{rll}
\max & \sum_{(u,v)\in E} -f_{u,v} & \\
\textit{subject to:} & f_{u,v} \geq 0 & \textit{for each } (u,v) \in E \\
& f_{\Delta,t} = 1 & \\
& f_{\mathcal{C},\blacktriangledown} = \mu(\mathcal{C}) & \textit{for each } \mathcal{C} \in S/\mathcal{R} \\
& \sum_{(u,v)\in E} f_{u,v} - \sum_{(v,w)\in E} f_{v,w} = 0 & \textit{for each } v \in V \setminus \{\Delta, \blacktriangledown\} \\
& f_{v^{tr},v'} - \rho(v') \cdot f_{v,v^{tr}} = 0 & \textit{for each } tr = v \xrightarrow{\tau} \rho \in T \textit{ and } v' \in \mathrm{Supp}(\rho) \\
& f_{v^{tr}_a,v'_a} - \rho(v') \cdot f_{v_a,v^{tr}_a} = 0 & \textit{for each } tr = v \xrightarrow{\tau} \rho \in T \textit{ and } v' \in \mathrm{Supp}(\rho) \\
& f_{v^{tr}_a,v'_a} - \rho(v') \cdot f_{v,v^{tr}_a} = 0 & \textit{for each } tr = v \xrightarrow{a} \rho \in T \textit{ and } v' \in \mathrm{Supp}(\rho)
\end{array}
$$

*When $a \in H$, the LP problem $LP(t, a, \mu, \mathcal{R})$ associated to $G(t, a, \mu, \mathcal{R})$ is defined as above without the last two groups of constraints:*

$$
\begin{array}{rll}
\max & \sum_{(u,v)\in E} -f_{u,v} & \\
\textit{subject to:} & f_{u,v} \geq 0 & \textit{for each } (u,v) \in E \\
& f_{\Delta,t} = 1 & \\
& f_{\mathcal{C},\blacktriangledown} = \mu(\mathcal{C}) & \textit{for each } \mathcal{C} \in S/\mathcal{R} \\
& \sum_{(u,v)\in E} f_{u,v} - \sum_{(v,w)\in E} f_{v,w} = 0 & \textit{for each } v \in V \setminus \{\Delta, \blacktriangledown\} \\
& f_{v^{tr},v'} - \rho(v') \cdot f_{v,v^{tr}} = 0 & \textit{for each } tr = v \xrightarrow{\tau} \rho \in T \textit{ and } v' \in \mathrm{Supp}(\rho)
\end{array}
$$

**Example 5.2.** *Consider again the automaton $\mathfrak{E}$ from Example 4.4 (depicted in Figure 4.4) and a weak transition $\bar{s} \overset{a}{\Longrightarrow}_{\mathrm{c}} \rho$ such that $\rho \; \mathcal{L}(\mathcal{R}) \; \mu$ where $\mu = \{(●, \frac{9}{50}), (▲, \frac{8}{50}), (●, \frac{33}{50})\}$ and $\mathcal{R} = \mathcal{I}$. As in Example 5.1, since $\mathcal{I}$ is the identity relation, we have that $\rho = \mu$. Denote as usual the transitions of $\mathfrak{E}$ as follows: $tr_0 = \bar{s} \xrightarrow{\tau} \{(r, 0.25), (y, 0.25), (g, 0.5)\}$, $tr_1 = r \xrightarrow{a} \delta_{●}$, $tr_2 = y \xrightarrow{a} \delta_{▲}$, $tr_3 = g \xrightarrow{a} \delta_{●}$, $tr_4 = r \xrightarrow{\tau} \delta_{\bar{s}}$, and $tr_5 = g \xrightarrow{\tau} \delta_{\bar{s}}$.*

*Besides the constraints for the non-negativity of the variables, the LP problem $LP(\bar{s}, a, \mu, \mathcal{R})$ has the following constraints:*

- *initial flow and challenging probabilities:*

$$
\begin{array}{lll}
f_{\Delta,\bar{s}} = 1 & f_{[●]_{\mathcal{R}},\blacktriangledown} = 9/50 & f_{[▲]_{\mathcal{R}},\blacktriangledown} = 8/50 \\
f_{[●]_{\mathcal{R}},\blacktriangledown} = 33/50 & f_{[\bar{s}]_{\mathcal{R}},\blacktriangledown} = 0 & f_{[r]_{\mathcal{R}},\blacktriangledown} = 0 \\
f_{[y]_{\mathcal{R}},\blacktriangledown} = 0 & f_{[g]_{\mathcal{R}},\blacktriangledown} = 0 &
\end{array}
$$

- *conservation of the flow for vertices in $S$:*

$$
\begin{array}{ll}
f_{\Delta,\bar{s}} + f_{r^{tr_4},\bar{s}} + f_{g^{tr_5},\bar{s}} - f_{\bar{s},\bar{s}^{tr_0}} = 0 & f_{\bar{s}^{tr_0},r} - f_{r,r^{tr_1}_a} - f_{r,r^{tr_4}} = 0 \\
f_{\bar{s}^{tr_0},y} - f_{y,y^{tr_2}_a} = 0 & f_{\bar{s}^{tr_0},g} - f_{g,g^{tr_3}_a} - f_{g,g^{tr_5}} = 0
\end{array}
$$

- *conservation of the flow for vertices in $S^{tr}$:*

$$
\begin{array}{ll}
f_{\bar{s},\bar{s}^{tr_0}} - f_{\bar{s}^{tr_0},r} - f_{\bar{s}^{tr_0},y} - f_{\bar{s}^{tr_0},g} = 0 & f_{r,r^{tr_4}} - f_{r^{tr_4},\bar{s}} = 0 \\
f_{g,g^{tr_5}} - f_{g^{tr_5},\bar{s}} = 0 &
\end{array}
$$

- *conservation of the flow for vertices in $S_a$:*

$$f_{r_a^{tr_4},\bar{s}_a} + f_{g_a^{tr_5},\bar{s}_a} - f_{\bar{s}_a,\bar{s}_a^{tr_0}} - f_{\bar{s}_a,[\bar{s}]_{\mathcal{R}}} = 0 \qquad f_{r_a^{tr_1},\bullet_a} - f_{\bullet_a,[\bullet]_{\mathcal{R}}} = 0$$
$$f_{\bar{s}_a^{tr_0},r_a} - f_{r_a,r_a^{tr_4}} - f_{r_a,[r]_{\mathcal{R}}} = 0 \qquad f_{y_a^{tr_2},\blacktriangle_a} - f_{\blacktriangle_a,[\blacktriangle]_{\mathcal{R}}} = 0$$
$$f_{\bar{s}_a^{tr_0},y_a} - f_{y_a,[y]_{\mathcal{R}}} = 0 \qquad f_{g_a^{tr_3},\bullet_a} - f_{\bullet_a,[\bullet]_{\mathcal{R}}} = 0$$
$$f_{\bar{s}_a^{tr_0},g_a} - f_{g_a,g_a^{tr_5}} - f_{g_a,[g]_{\mathcal{R}}} = 0$$

- *conservation of the flow for vertices in $S_a^{tr}$:*

$$f_{\bar{s}_a,\bar{s}_a^{tr_0}} - f_{\bar{s}_a^{tr_0},r_a} - f_{\bar{s}_a^{tr_0},y_a} - f_{\bar{s}_a^{tr_0},g_a} = 0 \qquad f_{r,r_a^{tr_1}} - f_{r_a^{tr_1},\bullet_a} = 0$$
$$f_{r_a,r_a^{tr_4}} - f_{r_a^{tr_4},\bar{s}_a} = 0 \qquad f_{y,y_a^{tr_2}} - f_{y_a^{tr_2},\blacktriangle_a} = 0$$
$$f_{g_a,g_a^{tr_5}} - f_{g_a^{tr_5},\bar{s}_a} = 0 \qquad f_{g,g_a^{tr_3}} - f_{g_a^{tr_3},\bullet_a} = 0$$

- *conservation of the flow for vertices in $S/\mathcal{R}$:*

$$f_{\bar{s}_a,[\bar{s}]_{\mathcal{R}}} - f_{[\bar{s}]_{\mathcal{R}},\blacktriangledown} = 0 \qquad f_{\bullet_a,[\bullet]_{\mathcal{R}}} - f_{[\bullet]_{\mathcal{R}},\blacktriangledown} = 0$$
$$f_{r_a,[r]_{\mathcal{R}}} - f_{[r]_{\mathcal{R}},\blacktriangledown} = 0 \qquad f_{\blacktriangle_a,[\blacktriangle]_{\mathcal{R}}} - f_{[\blacktriangle]_{\mathcal{R}},\blacktriangledown} = 0$$
$$f_{y_a,[y]_{\mathcal{R}}} - f_{[y]_{\mathcal{R}},\blacktriangledown} = 0 \qquad f_{\bullet_a,[\bullet]_{\mathcal{R}}} - f_{[\bullet]_{\mathcal{R}},\blacktriangledown} = 0$$
$$f_{g_a,[g]_{\mathcal{R}}} - f_{[g]_{\mathcal{R}},\blacktriangledown} = 0$$

- *balancing constraints for $\tau$-transitions generating $L_1$:*

$$f_{\bar{s}^{tr_0},r} - 0.3 \cdot f_{\bar{s},\bar{s}^{tr_0}} = 0 \qquad f_{\bar{s}^{tr_0},y} - 0.1 \cdot f_{\bar{s},\bar{s}^{tr_0}} = 0$$
$$f_{\bar{s}^{tr_0},g} - 0.6 \cdot f_{\bar{s},\bar{s}^{tr_0}} = 0 \qquad f_{r^{tr_4},\bar{s}} - 1 \cdot f_{r,r^{tr_4}} = 0$$
$$f_{g^{tr_5},\bar{s}} - 1 \cdot f_{g,g^{tr_5}} = 0$$

- *balancing constraints for $a$-transitions generating $L_a$:*

$$f_{r_a^{tr_1},\bullet_a} - 1 \cdot f_{r,r_a^{tr_1}} = 0 \qquad f_{y_a^{tr_2},\blacktriangle_a} - 1 \cdot f_{y,y_a^{tr_2}} = 0$$
$$f_{g_a^{tr_3},\bullet_a} - 1 \cdot f_{g,g_a^{tr_3}} = 0$$

- *balancing constraints for $\tau$-transitions generating $L_2$:*

$$f_{\bar{s}_a^{tr_0},r_a} - 0.3 \cdot f_{\bar{s}_a,\bar{s}_a^{tr_0}} = 0 \qquad f_{\bar{s}_a^{tr_0},y_a} - 0.1 \cdot f_{\bar{s}_a,\bar{s}_a^{tr_0}} = 0$$
$$f_{\bar{s}_a^{tr_0},g_a} - 0.6 \cdot f_{\bar{s}_a,\bar{s}_a^{tr_0}} = 0 \qquad f_{r_a^{tr_4},\bar{s}_a} - 1 \cdot f_{r_a,r_a^{tr_4}} = 0$$
$$f_{g_a^{tr_5},\bar{s}_a} - 1 \cdot f_{g_a,g_a^{tr_5}} = 0$$

*A solution that maximizes the objective function sets all variables to the value 0 except for the following variables:*

$$\begin{array}{llllll}
f_{\blacktriangle,\bar{s}} & = 50/50 & f_{\bar{s},\bar{s}^{tr0}} & = 80/50 & f_{\bar{s}^{tr_0},r} & = 24/50 \\
f_{\bar{s}^{tr_0},y} & = 8/50 & f_{\bar{s}^{tr_0},g} & = 48/50 & f_{r,r_a^{tr_1}} & = 9/50 \\
f_{r,r^{tr_4}} & = 15/50 & f_{y,y_a^{tr_2}} & = 8/50 & f_{g,g_a^{tr_3}} & = 33/50 \\
f_{g,g^{tr_5}} & = 15/50 & f_{r^{tr_4},\bar{s}} & = 15/50 & f_{g^{tr_5},\bar{s}} & = 15/50 \\
f_{r_a^{tr_1},\bullet_a} & = 9/50 & f_{y_a^{tr_2},\blacktriangle_a} & = 8/50 & f_{g_a^{tr_3},\bullet_a} & = 33/50 \\
f_{\bullet_a,[\bullet]_{\mathcal{R}}} & = 9/50 & f_{\blacktriangle_a,[\blacktriangle]_{\mathcal{R}}} & = 8/50 & f_{\bullet_a,[\bullet]_{\mathcal{R}}} & = 33/50 \\
f_{[\bullet]_{\mathcal{R}},\blacktriangledown} & = 9/50 & f_{[\blacktriangle]_{\mathcal{R}},\blacktriangledown} & = 8/50 & f_{[\bullet]_{\mathcal{R}},\blacktriangledown} & = 33/50
\end{array}$$

*It is worthwhile to note the value 80/50 for the variable $f_{\bar{s},\bar{s}^{tr_0}}$: this is caused by the fact that the arc $(\bar{s},\bar{s}^{tr_0})$ is part of a cycle and its flow value is greater than 1, confirming that 1, the maximum probability, in general is not a proper value for arc capacities, as discussed in [TH15].* ♦

In the LP problem described in Definition 5.4, the objective function maximizes the total sum of negated flow routed along the arcs of the network. In fact, the total flow is described as the sum of negated flow variables which are positive themselves. This prevents routing large amounts of flow over disconnected components of the network or over cycles that can be ignored. Furthermore, in the LP problem, there are two different sets of constraints. The first set is the ordinary set of flow conservation constraints which require the total flow incoming and outgoing a node of the network to be equal. The second set is the set of balancing constraints that require the entering amount of flow to a transition node to be distributed based on probabilities assigned to the outgoing arcs.

It is easy to observe that the $LP(t, a, \mu, \mathcal{R})$ LP problem has size that is quadratic in the size $N = |\mathfrak{P}|$: the number of variables is at most $3N^2 + 5N + 1$ while the number of constraints is at most $6N^2 + 11N + 2$. Moreover, it is also worthwhile to spell out the number of transition, state, and class nodes of the network $G(t, a, \mu, \mathcal{R})$: there are at most $2|T|$ transition nodes, at most $2|S|$ state nodes, and at most $|S|$ class nodes.

The equivalence of the LP problem and the weak transition is formalized by Theorem 9 and Corollary 12(1) of [TH15]:

**Proposition 5.2.** *A weak transition $t \overset{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ such that $\mu\ \mathcal{L}(\mathcal{R})\ \mu_t$ exists if and only if the LP problem $LP(t, a, \mu, \mathcal{R})$ has a feasible solution.*

**Remark 5.3.** *The LP problem construction proposed in Definition 5.4 is not easily extendable to weak non-combined transitions induced by a Dirac scheduler. In fact, in order to obtain for such setting a result equivalent to Proposition 5.2, we should enforce that the flow leaving the nodes $\nu$ and $\nu_a$ is not split among several outgoing arcs, but it is routed completely to a single arc. To obtain such a situation, we should replace, for each $\nu \in S \cup S_a$, the flow conservation constraint in Definition 5.4*

$$\textstyle\sum_{(u,\nu)\in E} f_{u,\nu} - \sum_{(\nu,w)\in E} f_{\nu,w} = 0 \quad \text{for each } \nu \in V \setminus \{\triangle, \blacktriangledown\}$$

*by the following set of constraints:*

$$\begin{array}{ll} \textstyle\sum_{(u,\nu)\in E} f_{u,\nu} - \sum_{(\nu,w)\in E} \alpha_{\nu,w} f_{\nu,w} = 0 & \text{for each } \nu \in V \setminus \{\triangle, \blacktriangledown\} \\ \textstyle\sum_{(\nu,w)\in E} \alpha_{\nu,w} = 1 & \text{for each } \nu \in V \setminus \{\triangle, \blacktriangledown\} \\ \alpha_{\nu,w} \in \{0, 1\} & \text{for each } (\nu, w) \in E \end{array}$$

*The latter ensures that the flow is sent through a single outgoing arc in its entirety. This change implies that the resulting problem is no longer a Linear Programming problem but a Mixed Integer Nonlinear Programming problem (MINLP), known to belong to the class of **NP**-complete problems [Sch03]. While it is rather easy to show that the problem of finding a weak transition induced by a Dirac scheduler is equivalent to the above MINLP problem (the proof is essentially the same of the one of Proposition 5.2, see [TH15, Lemmas 7 and 8]), such an equivalence is not sufficient to establish the **NP**-completeness of the problem. However, it is still possible to show such a result by a direct reduction from the 3-SAT problem.*

*We recall that a formula $\varphi$ is written in conjunctive normal form with three variables per clause (3-CNF) if $\varphi = \bigwedge_{i=1}^{n} Cl_i$ where each clause $Cl_i$ is a disjunction of three literals. To simplify the presentation, we assume that each clause contains distinct literals. Given a formula $\varphi$, we denote by $Var(\varphi)$ the set of variables occurring in $\varphi$, by $Lit(\varphi)$ the set of literals occurring in $\varphi$, by $Cl(\varphi)$ the set of clauses of $\varphi$, and, given a literal $l$, we denote by $Cl(\varphi, l)$ the set of clauses of $\varphi$ where $l$ occurs.*

**Proposition 5.3.** *Given a PA $\mathfrak{P}$, a state $s \in S$, an action $a \in \mathscr{A}$, and a probability measure $\mu \in \mathrm{Disc}(S)$, checking whether there exists a Dirac scheduler inducing $s \stackrel{a}{\Longrightarrow} \mu$ is **NP**-complete.*

*Proof.* To prove the claim, we have to show two results: the problem is **NP**-hard and belongs to **NP**.

The fact that the problem belongs to **NP** follows directly from the fact that the existence of a weak transition induced by a Dirac scheduler can be encoded as a MINLP problem, that is in **NP**.

For showing the **NP**-hardness, we provide a reduction from the 3-SAT problem. Let $\varphi = \bigwedge_{i=1}^{n} Cl_i$ be a 3-CNF formula, $n = |Cl(\varphi)|$, and $m = |Var(\varphi)|$.

Consider the *PA* $\mathfrak{P}_\varphi$ whose set of states is $S = \{\varphi, \nabla\} \cup Var(\varphi) \cup \{ v^{\mathbf{false}}, v^{\mathbf{true}} \mid v \in Var(\varphi) \} \cup Cl(\varphi)$, whose start state is $\varphi$, whose set of actions is $\mathscr{A} = \{\tau\}$, and whose transitions are:

$$\begin{aligned} T = {} & \{\varphi \xrightarrow{\tau} \upsilon_{Var(\varphi)}\} \cup \{ v \xrightarrow{\tau} \delta_{v^{\mathbf{true}}}, v \xrightarrow{\tau} \delta_{v^{\mathbf{false}}} \mid v \in Var(\varphi) \} \\ & \cup \{ v^{\mathbf{true}} \xrightarrow{\tau} \rho_v \mid v \in Lit(\varphi) \} \cup \{ v^{\mathbf{false}} \xrightarrow{\tau} \rho_{\neg v} \mid \neg v \in Lit(\varphi) \} \\ & \cup \{ Cl \xrightarrow{\tau} \{(Cl, \frac{1}{k}), (\nabla, \frac{k-1}{k})\} \mid Cl \in Cl(\varphi), k \in \{1,2,3\} \}, \end{aligned}$$

where, for a literal $l$, $\rho_l$ is defined as

$$\rho_l(t) = \begin{cases} \frac{1}{n} & \text{if } t \in Cl(\varphi, l), \\ \frac{|Cl(\varphi) \setminus Cl(\varphi,l)|}{n} & \text{if } t = \nabla, \\ 0 & \text{otherwise.} \end{cases}$$

We now prove that $\varphi$ is satisfiable if and only if $\mathfrak{P}_\varphi$ exhibits the weak transition $\varphi \stackrel{\tau}{\Longrightarrow} \mu$ where

$$\mu(t) = \begin{cases} \frac{1}{n \cdot m} & \text{if } t = Cl \text{ for some clause } Cl \in Cl(\varphi), \\ 1 - \frac{1}{m} & \text{if } t = \nabla, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Suppose that $\varphi$ is satisfiable; this implies that there exists a truth value assignment for the variables occurring in $\varphi$ that makes the formula true. Moreover, since $\varphi$ is satisfiable, it follows that at least one literal of each clause $Cl \in Cl(\varphi)$ has assignment **true**. Let $\sigma$ be the Dirac scheduler defined as follows:

$$\sigma(\alpha) = \begin{cases} \delta_{\varphi \xrightarrow{\tau} \upsilon_{Var(\varphi)}} & \text{if } \alpha = \varphi, \\ \delta_{v \xrightarrow{\tau} \delta_{v^{\mathbf{true}}}} & \text{if } \alpha = \varphi\tau v \text{ and } v \text{ is } \mathbf{true} \text{ in the assignment,} \\ \delta_{v \xrightarrow{\tau} \delta_{v^{\mathbf{false}}}} & \text{if } \alpha = \varphi\tau v \text{ and } v \text{ is } \mathbf{false} \text{ in the assignment,} \\ \delta_{v^{\mathbf{true}} \xrightarrow{\tau} \rho_v} & \text{if } \alpha = \varphi\tau v\tau v^{\mathbf{true}} \text{ and } v \in Lit(\varphi), \\ \delta_{v^{\mathbf{false}} \xrightarrow{\tau} \rho_{\neg v}} & \text{if } \alpha = \varphi\tau v\tau v^{\mathbf{false}} \text{ and } \neg v \in Lit(\varphi), \\ \delta_{Cl \xrightarrow{\tau} \{(Cl, \frac{1}{k}), (\nabla, \frac{k-1}{k})\}} & \text{if } \alpha \in \{\varphi\tau v\tau v^{\mathbf{true}}\tau Cl, \varphi\tau v\tau v^{\mathbf{false}}\tau Cl\} \text{ and exactly } k \text{ literals of } Cl \text{ are } \mathbf{true}, \\ \delta_\perp & \text{otherwise.} \end{cases}$$

It is rather easy to verify that $\sigma$ actually induces the weak transition $\varphi \stackrel{\tau}{\Longrightarrow} \mu$. Consider, for instance, a clause $Cl$; let $Cl = l_1 \vee l_2 \vee l_3$ and $v_i$ be the variable associated to the literal $l_i$. The probability of reaching $Cl$ is:

$$\begin{aligned}
&\mu_{\sigma,\varphi}(\{\,\alpha \in \mathit{frags}^*(\mathfrak{P}_\varphi) \mid \mathit{last}(\alpha) = Cl\,\}) \\
&= \mu_{\sigma,\varphi}(\{\varphi\tau v_1 \tau v_1^{\mathbf{v}} \tau Cl \tau Cl\}) && \text{if } l_1 = \textbf{true} \text{ and } \mathbf{v} \text{ is the assignment of } v_1 \\
&\quad + \mu_{\sigma,\varphi}(\{\varphi\tau v_2 \tau v_2^{\mathbf{v}} \tau Cl \tau Cl\}) && \text{if } l_2 = \textbf{true} \text{ and } \mathbf{v} \text{ is the assignment of } v_2 \\
&\quad + \mu_{\sigma,\varphi}(\{\varphi\tau v_3 \tau v_3^{\mathbf{v}} \tau Cl \tau Cl\}) && \text{if } l_3 = \textbf{true} \text{ and } \mathbf{v} \text{ is the assignment of } v_3
\end{aligned}$$

For each $i \in \{1,2,3\}$ such that $l_i = \textbf{true}$, we have that $\mu_{\sigma,\varphi}(\{\varphi\tau v_i \tau v_i^{\mathbf{v}} \tau Cl \tau Cl\}) = \frac{1}{m} \cdot \frac{1}{n} \cdot \frac{1}{k}$, where $k$ is the number of literals of $Cl$ that are **true**. In fact, for each $i \in \{1,2,3\}$ such that $l_i = \textbf{true}$,

$$\begin{aligned}
&\mu_{\sigma,\varphi}(\{\varphi\tau v_i \tau v_i^{\mathbf{v}} \tau Cl \tau Cl\}) \\
&= \mu_{\sigma,\varphi}(C_{\varphi\tau v_i \tau v_i^{\mathbf{v}} \tau Cl \tau Cl}) \cdot \sigma(\varphi\tau v_i \tau v_i^{\mathbf{v}} \tau Cl \tau Cl)(\bot) \\
&= \mu_{\sigma,\varphi}(C_\varphi) \cdot \Big( \sum_{tr \in T(\tau)} \sigma(\varphi)(tr) \cdot \mu_{tr}(v_i) \Big) \\
&\qquad \cdot \Big( \sum_{tr \in T(\tau)} \sigma(\varphi\tau v_i)(tr) \cdot \mu_{tr}(v_i^{\mathbf{v}}) \Big) \\
&\qquad \cdot \Big( \sum_{tr \in T(\tau)} \sigma(\varphi\tau v_i \tau v_i^{\mathbf{v}})(tr) \cdot \mu_{tr}(Cl) \Big) \\
&\qquad \cdot \Big( \sum_{tr \in T(\tau)} \sigma(\varphi\tau v_i \tau v_i^{\mathbf{v}} \tau Cl)(tr) \cdot \mu_{tr}(Cl) \Big) \\
&\qquad \cdot \sigma(\varphi\tau v_i \tau v_i^{\mathbf{v}} \tau Cl \tau Cl)(\bot)
\end{aligned}$$

(In the following step, we omit the transitions chosen by $\sigma$ with probability 0; for instance, $\varphi \xrightarrow{\tau} \upsilon_{Var(\varphi)}$ when $\alpha = \varphi\tau v_i$. For improving readability, we write $\theta_{Cl}$ for the distribution $\{(Cl, \frac{1}{k}), (\nabla, \frac{k-1}{k})\}$.)

$$\begin{aligned}
&= \mu_{\sigma,\varphi}(C_\varphi) \cdot \Big( \sigma(\varphi)(\varphi \xrightarrow{\tau} \upsilon_{Var(\varphi)}) \cdot \upsilon_{Var(\varphi)}(v_i) \Big) \\
&\qquad \cdot \Big( \sigma(\varphi\tau v_i)(v_i \xrightarrow{\tau} \delta_{v_i^{\mathbf{v}}}) \cdot \delta_{v_i^{\mathbf{v}}}(v_i^{\mathbf{v}}) \Big) \\
&\qquad \cdot \Big( \sigma(\varphi\tau v_i \tau v_i^{\mathbf{v}})(v_i^{\mathbf{v}} \xrightarrow{\tau} \rho_l) \cdot \rho_l(Cl) \Big) \\
&\qquad \cdot \Big( \sigma(\varphi\tau v_i \tau v_i^{\mathbf{v}} \tau Cl)(Cl \xrightarrow{\tau} \theta_{Cl}) \cdot \theta_{Cl}(Cl) \Big) \\
&\qquad \cdot \sigma(\varphi\tau v_i \tau v_i^{\mathbf{v}} \tau Cl \tau Cl)(\bot) \\
&= 1 \cdot \Big(\frac{1}{m}\Big) \cdot \Big(1\Big) \cdot \Big(\frac{1}{n}\Big) \cdot \Big(\frac{1}{k}\Big) \cdot 1 = \frac{1}{m} \cdot \frac{1}{n} \cdot \frac{1}{k}
\end{aligned}$$

Since $\mu_{\sigma,\varphi}(\{\varphi\tau v_i \tau v_i^{\mathbf{v}} \tau Cl \tau Cl\}) = \frac{1}{m} \cdot \frac{1}{n} \cdot \frac{1}{k}$ holds for each $i \in \{1,2,3\}$ such that $l_i = \textbf{true}$, it follows that, for $k$ literals being **true** in $Cl$, the overall probability assigned to the state $Cl$ is $k \cdot \frac{1}{m} \cdot \frac{1}{n} \cdot \frac{1}{k} = \frac{1}{n \cdot m}$ as required. Since this probability is independent from the particular $Cl$, the

overall probability assigned to $Cl(\varphi)$ is $n \times \frac{1}{n \cdot m} = \frac{1}{m}$; the remaining probability value $1 - \frac{1}{m}$ is assigned to $\nabla$, as required, as it can be easily checked in a similar way. The other properties the scheduler has to satisfy trivially follow from the previous one and the fact that the *PA* $\mathfrak{P}_\varphi$ has $E = \emptyset$, so $\sigma$ actually induces the weak transition $\varphi \stackrel{\tau}{\Longrightarrow} \mu$. This completes the proof that if $\varphi$ is satisfiable, then there is a Dirac scheduler inducing $\varphi \stackrel{\tau}{\Longrightarrow} \mu$.

Now, suppose that there exists a Dirac scheduler inducing $\varphi \stackrel{\tau}{\Longrightarrow} \mu$. We want to derive a logical value assignment such that the formula $\varphi$ holds. For each variable $v \in Var(\varphi)$, define the assignment $\theta(v)$ as follows:

$$\theta(v) = \begin{cases} \textbf{true} & \text{if } \sigma(\varphi \tau v) = \delta_{v \xrightarrow{\tau} \delta_{v\textbf{true}}}, \\ \textbf{false} & \text{otherwise.} \end{cases}$$

Since by hypothesis each clause $Cl$ is reached with probability $\frac{1}{n \cdot m}$, it means that there exists at least one finite execution fragment of the form $\varphi \tau v \tau v^{\mathbf{v}} \tau Cl \tau \overline{Cl}$ that occurs with non-zero probability. In particular,

$$\mathbf{v} = \begin{cases} \textbf{true} & \text{if } \sigma(\varphi \tau v) = \delta_{v \xrightarrow{\tau} \delta_{v\textbf{true}}}, \\ \textbf{false} & \text{otherwise,} \end{cases}$$

i.e., $v$ has truth value $\mathbf{v}$. Moreover, the existence of such execution fragment implies that the literal $v$ occurs in $Cl$ if $\mathbf{v} = \textbf{true}$ or the literal $\neg v$ occurs in $Cl$ if $\mathbf{v} = \textbf{false}$. The former case implies that $Cl = v \vee l' \vee l''$ for some literal $l'$ and $l''$ with $v = \textbf{true}$, while the latter case implies that $Cl = \neg v \vee l' \vee l''$ for some literal $l'$ and $l''$ with $v = \textbf{false}$. In both cases the clause $Cl$ is satisfied, hence $\varphi$ is satisfied as well since $Cl$ is a generic clause in $Cl(\varphi)$. This concludes the proof that if there exists Dirac scheduler inducing the weak transition $\varphi \stackrel{\tau}{\Longrightarrow} \mu$, then $\varphi$ is satisfiable.

Since we have shown that $\varphi$ is satisfiable if and only if $\mathfrak{P}_\varphi$ exhibits the weak transition $\varphi \stackrel{\tau}{\Longrightarrow} \mu$, in order to complete the reduction we have to show that the reduction is polynomial in the size of the formula $\varphi$: this follows immediately by the construction of $\mathfrak{P}_\varphi$ whose number of states and transitions is linear in the number of variables and clauses of $\varphi$. ■

### 5.3.3 Complexity Analysis of Deciding Weak Bisimulation

Proposition 5.2 allows us to verify the existence of a weak transition $t \stackrel{a}{\Longrightarrow}_c \mu_t$ such that $\mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t$ at line 3 of FindSplit efficiently: $W(N)$ is actually $p(N)$ for some polynomial $p$, hence the following result holds.

**Theorem 5.1.** *Given two PAs $\mathfrak{P}_1$ and $\mathfrak{P}_2$, let $N = |\mathfrak{P}_1| + |\mathfrak{P}_2|$. Checking $\mathfrak{P}_1 \approx \mathfrak{P}_2$ is polynomial in $N$.*

## 5.4 Efficiency of Solving the LP Problem

The analysis of the $LP(t, a, \mu, \mathcal{R})$ LP problem formalized in [TH15, Proposition 6] considers the theoretical complexity class the problem belongs to. It does not address how efficiently

the LP problem can indeed be solved in practice. Practical implementation aspects and empirical results will be presented in Section 5.5. To prepare for that, we first discuss abstract observations concerning the worst case running time needed to solve the LP problem. Then we recast the LP problem into a flow network problem and exploit the underlying network structure to arrive at an efficient LP problem solution approach harvesting an algorithm in the network optimization setting. We further discuss various alternative approaches to improve solution efficiency, including approximative methods.

### 5.4.1 Efficient Solution: Theory

Throughout this section, we define the dimension of an input to an algorithm as the number of data items in the input. The size of a rational number $p/q$ is defined as the length of its binary description, i.e., $|p/q| = \lceil \log_2(p+1) \rceil + \lceil \log_2(q+1) \rceil$, where $\lceil x \rceil$ denotes the smallest integer not less than $x$. The size of a rational vector or matrix is defined as the sum of the sizes of its entries.

Deciding the existence of a weak transition in a probabilistic automaton can be done in polynomial time [TH15, Proposition 6 and Theorem 8]. With the aim to refine this result, we discuss the problem in the context of the restricted class of rational probabilistic automata.

**Rational *PA*s.** We start our analysis with the class of rational *PA*s.

**Definition 5.5 (Rational *PA*s).** *Given a PA $\mathfrak{P}$, we say that $\mathfrak{P}$ is* rational *if for each $(s,a,\mu) \in T$ and $\nu \in \mathrm{Supp}(\mu)$, we have that $\mu(\nu) \in \mathbb{Q}$.*

For this class of *PA*s, we look for a tighter worst case complexity bound of solving the LP problem $LP(t,a,\mu,\mathcal{R})$. We proceed via a reformulation that reduces the size of $LP(t,a,\mu,\mathcal{R})$. This size reduction directly reduces the solution effort needed for the LP problem, since the latter depends on the number of variables and constraints, and this will indeed provide a tighter worst case bound. To reach our goal, we modify the network provided in Definition 5.3 and reformulate the original LP problem on the basis of these changes.

Consider the network $G(t,a,\mu,\mathcal{R})$ and let $\mathcal{G}(t,a,\mu,\mathcal{R})$ be a directed network which is generated from the network $G(t,a,\mu,\mathcal{R})$ by removing the source node $\triangle$ and the sink node $\blacktriangledown$; let $\mathcal{V} = V \setminus \{\triangle, \blacktriangledown\}$ and $\mathcal{E} = E \setminus (\{(\triangle, t)\} \cup \{(\mathcal{C}, \blacktriangledown) \mid \mathcal{C} \in S/\mathcal{R}\})$ be the set of vertices and directed arcs of $\mathcal{G}(t,a,\mu,\mathcal{R})$, respectively. Moreover, let $\bar{\mathcal{E}} \subseteq \mathcal{E}$ be the set $\bar{\mathcal{E}} = \{(\nu^{tr}, \nu'), (\nu_a^{tr}, \nu_a') \mid tr = \nu \xrightarrow{\tau} \rho \in T, \nu' \in \mathrm{Supp}(\rho)\} \cup \{(\nu_a^{tr}, \nu_a') \mid tr = \nu \xrightarrow{a} \rho \in T, \nu' \in \mathrm{Supp}(\rho)\}$. Then, we define $\rho_{i,j} = \mu_{tr}(\nu')$ as the proportionality coefficient corresponding to the arc $(i,j) \in \bar{\mathcal{E}}$ where $(i,j) = (\nu^{tr}, \nu')$ or $(i,j) = (\nu_a^{tr}, \nu_a')$. Since in both original and modified networks each arc in $\bar{\mathcal{E}}$ belongs to a single transition, the corresponding proportional coefficient is uniquely determined.

For each node $u \in \mathcal{V}$, let $b_u$ be a supply/demand value, that is, if $b_u > 0$ the node $u$ is a supply node and if $b_u < 0$ the node $u$ is a demand node. For the network $\mathcal{G}(t,a,\mu,\mathcal{R})$, we define $b_u$ for each node $u \in \mathcal{V}$ so as to take value 1 if $u = t$, value $-\mu(\mathcal{C})$ if $u = \mathcal{C} \in S/\mathcal{R}$ and 0 otherwise. It is immediate to see that $\sum_{u \in \mathcal{V}} b_u = 0$. This fact can be seen as a feasibility condition in the corresponding flow network [AMO93]. For $s \in \mathcal{T}$, assume $A_s$ to be the set of all arcs in the node-arc incidence matrix $A$ that should have proportional flow. We define $\tilde{A}$ to be the subset of arcs in $A$ that do not belong to any set $A_s$ for $s \in \mathcal{T}$. More precisely,

$\tilde{A} = A \setminus \bigcup_{s\in\mathscr{T}} A_s$. Based on the definitions, the $LP(t,a,\mu,\mathcal{R})$ LP problem can be reformulated as follows:

$$\begin{array}{rll} \text{LP1: min} & \sum_{(i,j)\in\mathscr{E}} f_{i,j} & \\ \text{subject to:} & \sum_{(i,j)\in\mathscr{E}} f_{i,j} - \sum_{(j,i)\in\mathscr{E}} f_{j,i} = b_i & \text{for each } i \in \mathscr{V} \\ & \frac{f_{i,j}}{\rho_{i,j}} \text{ are all equal} & s \in \mathscr{T}, (i,j) \in A_s \\ & f_{i,j} \geq 0 & \text{for each } (i,j) \in \mathscr{E} \end{array}$$

**Lemma 5.1.** *The $LP(t,a,\mu,\mathcal{R})$ LP problem and LP1 are equivalent.*

*Proof.* The statement follows immediately by a simple manipulation of the balancing constraints: consider the transition $tr = v \xrightarrow{\tau} \rho$; it is encoded in the network as the transition node $v^{tr}$ and the arcs $(v, v^{tr})$ and $(v^{tr}, v')$ for $v' \in \text{Supp}(\rho)$. The corresponding balancing constraints are $f_{v^{tr},v'} - \rho(v') \cdot f_{v,v^{tr}} = 0$, that is, $\frac{f_{v^{tr},v'}}{\rho(v')} = f_{v,v^{tr}}$. Since $f_{v,v^{tr}}$ is independent on $v'$, it follows that the ratio $\frac{f_{v^{tr},v'}}{\rho(v')}$ is equal for all $v' \in \text{Supp}(\rho)$, as required.

The same holds for the transition nodes $v_a^{tr}$ and $v_a^{tr'}$, the latter corresponding to the transition $tr' = v \xrightarrow{a} \gamma$. ■

By assuming the unit flow cost $c_{i,j} = 1$ for each arc $(i,j) \in \mathscr{E}$, the objective of this problem is to minimize the total cost of routing the flow on network arcs subject to the ordinary flow conservation constraints, the proportional flow constraints corresponding to the balancing constraints of the original LP problem, and the arc flow lower bounds.

It is worthwhile to note that there exists a proportional flow set for each transition node in the network and that each arc may belong to at most one proportional flow set. The flow on the arcs in each of these flow proportional sets can be regarded as a single decision variable. Using this intuition, let $a_{i,j}$ denote the column corresponding to the arc $(i,j)$ in the node-arc incidence matrix of the network $\mathscr{G}(t,a,\mu,\mathcal{R})$ and let $a_s = \sum_{(i,j)\in A_s} \rho_{i,j} \cdot a_{i,j}$ for each $s \in \mathscr{T}$. We denote by $a_s^k$ the $k$-th component of the vector $a_s$. Since the column vector $a_{i,j}$ in the node-arc incidence matrix includes only entities 0, +1 and -1 therefore, the $k$-th component of the vector $a_s$, i.e., $a_s^k$ can be equivalently written as $a_s^k = \sum_{(k,j)\in A_s} \rho_{k,j} - \sum_{(j,k)\in A_s} \rho_{j,k}$. By using the new notations, LP1 can be reformulated as the following LP problem which in turn can be regarded as an adaptation of the LP considered in [BF12].

$$\begin{array}{rll} \text{LP2: min} & \sum_{(i,j)\in\tilde{A}} f_{i,j} + \sum_{s\in\mathscr{T}} f_s & \\ \text{subject to:} & \sum_{(i,j)\in\tilde{A}} f_{i,j} - \sum_{(j,i)\in\tilde{A}} f_{j,i} + \sum_{s\in\mathscr{T}} a_s^i \cdot f_s = b_i & \text{for each } i \in \mathscr{V} \\ & f_{i,j} \geq 0 & \text{for each } (i,j) \in \tilde{A} \\ & f_s \geq 0 & \text{for each } s \in \mathscr{T} \end{array}$$

**Lemma 5.2.** *LP1 and LP2 are equivalent.*

*Proof.* Let $f = \{ f_{i,j} \mid (i,j) \in \tilde{A} \} \cup \{ f_s \mid s \in \mathscr{T} \}$ be a feasible solution for LP2. Define flow $\tilde{f}$ as follows:

$$\tilde{f}_{i,j} = \begin{cases} f_{i,j} & \text{if } (i,j) \in \tilde{A} \\ \rho_{i,j} \cdot f_s & \text{if } s \in \mathscr{T} \text{ and } (i,j) \in A_s \\ 0 & \text{otherwise.} \end{cases}$$

We claim that the flow $\tilde{f}$ satisfies the LP1 constrains. To show this, in the first set of constraints in LP1 and for each $i \in \mathcal{V}$ we get the following equivalences:

$$\begin{aligned}\sum_{(i,j)\in\mathcal{E}} \tilde{f}_{i,j} - \sum_{(j,i)\in\mathcal{E}} \tilde{f}_{j,i} &= \left(\sum_{(i,j)\in\tilde{A}} \tilde{f}_{i,j} + \sum_{(i,j)\in\mathcal{E}\setminus\tilde{A}} \tilde{f}_{i,j}\right) - \left(\sum_{(j,i)\in\tilde{A}} \tilde{f}_{j,i} + \sum_{(j,i)\in\mathcal{E}\setminus\tilde{A}} \tilde{f}_{j,i}\right) \\ &= \sum_{(i,j)\in\tilde{A}} \tilde{f}_{i,j} - \sum_{(j,i)\in\tilde{A}} \tilde{f}_{j,i} + \sum_{(i,j)\in\mathcal{E}\setminus\tilde{A}} \tilde{f}_{i,j} - \sum_{(j,i)\in\mathcal{E}\setminus\tilde{A}} \tilde{f}_{j,i} \\ &= \sum_{(i,j)\in\tilde{A}} \tilde{f}_{i,j} - \sum_{(j,i)\in\tilde{A}} \tilde{f}_{j,i} + \sum_{s\in\mathcal{T}} \sum_{(i,j)\in A_s} \tilde{f}_{i,j} - \sum_{s\in\mathcal{T}} \sum_{(j,i)\in A_s} \tilde{f}_{j,i}\end{aligned}$$

by definition of $\tilde{A}$

$$= \sum_{(i,j)\in\tilde{A}} f_{i,j} - \sum_{(j,i)\in\tilde{A}} f_{j,i} + \sum_{s\in\mathcal{T}} \sum_{(i,j)\in A_s} \rho_{i,j}\cdot f_s - \sum_{s\in\mathcal{T}} \sum_{(j,i)\in A_s} \rho_{j,i}\cdot f_s$$

by definition of $\tilde{f}$

$$= \sum_{(i,j)\in\tilde{A}} f_{i,j} - \sum_{(j,i)\in\tilde{A}} f_{j,i} + \sum_{s\in\mathcal{T}} f_s \cdot \left(\sum_{(i,j)\in A_s} \rho_{i,j} - \sum_{(j,i)\in A_s} \rho_{j,i}\right)$$

by simple term manipulation

$$= \sum_{(i,j)\in\tilde{A}} f_{i,j} - \sum_{(j,i)\in\tilde{A}} f_{j,i} + \sum_{s\in\mathcal{T}} a_s^i \cdot f_s$$

by definition of $a_s^i$

$$= b_i$$

by definition of LP2. Moreover, for each $s \in \mathcal{T}$ and $(i,j) \in A_s$, $\frac{\tilde{f}_{i,j}}{\rho_{i,j}} = \frac{\rho_{i,j}\cdot f_s}{\rho_{i,j}} = f_s$. This means that for each $s \in \mathcal{T}$ and for all $(i,j) \in A_s$, $\frac{\tilde{f}_{i,j}}{\rho_{i,j}}$ are all equal. Also, for each $(i,j) \in \tilde{A}$, $\tilde{f}_{i,j} = f_{i,j} \geq 0$ and for each $s \in \mathcal{T}$ and $(i,j) \in A_s$, $\tilde{f}_{i,j} = \rho_{i,j}\cdot f_s \geq 0$. Therefore, $\tilde{f}_{i,j}$ for $(i,j) \in \mathcal{E}$ is indeed a feasible solution for LP1. Next, consider the value of the objective function for LP1:

$$\begin{aligned}\sum_{(i,j)\in\mathcal{E}} \tilde{f}_{i,j} &= \sum_{(i,j)\in\tilde{A}} \tilde{f}_{i,j} + \sum_{(i,j)\in\mathcal{E}\setminus\tilde{A}} \tilde{f}_{i,j} \\ &= \sum_{(i,j)\in\tilde{A}} \tilde{f}_{i,j} + \sum_{s\in\mathcal{T}} \sum_{(i,j)\in A_s} \tilde{f}_{i,j}\end{aligned}$$

by definition of $\tilde{A}$

$$= \sum_{(i,j)\in\tilde{A}} f_{i,j} + \sum_{s\in\mathcal{T}} \sum_{(i,j)\in A_s} \rho_{i,j}\cdot f_s$$

by definition of $\tilde{f}$

$$= \sum_{(i,j)\in\tilde{A}} f_{i,j} + \sum_{s\in\mathscr{T}} f_s \cdot \overbrace{\sum_{(i,j)\in A_s} \rho_{i,j}}^{1}$$

by simple term manipulation and the fact that $\rho_{i,j} = \mu_{tr}(v')$ where $(i,j) = (v^{tr}, v')$ or $(i,j) = (v_a^{tr}, v_a')$

$$= \sum_{(i,j)\in\tilde{A}} f_{i,j} + \sum_{s\in\mathscr{T}} f_s.$$

Therefore, corresponding to this feasible solution, the value of the objective function of both LP problems are the same. For the reverse side, assume $\bar{f} = \{\bar{f}_{i,j} \mid (i,j) \in \mathscr{E}\}$ is a feasible solution for LP1. Define the flow $\hat{f} = \{\hat{f}_{i,j} \mid (i,j) \in \tilde{A}\} \cup \{\hat{f}_s \mid s \in \mathscr{T}\}$ where $\hat{f}_{i,j} = \bar{f}_{i,j}$ for $(i,j) \in \tilde{A}$ and $\hat{f}_s = \frac{\bar{f}_{i,j}}{\rho_{i,j}}$ for each $s \in \mathscr{T}$ where $(i,j) \in A_s$. In the following we show that $\hat{f}$ is a feasible solution for LP2. For each $i \in \mathscr{V}$, it holds:

$$\sum_{(i,j)\in\tilde{A}} \hat{f}_{i,j} - \sum_{(j,i)\in\tilde{A}} \hat{f}_{j,i} + \sum_{s\in\mathscr{T}} a_s^i \cdot \hat{f}_s$$

$$= \sum_{(i,j)\in\tilde{A}} \hat{f}_{i,j} - \sum_{(j,i)\in\tilde{A}} \hat{f}_{j,i} + \sum_{s\in\mathscr{T}} \left( \sum_{(i,j)\in A_s} \rho_{i,j} - \sum_{(j,i)\in A_s} \rho_{j,i} \right) \cdot \hat{f}_s$$

by definition of $a_s^i$

$$= \sum_{(i,j)\in\tilde{A}} \hat{f}_{i,j} - \sum_{(j,i)\in\tilde{A}} \hat{f}_{j,i} + \sum_{s\in\mathscr{T}} \sum_{(i,j)\in A_s} \rho_{i,j} \cdot \hat{f}_s - \sum_{s\in\mathscr{T}} \sum_{(j,i)\in A_s} \rho_{j,i} \cdot \hat{f}_s$$

by simple term manipulation

$$= \sum_{(i,j)\in\tilde{A}} \bar{f}_{i,j} - \sum_{(j,i)\in\tilde{A}} \bar{f}_{j,i} + \sum_{s\in\mathscr{T}} \sum_{(i,j)\in A_s} \bar{f}_{i,j} - \sum_{s\in\mathscr{T}} \sum_{(j,i)\in A_s} \bar{f}_{j,i}$$

by definition of $\hat{f}$

$$= \sum_{(i,j)\in\tilde{A}} \bar{f}_{i,j} + \sum_{s\in\mathscr{T}} \sum_{(i,j)\in A_s} \bar{f}_{i,j} - \left( \sum_{(j,i)\in\tilde{A}} \bar{f}_{j,i} + \sum_{s\in\mathscr{T}} \sum_{(j,i)\in A_s} \bar{f}_{j,i} \right)$$

by simple term manipulation

$$= \sum_{(i,j)\in\tilde{A}} \bar{f}_{i,j} + \sum_{(i,j)\in\mathscr{E}\setminus\tilde{A}} \bar{f}_{i,j} - \left( \sum_{(j,i)\in\tilde{A}} \bar{f}_{j,i} + \sum_{(i,j)\in\mathscr{E}\setminus\tilde{A}} \bar{f}_{j,i} \right)$$

by definition of $\tilde{A}$

$$= \sum_{(i,j)\in\mathscr{E}} \bar{f}_{i,j} - \sum_{(j,i)\in\mathscr{E}} \bar{f}_{j,i}$$
$$= b_i$$

by definition of LP1.

Moreover, for each $(i,j) \in \tilde{A}$, $\hat{f}_{i,j} = \bar{f}_{i,j} \geq 0$ and also for each $s \in \mathscr{T}$, $\hat{f}_s = \frac{\bar{f}_{i,j}}{\rho_{i,j}} \geq 0$. Therefore, $\hat{f}$ is a feasible solution for the LP2. The amount of the objective function of LP2 corresponding to this feasible solution is:

$$\sum_{(i,j)\in\tilde{A}} \hat{f}_{i,j} + \sum_{s\in\mathscr{T}} \hat{f}_s = \sum_{(i,j)\in\tilde{A}} \hat{f}_{i,j} + \sum_{s\in\mathscr{T}} 1 \cdot \hat{f}_s$$
$$= \sum_{(i,j)\in\tilde{A}} \hat{f}_{i,j} + \sum_{s\in\mathscr{T}} \left( \sum_{(i,j)\in A_s} \rho_{i,j} \right) \cdot \hat{f}_s$$

by the fact that $\rho_{i,j} = \mu_{tr}(v')$ where $(i,j) = (v^{tr}, v')$ or $(i,j) = (v_a^{tr}, v_a')$ and that $\sum_{(i,j)\in A_s} \rho_{i,j} = 1$

$$= \sum_{(i,j)\in\tilde{A}} \hat{f}_{i,j} + \sum_{s\in\mathscr{T}} \sum_{(i,j)\in A_s} \rho_{i,j} \cdot \hat{f}_s$$

by simple term manipulation

$$= \sum_{(i,j)\in\tilde{A}} \bar{f}_{i,j} + \sum_{s\in\mathscr{T}} \sum_{(i,j)\in A_s} \bar{f}_{i,j}$$

by definition of $\hat{f}_s$

$$= \sum_{(i,j)\in\tilde{A}} \bar{f}_{i,j} + \sum_{(i,j)\in\mathscr{E}\setminus\tilde{A}} \bar{f}_{i,j}$$

by definition of $\tilde{A}$

$$= \sum_{(i,j)\in\mathscr{E}} \bar{f}_{i,j}.$$

As a consequence, since every feasible solution for LP1 is a feasible solution for LP2 and vice versa, and the value of the objective functions is the same, we have that LP1 and LP2 are equivalent. ■

Since both LP1 and LP2 are equivalent to the $LP(t, a, \mu, \mathcal{R})$ LP problem, we exploit the structure of LP2 to improve the efficiency of checking for a solution of $LP(t, a, \mu, \mathcal{R})$. Simultaneously, we also improve the complexity of deciding weak bisimulation. Amongst all available versions of polynomial algorithms for solving a linear programming problem, we resort to a state-of-the-art polynomial interior point method [Ans99] which, to the best of our knowledge, is equipped with the tightest known worst case complexity.

**Theorem 5.2.** *Consider a rational PA $\mathfrak{P}$, the action $a$, the probability measure $\mu \in \mathrm{Disc}(S)$, the equivalence relation $\mathcal{R}$ on $S$ and a state $t \in S$. Let $N = |\mathfrak{P}|$. Then, checking the feasibility of the $LP(t, a, \mu, \mathcal{R})$ LP problem can be done in $\mathcal{O}(\frac{N^3}{\ln N} \cdot L)$ where $L$ is the bit size of the problem.*

*Proof.* By Lemmas 5.1 and 5.2, $LP(t, a, \mu, \mathcal{R})$ is feasible if and only if LP2 is feasible. Now, consider the dual of LP2; by assigning the dual variables $\pi_s$ for each $s \in \mathscr{V}$, hence $\mathcal{O}(N)$ variables, we get the following dual LP problem:

$$\begin{array}{rll} \text{DLP2: max} & \sum_{s\in\mathscr{V}} b_s \cdot \pi_s & \\ \text{subject to:} & \pi_i - \pi_j \leq 1 & \text{for each } (i,j) \in \tilde{A} \\ & \sum_{t\in\mathscr{V}} a_s^t \cdot \pi_t \leq 1 & \text{for each } s \in \mathscr{T}. \end{array}$$

By using a state-of-the-art preconditioned conjugate gradient (PCG) method with a partial updating procedure [Ans99], this LP problem can be solved optimally in $\mathcal{O}(\frac{N^3}{\ln N} \cdot L)$ where $L$ is the bit size of the problem. At termination of the algorithm, we have two possible cases:

1. The dual LP problem has a finite optimal objective value: by the strong duality Theorem 3.4, the original LP2 is feasible and also has a finite optimal objective value.
2. The dual LP problem is unbounded: by the strong duality Theorem the original LP2 is infeasible.

Thus, by solving the dual LP problem efficiently, we can verify the existence of a weak combined transition for the given *PA*. ■

Notably, if we were to use the interior point method directly on the original LP problem instead of LP2, we would face an extra factor $N$ in the complexity bound. This is because the running time of the method depends on the number of variables: The number of variables occurring in $LP(t, a, \mu, \mathcal{R})$ is $\mathcal{O}(N^2)$ while the number of variables in LP2 is $\mathcal{O}(N)$. This reduction directly translates into a reduced worst case complexity, and this is especially appreciable if working with large probabilistic automata.

**Corollary 5.1.** *Given two PAs $\mathfrak{P}_1$ and $\mathfrak{P}_2$, let $N = |\mathfrak{P}_1| + |\mathfrak{P}_2|$. Checking $\mathfrak{P}_1 \approx \mathfrak{P}_2$ can be done in time $\mathcal{O}(\frac{N^6}{\ln N} \cdot L)$ where $L$ is the maximum bit size of the LP problems solved in* FindSplit *and* Refine.

*Proof.* Immediate by Proposition 5.1 and Theorem 5.2. ■

Since the worst case runtime bound essentially depends on the type of the polynomial algorithm used to solve the LP problem, any advancement in LP problem solution complexity directly improves the complexity of the weak bisimulation decision problem.

**Remark 5.4.** *If considering the structure of the $LP(t, a, \mu, \mathcal{R})$ LP problem, one might observe that it is in essence a system of linear equations with non-negativity constraints. So, we may consider instead to use elimination techniques (inspired by Gaussian elimination) to reduce the number of variables and constraints we have in the LP problem:*

1. *take one of the linear equations, say $f_{v,v^{tr}} - \sum_{(v^{tr},u)\in E} f_{v^{tr},u} = 0$ and one variable occurring in it, say $f_{v,v^{tr}}$;*

2. *express the variable as linear combination of the other variables, i.e.,* $f_{v,v^{tr}} = \sum_{(v^{tr},u)\in E} f_{v^{tr},u}$;
3. *replace each occurrence of the variable with such combination, i.e.,* $f_{v,v^{tr}}$ *by* $\sum_{(v^{tr},u)\in E} f_{v^{tr},u}$.

*If we iterate this process until no more variables can be isolated at step 2, we obtain another LP problem that is equivalent to the original one.*

*Now, since we are not interested in the actual value of the variables, but only on whether the problem is feasible, we can eliminate the equations we considered at step 1 and the corresponding variables at step 2. This results in an LP problem no more equivalent to the original one, but it is easy to show that the latter is feasible if and only if the original problem is.*

*As an example, consider the following LP problem:*

$$\begin{array}{llll} f_0 - f_1 - f_2 - f_3 = 0 & f_0 = 1 & f_0 \geq 0 & f_1 \geq 0 \\ f_1 + f_2 - f_4 = 0 & f_4 = 0.5 & f_4 \geq 0 & f_2 \geq 0 \\ f_3 - f_5 = 0 & f_5 = 0.5 & f_5 \geq 0 & f_3 \geq 0 \end{array}$$

*In a single time, if we replace* $f_0$, $f_4$, *and* $f_5$ *with their respective values, we obtain:*

$$\begin{array}{llll} 1 - f_1 - f_2 - f_3 = 0 & f_0 = 1 & 1 \geq 0 & f_1 \geq 0 \\ f_1 + f_2 - 0.5 = 0 & f_4 = 0.5 & 0.5 \geq 0 & f_2 \geq 0 \\ f_3 - 0.5 = 0 & f_5 = 0.5 & 0.5 \geq 0 & f_3 \geq 0 \end{array}$$

*Now, by replacing* $f_3$ *with* $0.5$, *the system becomes:*

$$\begin{array}{llll} 1 - f_1 - f_2 - 0.5 = 0 & f_0 = 1 & 1 \geq 0 & f_1 \geq 0 \\ f_1 + f_2 - 0.5 = 0 & f_4 = 0.5 & 0.5 \geq 0 & f_2 \geq 0 \\ f_3 - 0.5 = 0 & f_5 = 0.5 & 0.5 \geq 0 & 0.5 \geq 0 \end{array}$$

*and by substituting* $f_1$ *with* $0.5 - f_2$*:*

$$\begin{array}{llll} 1 - 0.5 + f_2 - f_2 - 0.5 = 0 & f_0 = 1 & 1 \geq 0 & 0.5 - f_2 \geq 0 \\ f_1 = 0.5 - f_2 & f_4 = 0.5 & 0.5 \geq 0 & f_2 \geq 0 \\ f_3 = 0.5 & f_5 = 0.5 & 0.5 \geq 0 & 0.5 \geq 0 \end{array}$$

*that is,*

$$\begin{array}{llll} 0 = 0 & f_0 = 1 & 1 \geq 0 & 0.5 - f_2 \geq 0 \\ f_1 = 0.5 - f_2 & f_4 = 0.5 & 0.5 \geq 0 & f_2 \geq 0 \\ f_3 = 0.5 & f_5 = 0.5 & 0.5 \geq 0 & 0.5 \geq 0 \end{array}$$

*This system is feasible and it has a solution for each* $0 \leq f_2 \leq 0.5$.

*This approach looks promising, but in fact is much more expensive than the result achieved by Theorem 5.2: if we ignore the bit size of the problem, for an* $n \times n$ *matrix, the Gaussian elimination has complexity* $\mathcal{O}(n^3)$ *where* $n$ *is the number of variables in the system of equations (corresponding to the number of columns of the matrix). In our setting, we have an* $m \times n$ *matrix with* $m > n$, *thus the actual complexity is larger than* $\mathcal{O}(n^3)$. *If we now express the complexity of the Gaussian elimination approach in terms of* $N = |\mathfrak{P}|$, *since we have* $\mathcal{O}(N^2)$ *variables, the resulting complexity is at least* $\mathcal{O}(N^6)$, *without considering the complexity of solving the remaining LP problems.*

**Non-rational automata.** The class of rational probabilistic automata, to the best of our knowledge, encompasses all *PA*s that have appeared in practical applications. One may nevertheless consider relevant also the analysis of *PA*s with real valued probabilities.

One possible way to represent LP problems with real data is to use a model of computation that can perform any elementary arithmetic operation in constant time, regardless of the type of the operand. Another option is to encode reals as finite precision rationals. For a survey on the theory of computation over real numbers we refer the reader to [BSS89, Bel01].

When using finite precision rationals, the representation of the *PA* must become approximate, and still the size needed for this can no longer be guaranteed to be bounded by a polynomial. If assuming the rational approximation scheme being employed by the user, we are back to the rational setting for the LP problem solution process, and it is left to the user to interpret the outcome on the real valued *PA*. If instead the algorithm performs the approximation prior to solving the induced LP problem, the user may in general lack knowledge on how to transfer the result back to the original real valued *PA*.

### 5.4.2 Efficient Solution: Exploiting Structure

We now consider the practical efficiency of deciding probabilistic automata weak bisimulation. We first discuss available algorithms that can be employed. We show that the underlying structure of the problem enables us to check feasibility of the LP problem more efficiently than by just resorting to a general purpose LP solver implicitly finding the optimal solution. Afterwards we discuss other methods that are known to be more efficient in general but turn out to be unsuitable for solving the $LP(t, a, \mu, \mathcal{R})$ LP problem.

Working with a linear programming problem allows practitioners to use the omnipresent simplex method as an extremely efficient computational tool. It is worthwhile to note that the efficiency of the simplex method is measured as the number of pivots needed to solve the LP problem. Moreover, practical experiments show that although this method is highly efficient, there exist problems that require an exponential number of pivots. This means that the worst case theoretical complexity of the simplex method is exponential time [KM72]. However, computational experience on thousands of real-world problems reveals that the number of pivots is usually polynomial in the number of variables and of constraints. For a comprehensive survey on the efficiency of the simplex method, we refer the interested reader to [Sha87].

Since the LP1 problem is a minimum cost flow problem on the network $\mathcal{G}(t, a, \mu, \mathcal{R})$ extended with an additional set of proportional flow constraints, we consider the usage of efficient algorithms that solve the problem directly on the flow network itself. One such algorithm is the network simplex algorithm [BF12] for the minimum cost proportional flow problem that improves the per iteration running time considerably with respect to the simplex method, as long as the number of nodes in the network is at least an order of magnitude larger than the number of side constraints in the LP problem [Cal02, MSJ11, BF12, MSJ13]. So, the network simplex algorithm is a candidate for improving the running time required to solve LP1. However, the number of side constraints coincides with the number of transition nodes in the LP1 problem. Since the number of transitions in the automaton is usually larger than the number of states, we have that the number of side constraints is linear in the number of nodes, and thus the above assumption is not satisfied. Still, a more accurate analysis tells us that, in our setting, the resulting per iteration running time of both methods is in the same complexity class, as shown in Table 5.1. Since it is known that the network simplex algorithm without side constraints

| | | $LP(t, a, \mu, \mathcal{R})$ | LP1 | LP2 | DLP2 |
|---|---|---|---|---|---|
| Variables/Arcs | $n$ | $\mathcal{O}(N^2)$ | $\mathcal{O}(N^2)$ | $\mathcal{O}(N)$ | $\mathcal{O}(N)$ |
| Constraints | $m$ | $\mathcal{O}(N^2)$ | $\mathcal{O}(N)$ | $\mathcal{O}(N)$ | $\mathcal{O}(N)$ |
| Proportional Flow Sets | $p$ | not applicable | $\mathcal{O}(N)$ | not applicable | not applicable |
| Free Arcs | $n'$ | $\mathcal{O}(N^2)$ | $\mathcal{O}(N)$ | $\mathcal{O}(N)$ | $\mathcal{O}(N)$ |
| Simplex Method | $\mathcal{O}(nm)$ | $\mathcal{O}(N^4)$ | $\mathcal{O}(N^3)$ | $\mathcal{O}(N^2)$ | $\mathcal{O}(N^2)$ |
| Network Simplex Algorithm [MSJ13] | $\mathcal{O}(n' + mp + p^3)$ | not applicable | $\mathcal{O}(N^3)$ | not applicable | not applicable |

Table 5.1: Complexity comparison

performs better than the simplex method [AMO93], it is still worthwhile to consider its usage in an implementation.

Up to now, we have discussed that the simplex method and the network simplex algorithm [BF12] appear quite competitive in solving the LP1, and that the flow network structure underlying LP1 motivates the use of the network simplex algorithm. On the other hand, we can take the dual of the equivalent LP2. This allows us to deal with a smaller sized LP problem which is still close to a well known combinatorial problem by itself. To clarify the point, consider the dual DLP2 of the LP2 problem:

$$\begin{aligned} \text{DLP2: max} \quad & \sum_{s \in \mathcal{V}} b_s \cdot \pi_s \\ \text{subject to:} \quad & \pi_i - \pi_j \leq 1 \qquad \text{for each } (i,j) \in \tilde{A} \qquad (5.1) \\ & \sum_{t \in \mathcal{V}} a_s^t \cdot \pi_t \leq 1 \qquad \text{for each } s \in \mathcal{T} \qquad (5.2) \end{aligned}$$

The number of constraints in DLP2 is $\mathcal{O}(N)$, just as for LP1. The number of variables in DLP2 is $\mathcal{O}(N)$ which compares favorably with $\mathcal{O}(N^2)$, the number of variables in the original LP1. This observation is particularly important whenever the number of transitions is considerably larger than the number of states in the network. The dual LP problem can again be solved very efficiently using a state-of-the-art variant of the interior point method [Ans99]. This algorithm is a preconditioned conjugate gradient (PCG) method with a partial updating procedure which works excellent in practice as well. The algorithm is available in the software tools like CPLEX and LOQO. Furthermore, DLP2 has itself a combinatorial structure, i.e., it is the dual of the well known shortest path problem although with additional side constraints. Taking the advantage of this combinatorial property may help in the design of a more efficient algorithm to solve the problem.

Table 5.1 summarizes the size of the proposed LP problems and the per-iteration complexity of the simplex method and of the network simplex algorithm. Since each variable in the LP problem corresponds to an arc in the network, we identify by $n$ both variables and arcs; on networks, each arc either belongs to a proportional flow set or is a free arc. The computational comparison of three LP problems is described based on $N$ which is the size of the automaton $\mathfrak{P}$. It is immediate to see that LP2 and DLP2 are the smallest problems that are at least one degree smaller than the other LP problems, making them more suitable as input for the LP solvers.

### 5.4.3 Efficient Solution: Unsuitable Approaches

As we have seen, the $LP(t, a, \mu, \mathcal{R})$ LP problem can be solved efficiently using the simplex method or the network simplex algorithm.

Several other solutions have been proposed in the literature to solve variations of LP problems more efficiently: among others, there are approximation algorithms [Vaz04], electrical flow representation [CKMS11], network decomposition [Pul89], and Lagrangian relaxation [BT97]. As we will see in the remainder of this section, all these approaches are not suitable for solving the $LP(t, a, \mu, \mathcal{R})$ LP problem and its equivalent reformulations, but for different reasons: either because the corresponding model does not enable an encoding of the $LP(t, a, \mu, \mathcal{R})$ LP problem at hand, or the answer provided by the algorithm can not always be mapped into an answer to our problem, or the algorithm is prohibitively expensive in case of a positive answer.

Despite being unsuitable, we review our findings concerning these methods for two main motivations: the first is to clearly identify the specific characteristics of the problem faced, the second motivation is to propose a new challenging problem to both the optimization and the probabilistic automata world.

#### 5.4.3.1 Approximation Algorithms

As a first approach, we consider the use of an approximation algorithm to check the feasibility of the original LP problem by dropping the proportional flow constraints and then solving the remaining linear programming problem efficiently. This exploits that the remaining LP problem is actually a minimum cost flow problem on the network $\mathcal{G}(t, a, \mu, \mathcal{R})$ and the general network simplex algorithm can solve it extremely efficiently. If the relaxed problem is infeasible, so it is the original LP problem; otherwise, we get a feasible solution that may not be feasible for the original one. In this case, we assign fixed weights to each proportional flow constraint and increase the weight for the violated side constraints as a penalization. This procedure, known as the multiplicative weight update method [AHK12], is repeated until a feasible solution which is near optimal is found. The advantage of this approach is that in each iteration we deal with a well structured LP problem that can be solved efficiently.

The main problem of this approach is that in general a positive result does not imply the existence of the corresponding weak transition. Consider, for example, the automaton whose only transitions are $s \xrightarrow{a} \mu$ and $t \xrightarrow{a} \delta_v$ where $\mu = \{(v, p), (u, 1-p)\}$ for some $p \in [0, 1]$; suppose that $u \mathcal{R} v$. It is easy to verify that $\mu \not\mathcal{L}(\mathcal{R}) \delta_v$ for each $p \neq 1$, so there does not exist any weak transition $t \stackrel{a}{\Longrightarrow}_c \rho$ such that $\mu \mathcal{L}(\mathcal{R}) \rho$ since the only possible weak transition enabled by $t$ labelled by $a$ is $t \stackrel{a}{\Longrightarrow}_c \delta_v$. However the approximation algorithm gives us a positive answer whenever $p$ is close enough to 1, so a positive answer does not ensure the existence of the weak transition, unless we force the gap between optimal and near optimal objective values to be 0. But this may make the overall algorithm very expensive. However, for practical purposes, approximation algorithms can be used to refute the existence of a weak transition.

#### 5.4.3.2 Electrical Flows

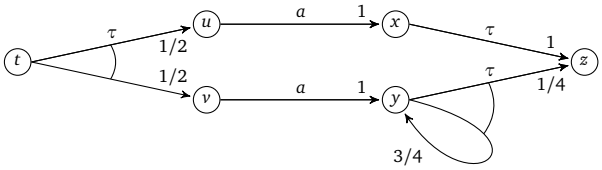
As a second approach we consider a physical metaphor for graphs by transforming the network $\mathcal{G}(t, a, \mu, \mathcal{R})$ as an electrical network. This comes with replacing the arcs of the network by resistors. Our goal is to arrive at a setting where we can use the state-of-the-art max flow algorithm [CKMS11] as an approximation algorithm to solve the original LP problem. The weakness of this approach is twofold: the approximate nature of the procedure has the same drawback as the previous approach, and furthermore the applicability of the transformation. Even if an efficient non-approximate algorithm was at hand, the transformation can not be applied, since it is restricted to undirected networks [CKMS11], while $G(t, a, \mu, \mathcal{R})$ is directed. Extending the results in [CKMS11] to directed networks is an open problem.

To make the network directed, we may represent each arc by a resistor and a diode that is a two-terminal component allowing the current to flow in a single direction. Even by using diodes to direct the network, we still need to solve two problems: cycles and nondeterminism. In an electrical network it is not possible to have the current going through a passive cycle since the overall potential difference in the cycle is zero, unless we use some fictitious voltage generator that breaks the cycle, while in probabilistic automata it is common to have internal cycles: consider for instance the transition $s \xrightarrow{\tau} \mu$ where $\mu = \{(t, 0.3), (s, 0.7)\}$; by using the determinate scheduler $\sigma$ that stops in $t$ and performs $s \xrightarrow{\tau} \mu$ in $s$, we obtain the weak transition $s \stackrel{\tau}{\Longrightarrow}_{\mathrm{c}} \delta_t$, that is, we eventually leave the self-loop with probability 1. In order to obtain a similar result in an electrical network, we have to add a fictitious voltage generator in the cycle corresponding to the self-loop that generates the correct potential difference. Finding such difference is essentially equivalent to defining the scheduler. The second problem is related to nondeterminism: suppose that we have two transitions $s \xrightarrow{\tau} \mu$ and $s \xrightarrow{\tau} \rho$ where $\mu = \{(u, 0.3), (v, 0.7)\}$ and $\rho = \{(u, 0.7), (v, 0.3)\}$, so we can reach $v$ with different probabilities by using two different transitions. When we encode these two transitions in the electrical network, we obtain two parallel paths from $s$ to $v$ that are subject to the same voltage difference $V_{sv}$, so the current flows in both paths according to Ohm's law. However the scheduler can choose to perform only $s \xrightarrow{\tau} \mu$ thus in the network we should have a non null current from $s$ to $v$ in the path modelling such transition and a null current in the path corresponding to $s \xrightarrow{\tau} \rho$, that is, the former requires $V_{sv} > 0$ while the latter requires $V_{sv} = 0$ and this is clearly impossible.

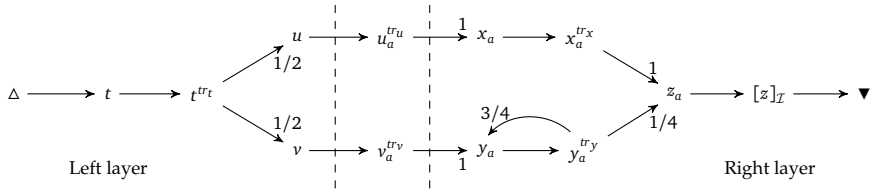
#### 5.4.3.3 Network Decomposition

As a third approach, we consider a natural decomposition of the state space of the underlying network. We aim at designing a parallel algorithm that speeds up the check for feasibility of the LP problem when $a \in E$. The underlying network can be seen as a network of three layers (here considered in horizontal layout): the left hand side and the right hand side layers correspond to the internal transitions (sets $L_1$ and $L_2$, respectively) while the central layer to the external transitions (set $L_a$). Moreover it is possible to change layer only from left to right. Using this intuition, each layer can be treated independently so that the network simplex algorithm instantiations can find the minimum cost flow in the left and the right layers in parallel. Then, to connect these two layers via the central one it is enough to solve a linear system of equations corresponding to the central arcs. This system can be solved in linear time.

However, this approach is not suitable as a negative answer does not imply the non-existence of the weak transition: consider the following *PA*.




It is immediate to see that $t \stackrel{a}{\Longrightarrow}_c \delta_z$; now, consider the identity relation over states $\mathcal{I}$ and the part of the corresponding layered network between source and sink, where numbers attached to arcs indicate probabilities, and are not part of the graph.




The solution of the left layer is unique and it assigns outgoing flow 1/2 to both vertices $u$ and $v$, while the optimal solution for the right layer assigns flow 1 to arcs $(x_a, x_a^{tr_x})$, $(x_a^{tr_x}, z_a)$ and flow 0 to arcs $(y_a, y_a^{tr_y})$, $(y_a^{tr_y}, z_a)$, and $(y_a^{tr_y}, y_a)$. By using these two optimal solutions, there is no way to obtain a solution for the central layer since there is only one path from $u$ to $x_a$ and flow requirements are different. However there exists a solution for the network as a whole that requires for the right layer the non-optimal feasible solution $f_{x_a, x_a^{tr_x}} = 1/2$, $f_{x_a^{tr_x}, z_a} = 1/2$, $f_{y_a, y_a^{tr_y}} = 4/2$, $f_{y_a^{tr_y}, z_a} = 1/2$, and $f_{y_a^{tr_y}, y_a} = 3/2$, thus a negative answer from the layered network decomposition does not imply that there does not exist a feasible solution for the whole network. The core reason is that the optimal solution of one layer may be not part of a feasible solution of the whole network: a feasible solution may only be induced by a sub-optimal layer solution.

#### 5.4.3.4 Lagrangian Relaxation

As a last approach, we consider a Lagrangian relaxation [BT97] algorithm to solve the dual LP problem efficiently. Consider again the DLP2 problem; in order to form a Lagrangian relaxation of DLP2, we multiply the set of constraints (5.2) by non-negative Lagrangian multipliers $\lambda_s$, $s \in \mathcal{T}$, and we add them to the objective function, obtaining the following relaxed dual LP (RDLP) problem:

$$L(\lambda)\text{: max} \quad \sum_{t \in \mathcal{V}} (b_t + \sum_{s \in \mathcal{T}} \lambda_s \cdot a_s^t) \cdot \pi_t - \sum_{s \in \mathcal{T}} \lambda_s$$

$$\text{subject to:} \quad \pi_i - \pi_j \leq 1 \qquad \text{for each } (i,j) \in \tilde{A}$$

For a fixed vector $\lambda$, this LP problem can be efficiently solved using the simple and fast algorithm described in [HN94].

Consider the Lagrangian dual LP (LDLP) problem:

$$\begin{aligned} \text{LDLP: min} \quad & L(\lambda) \\ \text{subject to:} \quad & \lambda \geq 0 \end{aligned}$$

The purpose of the LDLP problem is to find the tightest bound $L(\lambda)$ for the possible values of $\lambda$. Since RLDP is always feasible with solution $\pi_t = 0$ for each $t \in \mathcal{V}$, we can make no claims on the feasibility of the original DLP2 problem unless (1) actually finding a feasible solution for the original DLP2 problem, or (2) proving that the optimum solution for LDLP is bounded. This is the main point that induces the weakness of this approach as an efficient verification procedure.

## 5.5 Implementation of Minimization

The results presented thus far provide ample understanding for an implementation of a quotienting algorithm. This section presents and discusses such an implementation which is tailored to the computation of the minimal automaton that is weak probabilistic bisimilar to a given one [EHS+13]. In fact, some intricate problems remained to be overcome to make the approach effective and scalable. These problems are rooted in numerical aspects of the computations at hand as well as in the often excessive number of feasibility checks needed. Both these aspects are genuine to the setting considered and neither occur in the context of minimizing labelled transition systems, nor in other stochastic minimization contexts.

We here report on our second generation prototype minimizer, implemented in Java. It has a modular structure and it can delegate the feasibility checks either to an LP solver, or to an SMT solver. We use LpSolve [LpS] as LP solver, the GLP Kit [GLP] as exact arithmetic LP solver, and Z3 [dMB08] as SMT solver. We encode our SMT formulation according to the SMT-LIB format [BST10] allowing the use of other solvers. We can use an SMT solver instead of an LP solver since Proposition 5.2 relates the existence of the desired weak transition $t \stackrel{a}{\Longrightarrow} \mu_t$ such that $\mu \; \mathcal{L}(\mathcal{R}) \; \mu_t$ with the feasibility of the $LP(t, a, \mu, \mathcal{R})$ problem, so we are not interested in the optimal solution, but just in a solution.

We perform the feasibility check directly on the original $LP(t, a, \mu, \mathcal{R})$ problem. This allows us to maintain an undisguised view on the structure of the problem, which we considered important to ensure the correctness of the prototype implementation, and also to assess the relative share of the different algorithmic steps to the overall runtime and space requirements.

### 5.5.1 Implementation Details

In our prototype we have implemented several heuristics in order to minimize the number of solver calls needed to compute the coarsest weak bisimilarity. We can classify these heuristics in two classes: the *pre-bisimulation reductions* and the *in-loop optimizations*. Finally, we consider the extraction of exact solutions from inexact solutions to improve the in-loop optimizations and the parallelization of the solver calls.

#### 5.5.1.1 Pre-Bisimulation Reductions

We reduce the automaton before computing the weak probabilistic bisimulation by removing irrelevant transitions and collapsing states that are trivially bisimilar. In particular, we remove internal self loops (i.e., transitions as $s \xrightarrow{\tau} \delta_s$), we merge all deadlock states in a single one and each pair of states $(s,t)$ in $t$ such that the only transition enabled by $s$ is $s \xrightarrow{\tau} \delta_t$. Moreover, we merge states $s$ and $t$ if the transitions they enable reach the same distributions via the same labels. These reductions are sound since it is easy to prove that all these merged states are weak bisimilar.

It is also possible to apply a preliminary bisimulation reduction based on strong (probabilistic) bisimulation [Seg06, Seg95] that would collapse some more states; note however that such reduction does not cover all above reductions; for instance, it does not remove self-loops as well as usually it does not collapse transitions like $s \xrightarrow{\tau} \delta_t$.

#### 5.5.1.2 In-Loop Optimizations

We adopt several optimizations in order to reduce the number of weak combined transitions computed by calling the solver. In particular, we implement the *internal* optimization that allows us to skip the LP problem construction and solution whenever the challenging transition at line 3 of the FindSplit procedure is of the form $s \xrightarrow{\tau} \mu_s$ with $\mu_s([s]_{\mathcal{R}}) = 1$. Such transition is trivially matched by any $t \in [s]_{\mathcal{R}}$ by performing no transitions at all. Similarly, by using the *direct transition* optimization we save a solver call if the state $t$ directly enables a transition $t \xrightarrow{a} \mu_t$ matching $s \xrightarrow{a} \mu_s$, i.e., we check whether there exists $(t, a, \mu_t) \in T$ such that $\mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t$. Finally, we maintain a *transition cache* containing for each state $t$, the list of computed transitions $t \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ where the distribution $\mu_t$ has been generated by the solver on the $LP(t, a, \mu_s, \mathcal{R})$ problem for some challenging transition $s \xrightarrow{a} \mu_s$. This cache allows us to save a solver invocation whenever the cache contains a matching transition. Suppose that we have already used the LP or SMT solver in order to find a matching transition $t \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu'_t$ for $s \xrightarrow{a} \mu'_s$ such that $\mu'_s \; \mathcal{L}(\mathcal{R}') \; \mu'_t$ for some partitioning $\mathcal{R}'$. As long as $\mu_s \; \mathcal{L}(\mathcal{R}) \; \mu'_t$ holds for the current partitioning $\mathcal{R}$, there is no need to call again the solver since it is going to give a positive answer, so we can save such call. In order to be effective, we need to keep the cache updated and this can be achieved easily since the only operation we need is to add entries to the cache, provided that we store the computed $t \stackrel{a}{\Longrightarrow}_{\mathrm{c}} \mu_t$ transitions.

#### 5.5.1.3 Exact solutions from inexact solutions

The optimizations presented previously need to compare distributions, either when checking $\mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t$ or while searching for a cached weak combined transition. In order to be effective, rational numbers cannot be represented using floating-point numbers since small rounding errors may render floating-point comparisons to become incorrect. For the direct transition optimization we overcome the problem by using exact representations for probabilities, such as infinite precision integers. For the caching, we need to retrieve the optimal feasible solution from an LP or SMT solver. This is only possible if we use an SMT solver or an LP solver equipped with exact arithmetic, since floating-point based LP solvers only provide inexact solutions for the system of inequalities, making the cache rather useless. We solve this problem via the *inexact to exact* optimization, i.e., by finding the *exact* solu-

tion as long as the underlying rational number does not have a large denominator. Any number $p \in \mathbb{R}$ can uniquely be represented as a *simple continuous fraction* of the form:

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{\ddots}}$$

Therefore any number can be represented canonically by the sequence $a_0, a_1, \ldots$; note that such sequence is finite if and only if $p \in \mathbb{Q}$. The canonical representation can be obtained by using the following inductive definitions [JT80]:

$$a_i = \lfloor p_i \rfloor \qquad p_0 = p \qquad p_{i+1} = (p_i - \lfloor p_i \rfloor)^{-1}$$

Any number can be approximated by $c_n$, the continuous fraction obtained from the finite sequence $a_0, \ldots, a_n$. In fact, $c_n$ is the *best rational approximation*: any other rational number that is closer to $p$ has a denominator larger than $c_n$. Moreover, $c_{n+1}$ has a denominator larger than $c_n$, thus the sequence $\{c_i\}$ converges absolutely to $p$. We calculate the sequence of approximations $c_0, c_1, \ldots$ until we find a value $c_n$ such that $|p - c_n| < \varepsilon$ for a predefined $\varepsilon > 0$.

#### 5.5.1.4 Parallel solvers

In FindSplit, for each state $s$, for the current partition $\mathcal{R}$ we have to check whether all its enabled transitions have a matching weak transition in all the states in $[s]_{\mathcal{R}}$. During such a loop the partition $\mathcal{R}$ does not change, therefore we can generate all the LP problems in advance, so as to solve them in parallel. We implemented a simple thread pool where each single task checks all the matching transitions from a given $t \in [s]_{\mathcal{R}}$, i.e., the first task manages $t_1$, the second task $t_2$, and so on.

### 5.5.2 Case Studies

We evaluated our prototypical implementation by applying it to several cases studies taken from the literature. Experiments were run on four AMD® Opteron® 8350 (Quad-core) 2GHz with 120GB of RAM. We only used 14 out of 16 cores with the memory usage restricted to 8GB. Time-outs correspond to experiments that took more than 6 hours to complete. The models we considered are IEEE 802.3 CSMA/CD protocol, dining cryptographers, IEEE 1394 FireWire root contention protocol, IEEE 802.11 Wireless LAN, and IPv4 Zeroconf protocol that we have taken from the PRISM benchmark suite [PRI] and only minor changes have been made to manage the shared variables for synchronization. More information on the case studies and the choice of parameters is directly available from the benchmark suite [PRI]. We hide the action `time` in the variant with suffix "-nt" and we unify similar actions in the "-sa" variant; for example, we rename `send1` and `send2` to `send`.

#### 5.5.2.1 Additional Reductions

Given a *PA* $\mathfrak{P}$, we denote by $\mathfrak{P}_{\bowtie}$ the automaton resulting from the repeated application of the pre-bisimulation reductions until such reductions do not change the automaton anymore. The effectiveness of these reductions is shown in Table 5.2, where we report for

| Problem | $\|S\|$ | $\|T\|$ | $\|S_{\bowtie}\|$ | $\|T_{\bowtie}\|$ | $t_{\mathfrak{P}_{\bowtie}}$ | $\|[S]_{\approx}\|$ | $\|[T]_{\approx}\|$ | $t_{[\mathfrak{P}_{\bowtie}]_{\approx}}$ |
|---|---|---|---|---|---|---|---|---|
| csma2 | 1038 | 1054 | 835 | 849 | 1s | 449 | 459 | 1s |
| csma2-sa | 1038 | 1054 | 621 | 630 | 7s | 233 | 237 | < 1s |
| csma2-sa-nt | 1038 | 1054 | 91 | 98 | < 1s | 87 | 90 | < 1s |
| dining4 | 2165 | 4540 | 161 | 300 | < 1s | 1 | 1 | < 1s |
| firewire3 | 611 | 694 | 425 | 469 | 5s | 425 | 469 | 5s |
| firewire3-nt | 611 | 694 | 29 | 62 | < 1s | 4 | 4 | < 1s |
| wlan_dl0dl6 | 97 | 148 | 63 | 94 | < 1s | 59 | 86 | 1s |
| wlan0col0 | 2954 | 3972 | 1097 | 1591 | 14s | 798 | 1092 | 120s |
| zeroconf | 670 | 827 | 341 | 433 | < 1s | 334 | 420 | 14s |
| zeroconf-nt | 670 | 827 | 52 | 75 | < 1s | 41 | 52 | < 1s |

Table 5.2: Minimization overview

| Problem | $\|T_{\bowtie}\|$ | $\Delta$ | *DT* | *CH* | *TC* | $t_{\approx}$ | $t_{Solver}$ | $\|\approx\|$ |
|---|---|---|---|---|---|---|---|---|
| csma2 | 849 | 24297 | 82337 | 150 | 11700 | 288s | 243s | 449 |
| csma2-sa | 630 | 9111 | 66162 | 0 | 6067 | 97s | 83s | 233 |
| csma2-sa-nt | 98 | 1739 | 186 | 14 | 1118 | 12s | 6s | 87 |
| dining4 | 300 | 45440 | 240 | 2175 | 145 | 6s | 5s | 1 |
| firewire3 | 469 | 30842 | 34070 | 257 | 1311 | 44s | 36s | 425 |
| firewire3-nt | 62 | 664 | 833 | 48 | 166 | 2s | 1s | 4 |
| wlan_dl0dl6 | 94 | 107 | 277 | 46 | 405 | 4s | 1s | 59 |
| wlan0col0 | 1591 | 48284 | 102296 | 14902 | 30204 | 1h3m | 1h1m | 798 |
| zeroconf | 433 | 2063 | 30829 | 453 | 2065 | 59s | 47s | 334 |
| zeroconf-nt | 75 | 361 | 265 | 99 | 348 | 5s | 2s | 41 |

Table 5.3: Caching overview for SMT

several case studies the state and transitions space size of the original automaton $\mathfrak{P}$, of the corresponding $\mathfrak{P}_{\bowtie}$, and of the minimal automaton $[\mathfrak{P}]_{\approx}$. The columns $t_{\mathfrak{P}_{\bowtie}}$ and $t_{[\mathfrak{P}_{\bowtie}]_{\approx}}$ report the time needed to reduce $\mathfrak{P}$ to $\mathfrak{P}_{\bowtie}$ and to generate $[\mathfrak{P}_{\bowtie}]_{\approx}$ from $\mathfrak{P}_{\bowtie}$ and the given $\approx$, including the time for the post-quotient reductions of the automaton.

More precisely, we consider in $[\mathfrak{P}_{\bowtie}]_{\approx}$ also the time needed for removing redundant transitions (that requires further checks for the existence of the weak combined transitions) and for rescaling distributions (cf. Section 5.2.2). In particular, the former reduction requires to remove one transition at a time and check whether there exists a weak combined transition using only the remaining transitions that reaches the same distribution. The difference of time of the Wireless LAN case with respect to the other cases depends on the number of internal transitions still present in the quotient as well as the number of transitions leaving the state: if a state enables only one transition or only transitions with different external actions, then there is no need to try to remove such a transition, since we will obtain a negative answer for sure.

| Problem | $\lvert T_{\bowtie}\rvert$ | $\Delta$ | *DT* | *CH* | *TC* | $t_{\approx}$ | $t_{Solver}$ | $\lvert\approx\rvert$ |
|---|---|---|---|---|---|---|---|---|
| csma2 | 849 | 24289 | 73229 | 150 | 11650 | 560s | 556s | 449 |
| csma2-sa | 630 | 9111 | 67657 | 0 | 6078 | 121s | 118s | 233 |
| csma2-sa-nt | 98 | 1739 | 186 | 14 | 1118 | 6s | 6s | 87 |
| dining4 | 300 | 45440 | 240 | 2175 | 145 | 4s | 3s | 1 |
| firewire3 | 469 | 30842 | 33878 | 257 | 1311 | 68s | 66s | 425 |
| firewire3-nt | 62 | 664 | 833 | 48 | 166 | 1s | 1s | 4 |
| wlan_dl0dl6 | 94 | 107 | 275 | 45 | 410 | 1s | $<$ 1s | 59 |
| wlan0col0 | 1591 | 53768 | 109604 | 16584 | 30362 | 1h17m | 1h17m | 798 |
| zeroconf | 433 | 2258 | 35098 | 515 | 2063 | 71s | 69s | 334 |
| zeroconf-nt | 75 | 379 | 281 | 105 | 333 | 2s | 2s | 41 |

Table 5.4: Caching overview for LP with inexact to exact optimization

| Problem | $\lvert T_{\bowtie}\rvert$ | $\Delta$ | *DT* | *CH* | *TC* | $t_{\approx}$ | $t_{Solver}$ | $\lvert\approx\rvert$ |
|---|---|---|---|---|---|---|---|---|
| csma2 | 849 | 24297 | 55499 | 150 | 12139 | 1h2m | 1h2m | 449 |
| csma2-sa | 630 | 9111 | 66969 | 0 | 6136 | 555s | 544s | 233 |
| csma2-sa-nt | 98 | 1739 | 186 | 14 | 1118 | 18s | 17s | 87 |
| dining4 | 300 | 45440 | 240 | 2175 | 145 | 7s | 6s | 1 |
| firewire3 | 469 | 30842 | 34064 | 257 | 1311 | 404s | 399s | 425 |
| firewire3-nt | 62 | 664 | 833 | 48 | 166 | 2s | 2s | 4 |
| wlan_dl0dl6 | 94 | 107 | 275 | 45 | 410 | 2s | 2s | 59 |
| wlan0col0 | 1591 | —time-out— | | | | | | |
| zeroconf | 433 | 2009 | 29781 | 421 | 2069 | 207s | 200s | 334 |
| zeroconf-nt | 75 | 362 | 269 | 99 | 348 | 5s | 4s | 41 |

Table 5.5: Caching overview for GLP (exact solver)

#### 5.5.2.2 Quotient Performance

Tables 5.3, 5.4, and 5.5 show the effects of the different optimizations and the running time of the implementation of the Quotient procedure, where the solver used for checking $LP(t, a, \mu_s, \mathcal{R})$ is SMT, LP, and GLP, respectively.

After the columns with the problem and the number of transitions of $\mathfrak{P}_{\bowtie}$, the column $\Delta$ shows the number of challenging transitions verified via the internal optimization. Note that this condition trivially holds for each internal transition in the first round of the outer cycle since the initial partition contains only $S_{\bowtie}$ as class, so every internal transition reaches such class with probability 1. The column *DT* shows the number of times the defender $t$ has been able to use the direct transition optimization, i.e., by a transition $t \xrightarrow{a} \mu_t$, to match a challenging transition $s \xrightarrow{a} \mu_s$. Column *CH* reports the number of cache hits, that is, the challenging transitions $s \xrightarrow{a} \mu_s$ that have been matched by a transition stored in the transition cache. The column *TC* contains the number of challenging transitions for which we have solved the $LP(t, a, \mu_s, \mathcal{R})$ problem.

The following two columns show the time $t_{\approx}$ spent computing the weak bisimilarity $\approx$ including the time $t_{Solver}$ spent by the solvers for verifying all transitions counted in column *TC*. Since we use a pool of solvers running in parallel, $t_{Solver}$ is the time spent by

the slowest solvers in the pool, i.e., the time elapsing from the activation of the first solver to the completion of all solvers in the pool. Finally, the last column $|\approx|$ gives the size of the partition, i.e., the number of classes of bisimilar states. This value, decreased by 1, is also the number of refinements we perform in order to terminate the **while** loop of Quotient.

By comparing the running times for the SMT, LP, and GLP solvers, we can see that GLP is always the slowest one while SMT is the best performing among them. This can possibly be explained by the highly optimized code of Z3 and the remarkable results achieved by the SAT community on satisfaction modulo theory problems. The use of SMT, however, introduces an overhead in the computation, as highlighted by the comparison between the columns $t_{\approx}$ and $t_{Solver}$ of Table 5.3. Such overhead is mainly caused by the need of translating the LP problem construction into the textual SMT-LIB format [BST10] and then converting the solution (when the problem is satisfiable) back to numeric values. This induces a considerable usage of string operations and conversions that are not needed for the other solvers.

It is worthwhile to note that the values relative to the in-loop optimizations, including $\Delta$ and $DT$, strictly depend on the order in which we check the pairs of states belonging to the equivalence classes of the current partition $\mathcal{R}$. In fact, if we have a class $\mathcal{C}$ that has to be split in $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$, we may first split out $\mathcal{C}_1$, or $\mathcal{C}_2$, or $\mathcal{C}_3$. This means, for instance, that if we have to match from a state $t \in \mathcal{C}_1$ a transition $s \xrightarrow{\tau} \mu$ such that $\mu(\mathcal{C}_1 \cup \mathcal{C}_2) = 1$ but $\mu(\mathcal{C}_1) < 1$, only the latter split permits to increase $\Delta$. Moreover, the cache hits values are also affected by the fact that when we solve $LP(t, a, \mu_s, \mathcal{R})$, we look for one possible weak transition $t \stackrel{a}{\Longrightarrow}_c \mu_t$ such that $\mu_s \ \mathcal{L}(\mathcal{R}) \ \mu_t$ and in case there are several of them, we just store in the cache the transition computed by the solver. However there is no guarantee that such transition is the same for all solvers, thus the actual content of the cache can be different. This influences the successive cache hits, in particular after the split of one class.

It is worthwhile to remark that there are cases where an unlucky order of the partition splits may scale the transitions to check by a factor 20 that causes a significant increase of both $t_{\approx}$ and $t_{Solver}$. For this motivation, we do not fix an order on the pairs of states we check, and we let it depend on the nondeterministic insertion of states in the classes. In fact, for a fixed challenger state $s$, we generate the required $LP(t, a, \mu_s, \mathcal{R})$ problems for each challenging transition $s \xrightarrow{a} \mu_s$ and each defender state $t$ and then we use a pool of solvers running in parallel to verify them. In case of failure, we add the failing defender $t$ to $\mathcal{C}_f$ but the order of such additions is nondeterministic since it depends on the solver running time, the scheduling of the Java threads interacting with the solvers, the operating system scheduling of the solvers, and so on.

As the tables show, the in-loop optimizations reduce considerably the number of transitions that have to be checked by calling the LP or SMT solver, thus making the program faster. In fact, there is a strict correlation between the number of checked transitions and the time spent by the solver. This can be taken as a justification for the claim that weak bisimulation minimization does not scale very well to larger automata, unless the automaton is given as the composition of several smaller automata running in parallel.

In particular, the experiments demonstrate that our implementation uses a reasonable amount of time on automata whose size is the order of up to $3 \cdot 10^3$ states and transitions. Larger automata are likely to induce a time-out, as exemplified in Table 5.6. In these cases, compositional minimization is the suggested way to overcome this limitation, as we discuss in the next section.

| Components | $\lvert S_{\circlearrowleft}\rvert$ | $\lvert T_{\circlearrowleft}\rvert$ | $\lvert[S]_{\approx}\rvert$ | $\lvert[T]_{\approx}\rvert$ | $t_{\approx}$ |
|---|---|---|---|---|---|
| $c_3 \parallel p_1$ | 114 | 772 | 95 | 643 | 6s |
| $[c_3 \parallel p_1]_{\approx} \parallel p_2$ | 570 | 2502 | 285 | 1214 | 1h31m |
| $[[c_3 \parallel p_1]_{\approx} \parallel p_2]_{\approx} \parallel p_3$ | 1172 | 2352 | 1 | 1 | 1h38m |
| $c_3 \parallel p_1 \parallel p_2 \parallel p_3$ | 2720 | 5568 | —time-out— | | |

Table 5.6: Compositional minimization of consensus protocol for three parties

| Components | $\lvert S_{\circlearrowleft}\rvert$ | $\lvert T_{\circlearrowleft}\rvert$ | $\lvert[S]_{\approx}\rvert$ | $\lvert[T]_{\approx}\rvert$ | $t_{\approx}$ |
|---|---|---|---|---|---|
| $d_1 \parallel d_2$ | 33 | 76 | 6 | 14 | 2s |
| $[d_1 \parallel d_2]_{\approx} \parallel d_3$ | 39 | 92 | 6 | 14 | 4s |
| $[[d_1 \parallel d_2]_{\approx} \parallel d_3]_{\approx} \parallel d_4$ | 39 | 92 | 1 | 1 | 5s |
| $d_1 \parallel d_2 \parallel d_3 \parallel d_4$ | 2165 | 4540 | 1 | 1 | 5s |
| $d_1 \parallel d_2 \parallel \ldots \parallel d_8$ | 1687113 | 6952248 | 1 | 1 | 13s |
| $d_1 \parallel d_2 \parallel \ldots \parallel d_{10}$ | 42906171 | 220947474 | 1 | 1 | 18s |

Table 5.7: Compositional minimization of dining cryptographers: termination

| Components | $\lvert S_{\circlearrowleft}\rvert$ | $\lvert T_{\circlearrowleft}\rvert$ | $\lvert[S]_{\approx}\rvert$ | $\lvert[T]_{\approx}\rvert$ | $t_{\approx}$ |
|---|---|---|---|---|---|
| $i_1 = d_1 \parallel d_2$ | 41 | 92 | 20 | 41 | 4s |
| $i_2 = [i_1]_{\approx} \parallel d_3$ | 105 | 247 | 33 | 75 | 33s |
| $i_3 = [i_2]_{\approx} \parallel d_4$ | 180 | 482 | 45 | 107 | 330s |
| $i_4 = [i_3]_{\approx} \parallel d_5$ | 248 | 706 | 57 | 139 | 20m |
| $[i_4]_{\approx} \parallel d_6$ | 178 | 372 | 7 | 6 | 22m |
| $d_1 \parallel d_2 \parallel d_3 \parallel d_4$ | 2242 | 4708 | 5 | 4 | 39s |
| $d_1 \parallel d_2 \parallel \ldots \parallel d_5$ | 12042 | 31184 | 6 | 5 | 335s |
| $d_1 \parallel d_2 \parallel \ldots \parallel d_6$ | 63511 | 196642 | 7 | 6 | 22m |
| $d_1 \parallel d_2 \parallel \ldots \parallel d_7$ | 329784 | 1189626 | 8 | 7 | 59m |
| $d_1 \parallel d_2 \parallel \ldots \parallel d_8$ | 1689417 | 6961480 | 9 | 8 | 2h41m |

Table 5.8: Compositional minimization of dining cryptographers: Anonymity

### 5.5.3 Compositional Minimization

To show the practical effectiveness of the minimization in a compositional context, which we discussed in theory in Section 5.2.2, we consider two case studies that we fail to reduce otherwise, due to their prohibitive size: the Consensus Protocol with three parties and the Dining Cryptographers with four, eight, and ten cryptographers. For instance, by applying Definition 4.6, the four cryptographers case requires 38416 states and 6380 transitions; eight and ten cryptographers are essentially intractable since they involve around 1.5 and 300 billions states, respectively. We avoid this by constructing the model compositionally, applying weak bisimulation minimization on the intermediate automata. Moreover, to make this compositional minimization more effective, we use the hiding operator as soon as possible to restrict the visibility of the actions that are "private" between two automata.

Each of the Tables 5.6, 5.7 and 5.8 is split in two parts: the top part contains all intermediate steps performed by the compositional minimization leading to the minimization of

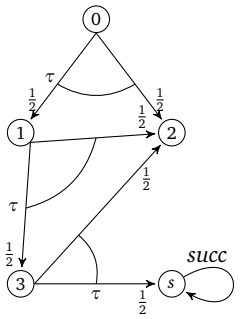



Figure 5.3: The minimized four dining cryptographers (anonymity)

the final automaton; in each row, the column $t_{\approx}$ includes the value of the previous row, thus reporting the total time used thus far. The bottom part of the table contains the number of states and transitions of the composed automata without intermediate minimization, and the time for the corresponding compositional minimization.

For the consensus protocol, we can see from the top part of Table 5.6 that the compositional minimization allows us to reduce the automaton to a single state and transition, representing the fact that the consensus is reached with probability 1, whereas the same reduction can not be obtained within the time-out by first composing the parties and then minimizing the composed automaton. The time required for the former approach actually depends on the intermediate step, where we reduce the automaton $[c_3 \parallel p_1]_{\approx} \parallel p_2$, that returns an automaton that is essentially half of the original one. The main motivation for this situation is that the intermediate automaton has still a lot of visible actions that can not be hidden since they are needed to synchronize with $p_3$.

On the contrary, the dining cryptographers protocol is a good example that shows how using the hiding operator as soon as possible permits to drastically reduce the size of the minimized automaton. In fact, since the synchronization happens only between cryptographers that are neighbors, such as $d_i$ and $d_{i+1}$, and such synchronization has to be secret, it makes sense to hide it just after having composed $d_i$ and $d_{i+1}$. Consider the termination of the dining cryptographers protocol with $n = 4$ cryptographers, as shown in the top part of Table 5.7: the proposed combination of hiding and compositional minimization permits to reduce any chain $d_1 \parallel \cdots \parallel d_l$, where $1 < l < n$, to an automaton with 6 states and 14 transitions. Then, for $l = n-1$, the synchronization of $d_1$ and $d_{n-1}$ with $d_n$ closes the circle of cryptographers that once minimized shows that the protocol terminates with probability 1.

For the anonymity property, the reduction of each chain does not lead to the same size but to an automaton whose size grows linearly with the number of cryptographers. This is caused by the fact that we have to keep track of the sequence of *agree*s announced by the cryptographers and this number clearly depends on the involved cryptographers. As in the PRISM benchmark, we assume that one cryptographer is paying and we check a particular outcome of the agreement, that is, we check that the probability of a given sequence of *agree*s and *disagree*s is $1/2^{n-1}$. It is immediate to see that the minimized automaton satisfies this property; see for instance the anonymity of four cryptographers in Figure 5.3, where the probability of reaching the state $s$ is $1/2^3$.

It is clear that for the dining cryptographers protocol the compositional minimization approach outperforms the minimization of the composition and we expect that this extends to all systems where few components share the same actions.

## 5.6 Concluding Remarks

In this chapter, we have considered efficiency analysis of deciding *PA* weak bisimulation which is known to be polynomial [TH15]. After a survey of available polynomial algorithms to solve an LP problem, we established an upper bound on the worst case complexity of the decision problem for general *PA*. We demonstrated that a small modification of the LP problem discussed in [TH15] enables taking advantage of the underlying network structure to improve the practical efficiency of solving the problem.

In addition, we have presented an implementation of the decision algorithm, in the form of a quotienting algorithm enabling to minimize probabilistic automata with respect to weak probabilistic bisimulation. We enhanced this algorithm with several heuristics that permit to reduce the running time of the program considerably, and have shown that minimization can be applied effectively to standard benchmark models. We have also investigated how compositional minimization techniques can be exploited for models consisting of several sub-automata running in parallel.

Although, probabilistic automata weak bisimulation admits an efficient decision algorithm and also is a congruence for parallel composition, hiding, and other operators on *PA*s; defining similar equivalence relations for the probabilistic systems with parametric uncertainty is not so straightforward in terms of computational complexity and compositionality. The latter will be at the core of our studies in the forthcoming chapters.

CHAPTER 6

# Compositional Minimization for Model Checking of Interval MDPs

In this chapter, we define the first bisimulation for model checking PCTL properties of interval *MDP*s which is in turn the first bisimulation for *MDP*s with uncertain transitions in general. Furthermore, we show how to compute the coarsest bisimulation by an algorithm based on comparing polytopes of probability distributions associated with each transition. We also discuss the worst case time complexity of the decision problem and show that it is **coNP**-complete. Afterwards, we build a bridge between *Probabilistic Verification* and *Robust Optimization* and establish a novel modelling of the probabilistic bisimulation problem for interval *MDP*s as an instance of an uncertain LP problem. In particular, we show that deciding bisimilarity of a pair of states can be encoded as the adjustable robust counterpart of an uncertain LP. We prove that using affine decision rules, probabilistic bisimulation relation can be approximated in polynomial time. We have implemented our approach and demonstrate its effectiveness on several case studies.

Finally, we address the key ingredients to build up the operations of parallel composition for composing interval MDP components at run-time. More precisely, we investigate how the parallel composition operator for interval MDPs can be defined so as to arrive at a congruence closure. As a result, we show that probabilistic bisimulation for interval MDPs is congruence with respect to two facets of parallelism, namely synchronous product and interleaving.

The material presented in this chapter is an extended version of the results reported in [HHK14, HHS+16b, HHHT16, HHT16].

**Organization of the chapter.** We start with Probabilistic Computation Tree Logic (PCTL) in Section 6.1 to express and analyse properties of *IMDP*s. In Section 6.2 we address the probabilistic bisimulation for model checking PCTL properties of *IMDP*s and afterwards, we focus on the computational complexity of the decision problem and the ways we can compute it algorithmically. In Section 6.3 we discuss compositionality methods for reasoning about *IMDP*s. Furthermore, we demonstrate the effectiveness of our approach on several case studies in Section 6.4. Finally, Section 6.5 concludes the chapter.

| | | |
|---|---|---|
| $s \models_{(\forall)} \mathit{true}$ | | |
| $s \models_{(\forall)} x$ | iff | $x \in L(s)$ |
| $s \models_{(\forall)} \neg\varphi$ | iff | $s \not\models_{(\forall)} \varphi$ |
| $s \models_{(\forall)} \varphi_1 \wedge \varphi_2$ | iff | $s \models_{(\forall)} \varphi_1 \wedge s \models_{(\forall)} \varphi_2$ |
| $s \models_{(\forall)} \mathbf{P}_{\bowtie p}(\psi)$ | iff | $\forall\sigma \in \Sigma, \forall\pi \in \Pi : \mathbf{Pr}_s^{\sigma,\pi}\left[\models_{(\forall)} \psi\right] \bowtie p$ |
| $\xi \models_{(\forall)} \mathbf{X}\varphi$ | iff | $s_2 \models \varphi$ |
| $\xi \models_{(\forall)} \varphi_1 \mathbf{U}^{\leq k} \varphi_2$ | iff | there exists $i \leq k$ such that $s_i \models_{(\forall)} \varphi_2$ and $s_j \models_{(\forall)} \varphi_1$ for every $1 \leq j < i$ |
| $\xi \models_{(\forall)} \varphi_1 \mathbf{U} \varphi_2$ | iff | there exists $k \in \mathbb{N}$ such that $\xi \models_{(\forall)} \varphi_1 \mathbf{U}^{\leq k} \varphi_2$ |

Table 6.1: PCTL semantics for model checking *IMDP*s

## 6.1 Probabilistic Computation Tree Logic (PCTL)

Formal verification of properties of *IMDP*s requires a proper language to precisely describe such properties. There are various ways to specify properties of *IMDP*s. Throughout this thesis, we focus on Probabilistic Computation Tree Logic (PCTL), a probabilistic logic derived from CTL [HJ94]. The syntax of PCTL state formulas $\varphi$ and PCTL path formulas $\psi$ is given by:

$$\varphi := \mathit{true} \mid x \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{P}_{\bowtie p}(\psi)$$
$$\psi := \mathbf{X}\varphi \mid \varphi_1 \mathbf{U} \varphi_2 \mid \varphi_1 \mathbf{U}^{\leq k} \varphi_2$$

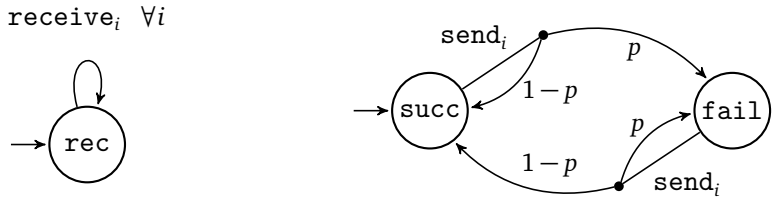
where $x \in \mathtt{AP}$, $p \in [0,1]$ is a rational constant, $\bowtie \in \{\leq, <, \geq, >\}$, and $k \in \mathbb{N}$.

The semantics of the logic PCTL depends essentially on the way nondeterminism is resolved for the probabilistic operator $\mathbf{P}_{\bowtie p}(\psi)$. In particular, in the setting of model checking we aim to check if the *IMDP* $\mathcal{M}$ satisfies the PCTL property $\varphi$ under all resolutions of nondeterminism, resolved by a scheduler and all resolutions of uncertainty, resolved by a nature. Thus, for the purpose of model checking PCTL properties, we quantify both the nondeterminisms universally and define the satisfaction relation $s \models_{(\forall)} \varphi$ as reported in Table 6.1. In the semantics description, $\models_{(\forall)} \psi$ denotes the set of infinite paths $\xi$ of the form $\xi = s_1 s_2 \cdots$ which satisfy $\psi$, i.e., $\{\xi \in \mathit{Paths}^{\mathit{inf}}_{\mathcal{M}} \mid \xi \models_{(\forall)} \psi\}$. It is easy to show that the set $\models_{(\forall)} \psi$ is measurable for any path formula $\psi$, hence the definition is correct. We will explain the PCTL semantics for other ways of resolving nondeterminisms in the next chapter.

## 6.2 Probabilistic Bisimulation for Model Checking IMDPs

In modelling real world systems, if uncertainty stems from a small number of different phenomena such as node failure or loss of a message, there would be a prohibitive redundancy over the states space. In the following example, we explain the redundancy of large models with a small source of uncertainty.

**Example 6.1.** *Consider a Wireless Sensor Network (WSN) containing $N$ sensors $S_1, S_2 \cdots S_N$ and a gateway $G$, all communicating over an unreliable channel. For simplicity, we assume that each sensor continuously sends some data to the gateway which are then pushed into an external server for further analysis. As the channel is unreliable, with some positive probability $p$ each message with data may get lost. The WSN can be seen as the parallel composition of gateway $G$ and sensors $S_i$ depicted below that synchronise over labels $\texttt{send}_i$'s and $\texttt{receive}_i$'s.*




(a) Gateway $G$ (b) Sensor $S_i$

*For instance environmental effects on radio transmission, mobility of sensor nodes or traffic burst (see e. g. [Rin09]) cause that the exact probability of failure is unknown. The estimation of this probability, e.g. by empirical data analysis, usually leads to an interval $p \in [\ell, u]$ which turns the model into an IMDP.*

*Let us stress that there is only one source of uncertainty appearing all over the state space no matter what is the number of sensors $N$. This makes many states of the model behave similarly. For example in the WSN, the parallel composition of the above model has $2^N$ states. However one can show that the bisimulation quotient has only $N + 1$ states. Indeed, all states that have the same number of failed sensors have the same behaviour. Thus, for limited source of uncertainty in a model obtained by compositional modelling, the state space reduction may be enormous.* ♦

In order to mitigate the impact of the resulting state space explosion phenomenon, we address probabilistic bisimulation to reduce the size of such an *IMDP* while preserving the PCTL properties it satisfies. To this end, we consider the notion of probabilistic bisimulation for the cooperative interpretation of *IMDP*s in which we assume that scheduler and nature are resolved *cooperatively* in the most *adversarial* way: in the game view of the bisimulation, challenging scheduler and nature work together in order to defeat the defender with a transition that can not be matched. As pointed out earlier, this is a natural semantics in verification of probabilistic systems.

Besides the cooperative behaviour, the choice of a probability distribution respecting the interval constraints can be done either *statically* [JL91], i.e., at the beginning once for all, or *dynamically* [SVA06, Iye05], i.e., independently at each computation step. In this chapter, we focus on the dynamic approach in resolving the stochastic nondeterminism: it is easier to work with algorithmically and can be seen as a relaxation of the static approach that is often intractable [BLW13, CSH08, DLL+11, GLD00].

Let $s \longrightarrow \mu_s$ denote a transition from $s$ to $\mu_s$ taken cooperatively, i.e., there is a scheduler $\sigma \in \Sigma$ and a nature $\pi \in \Pi$ such that $\mu_s = \sum_{a \in \mathscr{A}} \sigma(s)(a) \cdot \pi(s, a)$. In other words, $s \longrightarrow \mu_s$ if $\mu_s \in \mathrm{CH}(\bigcup_{a \in \mathscr{A}(s)} \mathscr{H}_s^a)$.

**Definition 6.1** (**Probabilistic bisimulation for model checking *IMDP*s**). *Given an IMDP $\mathcal{M}$, let $\mathcal{R} \subseteq S \times S$ be an equivalence relation. We say that $\mathcal{R}$ is a* probabilistic bisimulation *if for each $(s,t) \in \mathcal{R}$ we have that $L(s) = L(t)$ and*

$$\begin{array}{l} \text{for each} \quad s \longrightarrow \mu_s \\ \quad \text{there is} \quad t \longrightarrow \mu_t \quad \text{such that} \quad \mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t . \end{array}$$

*Furthermore, we write $s \sim_{(\forall)} t$ if there is a probabilistic bisimulation $\mathcal{R}$ such that $(s,t) \in \mathcal{R}$.*

Intuitively, each (cooperative) step of scheduler and nature from state $s$ needs to be matched by a (cooperative) step of scheduler and nature from state $t$; symmetrically, $s$ also needs to match $t$. In order to support the compositional reasoning, $\sim_{(\forall)}$ needs to be an equivalence relation. It is not difficult to see that $\sim_{(\forall)}$ is reflexive and symmetric. What remains is to show that it is also transitive. This is indeed a property of $\sim_{(\forall)}$, as stated by the following proposition:

**Theorem 6.1.** *Given three IMDPs $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$, if $\mathcal{M}_1 \sim_{(\forall)} \mathcal{M}_2$ and $\mathcal{M}_2 \sim_{(\forall)} \mathcal{M}_3$, then $\mathcal{M}_1 \sim_{(\forall)} \mathcal{M}_3$.*

*Proof.* Let $\mathcal{R}_{12}$ and $\mathcal{R}_{23}$ be the equivalence relations underlying $\mathcal{M}_1 \sim_{(\forall)} \mathcal{M}_2$ and $\mathcal{M}_2 \sim_{(\forall)} \mathcal{M}_3$, respectively. Let $\mathcal{R}_{13}$ be the symmetric and transitive closure of the set $\{(s_1,s_3) \mid \exists s_2 . s_1 \; \mathcal{R}_{12} \; s_2 \wedge s_2 \; \mathcal{R}_{23} \; s_3\} \cup \{(s_1,s_1') \in S_1 \times S_1 \mid (s_1,s_1') \in \mathcal{R}_{12}\} \cup \{(s_3,s_3') \in S_3 \times S_3 \mid (s_3,s_3') \in \mathcal{R}_{23}\}$. We claim that $\mathcal{R}_{13}$ is a probabilistic bisimulation justifying $\mathcal{M}_1 \sim_{(\forall)} \mathcal{M}_3$.

The fact that $\bar{s}_1 \; \mathcal{R}_{13} \; \bar{s}_3$ is trivial since by hypothesis we have that $\bar{s}_1 \; \mathcal{R}_{12} \; \bar{s}_2$ and $\bar{s}_2 \; \mathcal{R}_{23} \; \bar{s}_3$, so $(\bar{s}_1, \bar{s}_3) \in \mathcal{R}_{13}$ by construction.

In the following, assume that $s_1 \in S_1$ and $s_3 \in S_3$; the other cases are similar.

The labelling is respected: for each $s_1 \; \mathcal{R}_{13} \; s_3$, we have that there exists $s_2$ such that $s_1 \; \mathcal{R}_{12} \; s_2$ and $s_2 \; \mathcal{R}_{23} \; s_3$; this implies that $L_1(s_1) = L_2(s_2)$ and $L_2(s_2) = L_3(s_3)$, thus $L_1(s_1) = L_3(s_3)$ as required.

To complete the proof, consider $s_1 \; \mathcal{R}_{13} \; s_3$ and $s_1 \longrightarrow \mu_1$. By hypothesis, there exists $s_2$ such that $s_1 \; \mathcal{R}_{12} \; s_2$ and $s_2 \; \mathcal{R}_{23} \; s_3$; moreover, by $\mathcal{R}_{12}$ being a probabilistic bisimulation, we know that there exists $s_2 \longrightarrow \mu_2$ such that $\mu_1 \; \mathcal{L}(\mathcal{R}_{12}) \; \mu_2$. Since $\mathcal{R}_{23}$ is a probabilistic bisimulation, we have that there exists $s_3 \longrightarrow \mu_3$ such that $\mu_2 \; \mathcal{L}(\mathcal{R}_{23}) \; \mu_3$. By construction of $\mathcal{R}_{13}$ and the properties of lifting, it follows that $\mu_1 \; \mathcal{L}(\mathcal{R}_{13}) \; \mu_3$, as required. ∎

We show that the bisimulation $\sim_{(\forall)}$ also preserves the (cooperative) universally quantified PCTL satisfaction $\models_{(\forall)}$.

**Theorem 6.2** (**Soundness of $\sim_{(\forall)}$ with respect to the PCTL properties**). *For states $s \sim_{(\forall)} t$ and any PCTL formula $\varphi$, we have $s \models_{(\forall)} \varphi$ if and only if $t \models_{(\forall)} \varphi$.*

*Proof.* We use structural induction on the syntax of PCTL state formula $\varphi$ and PCTL path formula $\psi$. This means that we need to prove the following two results simultaneously:

1. $s \sim_{(\forall)} t$ implies that $s \models_{(\forall)} \varphi$ if and only if $t \models_{(\forall)} \varphi$ for any state formula $\varphi$
2. $\xi_1 \sim_{(\forall)} \xi_2$ implies that $\xi_1 \models_{(\forall)} \psi$ if and only if $\xi_2 \models_{(\forall)} \psi$ for any path formula $\psi$.

We consider the nontrivial part that is when $\varphi = \mathbf{P}_{\bowtie p}(\psi)$ where $\bowtie = \leq$ and $\psi = \varphi_1 \mathbf{U} \varphi_2$; because the other cases are similar. Assume $t \models_{(\forall)} \varphi$, we need to show that $s \models_{(\forall)} \varphi$. To drive a contradiction, assume $s \models_{(\forall)} \neg\varphi$. Therefore, there exist $\sigma \in \Sigma$ and $\pi \in \Pi$ such that $\mathbf{Pr}_s^{\sigma,\pi}\{\xi | \xi \models \psi\} > p$. By induction hypothesis $Sat(\varphi_1)$ and $Sat(\varphi_2)$ are $\sim_{(\forall)}$ closed and they are indeed union of equivalence classes induced by $\sim_{(\forall)}$. It is not difficult to see that the set $\{\xi | \xi \models \psi\}$ is $\sim_{(\forall)}$ closed and therefore, $\mathbf{Pr}_t^{\sigma,\pi}\{\xi | \xi \models \psi\} = \mathbf{Pr}_s^{\sigma,\pi}\{\xi | \xi \models \psi\} > p$. In other words, there exist $\sigma \in \Sigma$ and $\pi \in \Pi$ such that $t \models_{(\forall)} \neg\varphi$ which leads to a contradiction. ■

It is worthwhile to mention that the nondeterminism could also dually be resolved *existentially*. This corresponds to the setting where we want to synthesise both the scheduler $\sigma$ that controls the system and choice of feasible probability distributions $\pi$ such that $\sigma$ and $\pi$ together guarantee a specified behaviour $\varphi$. This setting is formalised by the satisfaction relation $\models_{(\exists)}$ which is defined like $\models_{(\forall)}$ except for the operator $\mathbf{P}_{\bowtie p}(\psi)$ where we set

$$s \models_{(\exists)} \mathbf{P}_{\bowtie p}(\psi) \quad \text{if} \quad \exists\sigma \in \Sigma\ \exists\pi \in \Pi : \quad \mathbf{Pr}_s^{\sigma,\pi}\left[\models_{(\exists)} \psi\right] \bowtie p.$$

Note that for any formula of the form $\mathbf{P}_{<p}(\psi)$, we have $s \models_{(\exists)} \mathbf{P}_{<p}(\psi)$ if and only if we have $s \models_{(\forall)} \neg\mathbf{P}_{\geq p}(\psi)$. This can be easily generalised: for each state formula $\varphi$ we obtain a state formula $\overline{\varphi}$ such that $s \models_{(\exists)} \varphi$ if and only if $s \models_{(\forall)} \overline{\varphi}$ for each state $s$. Hence $\sim_{(\forall)}$ also preserves $\models_{(\exists)}$.

**Corollary 6.1.** *For states $s \sim_{(\forall)} t$ and any PCTL formula $\varphi$, we have $s \models_{(\exists)} \varphi$ if and only if $t \models_{(\exists)} \varphi$.*

### 6.2.1 Complexity Analysis of Deciding $\sim_{(\forall)}$ for *IMDP*s

In the sequel, we first present a fixed-parameter tractable algorithm for computing probabilistic bisimulation $\sim_{(\forall)}$ which in turn provides enough understanding of the problem to derive the worst case complexity of the decision problem. We start by illustrating the idea on an example.

**Example 6.2.** *Consider a pair of IMDPs depicted in Figure 6.2. The general sketch of the algorithm is as follows. We need to construct the polytopes of probability distributions offered by the actions; in our examples the polytopes are just line segments in two-dimensional space. We get $s \sim_{(\forall)} \bar{s}$ since the* convex hull *of $P_1$ and $P_2$ equals to the* convex hull *of $P_3$ and $P_4$.* ♦

Let us state the results formally. To this end, we fix an *IMDP* $\mathcal{M} = (S, \bar{s}, \mathscr{A}, \mathtt{AP}, L, I)$. Computation of probabilistic bisimulation for *IMDP*s follows the standard partition refinement approach [CSKN05,KS90,PT87], formalized by the algorithm Bisimulation in Figure 6.3. In particular, we start with $\mathcal{R}$ being the complete relation and iteratively remove from $\mathcal{R}$ pairs of states that violate the definition of bisimulation with respect to $\mathcal{R}$. However, the core part of the algorithm is to find out whether two states "violate the definition of bisimulation". Verification of this violation, as described in procedure Violate$_{(\forall)}(s, t, \mathcal{R})$, amounts to checking inclusion of polytopes defined as follows.

Recall that for $s \in S$ and an action $a \in \mathscr{A}$, $\mathscr{H}_s^a$ denotes the polytope of feasible successor distributions over *states* with respect to taking the action $a$ in the state $s$. By $\mathscr{P}_{\mathcal{R}}^{s,a}$, we denote the polytope of feasible successor distributions over *equivalence classes* of $\mathcal{R}$ with respect to

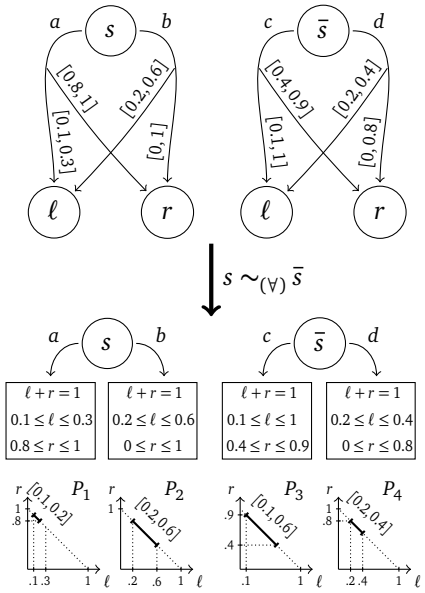



Figure 6.2: Description of the algorithm to decide if $s \sim_{(\forall)} \bar{s}$.

**Algorithm 4:** Bisimulation($\mathcal{M}$)

**Input:** A relation $\mathcal{R}$ on $S \times S$
**Output:** Return a probabilistic bisimulation $\mathcal{R}$

**begin**
  $\mathcal{R} \leftarrow \{(s,t) \in S \times S \mid L(s) = L(t)\}$;
  **repeat**
    $\mathcal{R}' \leftarrow \mathcal{R}$;
    **forall** $s \in S$ **do**
      $D \leftarrow \emptyset$;
      **forall** $t \in [s]_{\mathcal{R}}$ **do**
        **if** Violate$_{(\forall)}(s,t,\mathcal{R})$ **then**
          $D \leftarrow D \cup \{t\}$;
      split $[s]_{\mathcal{R}}$ in $\mathcal{R}$ into $D$ and $[s]_{\mathcal{R}} \setminus D$;
  **until** $\mathcal{R} = \mathcal{R}'$;
  **return** $\mathcal{R}$;

**Procedure 5:** Violate$_{(\forall)}(s,t,\mathcal{R})$

**Input:** States $s, t$ and relation $\mathcal{R}$
**Output:** Check if $s \sim_{(\forall)} t$

**begin**
  **return** $\mathscr{P}^{s}_{\mathcal{R}} \neq \mathscr{P}^{t}_{\mathcal{R}}$;

Figure 6.3: Decision algorithm to decide probabilistic bisimulation $\sim_{(\forall)}$ for *IMDP*s

taking the action $a$ in the state $s$. Formally, for $\mu \in \mathrm{Disc}(S/\mathcal{R})$ we set $\mu \in \mathscr{P}^{s,a}_{\mathcal{R}}$ if, for each

$\mathcal{C} \in S/\mathcal{R}$, we have $\mu(\mathcal{C}) \in I(s,a,\mathcal{C})$ where

$$I(s,a,\mathcal{C}) = \left[ \min\left\{1, \sum_{s' \in \mathcal{C}} \inf I(s,a,s')\right\}, \min\left\{1, \sum_{s' \in \mathcal{C}} \sup I(s,a,s')\right\}\right].$$

Note that we require that the probability of each class $\mathcal{C}$ must be in the interval of the sum of probabilities that can be assigned to states of $\mathcal{C}$. Furthermore, we define $\mathscr{P}^s_{\mathcal{R}} = \mathrm{CH}(\bigcup_{a \in \mathscr{A}(s)} \mathscr{P}^{s,a}_{\mathcal{R}})$. It is the set of feasible successor distributions over $S/\mathcal{R}$ with respect to taking an *arbitrary* distribution over actions in state $s$.

As specified in the procedure Violate$_{(\forall)}$, we show that it suffices to check equality of these polytopes.

**Proposition 6.1.** *We have $s \sim_{(\forall)} t$ if and only if $L(s) = L(t)$ and $\mathscr{P}^s_{\sim_{(\forall)}} = \mathscr{P}^t_{\sim_{(\forall)}}$.*

*Proof.* Let us first introduce one notation. For each distribution $\mu \in \mathrm{Disc}(S)$, let $\overline{\mu} \in \mathrm{Disc}(S/\sim_{(\forall)})$ denote the corresponding distribution such that $\overline{\mu}(\mathcal{C}) = \sum_{s \in \mathcal{C}} \mu(s)$. As regards the "if" part, for each choice $s \longrightarrow \mu$, we have $\overline{\mu} \in \mathscr{P}^s_{\sim_{(\forall)}}$. Similarly, for each $\rho \in \mathscr{P}^t_{\sim_{(\forall)}}$, there is a choice $t \longrightarrow \nu$ such that $\overline{\nu} = \rho$. Hence, $s \sim_{(\forall)} t$. As regards the "only if" part, let us assume that there is a distribution $\rho$ over equivalence classes such that, say $\rho \in \mathscr{P}^s_{\sim_{(\forall)}} \setminus \mathscr{P}^t_{\sim_{(\forall)}}$. There must be a choice $s \longrightarrow \mu$ such that $\overline{\mu} = \rho$ and there is no choice $t \longrightarrow \nu$ such that $\overline{\nu} = \rho$. Hence, $s \not\sim_{(\forall)} t$. ∎

Given an *IMDP* $\mathcal{M}$, let $|S| = n$, $|A| = m$ and $f$ be the maximal support of an action $\max_{s \in S, a \in \mathscr{A}(s)} |\{s' \mid I(s,a,s') \neq [0,0]\}|$. It is easy to see that the procedure Violate$_{(\forall)}$ is called at most $n^3$-times. Each polytope $\mathscr{P}^{s,a}_{\mathcal{R}}$ has at most $C = f \cdot 2^{f-1}$ corners, computing the convex hull $\mathscr{P}^s_{\mathcal{R}}$ takes $\mathcal{O}((bC)^2)$ time [CK70]. Checking inclusion of two polytopes then can be done in time polynomial [Sub09] in the number of corners of these two polytopes. In total, computing of $\sim_{(\forall)}$ can be done in time $|\mathcal{M}|^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(f)}$.

**Theorem 6.3.** *Given an IMDP $\mathcal{M}$, let $f$ be the maximal fanout, i.e., $f = \max_{s \in S, a \in \mathscr{A}(s)} |\{s' \in S \mid I(s,a,s') \neq [0,0]\}|$. Computing $\sim_{(\forall)}$ can be done in time $|\mathcal{M}|^{\mathcal{O}(1)} \cdot 2^{\mathcal{O}(f)}$.*

*Proof.* Immediate by the previous analysis. ∎

Given an *IMDP* $\mathcal{M}$, the computational complexity of computing $\sim_{(\forall)}$ strictly depends on checking bisimilarity of a pair of states as a core part. In the following, we will show that this verification routine is **coNP**-complete and therefore, the computation of $\sim_{(\forall)}$ as a whole is **coNP**-complete.

The definition of bisimulation can be reformulated equivalently as follows:

**Definition 6.2 (Reformulation of the bisimulation definition).** *Let $\mathcal{R} \subseteq S \times S$ be an equivalence relation. We say that $\mathcal{R}$ is a* probabilistic bisimulation *if $(s,t) \in \mathcal{R}$ implies that $L(s) = L(t)$ and $\mathscr{P}^s_{\mathcal{R}} = \mathscr{P}^t_{\mathcal{R}}$.*

As it is clear from Definition 6.2, the complexity of verifying bisimilarity of a pair of states $s$ and $t$ strictly depends on the complexity of the *Convex Hull Equivalence (CHE) problem* stated as follows:

**Definition 6.3** (**Convex hull equivalence problem**). *Given an IMDP $\mathcal{M}$, a pair of states $s$ and $t$, $n, n_s, n_t \in \mathbb{N}$, two sets $\{P^{s,i} \mid i \in \{1,\dots,n_s\}\}$ and $\{P^{t,i} \mid i \in \{1,\dots,n_t\}\}$ where for each $r \in \{s,t\}$ and $i \in \{1,\dots,n_r\}$, for given $\mathbf{l}^{r,i}, \mathbf{u}^{r,i} \in \mathbb{R}^n$, $P^{r,i}$ is the convex polyhedron*

$$P^{r,i} = \left\{ \mathbf{x}^{r,i} \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \mathbf{l}^{r,i} \leq \mathbf{x}^{r,i} \leq \mathbf{u}^{r,i} \\ \mathbf{1}^T \mathbf{x}^{r,i} = 1 \end{array} \right\},$$

*the CHE problem asks to determine whether* $\mathrm{CH}(\bigcup_{i=1}^{n_s} P^{s,i}) = \mathrm{CH}(\bigcup_{i=1}^{n_t} P^{t,i})$.

We show that CHE problem is in **coNP** by reducing it to the **coNP** quantified linear implication problem [ERSW14];

**Lemma 6.1.** *The CHE problem is in **coNP**.*

*Proof.* Consider $r \in \{s,t\}$ and the corresponding $n_r \in \mathbb{N}$ according to Definition 6.3. Let $\mathbf{y} \in \mathrm{CH}(\bigcup_{i=1}^{n_r} P^{r,i})$. Thus, $\mathbf{y}$ is a convex combination of the extreme points of $\mathrm{CH}(\bigcup_{i=1}^{n_r} P^{r,i})$. Let $W = \{\mathbf{w}^k \mid k \in \{1,\dots,m_r\}\}$ be the set of these extreme points. Thus, $\mathbf{y} = \sum_{k=1}^{m_r} \alpha_k \cdot \mathbf{w}^k$ where $\sum_{k=1}^{m_r} \alpha_k = 1$ and $\alpha_k \geq 0$ for each $k \in \{1,\dots,m_r\}$.

Since for each $k \in \{1,\dots,m_r\}$, $\mathbf{w}^k$ is an extreme point of $\mathrm{CH}(\bigcup_{i=1}^{n_r} P^{r,i})$, we know that there exists an unique $i$ such that $\mathbf{w}^k \in P^{r,i}$. Let $\{\mathbf{w}^{k_{i,1}}, \dots, \mathbf{w}^{k_{i,l}}\} \subseteq W$ be the set of extreme points that belong to $P^{r,i}$ and let $\alpha_{r,i} = \alpha_{k_{i,1}} + \ldots + \alpha_{k_{i,l}}$. We have that the vector $\mathbf{w}^{r,i}$ defined as

$$\mathbf{w}^{r,i} = \begin{cases} \frac{\alpha_{k_{i,1}}}{\alpha_{r,i}} \cdot \mathbf{w}^{k_{i,1}} + \ldots + \frac{\alpha_{k_{i,l}}}{\alpha_{r,i}} \cdot \mathbf{w}^{k_{i,l}} & \text{if } \alpha_{r,i} \neq 0 \\ \mathbf{w}^{k_{i,1}} & \text{otherwise} \end{cases}$$

belongs to $P^{r,i}$ and that

$$\left(\alpha_{k_{i,1}} \cdot \mathbf{w}^{k_{i,1}} + \ldots + \alpha_{k_{i,l}} \cdot \mathbf{w}^{k_{i,l}}\right) = \alpha_{r,i} \cdot \mathbf{w}^{r,i}.$$

Thus, we have that $\mathbf{y} = \sum_{i=1}^{n_r} \alpha_{r,i} \cdot \mathbf{w}^{r,i}$ and that $\sum_{i=1}^{n_r} \alpha_{r,i} = 1$.

This means that $\mathbf{y}$ can be written as a convex combination of *one* point from each $P^{r,i}$, $i \in \{1,\dots,n_r\}$. Let

$$Q^{r,i} = \left\{ \mathbf{z}^{r,i} \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \alpha_{r,i} \cdot \mathbf{l}^{r,i} \leq \mathbf{z}^{r,i} \leq \alpha_{r,i} \cdot \mathbf{u}^{r,i} \\ \mathbf{1}^T \mathbf{z}^{r,i} = \alpha_{r,i} \end{array} \right\}.$$

Let $\mathbf{z}^{r,i} = \alpha_{r,i} \cdot \mathbf{w}^{r,i} \in Q^{r,i}$, thus $\mathbf{y} = \sum_{i=1}^{n_r} \mathbf{z}^{r,i}$.

This means that, if $\mathbf{y} \in \mathrm{CH}(\bigcup_{i=1}^{n_r} P^{r,i})$, then there exist $\alpha_{r,i}$ and $\mathbf{z}^{r,i} \in Q^{r,i}$ for each $i \in \{1,\dots,n_r\}$ such that $\sum_{i=1}^{n_r} \alpha_{r,i} = 1$ and $\mathbf{y} = \sum_{i=1}^{n_r} \mathbf{z}^{r,i}$. Conversely, if there exist such $\alpha$s and $\mathbf{z}$s, then trivially $\mathbf{y} \in \mathrm{CH}(\bigcup_{i=1}^{n_r} P^{r,i})$.

Thus, we can represent $\mathbf{y} \in \mathrm{CH}(\bigcup_{i=1}^{n_r} P^{r,i})$ using the following linear programming problem, $LP^r$:

$$\exists \mathbf{y},\, \mathbf{z}^{r,1},\, \ldots,\, \mathbf{z}^{r,n_r},\, \alpha_{r,1},\, \ldots,\, \alpha_{r,n_r}$$

$$\alpha_{r,i} \cdot \mathbf{l}^{r,i} \leq \mathbf{z}^{r,i} \leq \alpha_{r,i} \cdot \mathbf{u}^{r,i} \quad \forall i \in \{1, \ldots, n_r\} \tag{6.1}$$

$$\mathbf{1}^T \mathbf{z}^{r,i} = \alpha_{r,i} \quad \forall i \in \{1, \ldots, n_r\} \tag{6.2}$$

$$\sum_{i=1}^{n_r} \mathbf{z}^{r,i} = \mathbf{y} \tag{6.3}$$

$$\sum_{i=1}^{n_r} \alpha_{r,i} = 1 \tag{6.4}$$

$$\alpha_{r,i} \geq 0 \quad \forall i \in \{1, \ldots, n_r\} \tag{6.5}$$

In $LP^r$, lines (6.1) and (6.2) ensure that $\mathbf{z}^{r,i} \in Q^{r,i}$. Similarly, lines (6.3) and (6.4) ensure that $\mathbf{y}$ is a convex combination of the $\mathbf{w}^{r,i}$s. Thus, $\mathbf{y}$ is part of a solution to $LP^r$ if and only if $\mathbf{y} \in \mathrm{CH}(\bigcup_{i=1}^{n_r} P^{r,i})$.

This means that

$$\mathrm{CH}\left(\bigcup_{i=1}^{n_r} P^{r,i}\right) = \left\{ \mathbf{y} \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} \exists \mathbf{z}^{r,1}, \ldots, \mathbf{z}^{r,n_r}, \alpha_{r,1}, \ldots, \alpha_{r,n_r} & \\ \quad \alpha_{r,i} \cdot \mathbf{l}^{r,i} \leq \mathbf{z}^{r,i} \leq \alpha_{r,i} \cdot \mathbf{u}^{r,i} & \forall i \in \{1, \ldots, n_r\} \\ \quad \mathbf{1}^T \mathbf{z}^{r,i} = \alpha_{r,i} & \forall i \in \{1, \ldots, n_r\} \\ \quad \sum_{i=1}^{n_r} \mathbf{z}^{r,i} = \mathbf{y} & \\ \quad \sum_{i=1}^{n_r} \alpha_{r,1} = 1 & \\ \quad \alpha_{r,i} \geq 0 & \forall i \in \{1, \ldots, n_r\} \end{array} \right\}.$$

Thus, $\mathrm{CH}(\bigcup_{i=1}^{n_s} P^{s,i}) \subseteq \mathrm{CH}(\bigcup_{i=1}^{n_t} P^{t,i})$ if and only if the quantified linear implication problem

$$\forall \mathbf{y}, \mathbf{z}^{s,1}, \ldots, \mathbf{z}^{s,n_s}, \alpha_{s,1}, \ldots, \alpha_{s,n_s} \; \exists \mathbf{z}^{t,1}, \ldots, \mathbf{z}^{t,n_t}, \alpha_{t,1}, \ldots, \alpha_{t,n_t} \;\; LP^s \rightarrow LP^t$$

holds.

Similarly, $\mathrm{CH}(\bigcup_{i=1}^{n_t} P^{t,i}) \subseteq \mathrm{CH}(\bigcup_{i=1}^{n_s} P^{s,i})$ if and only if the quantified linear implication problem

$$\forall \mathbf{y}, \mathbf{z}^{t,1}, \ldots, \mathbf{z}^{t,n_t}, \alpha_{t,1}, \ldots, \alpha_{t,n_t} \; \exists \mathbf{z}^{s,1}, \ldots, \mathbf{z}^{s,n_s}, \alpha_{s,1}, \ldots, \alpha_{s,n_s} \;\; LP^t \rightarrow LP^s$$

holds. These problems are known to be in **coNP** [ERSW14]. ■

Additionally, we show that CHE problem is **coNP**-hard by reducing the **coNP**-hard tautology problem for 3DNF [GJ90] to CHE.

**Lemma 6.2.** *The CHE problem is **coNP**-hard.*

*Proof.* We show our claim by reducing in polynomial time the tautology problem for 3DNF to the CHE problem. Since it is known [GJ90] that the tautology problem for 3DNF is **coNP**-hard, then it follows that also the CHE problem is **coNP**-hard. In particular, for a given instance $\varphi$ of 3DNF with $n$ variables and $m$ disjuncts, we construct in polynomial time two sets $\{P^s\}$ and $\{P^{t,i} \mid i \in \{1, \ldots, m\}\}$ such that their convex hulls are equal if and only if $\varphi$ is a tautology.

Let

$$P^s = \left\{ \mathbf{x}^s \in \mathbb{R}^{n+1} \;\middle|\; \begin{array}{ll} 0 \leq x_j^s \leq \frac{1}{n} & \forall j \in \{1, \ldots, n\} \\ 0 \leq x_{n+1}^s \leq 1 & \\ \mathbf{1}^T \mathbf{x}^s = 1 & \end{array} \right\}$$

Note that $\sum_{j=1}^{n} x_j^s \leq 1$ holds regardless of the actual value chosen for each element $x_j^s$ of $\mathbf{x}^s$ with $j \in \{1,\ldots,n\}$ (as long as it respects the first constraint), thus we can choose each element $x_j^s$ independently from the others and then set $x_{n+1}^s$ to be $1-\sum_{j=1}^{n} x_j^s$ to take up the slack so to satisfy $\mathbf{1}^T\mathbf{x}^s = 1$.

For each literal $l$ appearing in $\varphi$, let $p_l = 0$ if $l = \neg v$, and $p_l = \frac{1}{n}$ if $l = v$, for some variable $v$. Define for each disjunct $\varphi_i = (l_{i_1} \wedge l_{i_2} \wedge l_{i_3})$ of $\varphi$ the set

$$P^{t,i} = \left\{ \mathbf{x}^{t,i} \in \mathbb{R}^{n+1} \;\middle|\; \begin{array}{ll} 0 \leq x_j^{t,1} \leq \frac{1}{n} & \forall j \in \{1,\ldots,n\} \setminus \{i_1,i_2,i_3\} \\ p_{l_j} \leq x_j^{t,i} \leq p_{l_j} & \forall j \in \{i_1,i_2,i_3\} \\ 0 \leq x_{n+1}^{t,1} \leq 1 & \\ \mathbf{1}^T\mathbf{x}^{t,i} = 1 & \end{array} \right\}.$$

For example, if the $i^{\text{th}}$ disjunct of $\varphi$ is $\varphi_i = (v_{i_1} \wedge \neg v_{i_2} \wedge v_{i_3})$, then the corresponding $P^{t,i}$ is as follows:

$$P^{t,i} = \left\{ \mathbf{x}^{t,i} \in \mathbb{R}^{n+1} \;\middle|\; \begin{array}{ll} 0 \leq x_j^{t,1} \leq \frac{1}{n} & \forall j \in \{1,\ldots,n\} \setminus \{i_1,i_2,i_3\} \\ \frac{1}{n} \leq x_{i_1}^{t,i} \leq \frac{1}{n} & \\ 0 \leq x_{i_2}^{t,i} \leq 0 & \\ \frac{1}{n} \leq x_{i_3}^{t,i} \leq \frac{1}{n} & \\ 0 \leq x_{n+1}^{t,1} \leq 1 & \\ \mathbf{1}^T\mathbf{x}^{t,i} = 1 & \end{array} \right\}.$$

By construction, for each $i \in \{1,\ldots,n\}$, we have that $P^{t,i} \subseteq P^s$, thus it follows that $\mathrm{CH}(\bigcup_{i=1}^{m} P^{t,i}) \subseteq \mathrm{CH}(P^s) = P^s$.

To show equality we need to show that $P^s \subseteq \mathrm{CH}(\bigcup_{i=1}^{m} P^{t,i})$. This can be done by showing that each extreme point of $P^s$ is in $\mathrm{CH}(\bigcup_{i=1}^{m} P^{t,i})$.

Let $\mathbf{w}$ be an extreme point of $P^s$.

**Remark 6.1.** *We want to remark that an extreme point* $\mathbf{w}$ *of the polytope* $P^s$ *is such that* $\mathbf{w} = (w_1,\ldots,w_n,w_{n+1}) \in \{0,\frac{1}{n}\}^n \times \mathbb{R}$ *where the element* $w_{n+1}$ *is uniquely determined by the elements* $w_1,\ldots,w_n$. *In fact, let* $j \in \{1,\ldots,n\}$ *be such that* $0 < w_j < \frac{1}{n}$ *and suppose that* $\mathbf{w}$ *is an extreme point of* $P^s$. *Consider the following two points of* $P^s$ *differing from* $\mathbf{w}$ *only for the components* $j$ *and* $n+1$*:*

$$\mathbf{w}' = (w_1,\ldots,w_{j-1},0,w_{j+1},\ldots,w_n,w_{n+1}+w_j)$$
$$\mathbf{w}'' = (w_1,\ldots,w_{j-1},\frac{1}{n},w_{j+1},\ldots,w_n,w_{n+1}-\frac{1}{n}+w_j).$$

*We have that* $\mathbf{w} = (1-n\cdot w_j)\cdot\mathbf{w}' + n\cdot w_j\cdot\mathbf{w}''$. *This contradicts the fact that* $\mathbf{w}$ *is an extreme point of* $P^s$. *Thus, for each* $j \in \{1,\ldots,n\}$, $w_j \in \{0,\frac{1}{n}\}$, *hence* $\mathbf{w} = (w_1,\ldots,w_n,w_{n+1}) \in \{0,\frac{1}{n}\}^n \times \mathbb{R}$.

*Let* $\mathbf{w}$ *be a point in* $P^s$ *such that* $(w_1,\ldots,w_n,w_{n+1}) \in \{0,\frac{1}{n}\}^n \times \mathbb{R}$. *We have that* $\mathbf{w}$ *is at the intersection of* $(n+1)$ *hyperplanes defining* $P^s$, *hence* $\mathbf{w}$ *is an extreme point of* $P^s$. *This implies that* $\mathbf{w}$ *is an extreme point of* $P^s$ *if and only if* $\mathbf{w} = (w_1,\ldots,w_n,w_{n+1}) \in \{0,\frac{1}{n}\}^n \times \mathbb{R}$. ◊

If $\mathbf{w} \in \mathrm{CH}(\bigcup_{i=1}^{m} P^{t,i})$, then $\mathbf{w}$ is extreme point of $\mathrm{CH}(\bigcup_{i=1}^{m} P^{t,i})$. This holds because $\mathrm{CH}(\bigcup_{i=1}^{m} P^{t,i}) \subseteq P^s$. Thus, there has to be an $i$ such that $\mathbf{w} \in P^{t,i}$.

Let $\mathbf{v}$ be the boolean vector with $n$ components such that $v_j = \textbf{true}$ if $w_j = \frac{1}{n}$ and $v_j = \textbf{false}$ if $w_j = 0$, for $j \in \{1, \ldots, n\}$. Note that every possible $\mathbf{v}$ can be constructed in this way, by Remark 6.1. For instance, for $\varphi_i = (v_{i_1} \wedge v_{i_2} \wedge \neg v_{i_3})$ we have that

$$P^{t,i} = \left\{ \mathbf{x}^{t,i} \in \mathbb{R}^{n+1} \;\middle|\; \begin{array}{ll} 0 \leq x_j^{t,i} \leq \frac{1}{n} & j \in \{1, \ldots, n\} \setminus \{i_1, i_2, i_3\} \\ \frac{1}{n} \leq x_{i_1}^{t,i} \leq \frac{1}{n} & \\ \frac{1}{n} \leq x_{i_2}^{t,i} \leq \frac{1}{n} & \\ 0 \leq x_{i_3}^{t,i} \leq 0 & \\ 0 \leq x_{n+1}^{t,i} \leq 1 & \\ \mathbf{1}^T \mathbf{x}^{t,i} = 1 & \end{array} \right\}.$$

Thus, $\mathbf{w} \in P^{t,i}$ if and only if $w_{i_1} = \frac{1}{n}$, $w_{i_2} = \frac{1}{n}$, and $w_{i_3} = 0$. This corresponds to $v_{i_1} = \textbf{true}$, $v_{i_2} = \textbf{true}$, and $v_{i_3} = \textbf{false}$. Thus, $\mathbf{w} \in P^{t,i}$ if and only if $\mathbf{v}$ satisfies $\varphi_i$.

If every extreme point $\mathbf{w}$ of $P^s$ is in $\mathrm{CH}(\bigcup_{i=1}^m P^{t,i})$, then every possible $\mathbf{v}$ satisfies at least one disjunct of $\varphi$, namely the disjunct $\varphi_i$ so that $\mathbf{w} \in P^{t,i}$. This means that $P^s \subseteq \mathrm{CH}(\bigcup_{i=1}^m P^{t,i})$ implies that $\varphi$ is a tautology.

Let $\varphi$ be a tautology, and let $\mathbf{v}$ be a boolean vector. Thus, there exists a disjunct $\varphi_i$ of $\varphi$ such that $\mathbf{v}$ satisfies $\varphi_i$. From $\mathbf{v}$ we can construct the point $\mathbf{w}$ whose elements are defined as follows:

$$w_j = \begin{cases} 0 & \text{if } j \in \{1, \ldots, n\} \text{ and } v_j = \textbf{false}, \\ \frac{1}{n} & \text{if } j \in \{1, \ldots, n\} \text{ and } v_j = \textbf{true}, \\ 1 - \sum_{k=1}^n w_k & \text{if } j = n+1. \end{cases}$$

Note that $\mathbf{w}$ is an extreme point of $P^s$ and that every extreme point of $P^s$ can be constructed in this way (cf. Remark 6.1).

Since $\mathbf{v}$ satisfies $\varphi_i$, we have that $\mathbf{w} \in P^{t,i}$. Thus, $\mathbf{w} \in \mathrm{CH}(\bigcup_{i=1}^m P^{t,i})$. By the arguments made above, every extreme point of $P^s$ is in $\mathrm{CH}(\bigcup_{i=1}^m P^{t,i})$. This means that $P^s \subseteq \mathrm{CH}(\bigcup_{i=1}^m P^{t,i})$.

Thus, $\mathrm{CH}(P^s) = \mathrm{CH}(\bigcup_{i=1}^m P^{t,i})$ if and only if $\varphi$ is a tautology. Since the reduction from $\varphi$ to the problems $P^s$ and $P^{t,i}$, for $i \in \{1, \ldots, m\}$ is polynomial in the size $m$ and $n$ of $\varphi$, we have that deciding $\mathrm{CH}(P^s) = \mathrm{CH}(\bigcup_{i=1}^m P^{t,i})$ is **coNP**-hard as well. Note that this is a special case of the CHE problem where $n_s = 1$, $P^{s,1} = P^s$, and $n_t = m$, thus also the CHE problem is **coNP**-hard. ■

**Theorem 6.4.** *The CHE problem is **coNP**-complete.*

*Proof.* The proof follows directly from Lemma 6.1 and Lemma 6.2. ■

With this result at hand, together with Definition 6.2, we know that checking the bisimilarity of two states of an *IMDP* $\mathcal{M}$ is **coNP**-complete. Since the standard partition refinement algorithm performs this check a polynomial number of times (see, e.g., [CS02, TH15, CSKN05, KS90, PT87]), it follows that also computing $\sim_{(\forall)}$ is **coNP**-complete.

**Theorem 6.5.** *Given an IMDP $\mathcal{M}$, computing $\sim_{(\forall)}$ is **coNP**-complete.*

*Proof.* Immediate by the previous analysis. ■

### 6.2.2 Computational Tractability: An Approximation Algorithm

As discussed in the previous section, given an *IMDP*, the complexity of computing $\sim_{(\forall)}$ strictly depends on finding $t \longrightarrow \mu_t$: we show how a finer (sub-optimal) equivalence relation can be computed in polynomial time. The bisimulation in Definition 6.1 can be reformulated equivalently as follows:

**Definition 6.4** (**Equivalent form of *IMDP* probabilistic bisimulation**). *Let $\mathcal{R} \subseteq S \times S$ be an equivalence relation. We say that $\mathcal{R}$ is a* probabilistic bisimulation *if $(s,t) \in \mathcal{R}$ implies that $L(s) = L(t)$ and for each $a \in \mathscr{A}(s)$ and each $\mu_s \in \mathscr{P}_{\mathcal{R}}^{s,a}$, there exists $\mu_t \in \mathscr{P}_{\mathcal{R}}^{t}$ such that $\mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t$.*

Recall that a probabilistic bisimulation can be seen as a game between two players: in each round, the challenger, or attacker, $s$ proposes a transition, or step, that has to be matched by the defender $t$. The two states $s$ and $t$ are bisimilar if the defender is always able to match the challenging transitions proposed by the attacker, that is, the game can be played forever. Correspondingly, in our setting, probabilistic bisimulations require that each transition proposed by the challenger $s$ which is selected from the set $\mathscr{P}_{\mathcal{R}}^{s,a}$, is matched by the defender $t$ via a single (combined) transition. The above definition essentially disallows the state $s$ to randomize over the set of its available actions. Therefore, instead of allowing the challenger to pick a probability distribution from $\mathrm{CH}(\bigcup_{a \in \mathscr{A}(s)} \mathscr{P}_{\mathcal{R}}^{s,a})$, we restrict his choice to select a distribution for an action from the polytope $\mathscr{P}_{\mathcal{R}}^{s,a}$. This restriction does not lead to any loss of generality, since it is routine to check that the bisimulation $\mathcal{R}$ from Definition 6.4 satisfies the condition of Definition 6.1.

#### 6.2.2.1 Robust Methodologies for Probabilistic Bisimulation

We now discuss the key elements of a decision algorithm for probabilistic bisimulation on *IMDP*s. As we will see later in this section, the core part and also the main source of the exponential complexity of the exact decision algorithm in Figure 6.3 is the need to repeatedly verify the step condition, that is, given a challenging transition $\mu \in \mathscr{P}_{\mathcal{R}}^{s}$ and $(s,t) \in \mathcal{R}$, to check if there exists $t \longrightarrow \mu_t$ such that $\mu \; \mathcal{L}(\mathcal{R}) \; \mu_t$. We show that, using some inspiration from network flow problems, it is possible to treat a transition $t \longrightarrow \mu_t$ of the *IMDP* $\mathcal{M}$ as a flow where the initial probability mass $\delta_t$ flows and splits along transitions appropriately to the transition target distributions and the resolution of the nondeterminism fulfilled by the scheduler and nature. This intuition essentially enables us to model the probabilistic bisimulation problem as an *adjustable robust counterpart* of an uncertain LP problem that is intractable in general [BTGGN04, BTEGN09].

**Adjustable Robust Counterpart for Probabilistic Bisimulation.** From now on, we assume that the *IMDP* $\mathcal{M}$, the state $t$, the probability distribution $\mu$, and the equivalence relation $\mathcal{R}$ on $S$ are given. We intend to verify or refute the existence of a transition $t \longrightarrow \mu_t$ of $\mathcal{M}$ satisfying $\mu \; \mathcal{L}(\mathcal{R}) \; \mu_t$ via the construction of a flow through the network graph $G(t,\mathcal{R}) = (V,E)$ defined as follows: the set of vertices is $V = \{\triangle, \blacktriangledown, t\} \cup S_{\mathscr{A}} \cup S_{\mathcal{R}} \cup (S/\mathcal{R})$ where $S_{\mathscr{A}} = \{t_a \mid a \in \mathscr{A}(t)\}$ and $S_{\mathcal{R}} = \{s_{\mathcal{R}} \mid s \in S\}$, and the set of arcs is $E =$

$\{(\triangle, t)\} \cup \{(v_{\mathcal{R}}, \mathcal{C}), (\mathcal{C}, \blacktriangledown) \mid \mathcal{C} \in S/\mathcal{R}, v \in \mathcal{C}\} \cup \{(t, t_a), (t_a, v_{\mathcal{R}}) \mid a \in \mathscr{A}(t), v \in S\}$. In the flow network definition, $\triangle$ and $\blacktriangledown$ are the source node and the sink node of the network, respectively. The set of *transition nodes* $S_{\mathscr{A}}$ includes vertices that represent the interval transitions of the *IMDP* $\mathcal{M}$. More precisely, each transition labelled by $a$ enabled at state $t$ is represented by a transition node $t_a \in S_{\mathscr{A}}$. The set $S_{\mathcal{R}}$ is a copy of the state set $S$ that is used to represent the states reached after having performed the transition; for such states, we connect them to the equivalence class they belong to so to verify the condition of the lifting. The network construction can be seen as an adaptation to the strong case of flow networks used in Chapter 5.

We take advantage of the above transformation of the "*IMDP* into a network graph" to generate an optimization problem. To this aim, we adopt the same notation of the network optimization setting so we use $f_{u,v}$ to show the "flow" through the arc from $u$ to $v$. In formulating the optimization problem, we use in addition the so-called balancing constraints in order to reflect the probabilistic choices in the given *IMDP* $\mathcal{M}$ and to ensure the correct splitting of outgoing flows from the transition nodes in the set $S_{\mathscr{A}}$.

**Definition 6.5** (**Optimization problem for probabilistic bisimulation**). *The optimization problem associated to the network* $G(t, \mathcal{R}) = (V, E)$ *is defined as follows:*

$$
\begin{array}{rll}
\min_f & 0 & \\
\textit{subject to:} & f_{u,v} \geq 0 & \textit{for each } (u,v) \in E \\
& f_{\triangle,t} = 1 & \\
& f_{\mathcal{C},\blacktriangledown} = \mu(\mathcal{C}) & \textit{for each } \mathcal{C} \in S/\mathcal{R} \\
& \sum_{\{u \in V \mid (u,v) \in E\}} f_{u,v} - \sum_{\{w \in V \mid (v,w) \in E\}} f_{v,w} = 0 & \textit{for each } v \in V \setminus \{\triangle, \blacktriangledown\} \\
& f_{t_a, v_{\mathcal{R}}} - p_{a,v} \cdot f_{t,t_a} = 0 & \textit{for each } a \in \mathscr{A}(t) \textit{ and } v \in S \\
& p_{a,v} \in I(t,a,v) & \textit{for each } a \in \mathscr{A}(t) \textit{ and } v \in S
\end{array}
$$

It is not difficult to see that the optimization problem just defined is not an LP problem, as there are quadratic constraints where the flow variable $f_{t,t_a}$ is multiplied with the "probability" variable $p_{a,v}$. As a matter of fact, for a given $a \in \mathscr{A}(t)$, the variables $p_{a,v}$ have to lie in the interval defined by the interval transition $I(t,a,v)$ and they have to induce a probability distribution, i.e., $p_{a,v} \geq 0$ for each $v \in S$ and $\sum_{v \in S} p_{a,v} = 1$. The non-negativity of the variables comes for free from the constraints $p_{a,v} \in I(t,a,v)$ since $I(t,a,v) \subseteq [0,1]$; $\sum_{v \in S} p_{a,v} = 1$ follows by the flow conservation constraint $\sum_{\{u \in V \mid (u,v) \in E\}} f_{u,v} - \sum_{\{w \in V \mid (v,w) \in E\}} f_{v,w} = 0$ for $v = t_a$. Therefore, the optimization problem can be easily cast as an LP problem by replacing the pair of constraints $f_{t_a, v_{\mathcal{R}}} - p_{a,v} \cdot f_{t,t_a} = 0$ and $p_{a,v} \in I(t,a,v)$ with the pair of constraints $f_{t_a, v_{\mathcal{R}}} - \inf I(t,a,v) \cdot f_{t,t_a} \geq 0$ and $f_{t_a, v_{\mathcal{R}}} - \sup I(t,a,v) \cdot f_{t,t_a} \leq 0$, i.e., the state $v$ is reached from $t$ with probability $p_{a,v} = \frac{f_{t_a, v_{\mathcal{R}}}}{f_{t,t_a}}$ at least $\inf I(t,a,v)$ and at most $\sup I(t,a,v)$, as required. Taking this modification into account, we can reformulate the optimization problem in Definition 6.5 as the following LP problem.

**Definition 6.6** (**The** $LP(t, \mu, \mathcal{R})$ **LP problem**). *The* $LP(t, \mu, \mathcal{R})$ *LP problem associated to the*

*network graph $G(t,\mathcal{R}) = (V,E)$ is defined as follows:*

$$
\begin{array}{rll}
\min_f & 0 & \\
\textit{subject to:} & f_{u,v} \geq 0 & \textit{for each } (u,v) \in E \\
& f_{\triangle,t} = 1 & \\
& f_{\mathcal{C},\blacktriangledown} = \mu(\mathcal{C}) & \textit{for each } \mathcal{C} \in S/\mathcal{R} \\
& \sum_{\{u \in V \mid (u,v) \in E\}} f_{u,v} - \sum_{\{w \in V \mid (v,w) \in E\}} f_{v,w} = 0 & \textit{for each } v \in V \setminus \{\triangle, \blacktriangledown\} \\
& f_{t_a,\nu_{\mathcal{R}}} - \inf I(t,a,v) \cdot f_{t,t_a} \geq 0 & \textit{for each } a \in \mathscr{A}(t) \textit{ and } v \in S \\
& f_{t_a,\nu_{\mathcal{R}}} - \sup I(t,a,v) \cdot f_{t,t_a} \leq 0 & \textit{for each } a \in \mathscr{A}(t) \textit{ and } v \in S
\end{array}
$$

The feasibility of the resulting LP problem can be seen as an oracle to verify or refute the existence of a probabilistic transition $t \longrightarrow \mu_t$. Formally,

**Lemma 6.3.** *Given an IMDP $\mathcal{M}$, $t \in S$, $\mu \in \mathrm{Disc}(S)$, and an equivalence relation $\mathcal{R}$ on $S$, the LP$(t,\mu,\mathcal{R})$ LP problem has a feasible solution if and only if there exist $\sigma \in \Sigma$ and $\pi \in \Pi$ inducing $t \longrightarrow \mu_t$ such that $\mu\ \mathcal{L}(\mathcal{R})\ \mu_t$.*

*Proof.* For the first implication, suppose that there exist a scheduler $\sigma$ and a nature $\pi$ inducing $t \longrightarrow \mu_t$ such that $\mu\ \mathcal{L}(\mathcal{R})\ \mu_t$. Consider the LP problem $LP(t,\mu,\mathcal{R})$ and consider the following assignment for the variables: $f_{\triangle,t} = 1$, $f_{t,t_a} = \sigma(t)(a)$, $f_{t_a,\nu_{\mathcal{R}}} = \sigma(t)(a) \cdot \pi(t,a)(v)$ for each $a \in \mathscr{A}(t)$ and $v \in S$, and $f_{\nu_{\mathcal{R}},\mathcal{C}} = \sum_{a \in \mathscr{A}(t)} \sigma(t)(a) \cdot \pi(t,a)(v)$ and $f_{\mathcal{C},\blacktriangledown} = \mu_s(\mathcal{C})$ for each $\mathcal{C} \in S/\mathcal{R}$ and $v \in \mathcal{C}$.

It is immediate to see that such an assignment is a feasible solution of the $LP(t,\mu,\mathcal{R})$ problem: non-negativity of the variables is trivial since all values are sum of (products of) probabilities; the constraints $f_{\triangle,t} = 1$ and $f_{\mathcal{C},\blacktriangledown} = \mu_s(\mathcal{C})$ are obvious by the definition of the assignment; the flow conservation constraint for $t$ is immediate since

$$
\begin{aligned}
&\sum_{(u,t) \in E} f_{u,t} = f_{\triangle,t} = 1 = \sum_{a \in \mathscr{A}(t)} \sigma(t)(a) \\
&= \sum_{a \in \mathscr{A}(t)} f_{t,t_a} = \sum_{(t,u) \in E} f_{t,u};
\end{aligned}
$$

for each action $a \in \mathscr{A}(t)$, the flow conservation constraint for $t_a$ is clear since

$$
\begin{aligned}
&\sum_{(u,t_a) \in E} f_{u,t_a} = f_{t,t_a} = \sigma(t)(a) = \sigma(t)(a) \cdot \sum_{v \in S} \pi(t,a)(v) \\
&= \sum_{v \in S} \sigma(t)(a) \cdot \pi(t,a)(v) = \sum_{v \in S} f_{t_a,\nu_{\mathcal{R}}} = \sum_{(t_a,u) \in E} f_{t_a,u};
\end{aligned}
$$

for each state $v$, the flow conservation constraint for $\nu_{\mathcal{R}}$ is immediate since

$$
\begin{aligned}
&\sum_{(u,\nu_{\mathcal{R}}) \in E} f_{u,\nu_{\mathcal{R}}} = \sum_{a \in \mathscr{A}(t)} f_{t_a,\nu_{\mathcal{R}}} = \sum_{a \in \mathscr{A}(t)} \sigma(t)(a) \cdot \pi(t,a)(v) \\
&= f_{\nu_{\mathcal{R}},[v]_{\mathcal{R}}} = \sum_{(\nu_{\mathcal{R}},u) \in E} f_{\nu_{\mathcal{R}},u};
\end{aligned}
$$

and for each equivalence class $\mathcal{C}$, the flow conservation constraint for $\mathcal{C}$ is clear since

$$\sum_{(u,\mathcal{C})\in E} f_{u,\mathcal{C}} = \sum_{v\in\mathcal{C}} f_{v_{\mathcal{R}},\mathcal{C}} = \sum_{v\in\mathcal{C}} \sum_{a\in\mathscr{A}(t)} \sigma(t)(a)\cdot\pi(t,a)(v)$$
$$= \sum_{v\in\mathcal{C}} \mu_t(v) = \mu_s(\mathcal{C}) = f_{\mathcal{C},\blacktriangledown} = \sum_{(\mathcal{C},u)\in E} f_{\mathcal{C},u}.$$

The last constraints we have to verify are about the split of probability according to the transition: for $a \in \mathscr{A}(t)$ and $v \in S$,

$$f_{t_a,v_{\mathcal{R}}} = \sigma(t)(a)\cdot\pi(t,a)(v) = \pi(t,a)(v)\cdot\sigma(t)(a) = \pi(t,a)(v)\cdot f_{t,t_a}.$$

Since by hypothesis on $\pi$ we have that $\pi(t,a)(v) \in I(t,a,v)$, it follows that $\inf I(t,a,v) \leq \pi(t,a)(v) \leq \sup I(t,a,v)$, thus $\inf I(t,a,v)\cdot f_{t,t_a} \leq f_{t_a,v_{\mathcal{R}}} \leq \sup I(t,a,v)\cdot f_{t,t_a}$, as required. This completes the proof that if there exist a scheduler $\sigma$ and a nature $\pi$ inducing $t \longrightarrow \mu_t$ such that $\mu\ \mathcal{L}(\mathcal{R})\ \mu_t$, then the LP problem $LP(t,\mu,\mathcal{R})$ is feasible.

For the second implication, suppose that the LP problem $LP(t,\mu,\mathcal{R})$ has a feasible solution $f^*$; define the scheduler $\sigma$ and the nature $\pi$ as follows: $\sigma(t)(a) = f^*_{t,t_a}$ for each $a \in \mathscr{A}(t)$ and $\pi(t,a)(v) = \frac{f^*_{t_a,v_{\mathcal{R}}}}{f^*_{t,t_a}}$ for each $a \in \mathscr{A}(t)$ and $v \in S$ if $f^*_{t,t_a} \neq 0$, an arbitrary distribution in $\mathscr{H}^a_t$ otherwise. The fact that both $\sigma$ and $\pi$ are probability distributions is trivial: the non-negativity of $\sigma(t)(a)$ and $\pi(t,a)(v)$ for each $a \in \mathscr{A}(t)$ and $v \in S$ is ensured by the non-negativity of the feasible solution $f^*$; $\sum_{a\in\mathscr{A}(t)}\sigma(t)(a) = \sum_{a\in\mathscr{A}(t)} f^*_{t,t_a} = f^*_{\Delta,t} = 1$; if $f^*_{t,t_a} = 0$, then $\pi(t,a)$ is a probability distribution by the way it is chosen; if $f^*_{t,t_a} \neq 0$, then

$$\sum_{v\in S}\pi(t,a)(v) = \sum_{v\in S}\frac{f^*_{t_a,v_{\mathcal{R}}}}{f^*_{t,t_a}} = \frac{\sum_{v\in S} f^*_{t_a,v_{\mathcal{R}}}}{f^*_{t,t_a}} = \frac{f^*_{t,t_a}}{f^*_{t,t_a}} = 1.$$

The next step in the proof is to show that for each $a \in \mathscr{A}(t)$ and $v \in S$, we have that $\pi(t,a)(v) \in I(t,a,v)$. Fix $a \in \mathscr{A}(t)$ and $v \in S$ and suppose that $f^*_{t,t_a} \neq 0$. $f^*_{t_a,v_{\mathcal{R}}} - \inf I(t,a,v)\cdot f^*_{t,t_a} \geq 0$ implies that $f^*_{t_a,v_{\mathcal{R}}} \geq \inf I(t,a,v)\cdot f^*_{t,t_a}$, that is, $\frac{f^*_{t_a,v_{\mathcal{R}}}}{f^*_{t,t_a}} \geq \inf I(t,a,v)$, i.e., $\pi(t,a)(v) \geq \inf I(t,a,v)$; a similar argument shows that $\pi(t,a)(v) \leq \sup I(t,a,v)$.

The last step is about the condition of the lifting: fix an equivalence class $\mathcal{C} \in S/\mathcal{R}$;

$$\mu_t(\mathcal{C}) = \sum_{v\in\mathcal{C}}\mu_t(v) = \sum_{v\in\mathcal{C}}\sum_{a\in\mathscr{A}(t)}\sigma(t)(a)\cdot\pi(t,a)(v)$$
$$= \sum_{v\in\mathcal{C}}\sum_{a\in\mathscr{A}(t)} f^*_{t,t_a}\cdot\frac{f^*_{t_a,v_{\mathcal{R}}}}{f^*_{t,t_a}} = \sum_{v\in\mathcal{C}}\sum_{a\in\mathscr{A}(t)} f^*_{t_a,v_{\mathcal{R}}} = \sum_{v\in\mathcal{C}} f^*_{v_{\mathcal{R}},\mathcal{C}} = f^*_{\mathcal{C},\blacktriangledown} = \mu_s(\mathcal{C}),$$

as required. Note that here we are assuming that $f^*_{t,t_a} \neq 0$; if $f^*_{t,t_a} = 0$, then also $\sigma(t)(a) = 0$, hence we can just omit it from the sum. This completes the proof that if the LP problem $LP(t,\mu,\mathcal{R})$ is feasible, then there exist a scheduler $\sigma$ and a nature $\pi$ inducing $t \longrightarrow \mu_t$ such that $\mu\ \mathcal{L}(\mathcal{R})\ \mu_t$. ■

It is worthwhile to be noted that the resulting scheduler and nature are history-independent, i.e., they base their choice only on the current state (and action, for nature).

Moreover, solving the generated LP problem from Definition 6.6 can be done in polynomial time. The polynomial time complexity, however, is not preserved when uncertainty affects transition probabilities in the model. In fact, in presence of uncertainty, the step condition needs to be checked for any realization of the probability distribution $\mu_s \in \mathscr{P}_{\mathcal{R}}^{s,a}$. This fact is essentially the main barrier in designing efficient algorithms for probabilistic bisimulation on such uncertain systems which particularly leads the problem to be intractable. To this end, we first model the probabilistic bisimulation problem as the ARC of the uncertain $LP(t, \mu, \mathcal{R})$ LP problem in which the uncertain data is the probability distribution $\mu$. More precisely, by Lemma 6.3, we can replace in Definition 6.4 the matching transition $\mu_t \in \mathscr{P}_{\mathcal{R}}^{t}$ for $\mu_s \in \mathscr{P}_{\mathcal{R}}^{s,a}$ such that $\mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t$ with the check for feasibility of $LP(t, \mu_s, \mathcal{R})$.

Modelling this probabilistic bisimulation game as ARC of an uncertain LP allows the adjustable flow variables $f_{i,j}$ in the $LP(t, \mu, \mathcal{R})$ LP problem to tune themselves to the uncertain probability distribution $\mu$. However, the ARC is in general computationally hard. On the other hand, restricting the adjustable flow variables $f_{i,j}$ to be affinely dependent on the uncertain probability distributions $\mu$ allows us to model the bisimulation problem as *affinely adjustable robust counterpart* of an uncertain LP problem and thus to arrive at a polynomial time algorithm to compute the equivalence relation $\mathcal{R}$. From the game semantics viewpoint, such affine dependency restriction reduces the power of the defender to match the challenger's choices and therefore, it leads to a finer (sub-optimal) equivalence relation.

**Affinely Adjustable Robust Counterpart for Probabilistic Bisimulation.** In the sequel, we adapt the ARC theory presented in Chapter 3 (cf. Section 3.3) to the setting of probabilistic bisimulation by imposing a restriction on adjustable flow variables $f_{i,j}$ to tune themselves *affinely* upon the uncertain probability distribution $\mu$ in the challenger's uncertainty set $\mathscr{P}_{\mathcal{R}}^{s,a}$. Without loss of generality, we let $\mathcal{C}_1, \ldots, \mathcal{C}_n$ be the equivalence classes induced by $\mathcal{R}$. We encode the *affine dependence* in the network graph $G(t, \mathcal{R}) = (V, E)$ by restricting, for each arch $(i, j) \in E$, the flow variable $f_{i,j}$ to be

$$f_{i,j} = l_{i,j} + \sum_{k=1}^{n} w^k \cdot \mu(\mathcal{C}_k),$$

where the new optimization variables are considered in the vector $l$ and the matrix $W$. Plugging affine equivalences of flow variables, we end up with the affinely adjustable robust counterpart (AARC) of the ULP problem $\{LP(t, \mu, \mathcal{R})\}_{\mu \in \mathscr{P}_{\mathcal{R}}^{s,a}}$ shown in Figure 6.4.

In order to show the computational tractability of the AARC, we need to ensure that the uncertainty set $\mathscr{P}_{\mathcal{R}}^{s,a}$ is itself computationally tractable. Formally, a set $\mathscr{P}_{\mathcal{R}}^{s,a}$ is *computationally tractable* [GLS81] if for any vector $\mu$, there is a tractable "separation oracle" that either decides correctly $\mu \in \mathscr{P}_{\mathcal{R}}^{s,a}$ or otherwise, generates a *separator*, i.e., a non-zero vector $r$ such that $r^T \mu \geq \max_{\gamma \in \mathscr{P}_{\mathcal{R}}^{s,a}} r^T \gamma$.

**Proposition 6.2.** *For every state $s \in S$, action $a \in \mathscr{A}(s)$ and equivalence relation $\mathcal{R}$, the polytopic uncertainty set $\mathscr{P}_{\mathcal{R}}^{s,a}$ is computationally tractable.*

*Proof.* It is easy to see that the polytopic uncertainty set $\mathscr{P}_{\mathcal{R}}^{s,a}$, can be represented by finitely many linear inequalities. It is in fact a special case of ellipsoidal uncertainty that are known to be computationally tractable [BTN99, BTEGN09]. ■

$$
\begin{array}{rll}
\min_{l,w} & 0 & \\
\text{subject to:} & l_{u,v} + \sum_{k=1}^{n} w^k \cdot \mu(\mathcal{C}_k) \geq 0 & \text{for each } (u,v) \in E \\
& l_{\triangle,t} + \sum_{k=1}^{n} w^k \cdot \mu(\mathcal{C}_k) = 1 & \\
& l_{\mathcal{C},\blacktriangledown} + \sum_{k=1}^{n} w^k \cdot \mu(\mathcal{C}_k) = \mu(\mathcal{C}_i) & \text{for each } \mathcal{C}_i \in S/\mathcal{R}, i = 1,\ldots,n \\
& \sum_{\{u|(u,v)\in E\}} (l_{u,v} + \sum_{k=1}^{n} w^k \cdot \mu(\mathcal{C}_k)) - \sum_{\{u|(v,u)\in E\}} (l_{v,u} + \sum_{k=1}^{n} w^k \cdot \mu(\mathcal{C}_k)) = 0 & \\
& & \text{for each } v \in V \setminus \{\triangle, \blacktriangledown\} \\
& l_{t_a,v_{\mathcal{R}}} + \sum_{k=1}^{n} w^k \cdot \mu(\mathcal{C}_k) - \inf I(t,a,v) \cdot (l_{t,t_a} + \sum_{k=1}^{n} w^k \cdot \mu(\mathcal{C}_k)) \geq 0 & \\
& & \text{for each } a \in \mathscr{A}(t) \text{ and } v \in S \\
& l_{t_a,v_{\mathcal{R}}} + \sum_{k=1}^{n} w^k \cdot \mu(\mathcal{C}_k) - \sup I(t,a,v) \cdot (l_{t,t_a} + \sum_{k=1}^{n} w^k \cdot \mu(\mathcal{C}_k)) \leq 0 & \\
& & \text{for each } a \in \mathscr{A}(t) \text{ and } v \in S \\
& \forall \mu = (\mu(\mathcal{C}_1),\ldots,\mu(\mathcal{C}_n)) \in \mathscr{P}_{\mathcal{R}}^{s,a} &
\end{array}
$$

Figure 6.4: Affinely adjustable robust counterpart of the ULP $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}_{\mathcal{R}}^{s,a}}$.

Computational tractability of the polytopic uncertainty sets concludes immediately tractability of the AARC. Formally,

**Theorem 6.6.** *Given the fixed recourse ULP problem $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}_{\mathcal{R}}^{s,a}}$, the AARC of $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}_{\mathcal{R}}^{s,a}}$ is computationally tractable.*

*Proof.* We prove the theorem by getting inspiration from [Gus02]. The AARC of the uncertain LP $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}_{\mathcal{R}}^{s,a}}$ can be described in a closed form as the optimization problem

$$\min_{l,W} \left\{ 0 : V[l + W\mu] \leq b \quad \forall \mu \in \mathscr{P}_{\mathcal{R}}^{s,a} \right\}$$

where $V$ is the adjacency matrix and $l, W$ are the set of variables. It is not difficult to see that since the recourse matrix $V$ is fixed, the above optimization problem can be rewritten as

$$\min_{x=[l,W]} \left\{ 0 : A(\mu)x \leq b(\mu) \quad \forall \mu \in \mathscr{P}_{\mathcal{R}}^{s,a} \right\}$$

by appropriately chosen $A(\mu)$, $b(\mu)$ affinely dependent on $\mu$ and with $x = [l, W]$. The latter optimization can equivalently be written as

$$
\begin{array}{rl}
\min_x & \left\{ 0 : x \in S \right\} \\
& S = \left\{ x \;\middle|\; \forall \mu \in \mathscr{P}_{\mathcal{R}}^{s,a} : -A(\mu)x + b(\mu) \geq 0 \right\}
\end{array}
$$

This optimization problem is basically the robust counterpart of an uncertain LP with uncertain data that are affinely parametrized by $\mu \in \mathscr{P}_{\mathcal{R}}^{s,a}$ for which the computational tractability (and therefore, of AARC) is readily given in [BTN99]. ∎

It is not difficult to see that in the setting of probabilistic bisimulation, the polytopic uncertainty sets $\mathscr{P}_{\mathcal{R}}^{s}$ are closed, convex, and *well structured*, i.e., they can be described by a list of linear inequalities. Thus in our setting, the resulting AARC is also well structured and thus can be solved using highly efficient LP solvers (for instance, CPLEX [cpl] and Gurobi [GUR]) even for large-scale cases.

**Theorem 6.7.** *Given the fixed recourse ULP problem* $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}_{\mathcal{R}}^{s,a}}$, *the AARC of* $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}_{\mathcal{R}}^{s,a}}$ *is equivalent to an explicit LP program.*

*Proof.* We prove the theorem by getting inspiration from [Gus02]. The objective function of the AARC of the ULP $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}_{\mathcal{R}}^{s,a}}$ is linear from the beginning. It is convenient to assume that the right-hand side vector of the AARC is also affinely dependent on the uncertain probability distributions $\mu\in\mathscr{P}_{\mathcal{R}}^{s,a}$. Let us assume that $b\in\mathbb{R}^n$ denotes the right-hand side vector of the AARC. Thus, by affine dependency assumption, we let $b = b^0 + \sum_{k=1}^{n}\mu(\mathcal{C}_k)b^k$. Note that this assumption does not restrict generality as for every entry of the vector $b$ there is always values for $b^0$ and $b^k$, $k=1,\dots,n$ such that the above equation holds. It is also not difficult to see that the description of the feasibility set in the AARC is demonstrated through the system of (possibly infinite) systems of linear inequalities. Without loss of generality, we assume that the linear inequalities

$$V[l+\sum_{k=1}^{n}W^k\cdot\mu(\mathcal{C}_k)]\leq[b^0+\sum_{k=1}^{n}\mu(\mathcal{C}_k)b^k]\qquad\forall\mu\in\mathscr{P}_{\mathcal{R}}^{s,a}\tag{*}$$

represent the compact form of the AARC feasibility region in which $V$ is the adjacency matrix, $W^k$ is the $k^{th}$ column of the matrix $W\in\mathbb{R}_{\geq 0}^{|E|\times n}$ and $\chi=(l,W^1,\dots,W^n)$ is the set of variables. Moreover, we consider the closed form representation of the polytopic uncertainty set

$$\mathscr{P}_{\mathcal{R}}^{s,a}=\left\{\big(\mu(\mathcal{C}_1),\dots,\mu(\mathcal{C}_n)\big)\;\middle|\;\sum_{i=1}^{n}\mu(\mathcal{C}_i)=1,\forall i\in\{1,\dots,n\}:\mu(\mathcal{C}_i)\in I(s,a,\mathcal{C}_i)\right\}$$

as $\mathscr{P}_{\mathcal{R}}^{s,a}:=\{\,\mu=(\mu(\mathcal{C}_1),\dots,\mu(\mathcal{C}_n))\mid G\mu\geq d\,\}$. In the rest of the proof, we describe how the (possibly infinite) system of inequalities (*) can be recasted as an explicit finite system of linear inequalities. To this aim, suppose that

$$g_i^k\equiv g_i^k(\chi)\equiv\begin{cases}(-Vl+b^0)_i & k=0;i=1,\dots,m\\ (-VW^k+b^k)_i & k=1,\dots,n;i=1,\dots,m\end{cases}$$

where $n$ and $m$ are the number of equivalence classes induced by $\mathcal{R}$ and the number of constraints in AARC, respectively. The collection $\chi$ is feasible for (*) if and only if for every $i=1,\dots,m$, the following system holds true:

$$g_i^0(\chi)+\sum_{k=1}^{n}g_i^k(\chi)\geq 0\quad\forall\mu\in\mathscr{P}_{\mathcal{R}}^{s,a}.$$

Equivalently, $\chi$ is feasible for (*) if and only if the optimal value in $m$ LP problems

$$\begin{aligned}\mathrm{Opt}_i\equiv\quad &\mathrm{Min}_{\mu} && g_i^0(\chi)+\textstyle\sum_{k=1}^{n}g_i^k(\chi)\\ &\text{s.t.} && G\mu\geq d\end{aligned}\tag{$P_i[\chi]$}$$

in variable $\mu$ are non-negative. Note that $(P_i[\chi])$ is a feasible problem since $\mathscr{P}_{\mathcal{R}}^{s,a}$ is non-empty. By the linear programming duality Theorem 3.4, the optimal value in the feasible

minimization problem is non-negative if and only if the optimal value in the dual LP problem

$$\mathrm{Max}_y \left\{ g_i^0(\chi) + d^T y : G^T y = g_i(\chi) \equiv \left(g_i^1(\chi), \ldots, g_i^n(\chi)\right)^T, y \geq 0 \right\}$$

is non-negative. In other words:

$$\begin{array}{c} (P_i[\chi]) \text{ has a non-negative optimal value} \\ \Updownarrow \\ \exists y : G^T y = g_i(\chi) \equiv \left(g_i^1(\chi), \ldots, g_i^n(\chi)\right)^T, y \geq 0, g_i^0(\chi) + d^T y \geq 0 \end{array}$$

Therefore, $\chi = (l, W^1, \ldots, W^n)$ is feasible for (*) if and only if $\chi$ can be extended to a feasible solution for the following system of (in)equalities by appropriately chosen $y_1, \ldots, y_m$:

$$\begin{array}{ll} G^T y_i - g_i(l, W^1, \ldots, W^n) = 0 & i = 1, \ldots, m \\ d^T y_i + g_i^0(l, W^1, \ldots, W^n) \geq 0 & i = 1, \ldots, m \\ y_i \geq 0 & i = 1, \ldots, m \end{array} \qquad (**)$$

Since the quantities $g_i^0(l, W^1, \ldots, W^n)$ and $g_i(l, W^1, \ldots, W^n)$ are affine functions of $l, W^1, \ldots, W^n$, thus (**) is a system of linear (in)equalities in variables $l, W^1, \ldots, W^n, y_1, \ldots, y_m$. Since $\chi = (l, W^1, \ldots, W^n)$ is feasible for (*) if and only if $\chi$ can be extended to a feasible solution of (**), the AARC is thus equivalent to the problem of minimizing 0 over the set of feasible solutions of (**), which is an explicit LP program. ■

The "affine decision rules" used to derive the AARC counterpart of the probabilistic bisimulation problem allow us to compute a sub-optimal (finer) probabilistic bisimulation defined as follows.

**Definition 6.7 (AARC probabilistic bisimulation).** *Let $\mathcal{R} \subseteq S \times S$ be an equivalence relation. We say that $\mathcal{R}$ is an* AARC probabilistic bisimulation *if $(s,t) \in \mathcal{R}$ implies that $L(s) = L(t)$ and for each $a \in \mathscr{A}(s)$, the AARC of the ULP problem $\{LP(t, \mu, \mathcal{R})\}_{\mu \in \mathscr{P}_{\mathcal{R}}^{s,a}}$ is feasible.*
*Furthermore, we write $s \sim_{AARC} t$ if there exists an AARC probabilistic bisimulation $\mathcal{R}$ such that $(s,t) \in \mathcal{R}$.*

An immediate result relating $\sim_{AARC}$ and $\sim_{(\forall)}$ is that the former is a refinement of the latter, as formalized by the following proposition.

**Proposition 6.3.** *Given an IMDP $\mathcal{M}$, if $s \sim_{AARC} t$, then $s \sim_{(\forall)} t$, i.e., $\sim_{AARC} \subseteq \sim_{(\forall)}$.*

*Proof.* The result is trivial: given $s \sim_{AARC} t$ with underlying bisimulation $\mathcal{R}$, for each action $a \in \mathscr{A}(s)$, the AARC of the uncertain LP $\{LP(t, \mu, \mathcal{R})\}_{\mu \in \mathscr{P}_{\mathcal{R}}^{s,a}}$ is feasible. Using affine decision rules, the feasible solution for the AARC can be projected back to a feasible solution for the ARC of the uncertain LP problem $\{LP(t, \mu, \mathcal{R})\}_{\mu \in \mathscr{P}_{\mathcal{R}}^{s,a}}$, so by Definition 6.4 and Lemma 6.3 we have that $s \sim_{(\forall)} t$ with $\mathcal{R}$ as underlying probabilistic bisimulation.

■

**Algorithm 6:** Bisimulation($\mathcal{M}$)

**Input:** A relation $\mathcal{R}$ on $S \times S$
**Output:** Return a probabilistic bisimulation $\mathcal{R}$

```
1  begin
2      R ← {(s,t) ∈ S × S | L(s) = L(t)};
3      repeat
4          R' ← R;
5          forall s ∈ S do
6              D ← ∅;
7              forall t ∈ [s]_R do
8                  if Violate(t, R, P_R^{s,a}) then
9                      D ← D ∪ {t};
10             split [s]_R in R into D and [s]_R \ D;
11     until R = R';
12     return R;
```

**Procedure 7:** Violate($t, \mathcal{R}, \mathscr{P}_{\mathcal{R}}^{s,a}$)

**Input:** States $s, t$ and relation $\mathcal{R}$
**Output:** Check if $s \sim_{AARC} t$

```
1  begin
2      Construct the AARC of the
       ULP {LP(t, μ, R)}_{μ ∈ P_R^{s,a}} defined
       in Figure 6.4;
3      return is AARC not feasible?;
```

Figure 6.5: Decision algorithm to decide probabilistic bisimulation $\sim_{AARC}$ for *IMDP*s

**Decision Algorithm.** We now present a polynomial algorithm computing the probabilistic bisimulation $\sim_{AARC}$. The general idea of the algorithm follows the one of the algorithm in Figure 6.3 and involves the construction of the polytopes of the challenger's probability distributions.

In order to compute $\sim_{AARC}$ on *IMDP* $\mathcal{M} = (S, \bar{s}, \mathscr{A}, \mathtt{AP}, L, I)$, we follow the usual partition refinement approach formalized by the Bisimulation procedure in the algorithm depicted in Figure 6.5. The core part is to check whether two states "violate the definition of bisimulation". This is where the algorithm differs from the exact one depicted in Figure 6.3.

The violation is checked by the procedure Violate. We show that this amounts in solving the AARC of the uncertain LP problem $\{LP(t, \mu, \mathcal{R})\}_{\mu \in \mathscr{P}_{\mathcal{R}}^{s,a}}$ as follows. Recall that for $s \in S$ and an action $a \in \mathscr{A}(s)$, we denote by $\mathscr{P}_{\mathcal{R}}^{s,a}$ the polytope of feasible successor distributions over *equivalence classes* of $\mathcal{R}$ with respect to taking the action $a$ in the state $s$, as discussed in Section 6.2. Note that we require that the probability of each class $\mathcal{C}$ must be in the interval of the sum of probabilities that can be assigned to states of $\mathcal{C}$. As specified in the procedure Violate, we show that it suffices to check the feasibility of the resulting AARC of the constructed uncertain LP problem.

Given an *IMDP* $\mathcal{M}$, let $N = \max\{|S|, |A|\}$. It is not difficult to see that the procedure Violate is called at most $N^4$ times. In every call to this procedure, we need to generate and solve the explicit form of the AARC which is an LP according to Theorem 6.7, solvable in polynomial time $\mathcal{O}(poly(N))$. This means that computing $\sim_{AARC}$ can be done in time $|\mathcal{M}|^{\mathcal{O}(1)} \cdot \mathcal{O}(poly(N))$.

**Theorem 6.8.** *Algorithm in Figure 6.5 computes $\sim_{AARC}$ in time polynomial in $|\mathcal{M}|$.*

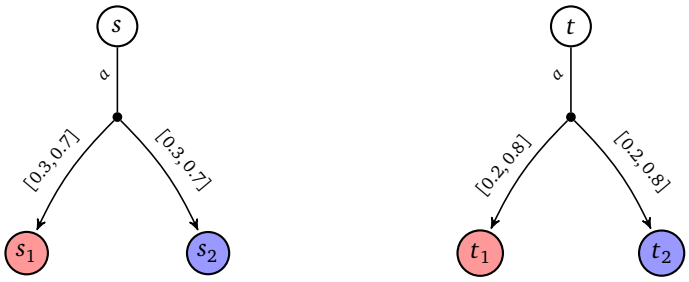



Figure 6.6: A pair of states and transitions in an *IMDP*

*Proof.* Immediate by the previous analysis. ■

**Example 6.3.** *The purpose of this example is to compare the behavior of* $\sim_{(\forall)}$ *and* $\sim_{AARC}$ *in splitting the states of a partition in the partition-refinement algorithm. As stated earlier,* $\sim_{(\forall)}$ *can alternatively be modelled as the ARC of the ULP* $\{LP(t, \mu, \mathcal{R})\}_{\mu \in \mathscr{P}_{\mathcal{R}}^{s,a}}$*. It is not difficult to see that the core difference between the approximation scheme* $\sim_{AARC}$ *and the exact* $\sim_{(\forall)}$ *algorithm can be tracked back to the procedure* Violate *to split a pair of states.*

*As described in Section 6.2, the* Violate *procedure in the exact* $\sim_{(\forall)}$ *algorithm amounts to check the equivalence of constructed convex hulls for a given pair of states that can alternatively be verified by checking the feasibility of the* $LP(t, \mu, \mathcal{R})$ *LP problem for every extreme point* $\mu \in \mathscr{P}_{\mathcal{R}}^{s,a}$ *which is in turn the ARC of the uncertain LP* $\{LP(t, \mu, \mathcal{R})\}_{\mu \in \mathscr{P}_{\mathcal{R}}^{s,a}}$*. Therefore, for a given pair of states* $(s, t)$*, we map the difference of the* Violate *procedure in both algorithms to the difference of* $\sim_{(\forall)}$ *and* $\sim_{AARC}$ *in splitting pairs of states.*

*Solving ARC or equivalently checking bisimilarity of a pair of states is in general computationally hard as indicated in Theorem 6.5. However, in our proposed approximation algorithm, the* Violate *procedure or ARC is approximated using* affine decision rules *and is solved in polynomial time, speaking about AARC. This computational tractability is not often the case as it heavily depends on the geometry of the uncertainty set. However, in the setting of probabilistic bisimulation for IMDPs, the polytopic uncertainty is computational tractable which consequently ensures the computational tractability.*

*In this example, we investigate the result of* $\sim_{(\forall)}$ *and* $\sim_{AARC}$ *for states* $(s, t)$ *in the IMDP* $\mathcal{M}$ *given in Figure 6.6. Let us assume that states with the same color/index are in the same class in the equivalence relation* $\mathcal{R}$*, that is, the set of equivalence classes induced by* $\mathcal{R}$ *is* $\{\{s, t\}, \{s_1, t_1\}, \{s_2, t_2\}\}$*. It is not difficult to see that*

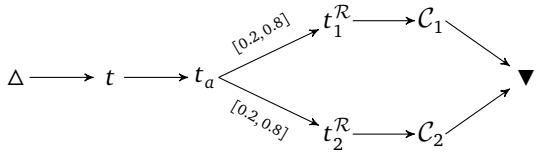
$$\mathscr{P}_{\mathcal{R}}^{s} = \mathrm{CH}(\bigcup_{a \in \mathscr{A}(s)} \mathscr{P}_{\mathcal{R}}^{s,a}) = \mathrm{CH}(\mathscr{P}_{\mathcal{R}}^{s,a}) =$$
$$\{\mu \in \mathrm{Disc}(S/\mathcal{R}) \mid \mu([s_1]_{\mathcal{R}}) \in [0.3, 0.7], \mu([s_2]_{\mathcal{R}}) \in [0.3, 0.7], \mu([s_1]_{\mathcal{R}}) + \mu([s_2]_{\mathcal{R}}) = 1\}$$

*and*

$$\mathscr{P}_{\mathcal{R}}^{t} = \mathrm{CH}(\bigcup_{a \in \mathscr{A}(t)} \mathscr{P}_{\mathcal{R}}^{t,a}) = \mathrm{CH}(\mathscr{P}_{\mathcal{R}}^{t,a}) =$$
$$\{\mu \in \mathrm{Disc}(S/\mathcal{R}) \mid \mu([t_1]_{\mathcal{R}}) \in [0.2, 0.8], \mu([t_2]_{\mathcal{R}}) \in [0.2, 0.8], \mu([t_1]_{\mathcal{R}}) + \mu([t_2]_{\mathcal{R}}) = 1\}.$$

*Therefore, the comparison of convex hulls results in the fact that the states s and t are not bisimilar, i.e., $s \not\sim_{(\forall)} t$, since for instance the distribution $\mu$ such that $\mu([t_1]_{\mathcal{R}}) = 0.75$ and $\mu([t_2]_{\mathcal{R}}) = 0.25$ belongs to $\mathscr{P}^t_{\mathcal{R}}$ but not to $\mathscr{P}^s_{\mathcal{R}}$.*

*In the next step, we check bisimilarity of states s and t based on $\sim_{AARC}$. To this aim, we need to solve AARC of the uncertain LPs $\{LP(s,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}^{t,a}_{\mathcal{R}}}$ and $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}^{s,a}_{\mathcal{R}}}$ extracted from $G(s,\mathcal{R}) = (V_s, E_s)$ and $G(t,\mathcal{R}) = (V_t, E_t)$ network graphs, respectively. Due to symmetry, we generate AARC of the uncertain LP $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}^{s,a}_{\mathcal{R}}}$ and only report the result of solving AARC of the other uncertain LP. To this end, we derive the uncertain LP $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}^{s,a}_{\mathcal{R}}}$ extracted from the $G(t,\mathcal{R}) = (V_t, E_t)$ network graph depicted as follows:*




*Note that in the $G(t,\mathcal{R})$ network graph, nodes $\mathcal{C}_1$ and $\mathcal{C}_2$ are equivalence classes including the red and blue states, respectively. The ARC of the ULP $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}^{s,a}_{\mathcal{R}}}$ problem after removal of redundant constraints is extracted as follows.*

| *ARC of the uncertain LP* $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}^{s,a}_{\mathcal{R}}}$ | | |
|---|---|---|
| $\min_f\{0 : \forall\mu\in\mathscr{P}^{s,a}_{\mathcal{R}}:$ | | |
| $f_{t_a,t_1^{\mathcal{R}}} + f_{t_a,t_2^{\mathcal{R}}} = 1$ | $f_{t_1^{\mathcal{R}},\mathcal{C}_1} = f_{t_a,t_1^{\mathcal{R}}}$ | $f_{t_2^{\mathcal{R}},\mathcal{C}_2} = f_{t_a,t_2^{\mathcal{R}}}$ |
| $f_{t_a,t_1^{\mathcal{R}}} - 0.2 \geq 0$ | $f_{t_a,t_1^{\mathcal{R}}} - 0.8 \leq 0$ | $f_{t_a,t_2^{\mathcal{R}}} - 0.2 \geq 0$ |
| $f_{t_a,t_2^{\mathcal{R}}} - 0.8 \leq 0$ | $f_{t_1^{\mathcal{R}},\mathcal{C}_1} = f_{\mathcal{C}_1,\blacktriangledown}$ | $f_{t_2^{\mathcal{R}},\mathcal{C}_2} = f_{\mathcal{C}_2,\blacktriangledown}$ |
| $f_{\mathcal{C}_1,\blacktriangledown} = \mu(\mathcal{C}_1)$ | $f_{\mathcal{C}_2,\blacktriangledown} = \mu(\mathcal{C}_2)$ | $f_{ij} \geq 0 \quad \forall(i,j)\in E_t\}$ |

*We proceed by generating AARC of the derived ULP $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}^{s,a}_{\mathcal{R}}}$ problem by using affine decision rules $f_{ij} = l_{ij} + w^1_{ij}\mu(\mathcal{C}_1) + w^2_{ij}\mu(\mathcal{C}_1)$ for all arcs $(i,j)\in E_t$ as described in the proof of Theorem 6.7. We have generated the AARC in Python and used PuLP [pul] as a front-end optimization modeling and Gurobi [GUR] as a back-end optimization solver to solve all linear programming problems. The extracted AARC is read as in Table 6.2. Note that in the extracted AARC, all variables are unrestricted in sign. Solving the AARC results in the following optimal solution indicating that the state t can simulate the state s.*

| *Optimal solution of the AARC (the value of non-reported variables is zero.)* | | | |
|---|---|---|---|
| $l_{\mathcal{C}_1,\blacktriangledown} = 0$ | $l_{\mathcal{C}_2,\blacktriangledown} = 0$ | $l_{t_a,t_1^{\mathcal{R}}} = 0.45$ | $l_{t_a,t_2^{\mathcal{R}}} = 0.55$ |
| $l_{t_1^{\mathcal{R}},\mathcal{C}_1} = 0.45$ | $l_{t_2^{\mathcal{R}},\mathcal{C}_2} = 0$ | $w^1_{\mathcal{C}_1,\blacktriangledown} = 1.0$ | $w^1_{t_a,t_1^{\mathcal{R}}} = 0.55$ |
| $w^1_{t_a,t_2^{\mathcal{R}}} = -0.55$ | $w^1_{t_1^{\mathcal{R}},\mathcal{C}_1} = 0.55$ | $w^1_{t2^{\mathcal{R}},\mathcal{C}_2} = 0.0$ | $w^2_{\mathcal{C}_2,\blacktriangledown} = 1$ |
| $w^2_{t_a,t_1^{\mathcal{R}}} = -0.45$ | $w^2_{t_a,t_2^{\mathcal{R}}} = 0.45$ | $w^2_{t_1^{\mathcal{R}},\mathcal{C}_1} = -0.45$ | $w^2_{t_2^{\mathcal{R}},\mathcal{C}_2} = 1$ |
| $y_{101} = 0$ | $y_{104} = 0$ | $y_{151} = 0.55$ | $y_{154} = 0.45$ |
| $y_{162} = 0.55$ | $y_{163} = 0.45$ | $y_{172} = 0.55$ | $y_{173} = 0.45$ |
| $y_{181} = 0.55$ | $y_{184} = 0.45$ | $y_{51} = 0$ | $y_{56} = 0.55$ |
| $y_{62} = 0$ | $y_{65} = 0.55$ | $y_{75} = 0.45$ | $y_{86} = 0.45$ |
| $y_{93} = 0$ | | | |

*Following the same procedure for the uncertain LP* $\{LP(s,\mu,\mathcal{R})\}_{\mu\in\mathscr{D}_{\mathcal{R}}^{t,a}}$ *results in the infeasibility of the extracted AARC and thus incapability of state* $s$ *to simulate state* $t$*. Therefore,* $s \not\sim_{AARC} t$*.*

♦

## 6.3 Compositional Reasoning for Interval Markov Decision Processes

In this section, we establishes a framework for compositional verification of complex systems with interval uncertainty. In particular, we provide a compositional reasoning over interval *MDP*s to understand better the ways how large models with interval uncertainties can be composed.

### 6.3.1 Action Agnostic Probabilistic Automata

We now introduce the action agnostic probabilistic automata which we use in this and the next chapter, based on the probabilistic automata framework, following the notation of [Seg06]. Note that the probabilistic automata we consider here correspond to the *simple* probabilistic automata of [Seg95]. In practice, we consider the subclass of (simple) probabilistic automata of [Seg95] having as set of actions the same singleton $\{f\}$, that is, all transitions are labelled by the same external action $f$. Since this action is unique, we just drop it from the definitions.

**Definition 6.8** (**Action agnostic *PA*s**). *An* action agnostic probabilistic automaton *(PA) is a tuple* $\mathfrak{P} = (S, \bar{s}, \mathtt{AP}, L, T)$*, where* $S$ *is a set of* states*,* $\bar{s} \in S$ *is the* start *state,* $\mathtt{AP}$ *is a finite set of* atomic propositions*,* $L : S \to 2^{\mathtt{AP}}$ *is a* labelling function*, and* $T \subseteq S \times \mathrm{Disc}(S)$ *is a* probabilistic transition relation*. We denote by* $[\mathfrak{P}]$ *the class of all finite-state finite-transition action agnostic probabilistic automata.*

We assume that each state in $S$ is reachable from $\bar{s}$. The start state is also called the *initial* state; we let $s$, $t$, $u$, $v$, and their variants with indices range over $S$. We also denote the generic elements of an action agnostic probabilistic automaton $\mathfrak{P}$ by $S$, $\bar{s}$, $\mathtt{AP}$, $L$, $T$, and we propagate primes and indices when necessary. Thus, for example, the action agnostic probabilistic automaton $\mathfrak{P}_i'$ has states $S_i'$, start state $\bar{s}_i'$, and transition relation $T_i'$.

We follow the same semantics as *PA*s for the action agnostic *PA*s. In particular, a transition $tr = (s,\mu) \in T$, also written as $s \longrightarrow \mu$, is said to *leave* from state $s$ and to *lead* to the measure $\mu$. We denote by $src(tr)$ the *source* state $s$ and by $trg(tr)$ the *target* measure $\mu$, also denoted by $\mu_{tr}$. We also say that $s$ enables the transition $(s,\mu)$ and that $(s,\mu)$ is enabled from $s$.

**Example 6.4.** *An example of an action agnostic PA is the one shown in Figure 6.7: the set of states is* $S = \{\bar{s}, r, y, g, \blacksquare, \blacktriangle, \bigcirc\}$*, the start state is* $\bar{s}$*, the set of atomic propositions is* $\mathtt{AP} = S$*, the labelling function* $L$ *is such that for each* $s \in S$*,* $L(s) = s$*, and the transition relation* $T$ *contains the following transitions:* $\bar{s} \longrightarrow \rho$ *with* $\rho = \{(r, 0.5), (y, 0.3), (g, 0.2)\}$*,* $r \longrightarrow \delta_{\blacksquare}$*,* $y \longrightarrow \delta_{\blacktriangle}$*,* $g \longrightarrow \delta_{\bigcirc}$*, and* $r \longrightarrow \delta_{\bar{s}}$*.*

♦

$\min_{w,y} 0$

$w^1_{t_a,t_1^{\mathcal{R}}} + w^1_{t_a,t_2^{\mathcal{R}}} + y_{11} - y_{12} + y_{15} - y_{16} = 0$ | $w^2_{t_a,t_1^{\mathcal{R}}} + w^2_{t_a,t_2^{\mathcal{R}}} + y_{13} - y_{14} + y_{15} - y_{16} = 0$

$-w^1_{t_a,t_1^{\mathcal{R}}} - w^1_{t_a,t_2^{\mathcal{R}}} + y_{21} - y_{22} + y_{25} - y_{26} = 0$ | $-w^2_{t_a,t_1^{\mathcal{R}}} - w^2_{t_a,t_2^{\mathcal{R}}} + y_{23} - y_{24} + y_{25} - y_{26} = 0$

$-w^1_{t_a,t_1^{\mathcal{R}}} + w^1_{t_1^{\mathcal{R}},\mathcal{C}_1} + y_{31} - y_{32} + y_{35} - y_{36} = 0$ | $-w^2_{t_a,t_1^{\mathcal{R}}} + w^2_{t_1^{\mathcal{R}},\mathcal{C}_1} + y_{33} - y_{34} + y_{35} - y_{36} = 0$

$w^1_{t_a,t_1^{\mathcal{R}}} - w^1_{t_1^{\mathcal{R}},\mathcal{C}_1} + y_{41} - y_{42} + y_{45} - y_{46} = 0$ | $w^2_{t_a,t_1^{\mathcal{R}}} - w^2_{t_1^{\mathcal{R}},\mathcal{C}_1} + y_{43} - y_{44} + y_{45} - y_{46} = 0$

$-w^1_{t_a,t_2^{\mathcal{R}}} + w^1_{t_2^{\mathcal{R}},\mathcal{C}_2} + y_{51} - y_{52} + y_{55} - y_{56} = 0$ | $-w^2_{t_a,t_2^{\mathcal{R}}} + w^2_{t_2^{\mathcal{R}},\mathcal{C}_2} + y_{53} - y_{54} + y_{55} - y_{56} = 0$

$w^1_{t_a,t_2^{\mathcal{R}}} - w^1_{t_2^{\mathcal{R}},\mathcal{C}_2} + y_{61} - y_{62} + y_{65} - y_{66} = 0$ | $w^2_{t_a,t_2^{\mathcal{R}}} - w^2_{t_2^{\mathcal{R}},\mathcal{C}_2} + y_{63} - y_{64} + y_{65} - y_{66} = 0$

$-w^1_{\mathcal{C}_1,\blacktriangledown} + w^1_{t_1^{\mathcal{R}},\mathcal{C}_1} + y_{71} - y_{72} + y_{75} - y_{76} = 0$ | $-w^2_{\mathcal{C}_1,\blacktriangledown} + w^2_{t_1^{\mathcal{R}},\mathcal{C}_1} + y_{73} - y_{74} + y_{75} - y_{76} = 0$

$w^1_{\mathcal{C}_1,\blacktriangledown} - w^1_{t_1^{\mathcal{R}},\mathcal{C}_1} + y_{81} - y_{82} + y_{85} - y_{86} = 0$ | $w^2_{\mathcal{C}_1,\blacktriangledown} - w^2_{t_1^{\mathcal{R}},\mathcal{C}_1} + y_{83} - y_{84} + y_{85} - y_{86} = 0$

$-w^1_{\mathcal{C}_2,\blacktriangledown} + w^1_{t_2^{\mathcal{R}},\mathcal{C}_2} + y_{91} - y_{92} + y_{95} - y_{96} = 0$ | $-w^2_{\mathcal{C}_2,\blacktriangledown} + w^2_{t_2^{\mathcal{R}},\mathcal{C}_2} + y_{93} - y_{94} + y_{95} - y_{96} = 0$

$w^1_{\mathcal{C}_2,\blacktriangledown} - w^1_{t_2^{\mathcal{R}},\mathcal{C}_2} + y_{101} - y_{102} + y_{105} - y_{106} = 0$ | $w^2_{\mathcal{C}_2,\blacktriangledown} - w^2_{t_2^{\mathcal{R}},\mathcal{C}_2} + y_{103} - y_{104} + y_{105} - y_{106} = 0$

$w^1_{\mathcal{C}_1,\blacktriangledown} + y_{111} - y_{112} + y_{115} - y_{116} = 1$ | $w^2_{\mathcal{C}_1,\blacktriangledown} + y_{113} - y_{114} + y_{115} - y_{116} = 0$

$-w^1_{\mathcal{C}_1,\blacktriangledown} + y_{121} - y_{122} + y_{125} - y_{126} = -1$ | $-w^2_{\mathcal{C}_1,\blacktriangledown} + y_{123} - y_{124} + y_{125} - y_{126} = 0$

$w^1_{\mathcal{C}_2,\blacktriangledown} + y_{131} - y_{132} + y_{135} - y_{136} = 0$ | $w^2_{\mathcal{C}_2,\blacktriangledown} + y_{133} - y_{134} + y_{135} - y_{136} = 1$

$-w^1_{\mathcal{C}_2,\blacktriangledown} + y_{141} - y_{142} + y_{145} - y_{146} = 0$ | $-w^2_{\mathcal{C}_2,\blacktriangledown} + y_{143} - y_{144} + y_{145} - y_{146} = -1$

$-w^1_{t_a,t_1^{\mathcal{R}}} + y_{151} - y_{152} + y_{155} - y_{156} = 0$ | $-w^2_{t_a,t_1^{\mathcal{R}}} + y_{153} - y_{154} + y_{155} - y_{156} = 0$

$w^1_{t_a,t_1^{\mathcal{R}}} + y_{161} - y_{162} + y_{165} - y_{166} = 0$ | $w^2_{t_a,t_1^{\mathcal{R}}} + y_{163} - y_{164} + y_{165} - y_{166} = 0$

$-w^1_{t_a,t_2^{\mathcal{R}}} + y_{171} - y_{172} + y_{175} - y_{176} = 0$ | $-w^2_{t_a,t_2^{\mathcal{R}}} + y_{173} - y_{174} + y_{175} - y_{176} = 0$

$w^1_{t_a,t_2^{\mathcal{R}}} + y_{181} - y_{182} + y_{185} - y_{186} = 0$ | $w^2_{t_a,t_2^{\mathcal{R}}} + y_{183} - y_{184} + y_{185} - y_{186} = 0$

$-l_{t_a,t_1^{\mathcal{R}}} - l_{t_a,t_2^{\mathcal{R}}} + 0.3y_{11} - 0.7y_{12} + 0.3y_{13} - 0.7y_{14} + y_{15} - y_{16} \geq -1$

$l_{t_a,t_1^{\mathcal{R}}} + l_{t_a,t_2^{\mathcal{R}}} + 0.3y_{21} - 0.7y_{22} + 0.3y_{23} - 0.7y_{24} + y_{25} - y_{26} \geq 1$

$l_{t_a,t_1^{\mathcal{R}}} - l_{t_1^{\mathcal{R}},\mathcal{C}_1} + 0.3y_{31} - 0.7y_{32} + 0.3y_{33} - 0.7y_{34} + y_{35} - y_{36} \geq 0$

$l_{t_a,t_1^{\mathcal{R}}} + l_{t_1^{\mathcal{R}},\mathcal{C}_1} + 0.3y_{41} - 0.7y_{42} + 0.3y_{43} - 0.7y_{44} + y_{45} - y_{46} \geq 0$

$l_{t_a,t_2^{\mathcal{R}}} - l_{t_2^{\mathcal{R}},\mathcal{C}_2} + 0.3y_{51} - 0.7y_{52} + 0.3y_{53} - 0.7y_{54} + y_{55} - y_{56} \geq 0$

$-l_{t_a,t_2^{\mathcal{R}}} + l_{t_2^{\mathcal{R}},\mathcal{C}_2} + 0.3y_{61} - 0.7y_{62} + 0.3y_{63} - 0.7y_{64} + y_{65} - y_{66} \geq 0$

$l_{\mathcal{C}_1,\blacktriangledown} - l_{t_1^{\mathcal{R}},\mathcal{C}_1} + 0.3y_{71} - 0.7y_{72} + 0.3y_{73} - 0.7y_{74} + y_{75} - y_{76} \geq 0$

$-l_{\mathcal{C}_1,\blacktriangledown} + l_{t_1^{\mathcal{R}},\mathcal{C}_1} + 0.3y_{81} - 0.7y_{82} + 0.3y_{83} - 0.7y_{84} + y_{85} - y_{86} \geq 0$

$l_{\mathcal{C}_2,\blacktriangledown} - l_{t_2^{\mathcal{R}},\mathcal{C}_2} + 0.3y_{91} - 0.7y_{92} + 0.3y_{93} - 0.7y_{94} + y_{95} - y_{96} \geq 0$

$-l_{\mathcal{C}_2,\blacktriangledown} + l_{t_2^{\mathcal{R}},\mathcal{C}_2} + 0.3y_{101} - 0.7y_{102} + 0.3y_{103} - 0.7y_{104} + y_{105} - y_{106} \geq 0$

$-l_{\mathcal{C}_1,\blacktriangledown} + 0.3y_{111} - 0.7y_{112} + 0.3y_{113} - 0.7y_{114} + y_{115} - y_{116} \geq 0$

$l_{\mathcal{C}_1,\blacktriangledown} + 0.3y_{121} - 0.7y_{122} + 0.3y_{123} - 0.7y_{124} + y_{125} - y_{126} \geq 0$

$-l_{\mathcal{C}_2,\blacktriangledown} + 0.3y_{131} - 0.7y_{132} + 0.3y_{133} - 0.7y_{134} + y_{135} - y_{136} \geq 0$

$l_{\mathcal{C}_2,\blacktriangledown} + 0.3y_{141} - 0.7y_{142} + 0.3y_{143} - 0.7y_{144} + y_{145} - y_{146} \geq 0$

$l_{t_a,t_1^{\mathcal{R}}} + 0.3y_{161} - 0.7y_{162} + 0.3y_{163} - 0.7y_{164} + y_{165} - y_{166} \geq 0.2$

$-l_{t_a,t_1^{\mathcal{R}}} + 0.3y_{161} - 0.7y_{162} + 0.3y_{163} - 0.7y_{164} + y_{165} - y_{166} \geq 0.8$

$l_{t_a,t_2^{\mathcal{R}}} + 0.3y_{171} - 0.7y_{172} + 0.3y_{173} - 0.7y_{174} + y_{175} - y_{176} \geq 0.2$

$-l_{t_a,t_2^{\mathcal{R}}} + 0.3y_{181} - 0.7y_{182} + 0.3y_{183} - 0.7y_{184} + y_{185} - y_{186} \geq -0.8$

Table 6.2: ARC of the uncertain LP $\{LP(t,\mu,\mathcal{R})\}_{\mu\in\mathscr{P}^{s,a}_{\mathcal{R}}}$

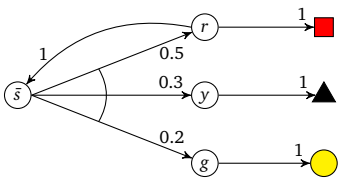



Figure 6.7: An example of action agnostic *PA*s: the *PA* $\mathfrak{E}$

From now on, we drop *action agnostic* since this is the only type of probabilistic automata we consider in this section.

#### 6.3.1.1 Synchronous Product

The following definition of synchronous product is a variation of the definition of parallel composition provided in [Seg95, Seg06], where the synchronization occurs for each pair of enabled transitions. This corresponds to the original definition of parallel composition for probabilistic automata having all transitions labelled by the same external action.

**Definition 6.9 (Synchronous product of *PA*s).** *Given two PAs* $\mathfrak{P}_1$ *and* $\mathfrak{P}_2$*, the* synchronous product *of* $\mathfrak{P}_1$ *and* $\mathfrak{P}_2$*, denoted by* $\mathfrak{P}_1 \otimes \mathfrak{P}_2$*, is the probabilistic automaton* $\mathfrak{P} = (S, \bar{s}, \mathtt{AP}, L, T)$ *where*

- $S = S_1 \times S_2$*;*
- $\bar{s} = (\bar{s}_1, \bar{s}_2)$*;*
- $\mathtt{AP} = \mathtt{AP}_1 \cup \mathtt{AP}_2$*;*
- *for each* $(s_1, s_2) \in S$*,* $L(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$*; and*
- $T = \{ ((s_1, s_2), \mu_1 \times \mu_2) \mid (s_1, \mu_1) \in T_1 \text{ and } (s_2, \mu_2) \in T_2 \}$.

For two *PA*s $\mathfrak{P}_1$ and $\mathfrak{P}_2$ and their synchronous product $\mathfrak{P}_1 \otimes \mathfrak{P}_2$, we refer to $\mathfrak{P}_1$ and $\mathfrak{P}_2$ as the component automata and to $\mathfrak{P}_1 \otimes \mathfrak{P}_2$ as the product automaton.

#### 6.3.1.2 Probabilistic Bisimulation

As for the definition of synchronous product, the following definition of (strong) probabilistic bisimulation is a variation of the definition provided in [Seg06], where all actions are treated as being the same external action. We first introduce the definition of combined transition.

**Definition 6.10 (*PA* combined transition).** *Given a PA* $\mathfrak{P}$ *and a state* $s$*, we say that* $s$ *enables a combined transition reaching the distribution* $\mu$*, denoted by* $s \longrightarrow_c \mu$*, if there exist a finite set of indexes* $I$*, a multiset of transitions* $\{ (s, \mu_i) \in T \mid i \in I \}$*, and a multiset of real values* $\{ p_i \in \mathbb{R}_{\geq 0} \mid i \in I \}$ *such that* $\sum_{i \in I} p_i = 1$ *and* $\mu = \sum_{i \in I} p_i \cdot \mu_i$.

The strong action agnostic probabilistic bisimulation for *PA*s is thus defined as follows:

**Definition 6.11 ((strong) (action agnostic) probabilistic bisimulation for *PA*s).** *Given a PA $\mathfrak{P}$, an equivalence relation $\mathcal{R} \subseteq S \times S$ is a* (strong) (action agnostic) probabilistic bisimulation *on $\mathfrak{P}$ if, for each $(s,t) \in \mathcal{R}$, $L(s) = L(t)$ and for each $s \longrightarrow \mu_s$, there exists a combined transition $t \longrightarrow_c \mu_t$ such that $\mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t$.*
*Given two states $s$ and $t$, we say that $s$ and $t$ are probabilistically bisimilar, denoted by $s \sim^p_{aa} t$, if there exists a probabilistic bisimulation $\mathcal{R}$ on $\mathfrak{P}$ such that $(s,t) \in \mathcal{R}$.*
*Given two PAs $\mathfrak{P}_1$ and $\mathfrak{P}_2$, we say that $\mathfrak{P}_1$ and $\mathfrak{P}_2$ are probabilistically bisimilar, denoted by $\mathfrak{P}_1 \sim^p_{aa} \mathfrak{P}_2$, if there exists a probabilistic bisimulation $\mathcal{R}$ on the disjoint union of $\mathfrak{P}_1$ and $\mathfrak{P}_2$ such that $(\bar{s}_1, \bar{s}_2) \in \mathcal{R}$.*

**Proposition 6.4.** *Given three PAs $\mathfrak{P}_1$, $\mathfrak{P}_2$, and $\mathfrak{P}_3$, if $\mathfrak{P}_1 \sim^p_{aa} \mathfrak{P}_2$, then $\mathfrak{P}_1 \otimes \mathfrak{P}_3 \sim^p_{aa} \mathfrak{P}_2 \otimes \mathfrak{P}_3$.*

*Proof.* The proof is a minor adaptation of the corresponding proof (cf. [Seg95]) for the original definition of probabilistic bisimulation and parallel composition of *PA*s.

In the following, we use the subscript "$j,3$" with $j \in \{1,2\}$ to refer to the component of the *PA* $\mathfrak{P}_{j,3} = \mathfrak{P}_j \otimes \mathfrak{P}_3$.

Let $\mathcal{R}$ be the probabilistic bisimulation justifying $\mathfrak{P}_1 \sim^p_{aa} \mathfrak{P}_2$ and $\mathcal{R}' = \mathcal{R} \times \mathcal{I}_{S_3}$; we claim that $\mathcal{R}'$ is a probabilistic bisimulation between $\mathfrak{P}_1 \otimes \mathfrak{P}_3$ and $\mathfrak{P}_2 \otimes \mathfrak{P}_3$. The fact that $\mathcal{R}'$ is an equivalence relation follows trivially by its definition and the fact that $\mathcal{R}$ is an equivalence relation. The fact that $((\bar{s}_1, \bar{s}_3), (\bar{s}_2, \bar{s}_3))$ follows immediately by the hypothesis that $(\bar{s}_1, \bar{s}_2) \in \mathcal{R}$ and $(\bar{s}_3, \bar{s}_3) \in \mathcal{I}_{S_3}$.

Let $((s_1, s_3), (s_2, s_3)) \in \mathcal{R}'$. Assume, without loss of generality, that $s_1 \in S_1$ and $s_2 \in S_2$; the other cases are similar. The fact that $L_{1,3}(s_1, s_3) = L_{2,3}(s_2, s_3)$ is straightforward, since by definition of synchronous product and the hypothesis that $s_1 \; \mathcal{R} \; s_2$, we have that

$$L_{1,3}(s_1, s_3) = L_1(s_1) \cup L_3(s_3) = L_2(s_2) \cup L_3(s_3) = L_{2,3}(s_2, s_3),$$

as required.

Consider now a transition $(s_1, s_3) \longrightarrow \mu_{1,3}$. By definition of synchronous product, there exist $\mu_1$ and $\mu_3$ such that $s_1 \longrightarrow \mu_1 \in T_1$, $s_3 \longrightarrow \mu_3 \in T_3$, and $\mu_{1,3} = \mu_1 \times \mu_3$. Since $s_1 \; \mathcal{R} \; s_2$, it follows that there exists a combined transition $s_2 \longrightarrow_c \mu_2$ such that $\mu_1 \; \mathcal{L}(\mathcal{R}) \; \mu_2$. Let $I$ be a the finite set of indexes, $\{ (s_2, \mu_{2,i}) \in T_2 \mid i \in I \}$ be a multiset of transitions, and $\{ p_i \in \mathbb{R}_{\geq 0} \mid i \in I \}$ be a multiset of real values such that $\sum_{i \in I} p_i = 1$ and $\mu_2 = \sum_{i \in I} p_i \cdot \mu_{2,i}$. By definition of synchronous product, it follows that for each $i \in I$, $(s_2, s_3) \longrightarrow \mu_{2,i} \times \mu_3 \in T_{2,3}$, hence we have the combined transition $(s_2, s_3) \longrightarrow_c \mu_2 \times \mu_3$. By standard properties of lifting (see, e.g., [TH14]), it follows that $\mu_1 \times \mu_3 \; \mathcal{L}(\mathcal{R}') \; \mu_2 \times \mu_3$, as required. ∎

### 6.3.2 *IMDP*s vs. *PA*s

A cornerstone towards establishing compositional reasoning for *IMDP*s essentially relies on *transformations* from *IMDP*s to *PA*s and vice versa. To this aim, we define two mappings namely, *unfolding* which unfolds a given *IMDP* as a *PA* and *folding* which transforms a given *PA* to an *IMDP*. Formally,

**Definition 6.12 (Unfolding mapping).** *An unfolding mapping $\mathtt{UF}\colon [\mathcal{M}] \to [\mathfrak{P}]$ is a function that maps a given IMDP $\mathcal{M} = (S, \bar{s}, \mathscr{A}, \mathtt{AP}, L, I)$ to the PA $\mathfrak{P} = (S, \bar{s}, \mathtt{AP}, L, T)$ where*

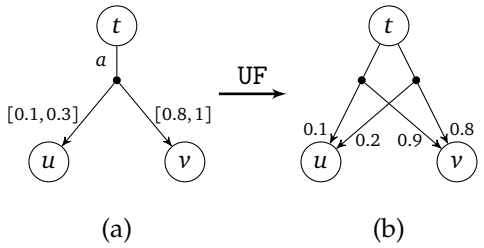



Figure 6.8: Unfolding *IMDP* $\mathcal{M}$ to *PA* $\mathfrak{P}$

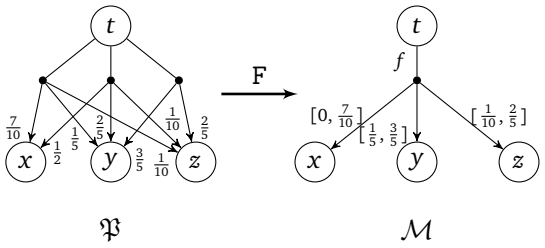



Figure 6.9: Folding a *PA* $\mathfrak{P}$ as an *IMDP* $\mathcal{M}$

$T = \{(s, \mu) \mid s \in S, \exists a \in \mathscr{A}(s) : \mu \in \mathtt{Ext}(\mathscr{H}_s^a)\}$.

It is worthy to note that the unfolding mapping might transform an *IMDP* to a *PA* with an exponentially larger size. This is in fact due to the exponential blow up in the number of transitions in the resultant *PA* which in turn depends on the number of extreme points of the polytope constructed for each state and action in the given *IMDP*. An example of unfolding is given in Figure 6.8.

In order to transform a given *PA* to an instance of *IMDP*s, we use the folding mapping defined as follows:

**Definition 6.13** (**Folding mapping**). *The folding mapping* $\mathtt{F}\colon [\mathfrak{P}] \to [\mathcal{M}]$ *transforms a PA* $\mathfrak{P} = (S, \bar{s}, \mathtt{AP}, L, T)$ *to the IMDP* $\mathcal{M} = (S, \bar{s}, \{f\}, \mathtt{AP}, L, I)$ *where, for each* $s, t \in S$, $I(s, f, t) = \mathrm{proj}_{\mathbf{e}_t}\,\mathrm{CH}(\{\mu \mid (s, \mu) \in T\})$, *where each component* $\mathbf{e_{uv}}$ *of the vector* $\mathbf{e_u} \in \mathbb{R}^{|S|}$ *is defined as* $\mathbf{e_{uv}} = \delta_u(v)$.

An example of the folding mapping is shown in Figure 6.9. The *PA* $\mathfrak{P}$ has three transitions from $t$ with label $a$; in particular, it is worthwhile to note that for all these transitions the probability of reaching $y$ is larger than the probability of reaching $z$, so this has to happen for every combined transition leaving $t$. According to Definition 6.13, the folding of $\mathfrak{P}$ is the *IMDP* $\mathcal{M}$. It is immediate to see that the folding mapping is not surjective as there may be some probabilistic transitions in the generated *IMDP* specification which cannot be mapped to a probability distribution in the given *PA*. In fact, one of such distributions is $\mu_o$ such that $\mu_o(x) = \frac{2}{5}$, $\mu_o(y) = \frac{1}{5}$, and $\mu_o(z) = \frac{2}{5}$ that clearly violates the condition $\mu_o(y) > \mu_o(z)$. This is better recognizable by comparing the corresponding polytopes in a graphical way.

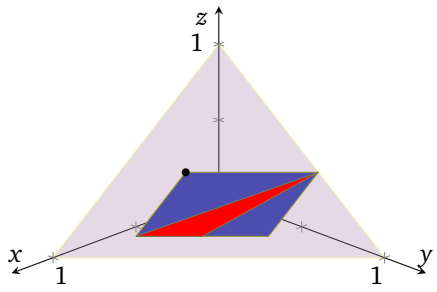



Figure 6.10: Comparison of polytopes resulted from folding mapping F

Figure 6.10 shows the three polytopes involved in $\mathcal{M}$: the purplish large triangular polytope is the standard *2-simplex* in the three dimensional space; the reddish small triangular and the bluish parallelogram-like polytopes represent the convex hull of $\{(\frac{7}{10}, \frac{1}{5}, \frac{1}{10}), (\frac{1}{2}, \frac{2}{5}, \frac{1}{10}), (0, \frac{3}{5}, \frac{2}{5})\}$ and the polytope $\mathscr{H}_t^f$, respectively, both being a sub-polytope of the 2-simplex. Clearly there are points in $\mathscr{H}_t^f$ that do not belong to the reddish polytope, such as the black dot corresponding to $\mu_o$.

**Lemma 6.4.** *Given the folding and unfolding mappings* $\mathtt{F}: [\mathfrak{P}] \to [\mathcal{M}]$ *and* $\mathtt{UF}: [\mathcal{M}] \to [\mathfrak{P}]$, *in general:*

1. $\mathtt{UF}(\mathtt{F}(\mathfrak{P})) \neq \mathfrak{P}$
2. $\mathtt{F}(\mathtt{UF}(\mathcal{M})) \neq \mathcal{M}$

*Proof.* Proof is straightforward. ■

As we will discuss later, the general incompleteness property of the folding mapping does not influence on the generality of our compositional reasoning for *IMDP* specifications. We will dive into this point later in Section 6.3.3.

### 6.3.3 Compositional Reasoning for *IMDP*s

The compositional reasoning is a widely used technique (see, e.g., [CGM+96, HK00, KKZJ07]) that permits to deal with large systems. In particular, a large system is decomposed into multiple components running in parallel; such components are then minimized by replacing each of them by a bisimilar but smaller one so that the overall behaviour remains unchanged. In order to apply this technique, bisimulation has first to be extended to pairs of components and then to be shown to be transitive and preserved by the synchronous product operator. The extension to a pair of components is trivial and commonly done (see, e.g., [CS02, Seg95]):

**Definition 6.14** (**Probabilistic bisimulation for *IMDP* components**). *Given two IMDPs* $\mathcal{M}_1$ *and* $\mathcal{M}_2$, *we say that they are* probabilistic bisimilar, *denoted by* $\mathcal{M}_1 \sim_{(\forall)} \mathcal{M}_2$, *if there*

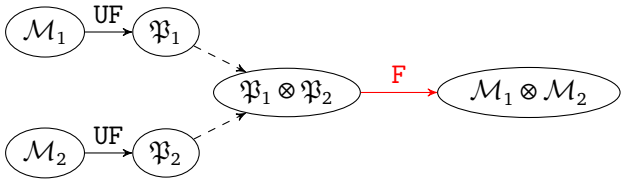



Figure 6.11: Schematic representation of the synchronous product of *IMDP*s $\mathcal{M}_1$ and $\mathcal{M}_2$. In this figure, we let $\mathfrak{P}_1 = \mathtt{UF}(\mathcal{M}_1)$ and $\mathfrak{P}_2 = \mathtt{UF}(\mathcal{M}_2)$.

*exists a probabilistic bisimulation on the disjoint union of $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $\bar{s}_1 \sim_{(\forall)} \bar{s}_2$.*

The next step is to define the synchronous product for *IMDP*s:

**Definition 6.15 (*IMDP*s synchronous product).** *Given two IMDPs $\mathcal{M}_1$ and $\mathcal{M}_2$, we define the* synchronous product *of $\mathcal{M}_1$ and $\mathcal{M}_2$ as*

$$\mathcal{M}_1 \otimes \mathcal{M}_2 := \mathtt{F}(\mathtt{UF}(\mathcal{M}_1) \otimes \mathtt{UF}(\mathcal{M}_2)).$$

A schematic representation of constructing the synchronous product of two *IMDP*s $\mathcal{M}_1$ and $\mathcal{M}_2$ is given in Figure 6.11. As discussed earlier, the folding mapping from *PA* to *IMDP*, i.e. the red arrow, is not complete and in principle, this transformation may add additional behavior to the resultant system. For each state and action in the resultant *IMDP*, these extra behaviors are essentially a set of probability distributions that do not belong to the convex hull of the enabled probability distributions for that state in the original *PA*. At first sight, these extra behaviors generated from the folding mapping might be seen as an impediment towards showing that $\sim_{(\forall)}$ is a congruence for the synchronous product. Fortunately, as it is shown by the next theorem, these extra probability distributions are in fact *spurious* and do not affect the congruence result.

To this aim and in order to pave the way for establishing the congruence result, we first prove two intermediate results stating that the folding and unfolding mappings preserve bisimilarity on the corresponding codomains.

**Lemma 6.5.** *Given two IMDPs $\mathcal{M}_1$ and $\mathcal{M}_2$, if $\mathcal{M}_1 \sim_{(\forall)} \mathcal{M}_2$, then $\mathtt{UF}(\mathcal{M}_1) \sim^{p}_{aa} \mathtt{UF}(\mathcal{M}_2)$.*

*Proof.* Let $\mathcal{R}$ be the probabilistic bisimulation justifying $\mathcal{M}_1 \sim_{(\forall)} \mathcal{M}_2$; we claim that $\mathcal{R}$ is also a *PA* probabilistic bisimulation for $\mathtt{UF}(\mathcal{M}_1)$ and $\mathtt{UF}(\mathcal{M}_2)$, that is, it justifies $\mathtt{UF}(\mathcal{M}_1) \sim^{p}_{aa} \mathtt{UF}(\mathcal{M}_2)$.

In the following we assume without loss of generality that $s_1 \in S_1$ and $s_2 \in S_2$; the other cases are similar. The fact that $\mathcal{R}$ is an equivalence relation and that for each $(s_1, s_2) \in \mathcal{R}$, $L_1(s_1) = L_2(s_2)$ follow directly by definition of $\sim_{(\forall)}$. Let $(s_1, \mu_1) \in T_1$: by definition of UF, it follows that $\mu_1 \in \mathtt{Ext}(\mathscr{H}^{a_1}_{s_1})$ for some $a_1 \in \mathscr{A}(s_1)$, thus in particular $\mu_1 \in \mathscr{H}^{a_1}_{s_1}$, hence $\mu_1 \in \mathrm{CH}(\bigcup_{a \in \mathscr{A}(s_1)} \mathscr{H}^{a}_{s_1})$. By hypothesis, we have that there exists $\mu_2 \in \mathrm{CH}(\bigcup_{a_2 \in \mathscr{A}(s_2)} \mathscr{H}^{a_2}_{s_2})$ such that $\mu_1 \, \mathcal{L}(\mathcal{R}) \, \mu_2$. Since $\mu_2 \in \mathrm{CH}(\cup_{a_2 \in \mathscr{A}(s_2)} \mathscr{H}^{a_2}_{s_2})$, it follows that there exist a multiset of real values $\{ p_{a_2} \in \mathbb{R}_{\geq 0} \mid a_2 \in \mathscr{A}(s_2) \}$ and a multiset of distributions $\{ \mu_{a_2} \in \mathscr{H}^{a_2}_{s_2} \mid a_2 \in \mathscr{A}(s_2) \}$ such that $\sum_{a_2 \in \mathscr{A}(s_2)} p_{a_2} = 1$ and $\mu_2 = \sum_{a_2 \in \mathscr{A}(s_2)} p_{a_2} \cdot \mu_{a_2}$. For each $a_2 \in \mathscr{A}(s_2)$, since

$\mu_{a_2} \in \mathscr{H}^{a_2}_{s_2}$, it follows that there exist a finite set of indexes $I_{a_2}$, a multiset of real values $\{ p_{a_2,i} \in \mathbb{R}_{\geq 0} \mid i \in I_{a_2} \}$ and a multiset of distributions $\{ \mu_{a_2,i} \in \mathtt{Ext}(\mathscr{H}^{a_2}_{s_2}) \mid i \in I_{a_2} \}$ such that $\sum_{i \in I_{a_2}} p_{a_2,i} = 1$ and $\mu_{a_2} = \sum_{i \in I_{a_2}} p_{a_2,i} \cdot \mu_{a_2,i}$. This means that

$$\mu_2 = \sum_{a_2 \in \mathscr{A}(s_2)} p_{a_2} \cdot \sum_{i \in I_{a_2}} p_{a_2,i} \cdot \mu_{a_2,i} = \sum_{a_2 \in \mathscr{A}(s_2)} \sum_{i \in I_{a_2}} p_{a_2} \cdot p_{a,i} \cdot \mu_{a_2,i}.$$

Since for each $a_2 \in \mathscr{A}(s_2)$ and $i \in I_{a_2}$ we have that $\mu_{a_2,i} \in \mathtt{Ext}(\mathscr{H}^{a_2}_{s_2})$, it follows that $(s_2, \mu_{a_2,i}) \in T_2$, thus we have the combined transition $s_2 \longrightarrow_{\mathrm{c}} \mu_2$ obtained by taking as set of indexes $I = \{ (a_2, i) \mid a_2 \in \mathscr{A}(s_2), i \in I_{a_2} \}$, as multiset of real values $\{ q_{a_2,i} \in \mathbb{R}_{\geq 0} \mid (a_2, i) \in I, q_{a_2,i} = p_{a_2} \cdot p_{a_2,i} \}$, and as multiset of transitions $\{ (s_2, \mu_{a_2,i}) \in T_2 \mid (a_2, i) \in I \}$: in fact, it is immediate to see that

$$\begin{aligned} \sum_{(a_2,i) \in I} q_{a_2,i} &= \sum_{(a_2,i) \in I} p_{a_2} \cdot p_{a_2,i} = \sum_{a_2 \in \mathscr{A}(s_2)} \sum_{i \in I_{a_2}} p_{a_2} \cdot p_{a_2,i} \\ &= \sum_{a_2 \in \mathscr{A}(s_2)} p_{a_2} \cdot \sum_{i \in I_{a_2}} p_{a_2,i} = \sum_{a_2 \in \mathscr{A}(s_2)} p_{a_2} \cdot 1 = 1 \end{aligned}$$

and that

$$\begin{aligned} &\sum_{(a_2,i) \in I} q_{a_2,i} \cdot \mu_{a_2,i} = \sum_{(a_2,i) \in I} p_{a_2} \cdot p_{a_2,i} \cdot \mu_{a_2,i} \\ &= \sum_{a_2 \in \mathscr{A}(s_2)} \sum_{i \in I_{a_2}} p_{a_2} \cdot p_{a_2,i} \cdot \mu_{a_2,i} = \sum_{a_2 \in \mathscr{A}(s_2)} p_{a_2} \cdot \sum_{i \in I_{a_2}} p_{a_2,i} \cdot \mu_{a_2,i} \\ &= \sum_{a_2 \in \mathscr{A}(s_2)} p_{a_2} \cdot \mu_{a_2} = \mu_2. \end{aligned}$$

Moreover, by hypothesis, we have $\mu_1 \, \mathcal{L}(\mathcal{R}) \, \mu_2$, as required. ■

Likewise computation of probabilistic bisimulation for *IMDP*s, we use the standard *partition refinement* approach as a ground procedure to compute $\sim^{p}_{aa}$ for *PA*s. Still the core part of the approach is to decide bisimilarity of a pair of states. For each state in the given *PA*, we construct a convex hull polytope which encodes all possible behaviors that can be taken by a scheduler. Hence, for a given pair of states, we show that verifying if two states are bisimilar can be reduced to comparison of their corresponding convex polytopes with respect to set inclusion. Strictly speaking, for an equivalence relation $\mathcal{R}$ on $S$ and $s \in S$, we denote by $\mathrm{P}^{s}_{\mathcal{R}}$ the polytope of feasible successor distributions over *equivalence classes* of $\mathcal{R}$ with respect to taking a transition in the state $s$. Formally,

$$\mathrm{P}^{s}_{\mathcal{R}} = \mathrm{CH}(\{ [\mu]_{\mathcal{R}} \mid (s, \mu) \in T \}),$$

where, for a given $\mu \in \mathrm{Disc}(S)$, $[\mu]_{\mathcal{R}} \in \mathrm{Disc}(S/\mathcal{R})$ is the probability distribution such that for each $\mathcal{C} \in S/\mathcal{R}$, it is $[\mu]_{\mathcal{R}}(\mathcal{C}) = \sum_{s' \in \mathcal{C}} \mu(s')$.

**Lemma 6.6** (cf. [CS02, Thm. 1]). *Given a PA $\mathfrak{P}$, there exists an equivalence relation $\mathcal{R}$ on $S$ such that for each pair of states $s, t \in S$, it holds that $s \sim^{p}_{aa} t$ if and only if $s \, \mathcal{R} \, t$, $L(s) = L(t)$, and $\mathrm{P}^{s}_{\mathcal{R}} = \mathrm{P}^{t}_{\mathcal{R}}$.*

To simplify the presentation of the proof, we first introduce some notation. Given an equivalence relation $\mathcal{R}$ on $S$, for each distribution $\mu \in \mathrm{Disc}(S)$, let $\bar{\mu} \in \mathrm{Disc}(S/\mathcal{R})$ denote the corresponding distribution $\bar{\mu} = [\mu]_{\mathcal{R}}$, i.e., $\bar{\mu}$ is such that $\bar{\mu}(\mathcal{C}) = \sum_{s' \in \mathcal{C}} \mu(s')$ for each $\mathcal{C} \in S/\mathcal{R}$.

*Proof.* We show the two implications separately. For the implication from left to right, suppose that $s \sim^{p}_{aa} t$; this implies that there exists a probabilistic bisimulation $\mathcal{R}$ such that $s \mathcal{R} t$ and $L(s) = L(t)$. We want to show that $\mathrm{P}^{s}_{\mathcal{R}} = \mathrm{P}^{t}_{\mathcal{R}}$ holds. To this aim, let $\eta \in \mathrm{P}^{s}_{\mathcal{R}}$. By definition of $\mathrm{P}^{s}_{\mathcal{R}}$, it follows that there exist a finite set of indexes $I_\eta$, a multiset of real values $\{ p_{\eta,i} \mid i \in I_\eta \}$ and a multiset of distributions

$$\{ \eta_i \in \mathrm{P}^{s}_{\mathcal{R}} \mid i \in I_\eta, \exists (s, \mu_{s,i}) \in T : \eta_i = [\mu_{s,i}]_{\mathcal{R}} \}$$

such that $\sum_{i \in I_\eta} p_{\eta,i} = 1$ and $\sum_{i \in I_\eta} p_{\eta,i} \cdot \eta_i = \eta$. Since $s \mathcal{R} t$ and $\mathcal{R}$ is a probabilistic bisimulation, it follows that for each $i \in I_\eta$ there exists a combined transition $t \longrightarrow_{\mathrm{c}} \mu_t$ such that $\mu_{s,i}\ \mathcal{L}(\mathcal{R})\ \mu_t$. By definition of combined transition, it follows that there exist a finite set of indexes $I_t$, a set of transitions $\{ (t, \mu_{t,i}) \in T \mid i \in I_t \}$ and a multiset of real values

$$\{ p_{t,i} \in \mathbb{R}_{\geq 0} \mid i \in I_t \}$$

such that $\sum_{i \in I_t} p_{t,i} = 1$ and $\mu_t = \sum_{i \in I_t} p_{t,i} \cdot \mu_{t,i}$. This implies that for each $i \in I_t$, $\bar{\mu}_{t,i} \in \mathrm{P}^{t}_{\mathcal{R}}$. Moreover, since by definition of lifting we have that for each $\mathcal{C} \in S/\mathcal{R}$, $\mu_s(\mathcal{C}) = \mu_t(\mathcal{C})$, it follows immediately that $\bar{\mu}_t = \bar{\mu}_s$, thus we have that $\eta = \bar{\mu}_s = \bar{\mu}_t \in \mathrm{P}^{t}_{\mathcal{R}}$, hence $\mathrm{P}^{s}_{\mathcal{R}} \subseteq \mathrm{P}^{t}_{\mathcal{R}}$. By swapping the roles of $s$ and $t$, we can show in the same way that $\mathrm{P}^{t}_{\mathcal{R}} \subseteq \mathrm{P}^{s}_{\mathcal{R}}$, hence $\mathrm{P}^{s}_{\mathcal{R}} = \mathrm{P}^{t}_{\mathcal{R}}$ as required.

For the implication from right to left, fix an equivalence relation $\mathcal{R}$ on $S$ such that for each $(s,t) \in \mathcal{R}$ it holds that $L(s) = L(t)$ and $\mathrm{P}^{s}_{\mathcal{R}} = \mathrm{P}^{t}_{\mathcal{R}}$; we want to show that $\mathcal{R}$ is a probabilistic bisimulation, i.e., whenever $s \mathcal{R} t$ and $s \longrightarrow \mu_s$ then there exists $t \longrightarrow_{\mathrm{c}} \mu_t$ such that $\mu_s\ \mathcal{L}(\mathcal{R})\ \mu_t$. Let $(s,t) \in \mathcal{R}$; if $\mathrm{P}^{s}_{\mathcal{R}} = \emptyset$, then the step condition of the probabilistic bisimulation is trivially verified since there is no transition $s \longrightarrow \mu_s$ from $s$ that needs to be matched by $t$. Suppose now that $\mathrm{P}^{s}_{\mathcal{R}} \neq \emptyset$ and consider a transition $s \longrightarrow \mu_s$ so that $\bar{\mu}_s \in \mathrm{P}^{s}_{\mathcal{R}}$. By hypothesis, $\bar{\mu}_s \in \mathrm{P}^{s}_{\mathcal{R}} = \mathrm{P}^{t}_{\mathcal{R}}$, thus there exist a finite set of indexes $I$, a multiset of distributions

$$\{ \mu_i \in \mathrm{P}^{t}_{\mathcal{R}} \mid i \in I \}$$

and a multiset of real values

$$\{ p_i \in \mathbb{R}_{\geq 0} \mid i \in I \}$$

such that $\sum_{i \in I} p_i = 1$ and $\sum_{i \in I} p_i \cdot \mu_i = \bar{\mu}_s$. This implies, for each $i \in I$, that there exist a finite set of indexes $J_i$, a multiset of real values

$$\{ p_{i,j} \in \mathbb{R}_{\geq 0} \mid j \in I_i \}$$

and a multiset of distributions

$$\{ \mu_{i,j} \in \mathrm{P}^{t}_{\mathcal{R}} \mid j \in J_i \}$$

such that $\sum_{j \in J_i} p_{i,j} = 1$, $\sum_{j \in J_i} p_{i,j} \cdot \mu_{i,j} = \mu_i$, and for each $j \in J_i$, $\mu_{i,j} = \bar{\mu}_{t,i,j}$ where $(t, \mu_{t,i,j}) \in T$. Consider now the combined transition $t \longrightarrow_{\mathrm{c}} \mu_t$ obtained by taking as set of indexes $J = \{ (i,j) \mid i \in I, j \in J_i \}$, as multiset of real values

$$\{ q_{i,j} \in \mathbb{R}_{\geq 0} \mid (i,j) \in J, q_{i,j} = p_i \cdot p_{i,j} \}$$

and as set of transitions $\{(t, \mu_{t,i,j}) \in T \mid (i,j) \in J\}$: we have that

$$\sum_{(i,j)\in J} q_{i,j} = \sum_{(i,j)\in J} p_i \cdot p_{i,j} = \sum_{i\in I}\sum_{j\in J_i} p_i \cdot p_{i,j}$$
$$= \sum_{i\in I} p_i \cdot \sum_{j\in J_i} p_{i,j} = \sum_{i\in I} p_i \cdot 1 = 1$$

and that

$$\mu_t = \sum_{(i,j)\in I} q_{i,j} \cdot \mu_{t,i,j} = \sum_{(i,j)\in I} p_i \cdot p_{i,j} \cdot \mu_{t,i,j}$$
$$= \sum_{i\in I}\sum_{j\in J_i} p_i \cdot p_{i,j} \cdot \mu_{t,i,j} = \sum_{i\in I} p_i \cdot \sum_{j\in J_i} p_{i,j} \cdot \mu_{t,i,j}.$$

To complete the proof, we have to show that $\mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t$, that is, for each $\mathcal{C} \in S/\mathcal{R}$, $\mu_s(\mathcal{C}) = \mu_t(\mathcal{C})$. Let $\mathcal{C} \in S/\mathcal{R}$: we have that

$$\mu_t(\mathcal{C}) = \sum_{c\in\mathcal{C}} \mu_t(c) = \sum_{c\in\mathcal{C}}\sum_{i\in I} p_i \cdot \sum_{j\in J_i} p_{i,j} \cdot \mu_{t,i,j}(c)$$
$$= \sum_{i\in I} p_i \cdot \sum_{j\in J_i} p_{i,j} \cdot \sum_{c\in\mathcal{C}} \mu_{t,i,j}(c) = \sum_{i\in I} p_i \cdot \sum_{j\in J_i} p_{i,j} \cdot \bar{\mu}_{t,i,j}(\mathcal{C})$$
$$= \sum_{i\in I} p_i \cdot \sum_{j\in J_i} p_{i,j} \cdot \mu_{i,j}(\mathcal{C}) = \sum_{i\in I} p_i \cdot \mu_i(\mathcal{C}) = \bar{\mu}_s(\mathcal{C})$$
$$= \sum_{c\in\mathcal{C}} \mu_s(c) = \mu_s(\mathcal{C}),$$

as required. ■

**Lemma 6.7.** *Given a PA $\mathfrak{P}$ and an equivalence relation $\mathcal{R}$ on $S$, for $n = |S/\mathcal{R}|$, it holds that for each $(s,t) \in \mathcal{R}$, if $\mathrm{P}^s_{\mathcal{R}} = \mathrm{P}^t_{\mathcal{R}}$ then $(\prod_{\mathcal{C}\in S/\mathcal{R}} \mathrm{proj}_{\mathbf{e}_\mathcal{C}} \mathrm{P}^s_{\mathcal{R}}) \cap \Delta_n = (\prod_{\mathcal{C}\in S/\mathcal{R}} \mathrm{proj}_{\mathbf{e}_\mathcal{C}} \mathrm{P}^t_{\mathcal{R}}) \cap \Delta_n$ where*

$$\Delta_n = \{(x_1, \ldots, x_n) \in \mathbb{R}^n_{\geq 0} \mid \sum_{i=1}^{n} x_i = 1\}$$

*is the standard probability simplex in $\mathbb{R}^n$.*

*Proof.* The proof is trivial, since by $\mathrm{P}^s_{\mathcal{R}} = \mathrm{P}^t_{\mathcal{R}}$ it follows that for each $\mathcal{C} \in S/\mathcal{R}$, $\mathrm{proj}_{\mathbf{e}_\mathcal{C}} \mathrm{P}^s_{\mathcal{R}} = \mathrm{proj}_{\mathbf{e}_\mathcal{C}} \mathrm{P}^t_{\mathcal{R}}$. This implies that $\prod_{\mathcal{C}\in S/\mathcal{R}} \mathrm{proj}_{\mathbf{e}_\mathcal{C}} \mathrm{P}^s_{\mathcal{R}} = \prod_{\mathcal{C}\in S/\mathcal{R}} \mathrm{proj}_{\mathbf{e}_\mathcal{C}} \mathrm{P}^t_{\mathcal{R}}$ thus $(\prod_{\mathcal{C}\in S/\mathcal{R}} \mathrm{proj}_{\mathbf{e}_\mathcal{C}} \mathrm{P}^s_{\mathcal{R}}) \cap \Delta_n = (\prod_{\mathcal{C}\in S/\mathcal{R}} \mathrm{proj}_{\mathbf{e}_\mathcal{C}} \mathrm{P}^t_{\mathcal{R}}) \cap \Delta_n$, as required. ■

**Lemma 6.8.** *Given two PAs $\mathfrak{P}_1$ and $\mathfrak{P}_2$, if $\mathfrak{P}_1 \sim^p_{aa} \mathfrak{P}_2$ then $\mathrm{F}(\mathfrak{P}_1) \sim_{(\forall)} \mathrm{F}(\mathfrak{P}_2)$.*

*Proof.* Let $\mathcal{R}$ be the equivalence relation justifying $\mathfrak{P}_1 \sim^{p}_{aa} \mathfrak{P}_2$; we claim that $\mathcal{R}$ is also an *IMDP* probabilistic bisimulation for $\mathtt{F}(\mathfrak{P}_1)$ and $\mathtt{F}(\mathfrak{P}_2)$, that is, it justifies $\mathtt{F}(\mathfrak{P}_1) \sim_{(\forall)} \mathtt{F}(\mathfrak{P}_2)$.

In the following we assume without loss of generality that $s_1 \in S_1$ and $s_2 \in S_2$; the other cases are similar. The fact that $\mathcal{R}$ is an equivalence relation and that for each $(s_1, s_2) \in \mathcal{R}$, $L_1(s_1) = L_2(s_2)$ follow directly by definition of $\sim^{p}_{aa}$. Since $\mathfrak{P}_1 \sim^{p}_{aa} \mathfrak{P}_2$, it follows from Lemma 6.6 that $\mathrm{P}^{s_1}_{\mathcal{R}} = \mathrm{P}^{s_2}_{\mathcal{R}}$. Additionally, it is not difficult to see that for $s_j \in \{s_1, s_2\}$,

$$\mathscr{P}^{s_j}_{\mathcal{R}} = \mathscr{P}^{s_j,f}_{\mathcal{R}} = (\prod_{i=1}^{n} \mathrm{proj}_{\mathbf{e}_i}\, \mathrm{P}^{s_j}_{\mathcal{R}}) \cap \Delta_n$$

where

$$\Delta_n = \{\, (x_1, \ldots, x_n) \in \mathbb{R}^n_{\geq 0} \mid \sum_{i=1}^{n} x_i = 1 \,\}$$

is the standard probability simplex in $\mathbb{R}^n$. By Lemma 6.7, this implies that $\mathscr{P}^{s_1,f}_{\mathcal{R}} = \mathscr{P}^{s_2,f}_{\mathcal{R}}$. Consider now $s_1 \longrightarrow \mu_1$ with $\mu_1 \in \mathscr{H}^{f}_{s_1}$: this implies that $[\mu_1]_{\mathcal{R}} \in \mathscr{P}^{s_1,f}_{\mathcal{R}}$ since $\mathscr{P}^{s_1,f}_{\mathcal{R}} = \mathscr{P}^{s_2,f}_{\mathcal{R}}$, it follows that $[\mu_1]_{\mathcal{R}} \in \mathscr{P}^{s_2,f}_{\mathcal{R}}$ as well, thus there exists $\mu_2 \in \mathscr{H}^{f}_{s_2}$ such that $[\mu_2]_{\mathcal{R}} = [\mu_1]_{\mathcal{R}}$. By definition of $[\,\cdot\,]_{\mathcal{R}}$, we have that for each $\mathcal{C} \in S/\mathcal{R}$,

$$\sum_{s \in \mathcal{C}} \mu_i(s) = [\mu_i]_{\mathcal{R}}$$

for $i \in \{1, 2\}$, thus $[\mu_2]_{\mathcal{R}} = [\mu_1]_{\mathcal{R}}$ implies that for each $\mathcal{C} \in S/\mathcal{R}$,

$$\sum_{s \in \mathcal{C}} \mu_1(s) = \sum_{s \in \mathcal{C}} \mu_2(s),$$

i.e., $\mu_1\ \mathcal{L}(\mathcal{R})\ \mu_2$. This means that we have found $s_2 \longrightarrow \mu_2$ with $\mu_1\ \mathcal{L}(\mathcal{R})\ \mu_2$, as required. ∎

By using Lemmas 6.5 and 6.8 and Proposition 6.4, we can now show that $\sim_{(\forall)}$ is preserved by the synchronous product operator introduced in Definition 6.15.

**Theorem 6.9.** *Given three IMDPs $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$, if $\mathcal{M}_1 \sim_{(\forall)} \mathcal{M}_2$, then $\mathcal{M}_1 \otimes \mathcal{M}_3 \sim_{(\forall)} \mathcal{M}_2 \otimes \mathcal{M}_3$.*

*Proof.* Assume that $\mathcal{M}_1 \sim_{(\forall)} \mathcal{M}_2$. By Lemma 6.5, it follows that $\mathtt{UF}(\mathcal{M}_1) \sim^{p}_{aa} \mathtt{UF}(\mathcal{M}_2)$, thus, by Proposition 6.4, we have that

$$\mathtt{UF}(\mathcal{M}_1) \otimes \mathtt{UF}(\mathcal{M}_3) \sim^{p}_{aa} \mathtt{UF}(\mathcal{M}_2) \otimes \mathtt{UF}(\mathcal{M}_3).$$

Lemma 6.8 now implies that

$$\mathtt{F}(\mathtt{UF}(\mathcal{M}_1) \otimes \mathtt{UF}(\mathcal{M}_3)) \sim_{(\forall)} \mathtt{F}(\mathtt{UF}(\mathcal{M}_2) \otimes \mathtt{UF}(\mathcal{M}_3)),$$

that is, $\mathcal{M}_1 \otimes \mathcal{M}_3 \sim_{(\forall)} \mathcal{M}_2 \otimes \mathcal{M}_3$, as required. ∎

### 6.3.4 Interleaved approach

In the previous section, we have considered the parallel composition via synchronous production, which is working by the definition of a folding collapsing all labels to a single transition. Here we consider the other extreme of the parallel composition: interleaving only.

**Definition 6.16 (*IMDP*s interleaving parallel composition).** *Given two IMDPs $\mathcal{M}_l$ and $\mathcal{M}_r$, we define the* interleaved composition *of $\mathcal{M}_l$ and $\mathcal{M}_r$, denoted by $\mathcal{M}_l \between \mathcal{M}_r$, as the IMDP $\mathcal{M} = (S, \bar{s}, \mathscr{A}, \mathtt{AP}, L, I)$ where*

- $S = S_l \times S_r$;
- $\bar{s} = (\bar{s}_l, \bar{s}_r)$;
- $\mathscr{A} = (\mathscr{A}_l \times \{l\}) \cup (\mathscr{A}_r \times \{r\})$;
- $\mathtt{AP} = \mathtt{AP}_l \cup \mathtt{AP}_r$;
- *for each $(s_l, s_r) \in S$, $L(s_l, s_r) = L_l(s_l) \cup L_r(s_r)$; and*
- $I((s_l, s_r), (a, i), (t_l, t_r)) = \begin{cases} I_l(s_l, a, t_l) & \text{if } i = l \text{ and } t_r = s_r, \\ I_r(s_r, a, t_r) & \text{if } i = r \text{ and } t_l = s_l, \\ [0, 0] & \text{otherwise.} \end{cases}$

**Theorem 6.10.** *Given three IMDPs $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$, if $\mathcal{M}_1 \sim_{(\forall)} \mathcal{M}_2$, then $\mathcal{M}_1 \between \mathcal{M}_3 \sim_{(\forall)} \mathcal{M}_2 \between \mathcal{M}_3$.*

*Proof.* Let $\mathcal{R}$ be the probabilistic bisimulation justifying $\mathcal{M}_1 \sim_{(\forall)} \mathcal{M}_2$ and define $\mathcal{R}' = \mathcal{R} \times \mathcal{I}_{s_3}$; we claim that $\mathcal{R}'$ is a probabilistic bisimulation between $\mathcal{M}_1 \between \mathcal{M}_3$ and $\mathcal{M}_2 \between \mathcal{M}_3$. The fact that $\mathcal{R}'$ is an equivalence relation follows trivially by its definition and the fact that $\mathcal{R}$ is an equivalence relation. The fact that $((\bar{s}_1, \bar{s}_3), (\bar{s}_2, \bar{s}_3))$ follows immediately by the hypothesis that $(\bar{s}_1, \bar{s}_2) \in \mathcal{R}$ and $(\bar{s}_3, \bar{s}_3) \in \mathcal{I}_{s_3}$.

Let $((s_1, s_3), (s_2, s_3)) \in \mathcal{R}'$. Assume, without loss of generality, that $s_1 \in S_1$ and $s_2 \in S_2$; the other cases are similar. The fact that $L_{1,3}(s_1, s_3) = L_{2,3}(s_2, s_3)$ is straightforward, since by definition of interleaved composition and the hypothesis that $s_1 \mathcal{R} s_2$, we have that

$$L_{1,3}(s_1, s_3) = L_1(s_1) \cup L_3(s_3) = L_2(s_2) \cup L_3(s_3) = L_{2,3}(s_2, s_3),$$

as required. Consider now a transition $(s_1, s_3) \longrightarrow \mu_{1,3}$. By definition, we have that $\mu_{1,3} \in \mathrm{CH}(\bigcup_{(a,i) \in \mathscr{A}(s_1,s_3)} \mathscr{H}^{(a,i)}_{(s_1,s_3)})$. This implies that there exist a multiset of distributions

$$\{ \mu_{a,i} \in \mathscr{H}^{(a,i)}_{(s_1,s_3)} \mid (a, i) \in \mathscr{A}(s_1, s_3) \}$$

and a multiset of real values

$$\{ p_{a,i} \in \mathbb{R}_{\geq 0} \mid (a, i) \in \mathscr{A}(s_1, s_3) \}$$

such that $\sum_{(a,i) \in \mathscr{A}(s_1,s_3)} p_{a,i} = 1$ and $\sum_{(a,i) \in \mathscr{A}(s_1,s_3)} p_{a,i} \cdot \mu_{a,i} = \mu_{1,3}$. Consider an action $(a, i) \in \mathscr{A}(s_1, s_3)$: by definition of interleaved composition, it is either of the form $(a, l) \in \mathscr{A}_l \times \{l\}$, or of the form $(a, r) \in \mathscr{A}_r \times \{r\}$. Consider the two cases separately:

**Case** $(a, i) \in \mathscr{A}_l \times \{l\}$**:** this means that $\mu_{a,i}$ is actually the distribution $\mu_{a,i} = \mu_a \times \delta_{s_3}$ where $\mu_a \in \mathscr{H}^a_{s_1}$ is such that for each $s_1' \in S_1$, $\mu_a(s_1') = \mu_{a,i}(s_1', s_3)$, thus $s_1 \longrightarrow \mu_a$. Since by hypothesis $(s_1, s_2) \in \mathcal{R}$ and $\mathcal{R}$ is a probabilistic bisimulation, there exists $\mu_{a,2} \in \mathrm{CH}(\bigcup_{b \in \mathscr{A}(s_2)} \mathscr{H}^b_{s_2})$ such that $\mu_a \; \mathcal{L}(\mathcal{R}) \; \mu_{a,2}$. This implies that there exist a multiset of distributions $\{ \mu_{a,2,b} \in \mathscr{H}^b_{s_2} \mid b \in \mathscr{A}(s_2) \}$ and a multiset of real values $\{ p_{a,2,b} \mid b \in \mathscr{A}(s_2) \}$ such that $\sum_{b \in \mathscr{A}(s_2)} p_{a,2,b} = 1$, $\sum_{b \in \mathscr{A}(s_2)} p_{a,2,b} \cdot \mu_{a,2,b} = \mu_{a,2}$,

| Model | $\lvert S_i\rvert$ | $\lvert I_i\rvert$ | $S/L$ | $t_\sim$ (s) | $\lvert S_\sim\rvert$ | $\lvert I_\sim\rvert$ |
|---|---|---|---|---|---|---|
| Consensus-Shared-Coin-3 | 5 216 | 13 380 | 2 | 0 | 787 | 1 770 |
| Consensus-Shared-Coin-4 | 43 136 | 144 352 | 2 | 2 | 2 189 | 5 621 |
| Consensus-Shared-Coin-5 | 327 936 | 1 363 120 | 2 | 23 | 5 025 | 14 192 |
| Consensus-Shared-Coin-6 | 2 376 448 | 11 835 456 | 2 | 219 | 10 173 | 30 861 |
| Crowds-5-10 | 111 294 | 261 444 | 2 | 1 | 107 | 153 |
| Crowds-5-20 | 2 061 951 | 7 374 951 | 2 | 17 | 107 | 153 |
| Crowds-5-30 | 12 816 233 | 61 511 033 | 2 | 116 | 107 | 153 |
| Crowds-5-40 | 44 045 030 | 266 812 421 | 2 | 464 | 125 | 198 |
| Mutual-Exclusion-PZ-3 | 3 008 | 10 868 | 2 | 0 | 1 123 | 3 939 |
| Mutual-Exclusion-PZ-4 | 48 128 | 231 040 | 2 | 0 | 7 319 | 32 630 |
| Mutual-Exclusion-PZ-5 | 770 048 | 4 611 072 | 2 | 7 | 32 053 | 168 151 |
| Mutual-Exclusion-PZ-6 | 3 377 344 | 25 470 144 | 2 | 98 | 109 986 | 649 360 |
| Dining-Phils-LR-nofair-4 | 9 440 | 40 120 | 4 | 0 | 1 232 | 5 037 |
| Dining-Phils-LR-nofair-5 | 93 068 | 494 420 | 4 | 1 | 9 408 | 49 467 |
| Dining-Phils-LR-nofair-6 | 917 424 | 5 848 524 | 4 | 14 | 76 925 | 487 620 |
| Dining-Phils-LR-nofair-7 | 9 043 420 | 67 259 808 | 4 | 173 | 646 928 | 4 804 695 |

Table 6.3: Experimental evaluation of the bisimulation computation

and $\mu_a \times \delta_{s_3}\ \mathcal{L}(\mathcal{R}')\ \mu_{a,2} \times \delta_{s_3}$. This means that for each $b \in \mathscr{A}(s_2)$, we have that $\mu_{a,2,b} \times \delta_{s_3} \in \mathscr{H}^{(b,l)}_{(s_2,s_3)}$, thus by taking $\mu_{a,2,3} = \sum_{b \in \mathscr{A}(s_2)} p_{a,2,b} \cdot (\mu_{a,2,b} \times \delta_{s_3})$ we have that $(s_2, s_3) \longrightarrow \mu_{a,2,3}$ and $\mu_{a,i}\ \mathcal{L}(\mathcal{R}')\ \mu_{a,2,3}$.

**Case** $(a,i) \in \mathscr{A}_r \times \{r\}$**:** this means that $\mu_{a,i}$ is actually the distribution $\mu_{a,i} = \delta_{s_1} \times \mu_a$ where $\mu_a \in \mathscr{H}^a_{s_3}$ is such that for each $s_3' \in S_3$, $\mu_a(s_3') = \mu_{a,i}(s_1, s_3')$, thus $s_3 \longrightarrow \mu_a$. This implies trivially that $(s_2, s_3) \longrightarrow \mu_{a,2,3}$ where $\mu_{a,2,3} = \delta_{s_2} \times \mu_a$ and $\mu_{a,i}\ \mathcal{L}(\mathcal{R}')\ \mu_{a,2,3}$.

From the analysis of the two cases, we have that for each $(a,i) \in \mathscr{A}(s_1, s_3)$, there exists a transition $(s_2, s_3) \longrightarrow \mu_{a,2,3}$ such that $\mu_{a,i}\ \mathcal{L}(\mathcal{R}')\ \mu_{a,2,3}$. This implies that

$$\mu_{2,3} = \sum_{(a,i) \in \mathscr{A}(s_1,s_3)} p_{a,i} \cdot \mu_{a,2,3} \in \mathrm{CH}\Big(\bigcup_{(b,j) \in \mathscr{A}(s_2,s_3)} \mathscr{H}^{(b,j)}_{(s_2,s_3)}\Big)$$

and $\mu_{1,3}\ \mathcal{L}(\mathcal{R}')\ \mu_{2,3}$, as required. ∎

## 6.4 Case Studies

We have written a prototypical implementation for computing the bisimulation presented in this chapter. Our tool reads a model specification in the input language of the probabilistic model checker PRISM [KNP11] (extended to support also intervals in the transitions), and constructs an explicit-state representation of the state space. Afterwards, it computes the quotient using algorithm depicted in Figure 6.5.

Table 6.3 shows the performance of our prototype on a number of case studies taken from the PRISM website [PRI], where we have relaxed some of the probabilistic choices to

intervals. The machine we used for the experiments is a 3.6 GHz Intel Core i7-4790 with 16 GB 1600 MHz DDR3 RAM of which 12GB were assigned to the tool. Despite using an explicit representation for the model, the prototype is able to manage cases studies in the order of millions of states and transitions (columns "Model", "$|S_i|$", and "$|I_i|$"). The time in seconds required to compute the bisimulation relation and the corresponding quotient *IMDP*, shown in columns "$t_\sim$", "$|S_\sim|$", and "$|I_\sim|$", is much less than the time expected from the theoretical analysis of the algorithm: this is motivated by the fact that we have implemented optimizations, such as caching equivalent LP problems, which improve the runtime of our algorithm in practice. Because of this, we never had to solve more than 30 LP problems in a single tool run, thereby avoiding the potentially costly solution of LP problems from becoming a bottleneck.

## 6.5 Concluding Remarks

In this chapter, we have studied the probabilistic bisimulation problem for *IMDP*s in order to speed up the run time of the PCTL model checking algorithms that often suffer from the state space explosion. *IMDP*s include two sources of nondeterminism for which we have considered the cooperative resolution in a dynamic setting.

We presented a fixed-parameter tractable decision algorithm to exactly decide the defined probabilistic bisimulation for *IMDP*s and showed that the worst case time complexity is **coNP**-complete. Afterwards, by exploiting techniques from robust optimisation setting, we showed that probabilistic bisimulation can be approximated in polynomial time. We have implemented our approach and demonstrate its effectiveness on several case studies.

Finally, we have provided a framework for compositional verification of complex systems with interval uncertainty. In particular, we have established the compositional reasoning by defining the parallel operators for *IMDP* models which preserve our notion of probabilistic bisimulation.

**Related work** Various probabilistic modelling formalisms with uncertain transitions are studied in the literature. Interval Markov chains [JL91, KU02] or Abstract Markov chains [FLW06] extend standard discrete-time Markov chains (*MC*) with interval uncertainties and thus do not feature the nondeterministic choice of transitions. Uncertain *MDP*s [NG05, WTM12, PLSVS13] allow more general sets of distributions to be associated with each transition, not only those described by intervals. Usually, they restrict to *rectangular uncertainty sets* requiring that the uncertainty is linear and independent for any two transitions of any two states. Our general algorithm working with polytopes can be easily adapted to this setting. Parametric *MDP*s [HHZ11a] to the contrary allow such dependencies as every probability is described as a rational function of a finite set of global parameters.

From the side of view of compositional specification, Interval Markov chains [JL91] and Abstract probabilistic automata [DKL$^+$11a, DKL$^+$11b] serve as specification theories for *MC* and *PA* featuring satisfaction relation, and various refinement relations. In order to be closed under parallel composition, Abstract *PA* allow general polynomial constraints on probabilities instead of interval bounds. Since for Interval *MC* it is not possible to explicitly construct parallel composition, the problem whether there is a common implemen-

tation of a set of Interval Markov chains is addressed instead [DLL+11]. To the contrary, interval bounds on *rates* of outgoing transitions work well with parallel composition in the continuous-time setting of Abstract interactive Markov chains [KKN09]. The reason is that unlike probabilities, the rates do not need to sum up to 1. A different way [Yi94] to successfully define parallel composition for interval models is to separate synchronising transitions from the transitions with uncertain probabilities.

We are not aware of any existing compositional bisimulation minimization for uncertain or parametric probabilistic models. Among similar concepts studied in the literature are simulation [Yi94] and refinement [JL91, DLL+11, DKL+11a] relations for previously mentioned models.

Many new verification algorithms for interval models appeared in last few years. Reachability and expected total reward is addressed for Interval *MC* [CHK13] as well as Interval *MDP* [WK08]. PCTL model checking and LTL model checking are studied for Interval *MC* [CSH08, CHK13, BLW13] and also for *IMDP* [PLSVS13, WTM12]. Among other technical tools, all these approaches make use of (robust) dynamic programming relying on the fact that transition probability distributions are resolved dynamically. For the static resolution of distributions, adaptive discretisation technique for PCTL parameter synthesis is given in [HHZ11a]. Uncertain models are also widely studied in the control community [GLD00, NG05, WK08], mainly interested in maximal expected finite-horizon reward or maximal expected discounted reward.

Finally, as regards the application of Robust Optimization in Probabilistic Verification community, to the best of our knowledge, we are not aware of any work in the literature. Therefore, our current approach is novel in this matter. On the other hand, the aforementioned theory has been adapted and applied successfully in control theory realm. For instance, Abate and El Ghaoui [AEG04] developed a robust modal predictive control using two-stage robust optimization.

CHAPTER 7

# Compositional Minimization for Optimal Control of Interval MDPs

In model checking PCTL properties of interval Markov decision processes, it is natural to assume that the scheduler and nature are resolved in a cooperative way. In Chapter 6 we discussed in detail a compositional specification theory based on the cooperative interpretation of nondeterminisms. In this chapter, we focus on the *competitive* semantics in interpreting the two sources of nondeterminisms in *IMDP*s. The competitive semantics has *different* interpretation in different applications. In control synthesis for systems with uncertain probabilities [WTM12] the transitions correspond to various control actions. We search for a choice of transitions that is *optimal* against an adversarial choice of probability distributions satisfying the interval bounds. In parameter synthesis for parallel systems [HHZ11a] the transition probabilities are underspecified to allow freedom in implementation of such a model. We search for a choice of probability distributions that is optimal for adversarial choice of transitions (again stemming from the possible interleaving).

We show that competitive interpretation of nondeterminisms yields two variants of alternating probabilistic bisimulations: one for models where the two nondeterminisms are resolved based on the *controller synthesis* semantics, another for models where they are resolved based on the *parameter synthesis* semantics. We prove that these two variants of alternating probabilistic bisimulations coincide and accordingly provide an efficient compositional minimization theory to reduce the size of the *IMDP*s with respect to the competitive semantics while preserving the probabilistic CTL properties it satisfies. We finally show promising results on a variety of case studies, obtained by prototypical implementations of all algorithms.

The material presented in this chapter is an extended version of the results reported in [HHK14, HTH+17].

**Organization of the chapter.** In Section 7.1, we give the definitions of alternating probabilistic bisimulation for *IMDP*s and discuss their properties. A polynomial time decision algorithm to decide alternating probabilistic bisimulation for *IMDP*s and also compositional reasoning are discussed in Sections 7.2 and 7.3, respectively. We present the practical

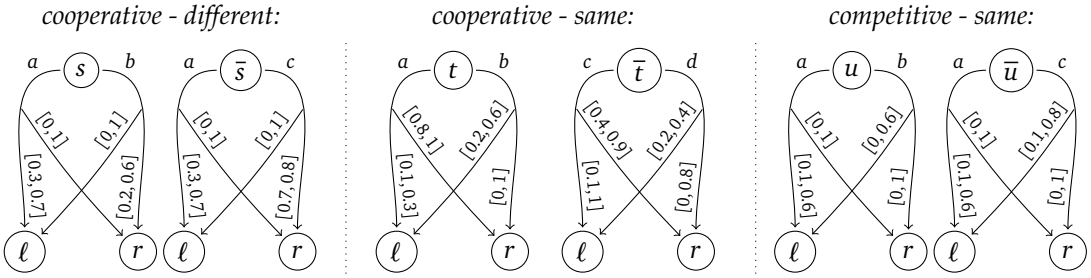
effectiveness of our proposed approaches by applying them on several case studies using a prototypical tool in Section 7.4. Finally, in Section 7.5 we conclude the chapter.

## 7.1 Alternating Probabilistic Bisimulation Relations for *IMDP*s

In the previous chapter, we discussed how cooperative interpretation of nondeterminisms yields a neat notion of probabilistic bisimulation to speed up the run time of model checking algorithms for PCTL properties of *IMDP*s. We now address the *competitive* way of resolving the sources of nondeterminisms and provide other notions of probabilistic bisimulation that can be leveraged in both controller synthesis and parameter synthesis of *IMDP* models. In the competitive semantics, the controller and nature are playing in a game *against* each other; therefore, they are resolved *competitively*. Likewise the previous chapter, we focus on the dynamic approach in resolving the stochastic nondeterminism.

We first illustrate how the cooperative and the competitive resolution of nondeterminism result in different behavioural equivalences.

**Example 7.1.** *Consider the three pair of states below.*




*As regards the cooperative nondeterminism, $s$ has not the same behaviour as $\bar{s}$ since $\bar{s}$ can move to $r$ with probability $0.8$ by choosing $c$ and $(\ell \mapsto 0.2, r \mapsto 0.8)$, which $s$ cannot simulate. So far the equivalence might seem easy to check. However, note that $t$ has the same behaviour as $\bar{t}$ even though the interval bounds for the transitions quite differ. Indeed, the sets of distributions satisfying the interval constraints are the same for $t$ and $\bar{t}$.*

*As regards the competitive nondeterminism, observe that $u$ and $\bar{u}$ have also the same behaviour. Indeed, the $a$ transitions coincide and both $b$ and $c$ offer a wider choice of probability distributions than $a$. If the most adversarial choice of the distribution scheduler lies in the difference $[b] \setminus [a]$ of the distributions offered by $b$ and $a$, the transition scheduler then never chooses $b$; hence $a$ in $\bar{u}$ can simulate both $a$ and $b$ in $u$. In the other direction it is similar and $u$ and $\bar{u}$ have the same behaviour although $[b] \neq [c]$.* ♦

As we already discussed, there are applications where it is natural to interpret the two sources of nondeterminism in a competitive way.

**Controller synthesis under uncertainty.** In this setting we search for a scheduler $\sigma$ such that for any nature $\pi$, a fixed PCTL property $\varphi$ is satisfied. This corresponds to the satisfaction relation $\models_{(\exists\sigma\forall)}$ defined as in Table 7.1.

| | | |
|---|---|---|
| $s \models_{(\exists\sigma\forall)} \mathit{true}$ | | |
| $s \models_{(\exists\sigma\forall)} x$ | iff | $x \in L(s)$ |
| $s \models_{(\exists\sigma\forall)} \neg\varphi$ | iff | $s \not\models_{(\exists\sigma\forall)} \varphi$ |
| $s \models_{(\exists\sigma\forall)} \varphi_1 \wedge \varphi_2$ | iff | $s \models_{(\exists\sigma\forall)} \varphi_1 \wedge s \models_{(\exists\sigma\forall)} \varphi_2$ |
| $s \models_{(\exists\sigma\forall)} \mathbf{P}_{\bowtie p}(\psi)$ | iff | $\exists\sigma \in \Sigma\ \forall\pi \in \Pi : \mathbf{Pr}_s^{\sigma,\pi}\left[\models_{(\exists\sigma\forall)} \psi\right] \bowtie p$ |
| $\xi \models_{(\exists\sigma\forall)} \mathbf{X}\varphi$ | iff | $s_2 \models \varphi$ |
| $\xi \models_{(\exists\sigma\forall)} \varphi_1 \mathbf{U}^{\leq k} \varphi_2$ | iff | there exists $i \leq k$ such that $s_i \models_{(\exists\sigma\forall)} \varphi_2$ and $s_j \models_{(\exists\sigma\forall)} \varphi_1$ for every $1 \leq j < i$ |
| $\xi \models_{(\exists\sigma\forall)} \varphi_1 \mathbf{U} \varphi_2$ | iff | there exists $k \in \mathbb{N}$ such that $\xi \models_{(\exists\sigma\forall)} \varphi_1 \mathbf{U}^{\leq k} \varphi_2$ |

Table 7.1: PCTL semantics for controller synthesis of *IMDP*s

As regards bisimulation, the competitive setting is not a common one. We define a bisimulation similar to the alternating bisimulation of [AHK98] applied to non-stochastic two-player games. For a decision $\rho \in \mathrm{Disc}(\mathscr{A})$ of $\Sigma$, let us denote by $s \xrightarrow{\rho} \mu$ that $\mu$ is a possible successor distribution, i.e. there are decisions $\mu_a$ of $\Pi$ for each $a$ such that $\mu = \sum_{a \in \mathscr{A}} \rho(a) \cdot \mu_a$.

**Definition 7.1 (Alternating probabilistic $(\exists\sigma\forall)$-bisimulation).** *Given an IMDP $\mathcal{M}$, let $\mathcal{R} \subseteq S \times S$ be an equivalence relation. We say that $\mathcal{R}$ is an alternating probabilistic $(\exists\sigma\forall)$-bisimulation if for any $(s,t) \in \mathcal{R}$ we have that $L(s) = L(t)$ and*

$$\begin{aligned} \textit{for each} \quad & \rho_s \in \mathrm{Disc}(\mathscr{A}(s)) \\ \textit{there is} \quad & \rho_t \in \mathrm{Disc}(\mathscr{A}(t)) \\ \textit{such that for each} \quad & t \xrightarrow{\rho_t} \mu_t \\ \textit{there is} \quad & s \xrightarrow{\rho_s} \mu_s \quad \textit{such that} \quad \mu_s \, \mathcal{L}(\mathcal{R}) \, \mu_t . \end{aligned}$$

*Furthermore, we write $s \sim_{(\exists\sigma\forall)} t$ if there is an alternating probabilistic $(\exists\sigma\forall)$-bisimulation $\mathcal{R}$ such that $(s,t) \in \mathcal{R}$.*

The exact alternation of quantifiers might be counter-intuitive at first sight. Note that it exactly corresponds to the situation in non-stochastic games [AHK98]. The defined bisimulation preserves the PCTL logic with $\models_{(\exists\sigma\forall)}$.

**Theorem 7.1 (Soundness of $\sim_{(\exists\sigma\forall)}$ with respect to the PCTL properties).** *For states $s \sim_{(\exists\sigma\forall)} t$ and any PCTL formula $\varphi$, we have $s \models_{(\exists\sigma\forall)} \varphi$ if and only if $t \models_{(\exists\sigma\forall)} \varphi$.*

*Proof.* We use structural induction on the syntax of PCTL state formula $\varphi$ and PCTL path formula $\psi$. That is, we need to prove the following two results simultaneously:

1. $s \sim_{(\exists\sigma\forall)} t$ implies that $s \models_{(\exists\sigma\forall)} \varphi$ if and only if $t \models_{(\exists\sigma\forall)} \varphi$ for any state formula $\varphi$
2. $\omega_1 \sim_{(\exists\sigma\forall)} \omega_2$ implies that $\omega_1 \models_{(\exists\sigma\forall)} \psi$ if and only if $\omega_2 \models_{(\exists\sigma\forall)} \psi$ for any path formula $\psi$.

| | | |
|---|---|---|
| $s \models_{(\exists\pi\forall)} \mathit{true}$ | | |
| $s \models_{(\exists\pi\forall)} x$ | iff | $x \in L(s)$ |
| $s \models_{(\exists\pi\forall)} \neg\varphi$ | iff | $s \not\models_{(\exists\pi\forall)} \varphi$ |
| $s \models_{(\exists\pi\forall)} \varphi_1 \wedge \varphi_2$ | iff | $s \models_{(\exists\pi\forall)} \varphi_1 \wedge s \models_{(\exists\pi\forall)} \varphi_2$ |
| $s \models_{(\exists\pi\forall)} \mathbf{P}_{\bowtie p}(\psi)$ | iff | $\exists\pi \in \Pi\ \forall\sigma \in \Sigma : \mathbf{Pr}_s^{\sigma,\pi}\left[\models_{(\exists\pi\forall)} \psi\right] \bowtie p$ |
| $\xi \models_{(\exists\pi\forall)} \mathbf{X}\varphi$ | iff | $s_2 \models \varphi$ |
| $\xi \models_{(\exists\pi\forall)} \varphi_1 \mathbf{U}^{\leq k} \varphi_2$ | iff | there exists $i \leq k$ such that $s_i \models_{(\exists\sigma\forall)} \varphi_2$ and $s_j \models_{(\exists\pi\forall)} \varphi_1$ for every $1 \leq j < i$ |
| $\xi \models_{(\exists\pi\forall)} \varphi_1 \mathbf{U} \varphi_2$ | iff | there exists $k \in \mathbb{N}$ such that $\xi \models_{(\exists\pi\forall)} \varphi_1 \mathbf{U}^{\leq k} \varphi_2$ |

Table 7.2: PCTL semantics for parameter synthesis of *IMDP*s

We consider the nontrivial part that is when $\varphi = \mathbf{P}_{\bowtie p}(\psi)$ where $\bowtie = \leq$ and $\psi = \varphi_1 \mathbf{U} \varphi_2$; because the other cases are similar. Assume $s \models_{(\exists\sigma\forall)} \varphi$, we need to show that $t \models_{(\exists\sigma\forall)} \varphi$. To drive a contradiction, assume $t \models_{(\exists\sigma\forall)} \neg\varphi$. Therefore, for each $\sigma \in \Sigma$ there exists $\pi \in \Pi$, $\mathbf{Pr}_t^{\sigma,\pi}\{\omega | \omega \models \psi\} > p$. By induction hypothesis $\mathit{Sat}(\varphi_1)$ and $\mathit{Sat}(\varphi_2)$ are $\models_{(\exists\sigma\forall)}$ closed and they are indeed union of equivalence classes induced by $\models_{(\exists\sigma\forall)}$. It is not difficult to see that the set $\{\omega | \omega \models \psi\}$ is $\models_{(\exists\sigma\forall)}$ closed and therefore, $\mathbf{Pr}_s^{\sigma,\pi}\{\omega | \omega \models \psi\} = \mathbf{Pr}_t^{\sigma,\pi}\{\omega | \omega \models \psi\} > p$. In other words, for each $\sigma \in \Sigma$ there exists $\pi \in \Pi$ such that $s \models_{(\exists\sigma\forall)} \neg\varphi$ which leads to a contradiction. ■

Similarly to Corollary 6.1, we could define a satisfaction relation with the alternation $\forall\sigma \in \Sigma\ \exists\pi \in \Pi$ that is then preserved by the same bisimulation $\sim_{(\exists\sigma\forall)}$. However, we see no natural application thereof.

**Parameter synthesis in parallel systems.** Another variant of the alternating probabilistic bisimulation is defined based on the *parameter synthesis* semantics. In the setting of parameter synthesis in parallel systems, we search for a resolution $\pi$ of the underspecified probabilities such that for any scheduler $\sigma$ resolving the interleaving nondeterminism, a fixed property $\varphi$ is satisfied. This corresponds to the satisfaction relation $\models_{(\exists\pi\forall)}$ defined as in Table 7.2.

This yields a definition of bisimulation similar to Definition 7.1. For a choice $(\mu_a)_{a \in \mathscr{A}}$ of underspecified probabilities, let us denote by $s \xrightarrow{(\mu_a)} \mu$ that $\mu$ is a possible successor distribution, i.e. there is a decision $\rho$ of $\Sigma$ such that $\mu = \sum_{a \in \mathscr{A}} \rho(a) \cdot \mu_a$.

**Definition 7.2 (Alternating probabilistic $(\exists\pi\forall)$-bisimulation).** *Given an IMDP $\mathcal{M}$, let $\mathcal{R} \subseteq S \times S$ be a symmetric relation. We say that $\mathcal{R}$ is an* alternating probabilistic $(\exists\pi\forall)$-bisimulation *if for any $(s,t) \in \mathcal{R}$ we have that $L(s) = L(t)$ and*

$$\begin{aligned} \textit{for each}\ \ & (\mu_a)_{a \in \mathscr{A}(s)}\ \ \textit{where}\ s \xrightarrow{a} \mu_a\ \textit{for each}\ a \in \mathscr{A}(s) \\ \textit{there is}\ \ & (\nu_a)_{a \in \mathscr{A}(t)}\ \ \textit{where}\ t \xrightarrow{a} \nu_a\ \textit{for each}\ a \in \mathscr{A}(t) \\ \textit{such that for each}\ \ & t \xrightarrow{(\nu_a)} \mu_t \end{aligned}$$

$$\text{there is } \; s \xrightarrow{(\mu_a)} \mu_s \;\; \text{such that } \; \mu_s \; \mathcal{L}(\mathcal{R}) \; \mu_t.$$

*Furthermore, we write $s \sim_{(\exists\pi\forall)} t$ if there is an alternating probabilistic $(\exists\pi\forall)$-bisimulation $\mathcal{R}$ such that $(s,t) \in \mathcal{R}$.*

The fact that this bisimulation preserves $\models_{(\exists\pi\forall)}$ can be proven analogously to Theorem 7.1.

**Theorem 7.2 (Soundness of $\sim_{(\exists\pi\forall)}$ with respect to the PCTL properties).** *For states $s \sim_{(\exists\pi\forall)} t$ and any PCTL formula $\varphi$, we have $s \models_{(\exists\pi\forall)} \varphi$ if and only if $t \models_{(\exists\pi\forall)} \varphi$.*

*Proof.* The proof is similar to the proof of Theorem 7.1. ∎

As a final result of this section, we show that these two bisimulations coincide. Formally,

**Theorem 7.3.** *We have $\sim_{(\exists\sigma\forall)} \; = \; \sim_{(\exists\pi\forall)}$.*

*Proof.* We only prove that $\sim_{(\exists\sigma\forall)} \subseteq \sim_{(\exists\pi\forall)}$. The other way around is proved similarly. Let $\mathcal{R} \subseteq S \times S$ be an equivalence relation that is alternating probabilistic $(\exists\sigma\forall)$-bisimulation. Let $(s,t) \in \mathcal{R}$ and let $\rho_s \in \text{Disc}(\mathscr{A})$. By definition of $\mathcal{R}$, $L(s) = L(t)$. Moreover, for the given choice $\rho_s$, there exists $\rho_t \in \text{Disc}(\mathscr{A})$ such that for each $t \xrightarrow{\rho_t} \nu$ there exists a transition $s \xrightarrow{\rho_s} \mu$ such that $\mu(\mathcal{C}) = \nu(\mathcal{C})$ for each equivalence class $\mathcal{C} \in S/\mathcal{R}$. Consider a transition $t \xrightarrow{\rho_t} \nu$. By definition of $(\exists\sigma\forall)$-bisimulation, there exists a choice $(\nu_a)_{a\in\mathscr{A}}$ of nature such that $\nu = \sum_{a\in\mathscr{A}} \rho_t(a)\,\nu_a$ and also there exists a probabilistic transition $s \xrightarrow{\rho_s} \mu$. Let us assume $(\mu_a)_{a\in\mathscr{A}}$ are the choice of nature such that $\mu = \sum_{a\in\mathscr{A}} \rho_s(a) \cdot \mu_a$. Define $(\eta_a)_{a\in\mathscr{A}} = (\nu_a)_{a\in\mathscr{A}}$. For such realization of uncertainty $(\eta_a)_{a\in\mathscr{A}}$, let $t \xrightarrow{\eta_a} \eta$. Since the realized uncertainty $(\eta_a)_{a\in\mathscr{A}}$ was an outcome of the choice $\rho_t$ of the controller, therefore, $\eta = \sum_{a\in\mathscr{A}} \rho_t(a) \cdot \eta_a$. We define $(\zeta_a)_{a\in\mathscr{A}} = (\mu_a)_{a\in\mathscr{A}}$, $\zeta = \sum_{a\in\mathscr{A}} \rho_s(a) \cdot \zeta_a$ and let $s \xrightarrow{\rho_s} \zeta$. For each $\mathcal{C} \in S/\mathcal{R}$, we get $\zeta(\mathcal{C}) = \sum_{a\in\mathscr{A}} \rho_s(a) \cdot \zeta_a(\mathcal{C}) = \sum_{a\in\mathscr{A}} \rho_s(a) \cdot \mu_a(\mathcal{C}) = \sum_{a\in\mathscr{A}} \rho_t(a) \cdot \nu_a(\mathcal{C}) = \sum_{a\in\mathscr{A}} \rho_t(a) \cdot \eta_a(\mathcal{C}) = \eta(\mathcal{C})$. Therefore, $(s,t) \in \mathcal{R}$ with respect to the definition of $(\exists\pi\forall)$-bisimulation. ∎

Thanks to this result, we denote from now on these coinciding bisimulations by $\sim_{(\exists\forall)}$.

**Remark 7.1.** *It is worthwhile to note that Definitions 6.1, 7.1 and 7.2 can be seen as the conservative extension of probabilistic bisimulation for (state-labelled) MDPs. To see that assume the set of uncertainty for every transition is a singleton. Since there is only one choice for the nature, the role of nature can be safely removed from the definitions. Moreover, it is worthwhile to note that Theorems 6.2, 7.1 and 7.2 show the soundness of the probabilistic bisimulation definitions with respect to PCTL. Unfortunately, it is shown in [Seg95, SL94] that probabilistic bisimulation for probabilistic automata is finer than PCTL equivalence which leads to the incompleteness in general. Since MDPs can be seen as a subclass of PAs, it is not difficult to see that the incompleteness holds also for MDPs.*

The notions $\sim_{(\forall)}$ and $\sim_{(\exists\forall)}$ are incomparable, as it is for instance observable in Example 7.1. It is shown in the example that $t \sim_{(\forall)} \overline{t}$ and $u \sim_{(\exists\forall)} \overline{u}$. However it is not hard to verify that $t \not\sim_{(\exists\forall)} \overline{t}$ and $u \not\sim_{(\forall)} \overline{u}$. For the latter, notice that for example $u$ can evolve to $r$ with

probability one by taking action $b$, whereas $\overline{u}$ cannot simulate. The former is noticeable in the situation where the controller wants to maximise the probability to reach $r$, but the nature declines. In this case $t$ chooses action $b$ and the nature let it go to $r$ with probability 0.8. Nevertheless the nature can prevent $\overline{t}$ to evolve into $r$ with probability more than 0.6, despite the fact which action has been chosen by $\overline{t}$.

## 7.2 A PTIME Decision Algorithm for Bisimulation Minimization

Computation of the alternating probabilistic bisimulation $\sim_{(\exists\forall)}$ for *IMDP*s follows the standard partition refinement approach [PT87, KS90, CS02, GHT14]. However, the core part is finding out whether two states "violate the definition of bisimulation". This verification routine amounts to check that $s$ and $t$ have the same set of *strictly minimal polytopes* detailed as follows.

For $s \in S$ and $a \in \mathscr{A}(s)$, recall that $\mathscr{H}_s^a$ denotes the polytope of feasible successor distributions over *states* with respect to taking the action $a$ in the state $s$. By $\mathscr{P}_{\mathcal{R}}^{s,a}$, we denote the polytope of feasible successor distributions over *equivalence classes* of $\mathcal{R}$ with respect to taking the action $a$ in the state $s$. Formally, for $\mu \in \mathrm{Disc}(S/\mathcal{R})$ we set $\mu \in \mathscr{P}_{\mathcal{R}}^{s,a}$ if, for each $\mathcal{C} \in S/\mathcal{R}$, we have $\mu(\mathcal{C}) \in I(s,a,\mathcal{C})$.

It is not difficult to see that each $\mathscr{P}_{\mathcal{R}}^{s,a}$ can be represented as an $\mathcal{H}$-polytope. To simplify our presentation, we shall fix an order over all equivalence classes in $S/\mathcal{R}$. By doing so, any distribution $\rho \in \mathrm{Disc}(S/\mathcal{R})$ can be seen as a vector $\mathbf{v}$ such that $\mathbf{v}_i = \rho(\mathcal{C}_i)$ for each $1 \leq i \leq n$, where $n = |S/\mathcal{R}|$, $\mathcal{C}_i$ is the $i$-th equivalence class, and $\mathbf{v}_i$ the $i$-th element in $\mathbf{v}$. For the above discussion, $\rho \in \mathscr{P}_{\mathcal{R}}^{s,a}$ iff $\rho(\mathcal{C}_i) \in \left[\mathbf{l}_i^{s,a}, \mathbf{u}_i^{s,a}\right]$ for any $1 \leq i \leq n$ and $\rho \in \mathrm{Disc}(S/\mathcal{R})$, where $\mathbf{l}^{s,a}$ and $\mathbf{u}^{s,a}$ are vectors such that $\mathbf{l}_i^{s,a} = \sum_{s' \in \mathcal{C}_i} \inf I(s,a,s')$ and $\mathbf{u}_i^{s,a} = \sum_{s' \in \mathcal{C}_i} \sup I(s,a,s')$ for each $1 \leq i \leq n$. Therefore, $\mathscr{P}_{\mathcal{R}}^{s,a}$ corresponds to an $\mathcal{H}$-polytope defined by:

$$\left\{ \mathbf{x}^{s,a} \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \mathbf{l}^{s,a} \leq \mathbf{x}^{s,a} \leq \mathbf{u}^{s,a} \\ \mathbf{1}^T \mathbf{x}^{s,a} = 1 \end{array} \right\}. \tag{7.1}$$

**Definition 7.3** (**Strictly minimal polytopes**). *Given an IMDP $\mathcal{M}$, a state $s$, an equivalence relation $\mathcal{R} \subseteq S \times S$ and a set $\{ \mathscr{P}_{\mathcal{R}}^{s,a} \mid a \in \mathscr{A}(s) \}$ where for each $a \in \mathscr{A}(s)$, for given $\mathbf{l}^{s,a}, \mathbf{u}^{s,a} \in \mathbb{R}^n$, $\mathscr{P}_{\mathcal{R}}^{s,a}$ is the convex polytope*

$$\mathscr{P}_{\mathcal{R}}^{s,a} = \left\{ \mathbf{x}^{s,a} \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \mathbf{l}^{s,a} \leq \mathbf{x}^{s,a} \leq \mathbf{u}^{s,a} \\ \mathbf{1}^T \mathbf{x}^{s,a} = 1 \end{array} \right\},$$

*a polytope $\mathscr{P}_{\mathcal{R}}^{s,a}$ is called* strictly minimal, *if for no $\rho \in \mathrm{Disc}(\mathscr{A}(s) \setminus \{a\})$, we have $\mathscr{P}_{\mathcal{R}}^{s,\rho} \subseteq \mathscr{P}_{\mathcal{R}}^{s,a}$ where $\mathscr{P}_{\mathcal{R}}^{s,\rho}$ is defined as*

$$\mathscr{P}_{\mathcal{R}}^{s,\rho} = \{ \mathbf{x}^{s,\rho} \in \mathbb{R}^n \mid \mathbf{x}^{s,\rho} = \sum_{b \in \mathscr{A}(s) \setminus \{a\}} \rho(b) \cdot \mathbf{x}^{s,b} \wedge \mathbf{x}^{s,b} \in \mathscr{P}_{\mathcal{R}}^{s,b} \}.$$

Thus, checking violation of a given pair of states amounts to check if the states have the same set of strictly minimal polytopes. Formally,

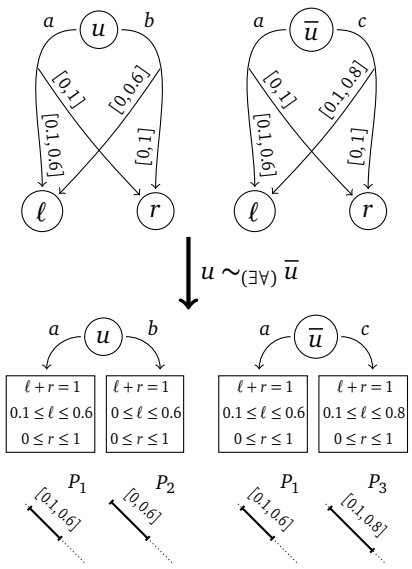



Figure 7.1: Description of the algorithm to decide if $u \sim_{(\exists\forall)} \overline{u}$.

**Lemma 7.1.** *We have* $s \sim_{(\exists\forall)} t$ *if and only if* $L(s) = L(t)$ *and* $\{\mathscr{P}^{s,a}_{\sim_{(\exists\forall)}} \mid a \in \mathscr{A}, \mathscr{P}^{s,a}_{\sim_{(\exists\forall)}}$ *is strictly minimal*$\} = \{\mathscr{P}^{t,a}_{\sim_{(\exists\forall)}} \mid a \in \mathscr{A}, \mathscr{P}^{t,a}_{\sim_{(\exists\forall)}}$ *is strictly minimal*$\}$.

*Proof.* We first address the "if" part. For each choice of nature $(\mu_a)_{a\in\mathscr{A}}$ where each $s \xrightarrow{a} \mu_a$, let $M = \{\overline{\mu}_a \mid a \in \mathscr{A}\}$ and $M' \subseteq M$ be the subset where each distribution lies within some strictly minimal polytope $\mathscr{P}^{s,b}_{\sim_{(\exists\forall)}}$. Because the strictly minimal polytopes coincide, we can construct a choice of nature $(\nu_a)_{a\in\mathscr{A}}$ such that $N = \{\overline{\nu}_a \mid a \in \mathscr{A}\} = M'$. Because $N \subseteq M$, it is easy to see that for each $t \xrightarrow{(\nu_a)} \nu$ there is $s \xrightarrow{(\mu_a)} \mu$ such that $\mu(\mathcal{C}) = \nu(\mathcal{C})$ for each $\mathcal{C} \in S/\mathcal{R}$.

As regards the "only if" part, let us assume that there is, say in $t$, a strictly minimal polytope $\mathscr{P}^{t,b}_{\sim_{(\exists\forall)}}$ that is not in the set of strictly minimal polytopes for $s$. There is a choice of nature $(\mu_a)_{a\in\mathscr{A}}$ for state $s$ such that no convex combination of elements of $M = \{\overline{\mu}_a \mid a \in \mathscr{A}\}$ lies in $\mathscr{P}^{t,b}_{\sim_{(\exists\forall)}}$; in particular no element of $M$ lies in $\mathscr{P}^{t,b}_{\sim_{(\exists\forall)}}$. For any choice of nature $(\nu_a)_{a\in\mathscr{A}}$ for state $t$, $\overline{\nu}_b$ is not a convex combination of elements from $M$. Thus, if scheduler chooses action $b$, there is no $s \xrightarrow{(\mu_a)} \mu$ such that $\mu(\mathcal{C}) = \nu_b(\mathcal{C})$ for each $\mathcal{C} \in S/\mathcal{R}$ and it does *not* hold $s \sim_{(\exists\forall)} t$. ■

**Example 7.2.** *Consider a pair of IMDPs depicted in Figure 7.1. The general sketch of the algorithm is as follows. We need to construct the polytopes of probability distributions offered by the actions; in our examples the polytopes are just line segments in two-dimensional space. We get* $u \sim_{(\exists\forall)} \overline{u}$ *since* $u$ *and* $\overline{u}$ *have the same set of strictly minimal polytopes w.r.t. set inclusion.* ♦

The bottleneck procedure in the analysis of the worst case time complexity of computing the coarsest alternating probabilistic bisimulation $\sim_{(\exists\forall)}$ is to check the strict minimality

of a polytope $\mathscr{P}_{\mathcal{R}}^{s,a}$ for an action $a \in \mathscr{A}(s)$. In the sequel, we give a polynomial time routine to verify the strict minimality of a polytope which in turn enables a polynomial time decision algorithm to decide $\sim_{(\exists\forall)}$. To this aim, we need the following equivalent form of the Farkas' Lemma:

**Lemma 7.2.** *Let $A$ be a real $m \times n$ matrix, $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{c} \in \mathbb{R}^n$. Then, $A\mathbf{x} \leq \mathbf{b}$ implies $\mathbf{c}^T\mathbf{x} \leq d$ if and only if there exists $\mu \in \mathbb{R}^m_{\geq 0}$ s.t. $A^T\mu = \mathbf{c}$ and $\mathbf{b}^T\mu \leq d$.*

*Proof.* We first address the "if" part. Assume that there exists $\mu \in \mathbb{R}^m_{\geq 0}$ such that $A^T\mu = \mathbf{c}$ and $\mathbf{b}^T\mu \leq d$. Let $A\mathbf{x} \leq \mathbf{b}$. Thus, $\mathbf{c}^T\mathbf{x} = (A^T\mu)^T\mathbf{x} = \mu^T(A\mathbf{x}) \leq \mu^T\mathbf{b} \leq d$. Therefore, $A\mathbf{x} \leq \mathbf{b}$ correctly implies $\mathbf{c}^T\mathbf{x} \leq d$. As regards the "only if" part, consider the following primal-dual LPs:

$$(\mathscr{P}) \quad \begin{array}{ll} \max & \mathbf{c}^T\mathbf{x} \\ \text{subject to:} & A\mathbf{x} \leq \mathbf{b} \end{array} \qquad (\mathscr{D}) \quad \begin{array}{ll} \min & \mathbf{b}^T\mathbf{y} \\ \text{subject to:} & A^T\mathbf{y} = \mathbf{c} \\ & \mathbf{y} \geq 0 \end{array}$$

Now suppose that the primal problem $\mathscr{P}$ is feasible. Therefore, there exists an $\mathbf{x} \in \mathbb{R}^n$ such that $A\mathbf{x} \leq \mathbf{b}$ and by assumption it holds that $\mathbf{c}^T\mathbf{x} \leq d$. Thus the primal LP has a finite optimal solution, i.e., there exists an optimal solution $\mathbf{x}^*$ such that $\max\ \mathbf{c}^T\mathbf{x} = \mathbf{c}^T\mathbf{x}^*$ and $\mathbf{c}^T\mathbf{x}^* \leq d$. By strong duality Theorem 3.4, the dual problem $\mathscr{D}$ has also a finite optimal solution and these values are equal. Thus there exists a $\mu \in \mathbb{R}^m_{\geq 0}$ such that $A^T\mu = \mathbf{c}$ and $\mathbf{b}^T\mu = \mathbf{c}^T\mathbf{x}^* \leq d$. This concludes the proof, as required. ■

**Theorem 7.4.** *Given an IMDP $\mathcal{M}$, a state $s \in S$, an equivalence relation $\mathcal{R} \subseteq S \times S$ and a set $\{\mathscr{P}_{\mathcal{R}}^{s,a} \mid a \in \mathscr{A}(s)\}$ defined as in Definition 7.3, checking if for each $a \in \mathscr{A}(s)$, the polytope $\mathscr{P}_{\mathcal{R}}^{s,a}$ is strictly minimal is in $\boldsymbol{P}$.*

*Proof.* Let $\mathscr{A}(s) = \{a_0, a_1, \ldots, a_m\}$, $n = |S/\mathcal{R}|$, and $P_i = \mathscr{P}_{\mathcal{R}}^{s,a_i}$ for $0 \leq i \leq m$. We describe the verification routine to check the strict minimality of $P_0$; the same routine applies to the other polytopes. We consider the converse of the strict minimality problem which asks to decide whether there exist $\lambda_1, \lambda_2, \ldots, \lambda_m \in \mathbb{R}_{\geq 0}$ such that $\sum_{i=1}^m \lambda_i = 1$ and $\sum_{i=1}^m \lambda_i P_i \subseteq P_0$. We show that the latter problem can be casted as an LP via Farkas' Lemma 7.2. To this aim, we alternatively reformulate the converse problem as "*do there exist $\lambda_1, \lambda_2, \ldots, \lambda_m \in \mathbb{R}_{\geq 0}$ with $\sum_{i=1}^m \lambda_i = 1$, such that $\mathbf{x}^i \in P_i$ for each $1 \leq i \leq m$ implies $\sum_{i=1}^m \lambda_i \mathbf{x}^i \in P_0$?*".

For every fixed $\lambda_1, \lambda_2, \ldots, \lambda_m \in \mathbb{R}_{\geq 0}$ with $\sum_{i=1}^m \lambda_i = 1$, the implication "$(\forall 1 \leq i \leq m : \mathbf{x}^i \in P_i) \implies \sum_{i=1}^m \lambda_i \mathbf{x}^i \in P_0$" can be written as the conjunction of $2n$ conditions:

$$\bigwedge_{i=1}^m \mathbf{l}^i \leq \mathbf{x}^i \leq \mathbf{u}^i \wedge \bigwedge_{i=1}^m \mathbf{1} \cdot \mathbf{x}^i = 1 \implies \sum_{i=1}^m \lambda_i \mathbf{x}_k^i \geq \mathbf{l}_k^0 \tag{7.2}$$

$$\bigwedge_{i=1}^m \mathbf{l}^i \leq \mathbf{x}^i \leq \mathbf{u}^i \wedge \bigwedge_{i=1}^m \mathbf{1} \cdot \mathbf{x}^i = 1 \implies \sum_{i=1}^m \lambda_i \mathbf{x}_k^i \leq \mathbf{u}_k^0 \tag{7.3}$$

for all $1 \leq k \leq n$. (Note that the condition $\mathbf{1} \cdot \sum_{i=1}^m \lambda_i \mathbf{x}^i = 1$ is trivially satisfied if $\mathbf{1} \cdot \mathbf{x}^i = 1$ for all $1 \leq i \leq m$.) Each of the conditions (7.2) and (7.3), by Farkas' Lemma, is equivalent to

**Algorithm 8:** Bisimulation($\mathcal{M}$)

**Input:** A relation $\mathcal{R}$ on $S \times S$
**Output:** Return a probabilistic bisimulation $\mathcal{R}$

**begin**
  $\mathcal{R} \leftarrow \{(s,t) \in S \times S \mid L(s) = L(t)\}$;
  **repeat**
    $\mathcal{R}' \leftarrow \mathcal{R}$;
    **forall** $s \in S$ **do**
      $D \leftarrow \emptyset$;
      **forall** $t \in [s]_{\mathcal{R}}$ **do**
        **if** Violate$_{(\exists\forall)}(s,t,\mathcal{R})$ **then**
          $D \leftarrow D \cup \{t\}$;
      split $[s]_{\mathcal{R}}$ in $\mathcal{R}$ into $D$ and $[s]_{\mathcal{R}} \setminus D$;
  **until** $\mathcal{R} = \mathcal{R}'$;
  **return** $\mathcal{R}$;

**Procedure 9:** Violate$_{(\exists\forall)}(s,t,\mathcal{R})$

**Input:** States $s, t$ and relation $\mathcal{R}$
**Output:** Checks if $s \sim_{\mathcal{R}} t$

**begin**
  $S, T \leftarrow \emptyset$;
  **forall** $a \in \mathscr{A}$ **do**
    **if** $\mathscr{P}_{\mathcal{R}}^{s,a}$ *is strictly minimal* **then**
      $S \leftarrow S \cup \{\mathscr{P}_{\mathcal{R}}^{s,a}\}$;
    **if** $\mathscr{P}_{\mathcal{R}}^{t,a}$ *is strictly minimal* **then**
      $T \leftarrow T \cup \{\mathscr{P}_{\mathcal{R}}^{t,a}\}$;
  **return** $S \neq T$;

Figure 7.2: Alternating probabilistic bisimulation algorithm for interval *MDP*s

the feasibility of a system of inequalities; for instance, for a given $k$, (7.2) is true if and only if there exist vectors $\mu^{k,i}, \nu^{k,i} \in \mathbb{R}^n_{\geq 0}$ and scalars $\theta^{k,i}, \eta^{k,i} \in \mathbb{R}_{\geq 0}$ for each $1 \leq i \leq m$ satisfying:

$$\mu^{k,i} - \nu^{k,i} + \theta^{k,i}\mathbf{1} - \eta^{k,i}\mathbf{1} = -\lambda_i \mathbf{e_k} \qquad \forall 1 \leq i \leq m \tag{7.4}$$

$$\sum_{i=1}^{m} \left(\mathbf{u}^i \cdot \mu^{k,i} - \mathbf{l}^i \cdot \nu^{k,i} + \theta^{k,i} - \eta^{k,i}\right) \leq -\mathbf{l}^0_k \tag{7.5}$$

Similarly, for a given $k$, (7.3) is true if and only if there exist vectors $\widehat{\mu}^{k,i}, \widehat{\nu}^{k,i} \in \mathbb{R}^n_{\geq 0}$ and scalars $\widehat{\theta}^{k,i}, \widehat{\eta}^{k,i} \in \mathbb{R}_{\geq 0}$ for each $1 \leq i \leq m$ satisfying:

$$\widehat{\mu}^{k,i} - \widehat{\nu}^{k,i} + \widehat{\theta}^{k,i}\mathbf{1} - \widehat{\eta}^{k,i}\mathbf{1} = \lambda_i \mathbf{e_k} \qquad \forall 1 \leq i \leq m \tag{7.6}$$

$$\sum_{i=1}^{m} (\mathbf{u}^i \cdot \widehat{\mu}^{k,i} - \mathbf{l}^i \cdot \widehat{\nu}^{k,i} + \widehat{\theta}^{k,i} - \widehat{\eta}^{k,i}) \leq \mathbf{u}^0_k \tag{7.7}$$

Thus, the converse problem we are aiming to solve reduces to checking the existence of vectors $\mu^{k,i}, \nu^{k,i}, \widehat{\mu}^{k,i}, \widehat{\nu}^{k,i} \in \mathbb{R}^n_{\geq 0}$ and scalars $\lambda_i, \theta^{k,i}, \eta^{k,i}, \widehat{\theta}^{k,i}, \widehat{\eta}^{k,i} \in \mathbb{R}_{\geq 0}$ for each $1 \leq i \leq m$ satisfying (7.4)-(7.7) and $\sum_{i=1}^{m} \lambda_i = 1$. That amounts to solve an LP problem, which is known to be in **P**. ■

As stated earlier, in order to compute $\sim_{(\exists\forall)}$ we follow the standard partition refinement approach formalized by the Bisimulation algorithm in Figure 7.2. Namely, we start with $\mathcal{R}$ being the complete relation and iteratively remove from $\mathcal{R}$ pairs of states that violate the definition of bisimulation with respect to $\mathcal{R}$. Clearly the core part of the algorithm is to check if two states "violate the definition of bisimulation". The violation of bisimilarity of $s$ and $t$ with respect to $\mathcal{R}$, which is addressed by the procedure Violate$_{(\exists\forall)}$, is checked

by verifying if states $s$ and $t$ have the same set of strictly minimal polytopes. As a result of Theorem 7.4, this verification routine can be checked in polynomial time. As regards the computational complexity of the algorithm, let $|S| = n$, $|A| = m$. The procedure Violate$_{(\exists\forall)}$ in Figure 7.2 is called at most $n^3$ times. The procedure Violate$_{(\exists\forall)}$ is then linear in $m$ and in the complexity of checking the strict minimality of polytopes which is $\mathcal{O}(|\mathcal{M}|^{\mathcal{O}(1)})$. Putting all these together, we get the following result.

**Theorem 7.5.** *Given an IMDP $\mathcal{M}$, computing of $\sim_{(\exists\forall)}$ can be done in time $\mathcal{O}(|\mathcal{M}|^{\mathcal{O}(1)})$.*

*Proof.* Immediate by the previous analysis. ■

## 7.3 Compositional Reasoning

In order to study the compositional minimization, that is, to split a complex *IMDP* as parallel composition of several simpler *IMDP*s and then to use the bisimulation as a means to reduce the size of each of these *IMDP*s before performing the model checking for a given PCTL formula $\varphi$, we have to extend the notion of bisimulation from one *IMDP* to a pair of *IMDP*s; we do this by following the usual construction (see, e.g., [CS02, Seg95]).

**Definition 7.4 (Alternating probabilistic bisimulation for *IMDP* components).** *Given two IMDPs $\mathcal{M}_1$ and $\mathcal{M}_2$, we say that they are* alternating probabilistic $(\exists\forall)$-bisimilar, *denoted by $\mathcal{M}_1 \sim_{(\exists\forall)} \mathcal{M}_2$, if there exists an alternating probabilistic $(\exists\forall)$-bisimulation on the disjoint union of $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $\bar{s}_1 \sim_{(\exists\forall)} \bar{s}_2$.*

With this definition at hand, we can now establish the first property needed for the compositional minimization, that is, transitivity of $\sim_{(\exists\forall)}$:

**Theorem 7.6.** *Given three IMDPs $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$, whenever $\mathcal{M}_1 \sim_{(\exists\forall)} \mathcal{M}_2$ and $\mathcal{M}_2 \sim_{(\exists\forall)} \mathcal{M}_3$, then $\mathcal{M}_1 \sim_{(\exists\forall)} \mathcal{M}_3$.*

*Proof.* Let $\mathcal{R}_{12}$ and $\mathcal{R}_{23}$ be the equivalence relations underlying $\mathcal{M}_1 \sim_{(\exists\forall)} \mathcal{M}_2$ and $\mathcal{M}_2 \sim_{(\exists\forall)} \mathcal{M}_3$, respectively. Let $\mathcal{R}_{13}$ be the symmetric and transitive closure of the set $\{(s_1, s_3) \mid \exists s_2 . s_1 \mathcal{R}_{12} s_2 \wedge s_2 \mathcal{R}_{23} s_3\} \cup (\mathcal{R}_{12} \cap S_1^2) \cup (\mathcal{R}_{23} \cap S_3^2)$. We claim that $\mathcal{R}_{13}$ is a probabilistic bisimulation justifying $\mathcal{M}_1 \sim_{(\exists\forall)} \mathcal{M}_3$.

The fact that $\bar{s}_1 \mathcal{R}_{13} \bar{s}_3$ is trivial since by hypothesis we have that $\bar{s}_1 \mathcal{R}_{12} \bar{s}_2$ and $\bar{s}_2 \mathcal{R}_{23} \bar{s}_3$, so $(\bar{s}_1, \bar{s}_3) \in \mathcal{R}_{13}$ by construction.

In the following, assume that $s_1 \in S_1$ and $s_3 \in S_3$; the other cases are similar.

The labelling is respected: for each $s_1 \mathcal{R}_{13} s_3$, we have that there exists $s_2$ such that $s_1 \mathcal{R}_{12} s_2$ and $s_2 \mathcal{R}_{23} s_3$; this implies that $L_1(s_1) = L_2(s_2)$ and $L_2(s_2) = L_3(s_3)$, thus $L_1(s_1) = L_3(s_3)$ as required.

To complete the proof, consider $s_1 \mathcal{R}_{13} s_3$. By hypothesis, there exists $s_2 \in S_2$ such that $s_1 \mathcal{R}_{12} s_2$ and $s_2 \mathcal{R}_{23} s_3$; since $\mathcal{R}_{12}$ is an alternating probabilistic $(\exists\forall)$-bisimulation, this implies that for each $\rho_1 \in \mathrm{Disc}(\mathscr{A}_1)$, there exists $\rho_{21} \in \mathrm{Disc}(\mathscr{A}_2)$ such that for each $s_2 \xrightarrow{\rho_{21}} \mu_{21}$ there exists $s_1 \xrightarrow{\rho_1} \mu_{12}$ such that $\mu_{12}\, \mathcal{L}(\mathcal{R}_{12})\, \mu_{21}$. Consider now $\rho_{21} \in \mathrm{Disc}(\mathscr{A}_2)$: since $s_2 \mathcal{R}_{23} s_3$ and $\mathcal{R}_{23}$ is an alternating probabilistic $(\exists\forall)$-bisimulation, it follows that

there exists $\rho_{321} \in \text{Disc}(\mathscr{A}_3)$ such that for each $s_3 \xrightarrow{\rho_{321}} \mu_{321}$ there exists $s_2 \xrightarrow{\rho_{21}} \mu_2$ such that $\mu_2 \; \mathcal{L}(\mathcal{R}_{23}) \; \mu_{321}$. This implies that for each $\rho_1 \in \text{Disc}(\mathscr{A}_1)$, there exists $\rho_3 \in \text{Disc}(\mathscr{A}_3)$ (namely, $\rho_{321}$) such that for each $s_3 \xrightarrow{\rho_3} \mu_3$ (that is, $s_3 \xrightarrow{\rho_{321}} \mu_{321}$) there exists $s_1 \xrightarrow{\rho_1} \mu_1$ (that is, $s_1 \xrightarrow{\rho_1} \mu_{12}$) such that $\mu_1 \; \mathcal{L}(\mathcal{R}_{13}) \; \mu_3$ (that follows from $\mu_1 = \mu_{12} \; \mathcal{L}(\mathcal{R}_{12}) \; \mu_{21}$ and $\mu_2 \; \mathcal{L}(\mathcal{R}_{23}) \; \mu_{321} = \mu_3$, with $\mu_2$ being one of the distributions $\mu_{21}$ for which $\mu_1 = \mu_{12} \; \mathcal{L}(\mathcal{R}_{12}) \; \mu_{21}$ holds, by construction of $\mathcal{R}_{13}$ and the properties of lifting (cf. [TH14])). ■

For the second property needed by the compositional minimization, that is, that $\sim_{(\exists\forall)}$ is preserved by the parallel composition operator, we first have to introduce such an operator; to this end, we consider a slight adaption of synchronous product of $\mathcal{M}_1$ and $\mathcal{M}_2$ as introduced in Chapter 6. Such a synchronous product makes use of a subclass of the Segala's (simple) probabilistic automata [Seg95, Seg06], called *action agnostic probabilistic automata* (cf. Definition 6.8), where each automaton has as set of actions the same singleton set $\{f\}$, that is, all transitions are labelled by the same external action $f$. Recall that an *(action agnostic) probabilistic automaton* (*PA*) is a tuple $\mathfrak{P} = (S, \bar{s}, \mathtt{AP}, L, T)$, where $S$ is a set of *states*, $\bar{s} \in S$ is the *start* state, $\mathtt{AP}$ is a finite set of *atomic propositions*, $L : S \to 2^{\mathtt{AP}}$ is a *labelling function*, and $T \subseteq S \times \text{Disc}(S)$ is a *probabilistic transition relation*.

**Definition 7.5 (*IMDP*s synchronous product).** *Given two IMDPs $\mathcal{M}_1$ and $\mathcal{M}_2$, we define the* synchronous product *of $\mathcal{M}_1$ and $\mathcal{M}_2$ as*

$$\mathcal{M}_1 \otimes \mathcal{M}_2 := \mathtt{F}(\mathtt{UF}(\mathcal{M}_1) \otimes \mathtt{UF}(\mathcal{M}_2))$$

*where*

- *the unfolding mapping $\mathtt{UF} : [\mathcal{M}] \to [\mathfrak{P}]$ is a function that maps a given IMDP $\mathcal{M} = (S, \bar{s}, \mathscr{A}, \mathtt{AP}, L, I)$ to the PA $\mathfrak{P} = (S, \bar{s}, \mathtt{AP}, L, T)$ where $T = \{ (s, \mu) \mid s \in S, \exists a \in \mathscr{A}(s) : \mu \in \mathtt{Ext}(\mathscr{H}_s^a) \wedge \mathscr{H}_s^a$ is a strictly minimal polytope $\}$;*
- *the folding mapping $\mathtt{F} : [\mathfrak{P}] \to [\mathcal{M}]$ transforms a PA $\mathfrak{P} = (S, \bar{s}, \mathtt{AP}, L, T)$ into the IMDP $\mathcal{M} = (S, \bar{s}, \{f\}, \mathtt{AP}, L, I)$ where, for each $s, t \in S$, $I(s, f, t) = \text{proj}_{\mathbf{e}_t} \text{CH}(\{ \mu \mid (s, \mu) \in T \})$, where each component $\mathbf{e}_{\mathbf{uv}}$ of the vector $\mathbf{e}_{\mathbf{u}} \in \mathbb{R}^{|S|}$ is defined as $\mathbf{e}_{\mathbf{uv}} = \delta_u(v)$;*
- *the* synchronous product *of two PAs $\mathfrak{P}_1$ and $\mathfrak{P}_2$, denoted by $\mathfrak{P}_1 \otimes \mathfrak{P}_2$, is the probabilistic automaton $\mathfrak{P} = (S, \bar{s}, \mathtt{AP}, L, T)$ where $S = S_1 \times S_2$, $\bar{s} = (\bar{s}_1, \bar{s}_2)$, $\mathtt{AP} = \mathtt{AP}_1 \cup \mathtt{AP}_2$, for each $(s_1, s_2) \in S$, $L(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$, and $T = \{ ((s_1, s_2), \mu_1 \times \mu_2) \mid (s_1, \mu_1) \in T_1$ and $(s_2, \mu_2) \in T_2 \}$, where $\mu_1 \times \mu_2 \in \text{Disc}(S_1 \times S_2)$ is defined for each $(t_1, t_2) \in S_1 \times S_2$ as $(\mu_1 \times \mu_2)(t_1, t_2) = \mu_1(t_1) \cdot \mu_2(t_2)$.*

As stated earlier, Definition 7.5 is slightly different with its counterpart Definition 6.15. As a matter of fact, due to the competitive semantics for resolving the nondeterminisms, only actions whose uncertainty set is a strictly minimal polytope play a role in deciding the alternating bisimulation relation $\sim_{(\exists\forall)}$. In particular, for the compositional reasoning keeping state actions whose uncertainty set is not strictly minimal induces spurious behaviors and therefore, influences on the soundness of the parallel operator definition. In order to avoid such redundancies, we can either preprocess the *IMDP*s before composing by removing state actions whose uncertainty set is not strictly minimal or restricting the unfolding mapping $\mathtt{UF}$ to unfold a given *IMDP* while ensuring that all extreme transitions

in the resultant probabilistic automaton correspond to extreme points of strictly minimal polytopes in the original *IMDP*. For the sake of simplicity, we choose the latter.

**Theorem 7.7.** *Given three IMDPs $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$, if $\mathcal{M}_1 \sim_{(\exists\forall)} \mathcal{M}_2$, then $\mathcal{M}_1 \otimes \mathcal{M}_3 \sim_{(\exists\forall)} \mathcal{M}_2 \otimes \mathcal{M}_3$.*

Before proving the theorem, we prove that alternating probabilistic $(\exists\forall)$-bisimilar *IMDP*s induce probabilistic bisimilar unfolded *PA*. Formally,

**Lemma 7.3.** *Given two IMDPs $\mathcal{M}_1$ and $\mathcal{M}_2$, if $\mathcal{M}_1 \sim_{(\exists\forall)} \mathcal{M}_2$, then $\mathtt{UF}(\mathcal{M}_1) \sim^{p}_{aa} \mathtt{UF}(\mathcal{M}_2)$.*

*Proof.* Due to the Theorem 7.3, we consider without loss of generality the alternating probabilistic bisimulation $\sim_{(\exists\sigma\forall)}$. Let $\mathcal{R}$ be the alternating probabilistic $(\exists\sigma\forall)$-bisimulation justifying $\mathcal{M}_1 \sim_{(\exists\sigma\forall)} \mathcal{M}_2$; we claim that $\mathcal{R}$ is also a probabilistic bisimulation justifying $\mathtt{UF}(\mathcal{M}_1) \sim^{p}_{aa} \mathtt{UF}(\mathcal{M}_2)$.

The fact that $\mathcal{R}$ is an equivalence relation on $S_1 \cup S_2$, that $\bar{s}_1 \ \mathcal{R} \ \bar{s}_2$, and that $L_1(s_1) = L_2(s_2)$ for each $s_1 \ \mathcal{R} \ s_2$ follows immediately by construction of $\mathcal{R}$ and the fact that $\mathcal{R}$ is the alternating probabilistic $(\exists\sigma\forall)$-bisimulation.

Consider $(s_1, s_2) \in \mathcal{R}$ and assume that $s_1 \in S_1$ and $s_2 \in S_2$; the other cases are similar. Consider $s_1 \longrightarrow \mu_1$. By definition of the unfolding mapping $\mathtt{UF}$, there exists an action $a \in \mathscr{A}(s_1)$ such that $\mu_1 \in \mathtt{Ext}(\mathscr{H}^a_{s_1})$ and $\mathscr{H}^a_{s_1}$ is a strictly minimal polytope. Since $s_1 \sim_{(\exists\sigma\forall)} s_2$ therefore, their sets of strictly minimal polytopes are equivalent with respect to set inclusion [cf. Lemma 7.1]. This implies that there exists an action $b \in \mathscr{A}(s_2)$ such that $\mathscr{H}^b_{s_2} = \mathscr{H}^a_{s_1}$, hence by taking $\mu_2 = \mu_1 \in \mathtt{Ext}(\mathscr{H}^a_{s_1}) = \mathtt{Ext}(\mathscr{H}^b_{s_2})$ we have that $s_2 \xrightarrow{b} \mu_2$. This trivially implies that $s_2 \longrightarrow_{\mathrm{c}} \mu_2$ with $\mu_1 \ \mathcal{L}(\mathcal{R}) \ \mu_2$. ∎

We are now ready to prove Theorem 7.7.

*Proof.* Due to the Theorem 7.3, we consider without loss of generality the alternating probabilistic bisimulation $\sim_{(\exists\sigma\forall)}$. Let $\mathcal{R}_{12}$ be the alternating probabilistic $(\exists\sigma\forall)$-bisimulation justifying $\mathcal{M}_1 \sim_{(\exists\sigma\forall)} \mathcal{M}_2$ and consider the relation $\mathcal{R} = \{((s_1, s_3), (s_2, s_3)) \mid (s_1, s_2) \in \mathcal{R}_{12}, s_3 \in S_3\}$; we claim that $\mathcal{R}$ is an alternating probabilistic $(\exists\sigma\forall)$-bisimulation between $\mathcal{M}_1 \otimes \mathcal{M}_3$ and $\mathcal{M}_2 \otimes \mathcal{M}_3$.

The fact that $\mathcal{R}$ is an equivalence relation on $(S_1 \times S_3) \cup (S_2 \times S_3)$, that $(\bar{s}_1, \bar{s}_3) \ \mathcal{R} \ (\bar{s}_2, \bar{s}_3)$, and that $L_{13}(s_1, s_3) = L_{23}(s_2, s_3)$ for each $(s_1, s_3) \ \mathcal{R} \ (s_2, s_3)$ follows immediately by construction of $\mathcal{R}$ and the fact that $\mathcal{R}_{12}$ is the alternating probabilistic $(\exists\sigma\forall)$-bisimulation.

Consider $(s_1, s_3) \ \mathcal{R} \ (s_2, s_3)$ and assume that $s_1 \in S_1$ and $s_2 \in S_2$; the other cases are similar. By definition of the folding function $\mathtt{F}$, it follows that the only action available from $(s_1, s_3)$ and from $(s_2, s_3)$ is $f$, i.e., $\mathscr{A}(s_1, s_3) = \mathscr{A}(s_2, s_3) = \{f\}$. This means that checking whether for each $\rho_{13} \in \mathrm{Disc}(\mathscr{A}(s_1, s_3))$ there exists $\rho_{2,3} \in \mathrm{Disc}(\mathscr{A}(s_2, s_3))$ such that for each $(s_2, s_3) \xrightarrow{\rho_{2,3}} \mu_{2,3}$ there exists $(s_1, s_3) \xrightarrow{\rho_{1,3}} \mu_{1,3}$ such that $\mu_{1,3} \ \mathcal{L}(\mathcal{R}) \ \mu_{2,3}$ reduces to check whether for each $(s_2, s_3) \xrightarrow{\delta_f} \mu_{2,3}$ there exists $(s_1, s_3) \xrightarrow{\delta_f} \mu_{1,3}$ such that $\mu_{1,3} \ \mathcal{L}(\mathcal{R}) \ \mu_{2,3}$, since $\rho_{13} = \rho_{23} = \delta_f$. This holds if and only if for each transition $((s_2, s_3), \mu_{2,3})$ of $\mathtt{UF}(\mathcal{M}_2) \otimes \mathtt{UF}(\mathcal{M}_3)$ there is a transition $((s_1, s_3), \mu_{1,3})$ of $\mathtt{UF}(\mathcal{M}_1) \otimes \mathtt{UF}(\mathcal{M}_3)$ such that $\mu_{1,3} \ \mathcal{L}(\mathcal{R}) \ \mu_{2,3}$, that is, if $\mathcal{R}$ is a probabilistic bisimulation between $\mathtt{UF}(\mathcal{M}_2) \otimes \mathtt{UF}(\mathcal{M}_3)$ and $\mathtt{UF}(\mathcal{M}_1) \otimes \mathtt{UF}(\mathcal{M}_3)$.

By inspecting the equivalence relations used in the proofs of Lemmas 7.3 and 6.4, it follows that $\mathcal{R}$ is indeed a probabilistic bisimulation between $\mathtt{UF}(\mathcal{M}_2) \otimes \mathtt{UF}(\mathcal{M}_3)$ and $\mathtt{UF}(\mathcal{M}_1) \otimes \mathtt{UF}(\mathcal{M}_3)$, thus $\mathcal{M}_1 \otimes \mathcal{M}_3 \sim_{(\exists\sigma\forall)} \mathcal{M}_2 \otimes \mathcal{M}_3$, as required. ■

So far we have considered the parallel composition via synchronous production, which is working by the definition of folding collapsing all labels to a single transition. Here we consider the other extreme of the parallel composition: interleaving only.

**Definition 7.6 (*IMDP*s interleaving parallel composition).** *Given two IMDPs $\mathcal{M}_l$ and $\mathcal{M}_r$, we define the* interleaved composition *of $\mathcal{M}_l$ and $\mathcal{M}_r$, denoted by $\mathcal{M}_l \curlywedge \mathcal{M}_r$, as the IMDP $\mathcal{M} = (S, \bar{s}, \mathscr{A}, \mathtt{AP}, L, I)$ where $S = S_l \times S_r$; $\bar{s} = (\bar{s}_l, \bar{s}_r)$; $\mathscr{A} = (\mathscr{A}_l \times \{l\}) \cup (\mathscr{A}_r \times \{r\})$; $\mathtt{AP} = \mathtt{AP}_l \cup \mathtt{AP}_r$; for each $(s_l, s_r) \in S$, $L(s_l, s_r) = L_l(s_l) \cup L_r(s_r)$; and*

$$I((s_l, s_r), (a, i), (t_l, t_r)) = \begin{cases} I_l(s_l, a, t_l) & \text{if } i = l \text{ and } t_r = s_r, \\ I_r(s_r, a, t_r) & \text{if } i = r \text{ and } t_l = s_l, \\ [0, 0] & \text{otherwise.} \end{cases}$$

**Theorem 7.8.** *Given three IMDPs $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$, if $\mathcal{M}_1 \sim_{(\exists\forall)} \mathcal{M}_2$, then $\mathcal{M}_1 \curlywedge \mathcal{M}_3 \sim_{(\exists\forall)} \mathcal{M}_2 \curlywedge \mathcal{M}_3$.*

*Proof.* Due to the Theorem 7.3, we consider without loss of generality the alternating probabilistic bisimulation $\sim_{(\exists\sigma\forall)}$. Let $\mathcal{R}$ be the alternating probabilistic $(\exists\sigma\forall)$-bisimulation justifying $\mathcal{M}_1 \sim_{(\exists\sigma\forall)} \mathcal{M}_2$ and define $\mathcal{R}' = \{((s_1, s_3), (s_2, s_3)) \mid (s_1, s_2) \in \mathcal{R}, s_3 \in S_3\}$; we claim that $\mathcal{R}'$ is an alternating probabilistic $(\exists\sigma\forall)$-bisimulation between $\mathcal{M}_1 \curlywedge \mathcal{M}_3$ and $\mathcal{M}_2 \curlywedge \mathcal{M}_3$. The fact that $\mathcal{R}'$ is an equivalence relation follows trivially by its definition and the fact that $\mathcal{R}$ is an equivalence relation. The fact that $((\bar{s}_1, \bar{s}_3), (\bar{s}_2, \bar{s}_3))$ follows immediately by the hypothesis that $(\bar{s}_1, \bar{s}_2) \in \mathcal{R}$ and $(\bar{s}_3, \bar{s}_3) \in \mathcal{I}_{S_3}$.

Let $((s_1, s_3), (s_2, s_3)) \in \mathcal{R}'$. Assume, without loss of generality, that $s_1 \in S_1$ and $s_2 \in S_2$; the other cases are similar. The fact that $L_{1,3}(s_1, s_3) = L_{2,3}(s_2, s_3)$ is straightforward, since by definition of interleaved composition and the hypothesis that $s_1 \ \mathcal{R} \ s_2$, we have that $L_{1,3}(s_1, s_3) = L_1(s_1) \cup L_3(s_3) = L_2(s_2) \cup L_3(s_3) = L_{2,3}(s_2, s_3)$, as required.

Consider now $\rho_{13} \in \mathrm{Disc}(\mathscr{A}(s_1, s_3))$: by definition of interleaved composition, it follows that each $(a, i) \in \mathrm{Supp}(\rho_{13})$ is either of the form $(a, l) \in \mathscr{A}_1 \times \{l\}$, or of the form $(a, r) \in \mathscr{A}_3 \times \{r\}$. Consider now the two distributions $\rho_1 \in \mathrm{Disc}(\mathscr{A}_1)$ and $\rho_3 \in \mathrm{Disc}(\mathscr{A}_3)$ defined as follows: $\rho_1$ is defined for each $a \in \mathscr{A}_1$ as $\rho_1(a) = \frac{\rho_{13}(a,l)}{\sum_{b \in \mathscr{A}_1} \rho_{13}(b,l)}$ if $\sum_{b \in \mathscr{A}_1} \rho_{13}(b, l) \neq 0$, otherwise $\rho_1$ is taken arbitrarily from $\mathrm{Disc}(\mathscr{A}(s_1))$; and similarly $\rho_3$ is defined for each $a \in \mathscr{A}_3$ as $\rho_3(a) = \frac{\rho_{13}(a,r)}{\sum_{b \in \mathscr{A}_3} \rho_{13}(b,r)}$ if $\sum_{b \in \mathscr{A}_3} \rho_{13}(b, r) \neq 0$, otherwise $\rho_3$ is taken arbitrarily from $\mathrm{Disc}(\mathscr{A}(s_3))$. It is easy to see that for each $(a, i) \in \mathscr{A}(s_1, s_3)$, we have $\rho_{13}(a, i) = \rho_1(a) \cdot \sum_{b \in \mathscr{A}_1} \rho_{13}(b, l)$ if $i = l$ and $\rho_{13}(a, i) = \rho_3(a) \cdot \sum_{b \in \mathscr{A}_3} \rho_{13}(b, r)$ if $i = r$.

Since $((s_1, s_3), (s_2, s_3)) \in \mathcal{R}'$, it follows that $(s_1, s_2) \in \mathcal{R}$, thus for the considered $\rho_1$, there exists $\rho_2 \in \mathrm{Disc}(\mathscr{A}(s_2))$ such that for each $s_2 \xrightarrow{\rho_2} \mu_2$ there exists $s_1 \xrightarrow{\rho_1} \mu_1$ such that $\mu_1 \ \mathcal{L}(\mathcal{R}) \ \mu_2$. Trivially, for the considered $\rho_3$, there exists $\rho_3' = \rho_3 \in \mathrm{Disc}(\mathscr{A}(s_3))$ such that for each $s_3 \xrightarrow{\rho_3'} \mu_3'$ there exists $s_3 \xrightarrow{\rho_3} \mu_3$ such that $\mu_3 \ \mathcal{L}(\mathcal{I}_3) \ \mu_3'$ where $\mu_3 = \mu_3'$ and $\mathcal{I}_3$ is the identity relation on $S_3$.

Consider now the distribution $\rho_{23} \in \text{Disc}(\mathscr{A}(s_2, s_3))$ defined for each $(a, i) \in \mathscr{A}(s_2, s_3)$ as $\rho_{23}(a, i) = \rho_2(a) \cdot \sum_{b \in \mathscr{A}_1} \rho_{13}(b, l)$ if $i = l$ and $\rho_{23}(a, i) = \rho_3(a) \cdot \sum_{b \in \mathscr{A}_3} \rho_{13}(b, r)$ if $i = r$. Essentially, $\rho_{23}$ combines the distributions $\rho_2$ and $\rho_3$ according to the weights of the left $\rho_1$ and right $\rho_3$ choices made in $\rho_{13}$. This means that, given $\rho_{13}$ and $\rho_{23}$, we have that for each $(s_2, s_3) \xrightarrow{\rho_{23}} \mu_{23}$ there exists $(s_1, s_3) \xrightarrow{\rho_{13}} \mu_{13}$ such that $\mu_{13}\ \mathcal{L}(\mathcal{R}')\ \mu_{23}$, by standard properties of lifting (see, e.g., [TH14]), as required: the required $\mu_{13}$ is obtained by combining the distributions $\mu_1$ and $\mu_3$ according to the weights of the left $\rho_2$ and right $\rho_3$ choices made in $\rho_{23}$ (that are actually the same as the weights of the left $\rho_1$ and right $\rho_3$ choices made in $\rho_{13}$). ■

## 7.4 Case Studies

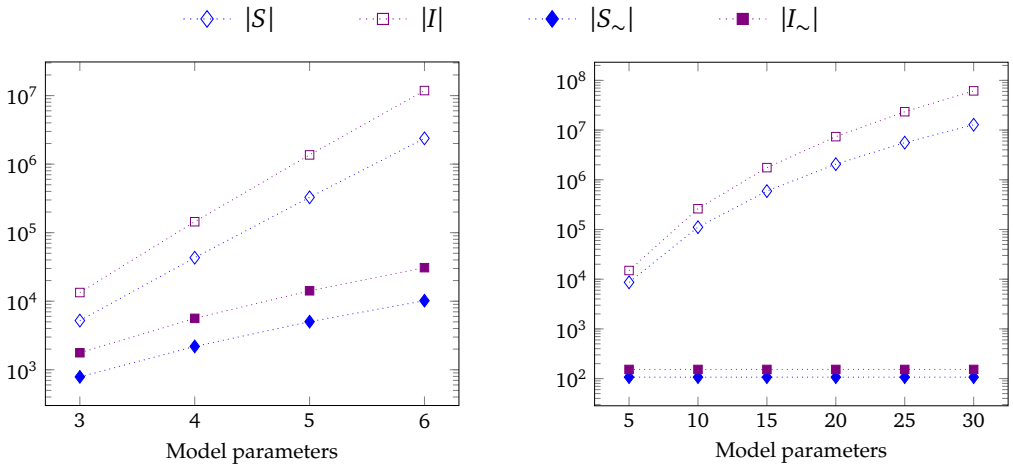
We implemented the proposed bisimulation minimization and applied them to several case studies. The goal of these experiments is to assess the impact of bisimulation minimization as a pre-processing step to minimize the *IMDP* models while preserving the PCTL properties (with respect to the controller (parameter) synthesis semantics) they satisfy.

In our experiments, we implemented in a prototypical tool the proposed bisimulation minimization algorithm and applied it to several case studies. The bisimulation algorithm is tested on several PRISM [KNP11] benchmarks extended to support also intervals in the transitions. For the evaluation, we have used a machine with a 3.6 GHz Intel i7-4790 with 16 GB of RAM of which 12 assigned to the tool; the timeout has been set to 30 minutes. Our tool reads a model specification in the PRISM input language and constructs an explicit-state representation of the state space. Afterwards, it computes the quotient using the algorithm in Figure 7.2.

Table 7.3 shows the performance of our prototype on a number of case studies taken from the PRISM website [PRI], where we have replaced some of the probabilistic choices with intervals. Despite using an explicit representation for the model, the prototype is able to manage case studies in the order of millions of states and transitions (columns "Model", "$|S|$", and "$|I|$"). The time in seconds required to compute the bisimulation relation and the size of the corresponding quotient *IMDP* are shown in columns "$t_{\sim}$", "$|S_{\sim}|$", and "$|I_{\sim}|$". In order to improve the performance of the tool, we have implemented optimizations, such as caching equivalent LP problems, which improve the runtime of our prototype. Because of this, we saved to solve several LP problems in each tool run, thereby avoiding the potentially costly solution of LP problems from becoming a bottleneck. However, the more refinements are needed, the more time is required to complete the minimization, since several new LP problems need to be solved. The plots in Figure 7.3 show graphically the number of states and transitions for the Consensus and Crowds experiments, where for the latter we have considered more instances than the ones reported in Table 7.3. As we can see, the bisimulation minimization is able to reduce considerably the size of the *IMDP*, by several orders of magnitude. Additionally, this reduction correlates positively with the number of model parameters as depicted in Figure 7.4.

Table 7.3: Experimental evaluation of the bisimulation computation

| Model | $\|S\|$ | $\|I\|$ | $t_\sim$ (s) | $\|S_\sim\|$ | $\|I_\sim\|$ |
|---|---|---|---|---|---|
| Consensus-Shared-Coin-3 | 5 216 | 13 380 | 1 | 787 | 1 770 |
| Consensus-Shared-Coin-4 | 43 136 | 144 352 | 3 | 2 189 | 5 621 |
| Consensus-Shared-Coin-5 | 327 936 | 1 363 120 | 26 | 5 025 | 14 192 |
| Consensus-Shared-Coin-6 | 2 376 448 | 11 835 456 | 238 | 10 173 | 30 861 |
| Crowds-5-10 | 111 294 | 261 444 | 1 | 107 | 153 |
| Crowds-5-20 | 2 061 951 | 7 374 951 | 20 | 107 | 153 |
| Crowds-5-30 | 12 816 233 | 61 511 033 | 149 | 107 | 153 |
| Crowds-5-40 | | –MO– | | | |
| Mutual-Exclusion-PZ-3 | 2 368 | 8 724 | 4 | 475 | 1 632 |
| Mutual-Exclusion-PZ-4 | 27 600 | 136 992 | 70 | 3 061 | 13 411 |
| Mutual-Exclusion-PZ-5 | 308 800 | 1 930 160 | 534 | 12 732 | 65 661 |
| Mutual-Exclusion-PZ-6 | 3 377 344 | 25 470 144 | | –TO– | |
| Dining-Phils-LR-nofair-3 | 956 | 3 048 | 1 | 172 | 509 |
| Dining-Phils-LR-nofair-4 | 9 440 | 40 120 | 14 | 822 | 3 285 |
| Dining-Phils-LR-nofair-5 | 93 068 | 494 420 | 622 | 5 747 | 29 279 |
| Dining-Phils-LR-nofair-6 | 917 424 | 5 848 524 | | –TO– | |

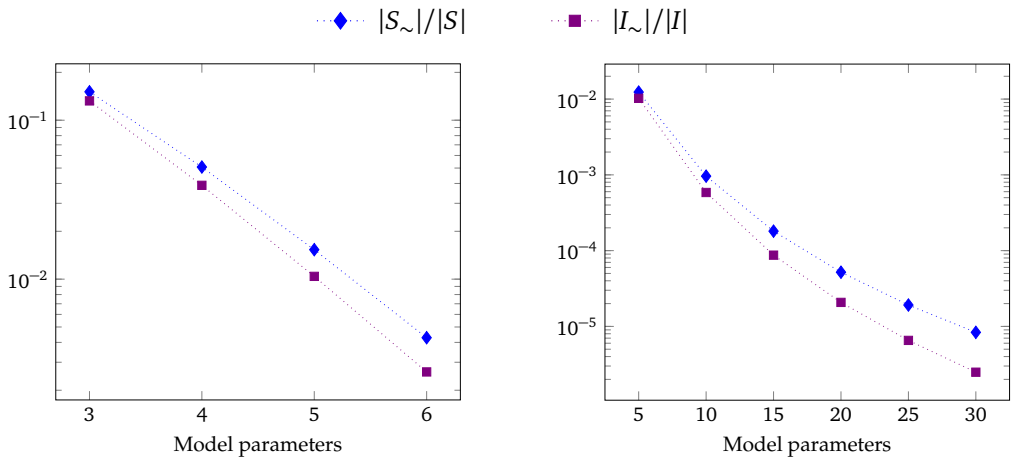



Consensus-Shared-Coin model Crowds model

Figure 7.3: Effectiveness of bisimulation minimization on model reduction

## 7.5 Concluding Remarks

In this chapter, we have analyzed interval Markov decision processes under controller (parameter) synthesis semantics in a dynamic setting. In particular, we provided an efficient compositional bisimulation minimization approach for *IMDP*s under competitive seman-




Consensus-Shared-Coin model Crowds model

Figure 7.4: State and transition reduction ratio by bisimulation minimization

tics. In this regard, we proved that the alternating probabilistic bisimulations for *IMDP*s can be decided in polynomial time. Moreover, from perspective of compositional reasoning, we showed that the alternating probabilistic bisimulations for *IMDP*s are congruences with respect to synchronous product and interleaving.

**Related work** Bisimulation minimization for uncertain or parametric probabilistic models has been studied in [HHS+16b, HHK14, HHHT16] where the authors explored the computational complexity and approximability of deciding probabilistic bisimulation for *IMDP*s with respect to the cooperative resolution of nondeterminisms. In this chapter, we showed that *IMDP*s can be minimized efficiently with respect to the competitive resolution of nondeterminisms. From the viewpoint of compositional minimization, *IMC*s [JL91] and abstract Probabilistic Automata (*PA*) [DKL+11a, DKL+11b] serve as specification theories for *MC* and *PA*, featuring satisfaction relation and various refinement relations. In [HHT16], the authors discuss the key ingredients to build up the operations of parallel composition for composing *IMDP* components at run-time. In this chapter we follow this spirit for alternating probabilistic bisimulation on *IMDP*s.

CHAPTER 8

# Multi-objective Robust Controller Synthesis for Interval MDPs

In this chapter, we present a novel technique for multi-objective controller synthesis for *IMDP*s. Our aim is to synthesize a robust controller that guarantees the satisfaction of the multi-objective property at the same time, despite the additional uncertainty over the transition probabilities in these models. Our approach relies on the controller synthesis semantics under which it views the uncertainty as making adversarial choices among the available transition probability distributions induced by the intervals, as the system evolves along state transitions. In this regard, we first analyze the problem complexity, showing that it is **PSPACE**-hard and then develop a value iteration-based decision algorithm to approximate the Pareto curve of achievable points. We finally show promising results on a variety of case studies, obtained by prototypical implementations of all algorithms.

The material presented in this chapter is an extended version of the results reported in [HHH+17b, HHH+17a].

**Organization of the chapter.** In Section 8.1, we introduce multi-objective robust controller synthesis for *IMDP*s and present our solution approach in detail. Afterwards in Section 8.2, we present the practical effectiveness of our proposed approaches by applying them on several case studies using a prototypical tool. Finally, in Section 8.3 we conclude the chapter.

## 8.1 Multi-objective Robust Controller Synthesis for IMDPs

In this section, we consider two main classes of properties for *IMDP*s; the *probability of reaching a target* and the *expected total reward*. The reason that we focus on these properties is that their algorithms usually serve as the basis for more complex properties. For instance, they can be easily extended to answer queries with linear temporal logic properties as shown in [EKVY07]. To this aim, we lift the satisfaction definitions of these two classes of properties from *MDP*s in [FKP12, FKN+11] to *IMDP*s by encoding the notion of robustness for controllers.

**Definition 8.1** (**Reachability predicate & its robust satisfaction**). *A* reachability predicate $[T]^{\leq k}_{\sim p}$ *consists of a set of target states* $T \subseteq S$*, a relational operator* $\sim \in \{\leq, \geq\}$*, a rational probability bound* $p \in [0,1] \cap \mathbb{Q}$ *and a time bound* $k \in \mathbb{N} \cup \{\infty\}$*. It indicates that the probability of reaching* $T$ *within* $k$ *time steps satisfies* $\sim p$*.*
*Robust satisfaction of* $[T]^{\leq k}_{\sim p}$ *by IMDP* $\mathcal{M}$ *under controller* $\sigma \in \Sigma$ *is denoted by* $\mathcal{M} \downharpoonright_{\sigma} \models_{\Pi} [T]^{\leq k}_{\sim p}$ *and indicates that the probability of the set of all paths that reach* $T$ *under* $\sigma$ *satisfies the bound* $\sim p$ *for every choice of nature* $\pi \in \Pi$*. Formally,*

$$\mathcal{M} \downharpoonright_{\sigma} \models_{\Pi} [T]^{\leq k}_{\sim p} \text{ iff } \mathbf{Pr}^{\sigma}_{\mathcal{M}}(\Diamond^{\leq k}\, T) \sim p$$

*where*

$$\mathbf{Pr}^{\sigma}_{\mathcal{M}}(\Diamond^{\leq k}\, T) := \operatorname*{opt}_{\pi \in \Pi} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}\{\, \xi \in Paths^{inf}_{\mathcal{M}} \mid \exists i \leq k : \xi[i] \in T \,\}$$

*and*

$$\text{opt} = \begin{cases} \min & \text{if } \sim = \geq \\ \max & \text{if } \sim = \leq \end{cases}$$

*Furthermore,* $\sigma$ *is referred to as a robust controller.*

In order to model reward predicates for an *IMDP*, we associate a reward to actions available in each state. This is done by introducing a *reward structure*:

**Definition 8.2** (***IMDP* reward structure**). *A* reward structure *for an IMDP is a function* $\mathtt{r} : S \times \mathscr{A} \to \mathbb{R}$ *that assigns to each state-action pair* $(s,a)$*, where* $s \in S$ *and* $a \in \mathscr{A}(s)$*, a reward* $\mathtt{r}(s,a) \in \mathbb{R}$*. Given a (possibly infinite) path* $\xi$ *and a step number* $k \in \mathbb{N} \cup \{\infty\}$*, the total accumulated reward in* $k$ *steps for* $\xi$ *over* $\mathtt{r}$ *is* $\mathtt{r}[k](\xi) := \sum_{i=0}^{k-1} \mathtt{r}(\xi[i], \xi(i))$*.*

Note that we allow negative rewards in this definition, but that due to later assumptions their use is restricted.

**Definition 8.3** (**Reward predicate & its robust satisfaction**). *A* reward predicate $[\mathtt{r}]^{\leq k}_{\sim r}$ *consists of a reward structure* $\mathtt{r}$*, a time bound* $k \in \mathbb{N} \cup \{\infty\}$*, a relational operator* $\sim \in \{\leq, \geq\}$ *and a reward bound* $r \in \mathbb{Q}$*. It indicates that the expected total accumulated reward within* $k$ *steps satisfies* $\sim r$*.*
*Robust satisfaction of* $[\mathtt{r}]^{\leq k}_{\sim r}$ *by IMDP* $\mathcal{M}$ *under controller* $\sigma \in \Sigma$ *is denoted by* $\mathcal{M} \downharpoonright_{\sigma} \models_{\Pi} [\mathtt{r}]^{\leq k}_{\sim r}$ *and indicates that the expected total reward over the set of all paths under* $\sigma$ *satisfies the bound* $\sim r$ *for every choice of nature* $\pi \in \Pi$*. Formally,*

$$\mathcal{M} \downharpoonright_{\sigma} \models_{\Pi} [\mathtt{r}]^{\leq k}_{\sim r} \text{ iff } ExpTot^{\sigma,k}_{\mathcal{M}}[\mathtt{r}] \sim r$$

*where*

$$ExpTot^{\sigma,k}_{\mathcal{M}}[\mathtt{r}] := \operatorname*{opt}_{\pi \in \Pi} \int_{\xi \in Paths^{inf}_{\mathcal{M}}} \mathtt{r}[k](\xi)\, \mathrm{d}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}$$

*and*

$$\text{opt} = \begin{cases} \min & \text{if } \sim = \geq \\ \max & \text{if } \sim = \leq \end{cases}$$

*Furthermore,* $\sigma$ *is referred to as the robust controller.*

For the purpose of algorithm design, we also consider weighted sum of rewards. Formally,

**Definition 8.4 (Weighted reward sum).** *Given a weight vector $\mathbf{w} \in \mathbb{R}^n$, vector of time bounds $\mathbf{k} = (k_1, \ldots, k_n) \in (\mathbb{N} \cup \{\infty\})^n$ and reward structures $\mathtt{r} = (\mathtt{r}_1, \ldots, \mathtt{r}_n)$ for IMDP $\mathcal{M}$, the* weighted reward sum $\mathbf{w}\cdot\mathtt{r}[\mathbf{k}]$ *over a path $\xi$ is defined as $\mathbf{w}.\mathtt{r}[\mathbf{k}](\xi) = \sum_{i=1}^{n} w_i.\mathtt{r}_i[k_i](\xi)$. The* expected total weighted sum *is defined as $ExpTot^{\sigma,\mathbf{k}}_{\mathcal{M}}[\mathbf{w}\cdot\mathtt{r}] = \max_{\pi\in\Pi} \int_{\xi} \mathbf{w}.\mathtt{r}[\mathbf{k}](\xi)\, \mathrm{d}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}$ for bounds $\leq$ and accordingly minimises over natures for $\geq$; for a given controller $\sigma$, we have:*

$$ExpTot^{\sigma,\mathbf{k}}_{\mathcal{M}}[\mathbf{w}\cdot\mathtt{r}] = \sum_{i=1}^{n} w_i \cdot ExpTot^{\sigma,k_i}_{\mathcal{M}}[\mathtt{r}_i].$$

### 8.1.1 Multi-objective Queries

Multi-objective properties for *IMDP*s essentially require multiple predicates to be satisfied at the same time under the same controller for every choice of the nature. We now explain how to formalise multi-objective queries for *IMDP*s.

**Definition 8.5 (Multi-objective predicate).** *A* multi-objective predicate *is a vector $\varphi = (\varphi_1, \ldots, \varphi_n)$ of reachability or reward predicates. We say that $\varphi$ is satisfied by IMDP $\mathcal{M}$ under controller $\sigma$ for every choice of nature $\pi \in \Pi$, denoted by $\mathcal{M} \downharpoonright_{\sigma} \models_{\Pi} \varphi$ if, for each $1 \leq i \leq n$, we have $\mathcal{M} \downharpoonright_{\sigma} \models_{\Pi} \varphi_i$. We refer to $\sigma$ as a robust controller. Furthermore, we call $\varphi$ a basic multi-objective predicate if it is of the form $([\mathtt{r}_1]^{\leq k_1}_{\geq r_1}, \ldots, [\mathtt{r}_n]^{\leq k_n}_{\geq r_n})$, i.e., it includes only lower-bounded reward predicates.*

We formulate multi-objective queries for *IMDP*s in three ways, namely *synthesis queries*, *quantitative queries* and *Pareto queries*. We first focus on the synthesis queries and discuss later the other types of queries. We formulate multi-objective synthesis queries for *IMDP*s as follows.

**Definition 8.6 (Synthesis Query).** *Given an IMDP $\mathcal{M}$ and a multi-objective predicate $\varphi$, the* synthesis query *asks if there exists a robust controller $\sigma \in \Sigma$ such that $\mathcal{M} \downharpoonright_{\sigma} \models_{\Pi} \varphi$.*

Note that the synthesis queries check for the existence of a robust controller that satisfies a multi-objective predicate $\varphi$ for every resolution of nature.

In order to avoid unusual behaviors in controller synthesis such as infinite total expected reward, we need to limit the usage of rewards by assuming reward-finiteness for the controllers that satisfy the reachability predicates in the given multi-objective query $\varphi$.

**Assumption 8.1 (Reward-finiteness).** *Suppose that an IMDP $\mathcal{M}$ and a synthesis query $\varphi$ are given. Let $\varphi = ([T_1]^{\leq k_1}_{\sim p_1}, \ldots, [T_n]^{\leq k_n}_{\sim p_n}, [\mathtt{r}_{n+1}]^{\leq k_{n+1}}_{\sim r_{n+1}}, \ldots, [\mathtt{r}_m]^{\leq k_m}_{\sim r_m})$. We say that $\varphi$ is reward-finite if for each $n+1 \leq i \leq m$ such that $k_i = \infty$, $\sup\{ExpTot^{\sigma,k_i}_{\mathcal{M}}[\mathtt{r}_i] \mid \mathcal{M} \downharpoonright_{\sigma} \models_{\Pi} ([T_1]^{\leq k_1}_{\sim p_1}, \ldots, [T_n]^{\leq k_n}_{\sim p_n})\} < \infty$.*

In the following, we discuss in detail how reward-finiteness assumption for a given *IMDP* $\mathcal{M}$ and a synthesis query $\varphi$ can be ensured and also provide a preprocessing procedure that removes actions with non-zero rewards from the end components of the *IMDP*.

In order to describe the procedure that checks Assumption 8.1, we first need to define a counterpart of end components of *MDP*s for *IMDP*s, to which we refer as a *strong end-component* (SEC). Intuitively, a SEC of an *IMDP* is a sub-*IMDP* for which there exists a controller that forces the sub-*IMDP* to remain in the end component and visit all its states infinitely often under any nature. It is referred to as strong because it is independent of the choice of nature. Formally,

**Definition 8.7 (Strong End-Component).** *A* strong end-component *(SEC) of an IMDP* $\mathcal{M}$ *is* $E_{\mathcal{M}} = (S', \mathscr{A}')$*, where* $S' \subseteq S$ *and* $\mathscr{A}' \subseteq \bigcup_{s \in S'} \mathscr{A}(s)$ *such that (1)* $\sum_{s' \in S'} \mathfrak{h}^a_{ss'} = 1$ *for every* $s \in S'$ *and* $a \in \mathscr{A}'(s)$*, and all* $\mathfrak{h}^a_s \in \mathscr{H}^a_s$*, and (2) for all* $s, s' \in S'$ *there is a finite path* $\xi = \xi[0] \cdots \xi[n]$ *such that* $\xi[0] = s$*,* $\xi[n] = s'$ *and for all* $0 \leq i \leq n-1$ *we have* $\xi[i] \in S'$ *and* $\xi(i) \in \mathscr{A}'$*.*

**Remark 8.1.** *The SECs of an IMDP* $\mathcal{M}$ *can be identified by using any end-component-search algorithm of MDPs on its underlying graph structure. That is, since the lower transition probability bounds of* $\mathcal{M}$ *are strictly greater than zero for the transitions whose upper probability bounds are non-zero, the underlying graph structure of* $\mathcal{M}$ *is identical to the graph structure of every MDP it contains. Therefore, a SEC of* $\mathcal{M}$ *is an end-component of every contained MDP, and vice versa.*

**Lemma 8.1.** *If state-action pair* $(s, a)$ *is not contained in a SEC, then*

$$\sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} occ^{\sigma}_{\pi}((s, a)) < \infty,$$

*where* $occ^{\sigma}_{\pi}((s, a))$ *denotes the expected total number of occurrences of* $(s, a)$ *under* $\sigma$ *and* $\pi$*.*

*Proof.* If $(s, a)$ is not contained in a SEC of $\mathcal{M}$, then starting from $s$ and under action $a$, the probability of returning to $s$ is less than one, independent of the choice of controller and nature. Then, the proof follows from basic results of probability theory. ■

**Proposition 8.1.** *Let* $E_{\mathcal{M}} = (S', \mathscr{A}')$ *denote a SEC of IMDP* $\mathcal{M}$*. Then, we have* $\sup_\sigma \{ ExpTot^{\sigma,\infty}_{\mathcal{M}}[\mathtt{r}] \mid \mathcal{M} \downharpoonright_\sigma \models_\Pi ([T_1]^{\leq k_1}_{\sim p_1}, \ldots, [T_n]^{\leq k_n}_{\sim p_n}) \} = \infty$ *for a reward structure* $\mathtt{r}$ *of* $\mathcal{M}$ *if and only if there is a controller* $\sigma$ *of* $\mathcal{M}$ *that* $\mathcal{M} \downharpoonright_\sigma \models_\Pi ([T_1]^{\leq k_1}_{\sim p_1}, \ldots, [T_n]^{\leq k_n}_{\sim p_n})$*,* $E_{\mathcal{M}}$ *is reachable under* $\sigma$*, and* $\mathtt{r}(\xi[i], \xi(i)) > 0$*, where* $\xi$ *is a path under* $\sigma$ *with* $\xi[i] \in S'$ *and* $\xi(i) \in \mathscr{A}'(\xi[i])$ *for some* $i \geq 0$*.*

*Proof.* We prove this proposition by adapting the proof from [FKN+11, Proposition 1].

Direction $\Rightarrow$. Assume that, for a reward structure $\mathtt{r}$, $\sup\{ ExpTot^{\sigma,\infty}_{\mathcal{M}}[\mathtt{r}] \mid \mathcal{M} \downharpoonright_\sigma \models_\Pi ([T_1]^{\leq k_1}_{\sim p_1}, \ldots, [T_n]^{\leq k_n}_{\sim p_n}) \} = \infty$. From Lemma 8.1, it follows that if state-action pair $(s, a)$ occurs infinitely often, $s$ and $a$ are contained in a SEC $E_{\mathcal{M}}$. Therefore, to satisfy the assumed condition, there must exist some controller $\sigma$ such that $\mathcal{M} \downharpoonright_\sigma \models_\Pi ([T_1]^{\leq k_1}_{\sim p_1}, \ldots, [T_n]^{\leq k_n}_{\sim p_n})$ and

a SEC is reachable, in which $\sigma$ picks action $a$ at reachable state $s$ with positive probability, and $\mathtt{r}(s,a) > 0$.

Direction $\Leftarrow$. Assume that there is a controller $\sigma$ such that $\mathcal{M} \downharpoonright_{\sigma} \models_{\Pi} ([T_1]^{\leq k_1}_{\sim p_1}, \ldots, [T_n]^{\leq k_n}_{\sim p_n})$, a SEC $E_{\mathscr{M}} = (S', \mathscr{A}')$ is reachable, and $\mathtt{r}(\xi[n], \xi(n)) > 0$, where $\xi$ is a finite path of length $n+1$ under $\sigma$ with $\xi[n] \in S'$ and $\xi(n) \in \mathscr{A}'(\xi[n])$ for some $n \geq 0$. To complete the proof, it is enough to show that there is a sequence of controllers $\{\sigma_k\}_{k\in\mathbb{N}}$ under which (i) the probabilistic predicates $[T_1]^{\leq k_1}_{\sim p_1}, \ldots, [T_n]^{\leq k_n}_{\sim p_n}$ are satisfied and (ii) $\lim_{k\to\infty} \mathit{ExpTot}^{\sigma_k,k}_{\mathcal{M}}[\mathtt{r}] = \infty$.
(i) Let $\xi[n] = s$ and $\xi(n) = a$. For $k \in \mathbb{N}$ consider $\sigma_k$ that

- for the paths that do not have the prefix $\xi$, $\sigma_k$ emulates $\sigma$.
- when the path $\xi$ is performed, $\sigma_k$ forces the system to stay in $E_{\mathscr{M}}$ containing $(s,a)$. After $k$ occurrences of $(s,a)$, the next time $s$ is visited, the controller $\sigma_k$ emulates $\sigma$ again as if the performed path segment after $\xi[n]$ was never executed.

Under $\sigma_k$, the reachability predicates are satisfied for any $k \in \mathbb{N}$. To see this, consider $\theta_k$ that maps each path $\xi$ of $\sigma$ to the paths of $\sigma_k$. We now have $\theta(\xi) \cap \theta(\xi') = \emptyset$ for all $\xi \neq \xi'$, and for all sets $\Omega$ and two natures $\pi$ and $\pi_k$, where $\pi_k$ emulates $\pi$ the same way $\sigma_k$ emulates $\sigma$, we have $\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}(\Omega) = \mathbf{Pr}^{\sigma_k,\pi_k}_{\mathcal{M}}(\theta(\Omega))$, independent of the choice of $\pi_k$ during the execution of the path segment that $\sigma_k$ forces the stay in $E_{\mathscr{M}}$. The satisfaction of the reachability predicates under each $\sigma_k$ follows from the fact that, for any path $\xi$ of $\sigma$, $\xi$ satisfies a reachability predicate iff each path in $\theta(\Omega)$ satisfies the reachability predicate.
(ii) To show that $\lim_{k\to\infty} \mathit{ExpTot}^{\sigma_k,k}_{\mathcal{M}}[\mathtt{r}] = \infty$, recall that the probability of reaching $(s,a)$ under $\sigma_k$ for the first time is some positive value $p_1$. From the properties of SEC, the probability of returning to $s$ within $l$ steps, where $l = |S|$, is also some positive value $p_2$. By construction, $(s,a)$ is picked $k$ times, therefore, $\mathit{ExpTot}^{\sigma_k,k}_{\mathcal{M}}[\mathtt{r}] \geq p_1 p_2 \frac{k}{l} \mathtt{r}(s,a)$, and hence, $\lim_{k\to\infty} \mathit{ExpTot}^{\sigma_k,k}_{\mathcal{M}}[\mathtt{r}] = \infty$. ■

We can now construct, from $\mathcal{M}$, an *IMDP* $\bar{\mathcal{M}}$ that is equivalent to $\mathcal{M}$ in terms of satisfaction of $\varphi$ but does not include actions with positive rewards in its SEC. The algorithm is similar to the one introduced in [FKN+11] for *MDP*s and is as follows. First, remove action $a$ from $\mathscr{A}(s)$ if $(s,a)$ is contained in a SEC and $\mathtt{r}(s,a) > 0$ for some maximizing reward structure $\mathtt{r}$. Second, recursively remove states with no outgoing transitions and transitions that lead to non-existent states until a fixpoint is reached.

**Proposition 8.2.** *There is a controller $\sigma$ of $\mathcal{M}$ such that $\mathit{ExpTot}^{\sigma,\infty}_{\mathcal{M}}[\mathtt{r}] = x < \infty$ and $\mathcal{M} \downharpoonright_{\sigma} \models_{\Pi} \varphi$ if and only if there is a controller $\bar{\sigma}$ of $\bar{\mathcal{M}}$ such that $\mathit{ExpTot}^{\bar{\sigma},\infty}_{\bar{\mathcal{M}}}[\mathtt{r}] = x$ and $\bar{\mathcal{M}} \downharpoonright_{\bar{\sigma}} \models_{\Pi} \varphi$.*

*Proof.* The proof follows straightforwardly from Proposition 8.1. ■

Due to Assumption 8.1, in the rest of this section we assume that all queries are reward-finite. Furthermore, for the soundness of our analysis we also require that for any *IMDP* $\mathcal{M}$ and $\varphi$ given as in Assumption 8.1: *(i)* each reward structure $\mathtt{r}_i$ assigns only non-negative values; *(ii)* $\varphi$ is reward-finite; and *(iii)* for indices $n+1 \leq i \leq m$ such that $k_i = \infty$, either all $\sim_i$s are $\leq$ or all are $\geq$.

### 8.1.2 Robust Controller Synthesis

We first study the computational complexity of multi-objective robust controller synthesis problem for *IMDP*s. Formally,

**Theorem 8.1.** *Given an IMDP $\mathcal{M}$ and a multi-objective predicate $\varphi$, the problem of synthesizing a controller $\sigma \in \Sigma$ such that $\mathcal{M} \downarrow_{\sigma} \models_{\Pi} \varphi$ is **PSPACE**-hard.*

In order to prove the theorem, we need to define the multiple reachability problem for *MDP*s. Formally,

**Definition 8.8 (Multiple reachability problem for *MDP*s).** *Given an MDP $\mathscr{M}$ and a reachability predicate described as a vector $\varphi = (\varphi_1, \ldots, \varphi_n)$ where $\varphi_j = [T_j]^{\leq k_j}_{\sim p_j}$ for $j \in \{1, \ldots, n\}$, the multiple reachability problem asks to check if there exists a controller $\sigma$ of $\mathscr{M}$ such that $\mathscr{M}, \sigma \models_{\Pi} \varphi$. The almost-sure multiple reachability problem restricts to $\sim = \geq$ and $p_j = 1$ for all $j \in \{1, \ldots, n\}$.*

The proof makes use of the following lemma:

**Lemma 8.2 (Complexity of the multiple reachability problem for *MDP*s [RRS15]).** *Given an MDP $\mathscr{M}$, the almost-sure multiple reachability problem is **PSPACE**-complete and controllers need exponential memory in the query size.*

*Proof.* We reduce the problem in Lemma 8.2 to the one under our analysis. In fact, any instance of the multiple reachability problem for *MDP* $\mathscr{M}$ can be seen as an instance of the multi-objective robust controller synthesis problem for an *IMDP* $\mathcal{M}$ generated from $\mathscr{M}$ by replacing all probability values with point intervals. Since the multiple reachability problem for *MDP*s is **PSPACE**-complete and the reduction is performed in polynomial time therefore, solving the robust controller synthesis problem for *IMDP*s is at least **PSPACE**-hard. ■

As the first step towards derivation of a solution approach for the robust controller synthesis problem, we need to convert all reachability predicates to reward predicates and therefore to transform an arbitrarily given query to a query over a basic predicate on a modified *IMDP*. This can be simply done by adding, once for all, a reward of one at the time of reaching the target set and also negating the objective of predicates with upper-bounded relational operators. We correct and extend the procedure in [FKP12] to reduce a general multi-objective predicate on an *IMDP* model to a basic form on a modified *IMDP*.

**Proposition 8.3 (Reduction to basic form).** *Given an IMDP $\mathcal{M} = (S, \bar{s}, \mathscr{A}, \mathtt{AP}, L, I)$ and a multi-objective predicate $\varphi = ([T_1]^{\leq k_1}_{\sim_1 p_1}, \ldots, [T_n]^{\leq k_n}_{\sim_n p_n}, [\mathtt{r}_{n+1}]^{\leq k_{n+1}}_{\sim_{n+1} r_{n+1}}, \ldots, [\mathtt{r}_m]^{\leq k_m}_{\sim_m r_m})$, let $\mathcal{M}' = (S', \bar{s}', \mathscr{A}', \mathtt{AP}', L', I')$ be the IMDP whose components are defined as follows:*

- $S' = S \times 2^{\{1,\ldots,n\}}$;
- $\bar{s}' = (\bar{s}, \emptyset)$;
- $\mathscr{A}' = \mathscr{A} \times 2^{\{1,\ldots,n\}}$;
- $\mathtt{AP}' = \mathtt{AP}$;

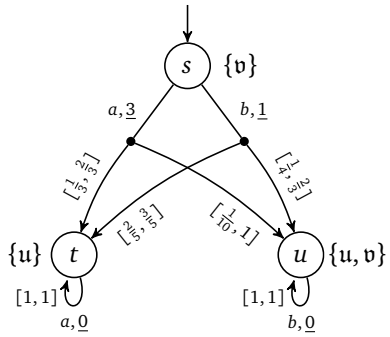



Figure 8.1: An Example of *IMDP*s.

- *for all* $s \in S$ *and* $\nu \subseteq \{1, \dots, n\}$, $L'(s, \nu) = L(s)$; *and*
- *for all* $s, s' \in S$, $a \in \mathscr{A}$, *and* $\nu, \nu', \nu'' \subseteq \{1, \dots, n\}$,

$$I'((s,\nu),(a,\nu'),(s',\nu'')) = \begin{cases} I(s,a,s') & \text{if } \nu' = \{\, i \mid s \in T_i \,\} \setminus \nu \text{ and } \nu'' = \nu \cup \nu', \\ 0 & \text{otherwise.} \end{cases}$$

*Now, let* $\varphi' = ([\mathtt{r}_{T_1}]^{\leq k_1+1}_{\geq p'_1}, \dots, [\mathtt{r}_{T_n}]^{\leq k_n+1}_{\geq p'_n}, [\bar{\mathtt{r}}_{n+1}]^{\leq k_{n+1}}_{\geq r'_{n+1}}, \dots, [\bar{\mathtt{r}}_m]^{\leq k_m}_{\geq r'_m})$ *where, for each* $i \in \{1, \dots, n\}$,

$$p'_i = \begin{cases} p_i & \text{if } \sim_i = \geq, \\ -p_i & \text{if } \sim_i = \leq; \end{cases} \quad \text{and} \quad \mathtt{r}_{T_i}((s,\nu),(a,\nu')) = \begin{cases} 1 & \text{if } i \in \nu' \text{ and } \sim_i = \geq, \\ -1 & \text{if } i \in \nu' \text{ and } \sim_i = \leq, \\ 0 & \text{otherwise;} \end{cases}$$

*and, for each* $j \in \{n+1, \dots, m\}$,

$$r'_j = \begin{cases} r_j & \text{if } \sim_j = \geq, \\ -r_j & \text{if } \sim_j = \leq; \end{cases} \quad \text{and} \quad \bar{\mathtt{r}}_j((s,\nu),(a,\nu')) = \begin{cases} \mathtt{r}_j(s,a) & \text{if } \sim_j = \geq, \\ -\mathtt{r}_j(s,a) & \text{if } \sim_j = \leq. \end{cases}$$

*Then* $\varphi$ *is satisfiable in* $\mathcal{M}$ *if and only if* $\varphi'$ *is satisfiable in* $\mathcal{M}'$.

Before proving the proposition formally, we first illustrate the reduction by an example.

**Example 8.1.** *Consider the IMDP* $\mathcal{M}$ *depicted in Figure 8.1. The set of states is* $S = \{s, t, u\}$ *with* $s$ *being the initial one. The set of actions is* $\mathscr{A} = \{a, b\}$, *and the non-zero transition probability intervals are* $I(s,a,t) = [\frac{1}{3}, \frac{2}{3}]$, $I(s,a,u) = [\frac{1}{10}, 1]$, $I(s,b,t) = [\frac{2}{5}, \frac{3}{5}]$, $I(s,b,u) = [\frac{1}{4}, \frac{2}{3}]$, *and* $I(t,a,t) = I(u,b,u) = [1,1]$. *Letters in curly brackets besides each circle denote the labels of each state and the underlined numbers indicate the reward structure* $\mathtt{r}$ *with* $\mathtt{r}(s,a) = 3$, $\mathtt{r}(s,b) = 1$, *and* $\mathtt{r}(t,a) = \mathtt{r}(u,b) = 0$. *Among the uncountable many distributions belonging to* $\mathscr{H}^a_s$, *two possible choices for nature* $\pi$ *on* $s$ *and* $a$ *are* $\pi(s,a) = \{(t, \frac{3}{5}), (u, \frac{2}{5})\}$ *and* $\pi(s,a) = \{(t, \frac{1}{3}), (u, \frac{2}{3})\}$. *Let us assume that the target set is* $T = \{t_1\}$ *and also consider* $\varphi = ([T]^{\leq 1}_{\geq \frac{1}{3}}, [\mathtt{r}]^{\leq 1}_{\geq \frac{1}{4}})$. *Execution of the reduction in Proposition 8.3, converts* $\varphi$ *to the property* $\varphi' = ([\mathtt{r}_T]^{\leq 2}_{\geq \frac{1}{3}}, [\mathtt{r}]^{\leq 1}_{\geq \frac{1}{4}})$ *on the modified*

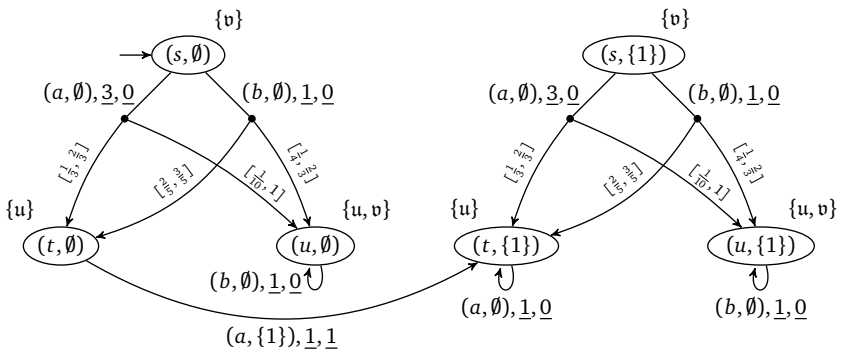



Figure 8.2: The *IMDP* $\mathcal{M}'$ generated from $\mathcal{M}$

*$\mathcal{M}'$ depicted in Figure 8.2. We show two different reward structure $\bar{\mathtt{r}}$ and $\mathtt{r}_T$ besides each action, respectively.* ♦

We now prove the proposition as follows.

*Proof.* Given a state $(s, \nu) \in S'$, let $\nu_e = \{\, i \in \{1, \dots, n\} \mid s \in T_i \,\} \setminus \nu$. By definition of the transition probability function, it follows that the only successors $(s', \nu')$ that can be reached from $(s, \nu)$ must have $\nu' = \nu \cup \nu_e$; moreover, the action performed for such a transition must be of the form $(a, \nu_e)$. This means that the sets $\nu_e$ and $\nu'$ are uniquely determined by the current state $(s, \nu)$; let $\nu \colon S' \to 2^{\{1,\dots,n\}}$ be the function such that $\nu(s, \nu) = \{\, i \in \{1, \dots, n\} \mid s \in T_i \,\} \setminus \nu$ for each $(s, \nu) \in S'$, $\nu_{\mathscr{A}} \colon S' \times \mathscr{A} \to \mathscr{A}'$ be the function such that $\nu_{\mathscr{A}}((s, \nu), a) = (a, \nu(s, \nu))$ for each $(s, \nu) \in S'$ and $a \in \mathscr{A}$, and $\nu_S \colon S' \times S \to S'$ be the function such that $\nu_S((s, \nu), s') = (s', \nu \cup \nu(s, \nu))$ for each $(s, \nu) \in S'$ and $s' \in S$.

It is immediate to see that every path $\xi'$ of $\mathcal{M}'$,

$$\xi' = (s_0, \nu_0)\, \mathfrak{h}^{(a_0, \nu_0')}_{(s_0,\nu_0)(s_1,\nu_1)}\, (s_1, \nu_1)\, \mathfrak{h}^{(a_1, \nu_1')}_{(s_1,\nu_1)(s_2,\nu_2)}\, (s_2, \nu_2) \dots,$$

is actually of the form

$$\xi' = (s_0, \nu_0)\, \mathfrak{h}^{\nu_{\mathscr{A}}((s_0,\nu_0),a_0)}_{(s_0,\nu_0)\nu_S((s_0,\nu_0),s_1)}\, (s_1, \nu_1)\, \mathfrak{h}^{\nu_{\mathscr{A}}((s_1,\nu_1),a_1)}_{(s_1,\nu_1)\nu_S((s_1,\nu_1),s_2)}\, (s_2, \nu_2) \dots$$

where $(s_{j+1}, \nu_{j+1}) = \nu_S((s_j, \nu_j), s_{j+1})$ for each $j \in \mathbb{N}$, i.e., $\nu_{j+1} = \nu_j \cup \nu(s_j, \nu_j)$. This means that we can define a bijection $\sharp \colon \mathit{Paths}^{\mathit{inf}}_{\mathcal{M}} \to \mathit{Paths}^{\mathit{inf}}_{\mathcal{M}'}$ as follows: given a path $\xi = s_0\, \mathfrak{h}^{a_0}_{s_0 s_1}\, s_1\, \mathfrak{h}^{a_1}_{s_1 s_2}\, s_2 \dots$ of $\mathcal{M}$, $\sharp(\xi)$ is defined as

$$\sharp(\xi) = (s_0, \nu_0)\, \mathfrak{h}^{(a_0, \nu_0')}_{(s_0,\nu_0)(s_1,\nu_1)}\, (s_1, \nu_1)\, \mathfrak{h}^{(a_1, \nu_1')}_{(s_1,\nu_1)(s_2,\nu_2)}\, (s_2, \nu_2) \dots$$

where $\nu_0 = \emptyset$ and for each $j \in \mathbb{N}$, $(a_j, \nu_j') = \nu_{\mathscr{A}}((s_j, \nu_j), a_j)$ and $(s_{j+1}, \nu_{j+1}) = \nu_S((s_j, \nu_j), s_j)$.

The inverse $\flat \colon \mathit{Paths}^{\mathit{inf}}_{\mathcal{M}'} \to \mathit{Paths}^{\mathit{inf}}_{\mathcal{M}}$ of $\sharp$ is just the projection on $\mathcal{M}$: given a path $\xi' = (s_0, \nu_0)\, \mathfrak{h}^{(a_0, \nu_0')}_{(s_0,\nu_0)(s_1,\nu_1)}\, (s_1, \nu_1)\, \mathfrak{h}^{(a_1, \nu_1')}_{(s_1,\nu_1)(s_2,\nu_2)}\, (s_2, \nu_2) \dots$ of $\mathcal{M}'$, $\flat(\xi')$ is defined as

$$\flat(\xi') = s_0\, \mathfrak{h}^{a_0}_{s_0 s_1}\, s_1\, \mathfrak{h}^{a_1}_{s_1 s_2}\, s_2 \dots.$$

Moreover, since the sequence of sets $v_0 v_1 v_2 \dots$ is monotonic non-decreasing with respect to the subset inclusion partial order, we have that, for a given $i \in \{1, \dots, n\}$, if $i \in v_N$ for some $N \in \mathbb{N}$, then there exists exactly one $l \in \mathbb{N}$ such that $i \notin v_j$ for each $0 \leq j < l$ and $i \in v_j$ for each $j \geq l$, i.e., $s_l$ is the first time a state $s \in T_i$ occurs along $\flat(\xi')$. Therefore, it follows that $i \in \nu(s_l, v_l)$ while $i \notin \nu(s_j, v_j)$ for each $j \in \mathbb{N} \setminus \{l\}$. This implies that $\mathbf{r}_{T_i}(\xi'[l], \xi'(l)) = 1$ if $\sim_i = \geq$ or $\mathbf{r}_{T_i}(\xi'[l], \xi'(l)) = -1$ if $\sim_i = \leq$ while $\mathbf{r}_{T_i}(\xi'[j], \xi'(j)) = 0$ for each $j \in \mathbb{N} \setminus \{l\}$, thus

$$\mathbf{r}_{T_i}[k](\xi') = \begin{cases} 1 & \text{if } l < k \text{ and } \sim_i = \geq, \\ -1 & \text{if } l < k \text{ and } \sim_i = \leq, \\ 0 & \text{otherwise.} \end{cases}$$

Note that, if $i \notin v_j$ for each $j \in \mathbb{N}$, then this means that $i \notin \nu(s_j, v_j)$ for each $j \in \mathbb{N}$, thus $\mathbf{r}_{T_i}(\xi'[j], \xi'(j)) = 0$ for each $j \in \mathbb{N}$ and $\mathbf{r}_{T_i}[k](\xi') = 0$.

Similarly, for each $h \in \{n+1, \dots, m\}$, we get that $\bar{\mathbf{r}}_h[k](\xi') = \mathbf{r}_h[k](\xi)$ if $\sim_h = \geq$ and $\bar{\mathbf{r}}_h[k](\xi') = -\mathbf{r}_h[k](\xi)$ if $\sim_h = \leq$.

We are now ready to prove the statement of the proposition, by considering the two implications separately.

Suppose that $\varphi$ is satisfiable in $\mathcal{M}$: by definition, it follows that there exists a controller $\sigma$ of $\mathcal{M}$ such that $\mathcal{M} \downharpoonright_\sigma \models_\Pi \varphi$, that is, $\mathcal{M} \downharpoonright_\sigma \models_\Pi [T_i]^{\leq k_i}_{\sim_i p_i}$ for each $i \in \{1, \dots, n\}$ and $\mathcal{M} \downharpoonright_\sigma \models_\Pi [\mathbf{r}_h]^{\leq k_h}_{\sim_h r_h}$ for each $h \in \{n+1, \dots, m\}$. Let $\sigma'$ be the controller of $\mathcal{M}'$ such that, for each finite path $\xi' \in Paths^{fin}_{\mathcal{M}'}$ and action $a \in \mathscr{A}$, $\sigma(\xi')(\nu_{\mathscr{A}}(last(\xi'), a)) = \sigma(\flat(\xi'))(a)$, 0 otherwise. Intuitively, $\sigma'$ chooses the next action $(a, \nu)$ exactly as $\sigma$ chooses $a$ since $\nu$ is uniquely determined by $\xi'$. We claim that $\sigma'$ is such that $\mathcal{M}' \downharpoonright_{\sigma'} \models_\Pi \varphi'$.

Let $i \in \{1, \dots, n\}$ and consider $\varphi'_i = [\mathbf{r}_{T_i}]^{\leq k_i + 1}_{\geq p'_i}$: there are two cases depending on the original bound $\sim_i$.

If $\sim_i = \geq$, then $[\mathbf{r}_{T_i}]^{\leq k_i+1}_{\geq p'_i} = [\mathbf{r}_{T_i}]^{\leq k_i+1}_{\geq p_i}$; $\mathcal{M}' \downharpoonright_{\sigma'} \models_{\Pi'} [\mathbf{r}_{T_i}]^{\leq k_i+1}_{\geq p_i}$ if and only if $\min_{\pi' \in \Pi'} \int_{\xi'} \mathbf{r}_{T_i}[k_i + 1](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma', \pi'}_{\mathcal{M}'} \geq p_i$. Since for each path $\xi' \in Paths^{inf}_{\mathcal{M}'}$, $\mathbf{r}_{T_i}[k_i + 1](\xi') = 1$ if there exists $l < k_i + 1$ such that $\flat(\xi')[l] \in T_i$, $\mathbf{r}_{T_i}[k_i + 1](\xi') = 0$ otherwise, by the way $I'$ and $\sigma'$ are defined it follows that $\min_{\pi' \in \Pi'} \int_{\xi'} \mathbf{r}_{T_i}[k_i + 1](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma', \pi'}_{\mathcal{M}'} = \min_{\pi \in \Pi} \mathbf{Pr}^{\sigma, \pi}_{\mathcal{M}} \{ \xi \in Paths^{inf}_{\mathcal{M}} \mid \exists l \leq k : \xi[l] \in T_i \}$. Since by hypothesis $\varphi$ is satisfiable in $\mathcal{M}$, then it follows that $\min_{\pi \in \Pi} \mathbf{Pr}^{\sigma, \pi}_{\mathcal{M}} \{ \xi \in Paths^{inf}_{\mathcal{M}} \mid \exists l \leq k : \xi[l] \in T_i \} \geq p_i$, thus $\min_{\pi' \in \Pi'} \int_{\xi'} \mathbf{r}_{T_i}[k_i + 1](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma', \pi'}_{\mathcal{M}'} \geq p_i$ holds as well, hence $\mathcal{M}' \downharpoonright_{\sigma'} \models_{\Pi'} [\mathbf{r}_{T_i}]^{\leq k_i+1}_{\geq p_i} = [\mathbf{r}_{T_i}]^{\leq k_i+1}_{\geq p'_i}$ is satisfied, as required.

Consider now the second case: if $\sim_i = \leq$, then $[\mathbf{r}_{T_i}]^{\leq k_i+1}_{\geq p'_i} = [\mathbf{r}_{T_i}]^{\leq k_i+1}_{\geq -p_i}$; $\mathcal{M}' \downharpoonright_{\sigma'} \models_{\Pi'} [\mathbf{r}_{T_i}]^{\leq k_i+1}_{\geq -p_i}$ if and only if $\min_{\pi' \in \Pi'} \int_{\xi'} \mathbf{r}_{T_i}[k_i + 1](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma', \pi'}_{\mathcal{M}'} \geq -p_i$. Since for each path $\xi' \in Paths^{inf}_{\mathcal{M}'}$, $\mathbf{r}_{T_i}[k_i + 1](\xi') = -1$ if there exists $l < k_i + 1$ such that $\flat(\xi')[l] \in T_i$, $\mathbf{r}_{T_i}[k_i + 1](\xi') = 0$ otherwise, by the way $I'$ and $\sigma'$ are defined it follows that $\min_{\pi' \in \Pi'} \int_{\xi'} \mathbf{r}_{T_i}[k_i + 1](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma', \pi'}_{\mathcal{M}'} = -\max_{\pi \in \Pi} \mathbf{Pr}^{\sigma, \pi}_{\mathcal{M}} \{ \xi \in Paths^{inf}_{\mathcal{M}} \mid \exists l \leq k : \xi[l] \in T_i \}$. Since by hypothesis we have that $\varphi$ is satisfiable in $\mathcal{M}$, then it follows that $\max_{\pi \in \Pi} \mathbf{Pr}^{\sigma, \pi}_{\mathcal{M}} \{ \xi \in Paths^{inf}_{\mathcal{M}} \mid \exists l \leq k : \xi[l] \in T_i \} \leq p_i$, thus $\min_{\pi' \in \Pi'} \int_{\xi'} \mathbf{r}_{T_i}[k_i + 1](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma', \pi'}_{\mathcal{M}'} \geq -p_i$ holds as well, hence $\mathcal{M}' \downharpoonright_{\sigma'} \models_{\Pi'} [\mathbf{r}_{T_i}]^{\leq k_i+1}_{\geq -p_i} = [\mathbf{r}_{T_i}]^{\leq k_i+1}_{\geq p'_i}$ is satisfied, as required.

This completes the analysis of the case $\varphi'_i = [\mathbf{r}_{T_i}]^{\leq k_i+1}_{\geq p'_i}$ for each $i \in \{1,\dots,n\}$.

Let $h \in \{n+1,\dots,m\}$ and consider $\varphi'_h = [\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq r'_h}$: there are two cases depending on the original bound $\sim_h$.

If $\sim_h = \geq$, then $[\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq r'_h} = [\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq r_h}$; $\mathcal{M}' \downarrow_{\sigma'} \models_{\Pi'} [\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq r_h}$ holds if and only if $\min_{\pi' \in \Pi'} \int_{\xi'} \bar{\mathbf{r}}_h[k_h](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'} \geq r_h$ holds. Since for each path $\xi' \in \mathit{Paths}^{\mathit{inf}}_{\mathcal{M}'}$, $\bar{\mathbf{r}}_h[k](\xi') = \mathbf{r}_h[k](\flat(\xi'))$, by the way the components $I'$, $\bar{\mathbf{r}}_h$, and $\sigma'$ are defined it follows that $\min_{\pi' \in \Pi'} \int_{\xi'} \bar{\mathbf{r}}_h[k_h](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'} = \min_{\pi \in \Pi} \int_{\xi} \mathbf{r}_h[k_h](\xi) \, \mathrm{d}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}$. Since by hypothesis $\varphi$ is satisfiable in $\mathcal{M}$, then it follows that $\min_{\pi \in \Pi} \int_{\xi} \mathbf{r}_h[k_h](\xi) \, \mathrm{d}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}} \geq r_h$, thus $\min_{\pi' \in \Pi'} \int_{\xi'} \bar{\mathbf{r}}_h[k_h](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'} \geq r_h$ holds as well, hence $\mathcal{M}' \downarrow_{\sigma'} \models_{\Pi'} [\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq r_h} = [\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq r'_h}$ is satisfied, as required.

Consider now the second case: if $\sim_h = \leq$, then $[\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq r'_h} = [\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq -r_h}$; $\mathcal{M}' \downarrow_{\sigma'} \models_{\Pi'} [\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq -r_h}$ if and only if $\min_{\pi' \in \Pi'} \int_{\xi'} \bar{\mathbf{r}}_h[k_h](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'} \geq -r_h$. Since for each path $\xi' \in \mathit{Paths}^{\mathit{inf}}_{\mathcal{M}'}$, $\bar{\mathbf{r}}_h[k](\xi') = -\mathbf{r}_h[k](\flat(\xi'))$, by the way $I'$, $\bar{\mathbf{r}}_h$, and $\sigma'$ are defined it follows that $\min_{\pi' \in \Pi'} \int_{\xi'} \bar{\mathbf{r}}_h[k_h](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'} = -\max_{\pi \in \Pi} \int_{\xi} \mathbf{r}_h[k_h](\xi) \, \mathrm{d}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}$. Since by hypothesis $\varphi$ is satisfiable in $\mathcal{M}$, then it follows that $\max_{\pi \in \Pi} \int_{\xi} \mathbf{r}_h[k_h](\xi) \, \mathrm{d}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}} \leq r_h$, thus $\min_{\pi' \in \Pi'} \int_{\xi'} \bar{\mathbf{r}}_h[k_h](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'} \geq -r_h$ holds as well, hence $\mathcal{M}' \downarrow_{\sigma'} \models_{\Pi'} [\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq -r_h} = [\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq r'_h}$ is satisfied, as required.

This completes the analysis of the case $\varphi'_h = [\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq r'_h}$ for each $h \in \{n+1,\dots,m\}$; since $\mathcal{M}' \downarrow_{\sigma'} \models_{\Pi'} \varphi'_j$ for each $j \in \{1,\dots,m\}$, it follows that $\varphi$ is satisfiable in $\mathcal{M}'$, as required to prove that "if $\varphi$ is satisfiable in $\mathcal{M}$, then $\varphi'$ is satisfiable in $\mathcal{M}'$".

Suppose now the other implication, namely "if $\varphi'$ is satisfiable in $\mathcal{M}'$, then $\varphi$ is satisfiable in $\mathcal{M}$" and assume that $\varphi'$ is satisfiable in $\mathcal{M}'$: by definition, it follows that there exists a controller $\sigma'$ of $\mathcal{M}'$ such that $\mathcal{M}' \downarrow_{\sigma'} \models_{\Pi'} \varphi'$, that is, $\mathcal{M}' \downarrow_{\sigma'} \models_{\Pi'} [\mathbf{r}_{T_i}]^{\leq k_i+1}_{\geq p'_i}$ for each $i \in \{1,\dots,n\}$ and $\mathcal{M}' \downarrow_{\sigma'} \models_{\Pi'} [\bar{\mathbf{r}}_h]^{\leq k_h}_{\geq r'_h}$ for each $h \in \{n+1,\dots,m\}$. Let $\sigma$ be the controller of $\mathcal{M}$ such that, for each finite path $\xi \in \mathit{Paths}^{\mathit{fin}}_{\mathcal{M}}$ and action $a \in \mathscr{A}$, $\sigma(\xi)(a) = \sigma'(\sharp(\xi))(a,\nu)$, 0 otherwise, where $(a,\nu) = \nu_{\mathscr{A}}(\mathit{last}(\sharp(\xi)), a)$. Intuitively, $\sigma$ chooses the next action $a$ exactly as $\sigma'$ chooses $(a,\nu)$ since $\nu$ is uniquely determined by $\xi'$. We claim that $\sigma$ is such that $\mathcal{M} \downarrow_{\sigma} \models_{\Pi} \varphi$.

Let $i \in \{1,\dots,n\}$ and consider $\varphi_i = [T_i]^{\leq k_i}_{\sim_i p_i}$: there are two cases depending on the bound $\sim_i$.

If $\sim_i = \geq$, then $\mathcal{M} \downarrow_{\sigma} \models_{\Pi} [T_i]^{\leq k_i}_{\geq p_i}$ if and only if $\min_{\pi \in \Pi} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}\{\, \xi \in \mathit{Paths}^{\mathit{inf}}_{\mathcal{M}} \mid \exists l \leq k : \xi[l] \in T_i \,\} \geq p_i$. Since for each path $\xi \in \mathit{Paths}^{\mathit{inf}}_{\mathcal{M}}$, $\mathbf{r}_{T_i}[k_i+1](\sharp(\xi)) = 1$ if there exists $l < k_i + 1$ such that $\xi[l] \in T_i$, $\mathbf{r}_{T_i}[k_i+1](\sharp(\xi)) = 0$ otherwise, by the way $I'$ and $\sigma$ are defined it follows that $\min_{\pi \in \Pi} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}\{\, \xi \in \mathit{Paths}^{\mathit{inf}}_{\mathcal{M}} \mid \exists l \leq k : \xi[l] \in T_i \,\} = \min_{\pi' \in \Pi'} \int_{\xi'} \mathbf{r}_{T_i}[k_i+1](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'}$. Since by hypothesis $\varphi'$ is satisfiable in $\mathcal{M}'$, then it follows that $\min_{\pi' \in \Pi'} \int_{\xi'} \mathbf{r}_{T_i}[k_i+1](\xi') \, \mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'} \geq p_i$, thus $\min_{\pi \in \Pi} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}\{\, \xi \in \mathit{Paths}^{\mathit{inf}}_{\mathcal{M}} \mid \exists l \leq k : \xi[l] \in T_i \,\} \geq p_i$ holds as well, hence $\mathcal{M} \downarrow_{\sigma} \models_{\Pi} [T_i]^{\leq k_i}_{\geq p_i} = [T_i]^{\leq k_i}_{\sim_i p_i}$ is satisfied, as required.

Consider now the second case: If $\sim_i = \leq$, then $\mathcal{M} \downarrow_{\sigma} \models_{\Pi} [T_i]^{\leq k_i}_{\leq p_i}$ if and only if

$\max_{\pi\in\Pi} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}\{\xi \in Paths^{inf}_{\mathcal{M}} \mid \exists l \leq k : \xi[l] \in T_i\} \leq p_i$. Since for each path $\xi \in Paths^{inf}_{\mathcal{M}}$, $\mathtt{r}_{T_i}[k_i+1](\sharp(\xi)) = -1$ if there exists $l < k_i + 1$ such that $\xi[l] \in T_i$, $\mathtt{r}_{T_i}[k_i+1](\sharp(\xi)) = 0$ otherwise, by the way $I'$ and $\sigma$ are defined it follows that $\max_{\pi\in\Pi} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}\{\xi \in Paths^{inf}_{\mathcal{M}} \mid \exists l \leq k : \xi[l] \in T_i\} = -\min_{\pi'\in\Pi'} \int_{\xi'} \mathtt{r}_{T_i}[k_i+1](\xi')\,\mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'}$. Since by hypothesis $\varphi'$ is satisfiable in $\mathcal{M}'$, then it follows that $\min_{\pi'\in\Pi'} \int_{\xi'} \mathtt{r}_{T_i}[k_i+1](\xi')\,\mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'} \geq -p_i$, thus $\max_{\pi\in\Pi} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}\{\xi \in Paths^{inf}_{\mathcal{M}} \mid \exists l \leq k : \xi[l] \in T_i\} \leq p_i$ holds as well, hence $\mathcal{M} \downharpoonright_\sigma \models_\Pi [T_i]^{\leq k_i}_{\leq p_i} = [T_i]^{\leq k_i}_{\sim_i p_i}$ is satisfied, as required.

This completes the analysis of the case $\varphi_i = [T_i]^{\leq k_i}_{\sim_i p_i}$ for each $i \in \{1,\ldots,n\}$.

Let $h \in \{n+1,\ldots,m\}$ and consider $\varphi_h = [\mathtt{r}_h]^{\leq k_h}_{\sim_h r_h}$: there are two cases depending on the original bound $\sim_h$.

If $\sim_h = \geq$, then $\mathcal{M} \downharpoonright_\sigma \models_\Pi [\mathtt{r}_h]^{\leq k_h}_{\geq r_h}$ if and only if $\min_{\pi\in\Pi} \int_\xi \mathtt{r}_h[k_h](\xi)\,\mathrm{d}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}} \geq r_h$. Since for each path $\xi \in Paths^{inf}_{\mathcal{M}}$, $\bar{\mathtt{r}}_h[k](\sharp(\xi)) = \mathtt{r}_h[k](\xi)$, by the way $I'$, $\bar{\mathtt{r}}_h$, and $\sigma$ are defined it follows that $\min_{\pi\in\Pi} \int_\xi \mathtt{r}_h[k_h](\xi)\,\mathrm{d}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}} = \min_{\pi'\in\Pi'} \int_{\xi'} \bar{\mathtt{r}}_h[k_h](\xi')\,\mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'}$. Since by hypothesis $\varphi'$ is satisfiable in $\mathcal{M}'$, then $\min_{\pi'\in\Pi'} \int_{\xi'} \bar{\mathtt{r}}_h[k_h](\xi')\,\mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'} \geq r_h$, thus $\min_{\pi\in\Pi} \int_\xi \mathtt{r}_h[k_h](\xi)\,\mathrm{d}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}} \geq r_h$ holds as well, hence $\mathcal{M} \downharpoonright_\sigma \models_\Pi [\mathtt{r}_h]^{\leq k_h}_{\geq r_h} = [\mathtt{r}_h]^{\leq k_h}_{\sim_h r_h}$ is satisfied, as required.

Consider now the second case: if $\sim_h = \leq$, then $\mathcal{M} \downharpoonright_\sigma \models_\Pi [\mathtt{r}_h]^{\leq k_h}_{\leq r_h}$ if and only if $\max_{\pi\in\Pi} \int_\xi \mathtt{r}_h[k_h](\xi)\,\mathrm{d}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}} \leq r_h$. Since for each path $\xi \in Paths^{inf}_{\mathcal{M}}$, $-\bar{\mathtt{r}}_h[k](\sharp(\xi)) = \mathtt{r}_h[k](\xi)$, by the definition of the components $I'$, $\bar{\mathtt{r}}_h$, and $\sigma$ it is the case that $\max_{\pi\in\Pi} \int_\xi \mathtt{r}_h[k_h](\xi)\,\mathrm{d}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}} = -\min_{\pi'\in\Pi'} \int_{\xi'} \bar{\mathtt{r}}_h[k_h](\xi')\,\mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'}$. Since by hypothesis $\varphi'$ is satisfiable in $\mathcal{M}'$, then $\min_{\pi'\in\Pi'} \int_{\xi'} \bar{\mathtt{r}}_h[k_h](\xi')\,\mathrm{d}\mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M}'} \geq -r_h$, thus $\max_{\pi\in\Pi} \int_\xi \mathtt{r}_h[k_h](\xi)\,\mathrm{d}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}} \leq r_h$ holds as well, hence $\mathcal{M} \downharpoonright_\sigma \models_\Pi [\mathtt{r}_h]^{\leq k_h}_{\leq r_h} = [\mathtt{r}_h]^{\leq k_h}_{\sim_h r_h}$ is satisfied, as required.

This completes the analysis of the case $\varphi_h = [\mathtt{r}_h]^{\leq k_h}_{\sim_h r_h}$ for each $h \in \{n+1,\ldots,m\}$; since $\mathcal{M} \downharpoonright_\sigma \models_\Pi \varphi_j$ for each $j \in \{1,\ldots,m\}$, it follows that $\varphi$ is satisfiable in $\mathcal{M}$, as required to prove that "if $\varphi'$ is satisfiable in $\mathcal{M}'$, then $\varphi$ is satisfiable in $\mathcal{M}$".

Having proved both implications, the statement of the proposition "$\varphi$ is satisfiable in $\mathcal{M}$ if and only if $\varphi'$ is satisfiable in $\mathcal{M}'$" holds, as required. ■

By means of Proposition 8.3, for robust strategy synthesis we therefore need to only consider the basic multi-objective predicates of the form $([\mathtt{r}_1]^{\leq k_1}_{\geq r_1}, \ldots, [\mathtt{r}_n]^{\leq k_n}_{\geq r_n})$ for our purpose of robust controller synthesis. For a basic multi-objective predicate, we define its Pareto curve as follows.

**Definition 8.9** (**Pareto curve of a multi-objective predicate**). *Given an IMDP $\mathcal{M}$ and a basic multi-objective predicate $\varphi = ([\mathtt{r}_1]^{\leq k_1}_{\geq r_1}, \ldots, [\mathtt{r}_n]^{\leq k_n}_{\geq r_n})$, we define the set of achievable values with respect to $\varphi$ as $A_{\mathcal{M},\varphi} := \{(r_1,\ldots,r_n) \in \mathbb{R}^n \mid ([\mathtt{r}_1]^{\leq k_1}_{\geq r_1}, \ldots, [\mathtt{r}_n]^{\leq k_n}_{\geq r_n})$ is satisfiable$\}$. We define the Pareto curve of $\varphi$ to be the Pareto curve of $A_{\mathcal{M},\varphi}$ and denote it by $\mathcal{P}_{\mathcal{M},\varphi}$.*

**Example 8.2.** *In Figure 8.3 we show the Pareto curve for the property specified in Example 8.1. As we see, until required probability $\frac{1}{3}$ to reach $T$, the maximal reward value is 3. Afterwards, the reward obtainable linearly decreases, until at required probability $\frac{2}{5}$ it is just 1. For higher required probabilities, the problem becomes infeasible. The reason for this behaviour is that, up to minimal*

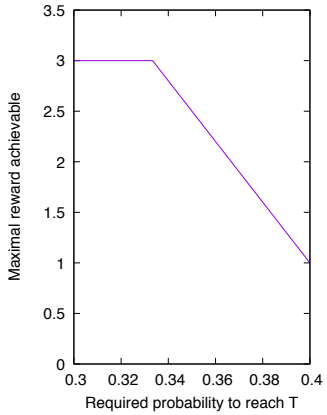



Figure 8.3: Pareto curve for the property $([\mathtt{r}_T]^{\leq 2}_{\max}, [\mathtt{r}]^{\leq 1}_{\max})$.

**Algorithm 10:** Algorithm for solving robust synthesis queries

**Input:** An *IMDP* $\mathcal{M}$, multi-objective predicate $\varphi = ([\mathtt{r}_1]^{\leq k_1}_{\geq r_1}, \ldots, [\mathtt{r}_n]^{\leq k_n}_{\geq r_n})$
**Output:** true if there exists a controller $\sigma \in \Sigma$ such that $\mathcal{M}\downarrow_\sigma \models_\Pi \varphi$, false if not.

```
1  begin
2      X = ∅; r = (r_1, ..., r_n);
3      k = (k_1, ..., k_n); r = (r_1, ..., r_n);
4      while r ∉ X↓ do
5          Find w separating r from X↓;
6          Find controller σ maximizing ExpTot^{σ,k}_M[w · r];
7          g := (ExpTot^{σ,k_i}_M[r_i])_{1≤i≤n};
8          if w · g < w · r then
9              return false;
10         X = X ∪ {g};
11     return true;
```

*probability* $\frac{1}{3}$*, action* $a$ *can be chosen in state* $s$*, because the lower interval bound to reach* $t$ *is* $\frac{1}{3}$*, which in turn leads to a reward of* 3 *being obtained. For higher reachability probabilities, choosing action* $b$ *with a certain probability is required, which however provides a lower reward. There is no controller with which* $t$ *is reached with a probability larger than* $\frac{2}{5}$*.* ♦

It is not difficult to see that the Pareto curve is in general an infinite set, and therefore, it is not possible to derive an exact representation of it in polynomial time. However, it can be shown that an $\varepsilon$-approximation of it can be computed efficiently [EKVY07].

In the rest of the section, we describe an algorithm to solve the synthesis query. We follow the well-known *normalization* approach in order to solve the multi-objective predicate which is essentially based on normalizing multiple objectives into one single objective. It is known that the optimal solution of the normalized (single-objective) predicate, if it exists, is the Pareto optimal solution of the multi-objective predicate [Ehr06].

The robust synthesis procedure is detailed in Algorithm 10. This algorithm basically aims to construct a sequential approximation to the Pareto curve $\mathcal{P}_{\mathcal{M},\varphi}$ while the quality of approximations gets better and more precise along the iterations. In other words, along the course of Algorithm 10 a sequence of weight vectors $\mathbf{w}$ are generated and corresponding to each of them, a $\mathbf{w}$-weighted sum of $n$ objectives is optimized through lines 6-7. The optimal controller $\sigma$ is then used in order to generate a point $\mathbf{g}$ on the Pareto curve $\mathcal{P}_{\mathcal{M},\varphi}$. We collect all these points in the set $X$. The multi-objective predicate $\varphi$ is satisfiable once we realize that $\mathbf{r}$ belongs to $X{\downarrow}$.

The optimal controllers for the multi-objective robust synthesis queries are constructed following the approach of [FKP12] and as a result of termination of Algorithm 10. In particular, when Algorithm 10 terminates, a sequence of points $\mathbf{g}^1, \ldots, \mathbf{g}^t$ on the Pareto curve $\mathcal{P}_{\mathcal{M},\varphi}$ are generated each of which corresponds to a deterministic controller $\sigma_{\mathbf{g}^j}$ for the current point $\mathbf{g}^j$. The resulting optimal controller $\sigma_{opt}$ is subsequently constructed from these using a randomized weight vector $\alpha \in \mathbb{R}^t$ satisfying $r_i \leq \sum_{j=1}^{t} \alpha_i \cdot {g_i}^j$. We will discuss the generation of randomized controllers in detail later in section 8.1.4.

**Remark 8.2.** *It is worthwhile to mention that the synthesis query for IMDPs cannot be solved on the MDPs generated from IMDPs by computing all feasible extreme transition probabilities and then applying the algorithm in [FKP12]. The latter is a valid approach provided the cooperative semantics is applied for resolving the two sources of nondeterminisms in IMDPs. With respect to the competitive semantics needed here, one can instead transform IMDPs to $2\frac{1}{2}$-player games [BKW14] and then along the lines of the previous approach apply the algorithm in [CFK+13]. Unfortunately, the transformation to (MDPs or) $2\frac{1}{2}$-player games induces an exponential blow up, adding an exponential factor to the worst case time complexity of the decision problem. Our algorithm avoids this by solving the robust synthesis problem directly on the IMDP so that the core part, i.e., lines 6- 7 of Algorithm 10 can be solved with time complexity polynomial in $|\mathcal{M}|$.*

Algorithm 11 represents a value iteration-based algorithm which slightly extends the value iteration-based algorithm in [FKP12] and adjusts it for *IMDP* models by encoding the notion of robustness. More precisely, the core difference is indicated in lines 7 and 17 where the optimal controller is computed so as to be robust against any choice of nature.

**Theorem 8.2.** *Algorithm 10 is sound, complete and has runtime exponential in $|\mathcal{M}|$, $\mathbf{k}$, and $n$.*

*Proof.* In every iteration of the loop in Algorithm 10, a point $\mathbf{g}$ on a unique face of the Pareto curve is identified. The number of faces of the Pareto curve $\mathcal{P}_{\mathcal{M},\varphi}$ is, in the worst case, exponential in $|\mathcal{M}|$, $\mathbf{k}$, and $n$ [EKVY07]. Therefore, termination of Algorithm 10 is guaranteed and the correctness is ensured as a result of the correctness of Algorithm 1 in [FKP12]. The soundness and completeness of the Algorithm 10 is followed by the fact that in every iteration of the algorithm through lines 6-7, the individual model checking problems can be solved in polynomial time in $|\mathcal{M}|$ by formulating the weighted sum of $n$ objectives as a linear programming problem. To see this, without loss of generality, assume that $k_i = \infty$ for all $i \in \{1, \ldots, n\}$. Therefore, following the approach in [Pug14], the problem of maximizing the $ExpTot^{\sigma,\mathbf{k}}_{\mathcal{M}}[\mathbf{w} \cdot \mathbf{r}]$ across the range of controllers $\sigma \in \Sigma$ can be formulated

**Algorithm 11:** Value iteration-based algorithm to solve lines 6-7 of Algorithm 10

**Input:** An *IMDP* $\mathcal{M}$, weight vector $w$, reward structures $\mathtt{r} = (\mathtt{r}_1, \ldots, \mathtt{r}_n)$, time-bound vector $\mathbf{k} \in (\mathbb{N} \cup \{\infty\})^n$, threshold $\varepsilon$

**Output:** controller $\sigma$ maximizing $ExpTot^{\sigma,\mathbf{k}}_{\mathcal{M}}[\mathbf{w} \cdot \mathtt{r}]$, $\mathbf{g} := (ExpTot^{\sigma,k_i}_{\mathcal{M}}[\mathtt{r}_i])_{1 \le i \le n}$

1 **begin**
2 $\quad \mathbf{x} = 0$; $\mathbf{x}^1 := 0$; …; $\mathbf{x}^n := 0$;
3 $\quad \mathbf{y} = 0$; $\mathbf{y}^1 := 0$; …; $\mathbf{y}^n := 0$;
4 $\quad \sigma^\infty(s) = \bot$ for all $s \in S$
5 $\quad$ **while** $\delta > \varepsilon$ **do**
6 $\quad\quad$ **foreach** $s \in S$ **do**
7 $\quad\quad\quad$ $y_s := \max_{a \in \mathscr{A}(s)}(\sum_{\{i \mid k_i = \infty\}} w_i \cdot \mathtt{r}_i(s,a) + \min_{\mathfrak{h}^a_s \in \mathscr{H}^a_s} \sum_{s' \in S} \mathfrak{h}^a_s(s') \cdot x_{s'})$;
8 $\quad\quad\quad$ $\sigma^\infty(s) := \arg\max_{a \in \mathscr{A}(s)}(\sum_{\{i \mid k_i = \infty\}} w_i \cdot \mathtt{r}_i(s,a) + \min_{\mathfrak{h}^a_s \in \mathscr{H}^a_s} \sum_{s' \in S} \mathfrak{h}^a_s(s') \cdot x_{s'})$
9 $\quad\quad\quad$ $\bar{\mathfrak{h}}^{\sigma^\infty(s)}_s(s') := \arg\min_{\mathfrak{h}^a_s \in \mathscr{H}^a_s} \sum_{s' \in S} \mathfrak{h}^a_s(s') \cdot x_{s'}$
10 $\quad\quad$ $\delta := \max_{s \in S}(y_s - x_s)$; $\mathbf{x} := \mathbf{y}$;
11 $\quad$ **while** $\delta > \varepsilon$ **do**
12 $\quad\quad$ **foreach** $s \in S$ *and* $i \in \{1, \ldots, n\}$ *where* $k_i = \infty$ **do**
13 $\quad\quad\quad$ $y^i_s := \mathtt{r}_i(s, \sigma^\infty(s)) + \sum_{s' \in S} \bar{\mathfrak{h}}^{\sigma^\infty(s)}_s(s') \cdot x^i_{s'}$;
14 $\quad\quad$ $\delta := \max_{i=1}^n \max_{s \in S}(y^i_s - x^i_s)$; $\mathbf{x}^1 := \mathbf{y}^1$; …; $\mathbf{x}^n := \mathbf{y}^n$;
15 $\quad$ **for** $j = \max\{\, k_b < \infty \mid b \in \{1, \ldots, n\}\,\}$ *down to 1* **do**
16 $\quad\quad$ **foreach** $s \in S$ **do**
17 $\quad\quad\quad$ $y_s := \max_{a \in \mathscr{A}(s)}(\sum_{\{i \mid k_i \ge j\}} w_i \cdot \mathtt{r}_i(s,a) + \min_{\mathfrak{h}^a_s \in \mathscr{H}^a_s} \sum_{s' \in S} \mathfrak{h}^a_s(s') \cdot x_{s'})$;
18 $\quad\quad\quad$ $\sigma^j(s) := \arg\max_{a \in \mathscr{A}(s)}(\sum_{\{i \mid k_i \ge j\}} w_i \cdot \mathtt{r}_i(s,a) + \min_{\mathfrak{h}^a_s \in \mathscr{H}^a_s} \sum_{s' \in S} \mathfrak{h}^a_s(s') \cdot x_{s'})$;
19 $\quad\quad\quad$ $\bar{\mathfrak{h}}^{\sigma^j(s)}_s(s') := \arg\min_{\mathfrak{h}^a_s \in \mathscr{H}^a_s} \sum_{s' \in S} \mathfrak{h}^a_s(s') \cdot x_{s'}$;
20 $\quad\quad\quad$ **foreach** $i \in \{1, \ldots, n\}$ *where* $k_i \ge j$ **do**
21 $\quad\quad\quad\quad$ $y^i_s := \mathtt{r}_i(s, \sigma^j(s)) + \sum_{s' \in S} \bar{\mathfrak{h}}^{\sigma^j(s)}_s(s') \cdot x^i_{s'}$;
22 $\quad\quad\quad$ $\mathbf{x} := \mathbf{y}$; $\mathbf{x}^1 := \mathbf{y}^1$; …; $\mathbf{x}^n := \mathbf{y}^n$;
23 $\quad$ **foreach** $i \in \{1, \ldots, n\}$ **do**
24 $\quad\quad$ $g_i := y^i_{\bar{s}}$;
25 $\quad$ $\sigma$ acts as $\sigma^j$ in $j^{th}$ step when $j < \max_{i \in \{1,\ldots,n\}} k_i$ and as $\sigma^\infty$ afterwards;
26 $\quad$ **return** $\sigma, \mathbf{g}$

as the following optimization problem:

$$\begin{aligned} \min_{x} \quad & \mathbf{x}^T \mathbf{1} \\ \text{subject to:} \quad & x_s \ge \textstyle\sum_{i=1}^n w_i \cdot \mathtt{r}_i(s,a) + \min\limits_{\mathfrak{h}^a_s \in \mathscr{H}^a_s} \mathbf{x}^T \mathfrak{h}^a_s \qquad \forall s \in S, \forall a \in \mathscr{A}(s) \end{aligned}$$

We now modify the above optimization problem to simplify derivation of the LP problem. To this aim, we transform the optimization operator "min" to "max". Therefore, we get the

following optimization problem:

$$\begin{aligned} \max_{x} \quad & -\mathbf{x}^T\mathbf{1} \\ \text{subject to:} \quad & x_s \geq \sum_{i=1}^{n} w_i \cdot \mathtt{r}_i(s,a) + \min_{\mathfrak{h}_s^a \in \mathscr{H}_s^a} \mathbf{x}^T \mathfrak{h}_s^a \qquad \forall s \in S, \forall a \in \mathscr{A}(s) \end{aligned}$$

As it is clear from the set of constraints in the latter optimization problem, the inner optimization problem is not linear. In order to overcome this difficulty and induce the LP formulation, we use dual of the inner optimization problem. To this aim, consider the inner optimization problem with fixed $\mathbf{x}$:

$$P(\mathbf{x}) := \min_{\mathfrak{h}_s^a \in \mathscr{H}_s^a} \mathbf{x}^T \mathfrak{h}_s^a$$

Based on the general description of the interval uncertainty set $\mathscr{H}_s^a = \{\, \mathfrak{h}_s^a \mid \mathbf{0} \leq \underline{\mathfrak{h}_s^a} \leq \mathfrak{h}_s^a \leq \overline{\mathfrak{h}_s^a} \leq \mathbf{1}, \mathbf{1}^T\mathfrak{h}_s^a = 1 \,\}$ in which $\underline{\mathfrak{h}_s^a}$ and $\overline{\mathfrak{h}_s^a}$ are respectively the lower bound and the upper bound of interval uncertainty; we can rewrite the latter inner optimization problem as:

$$\begin{aligned} P(\mathbf{x}) := \min \quad & \mathbf{x}^T \mathfrak{h}_s^a \\ \text{subject to:} \quad & \mathbf{1}^T \mathfrak{h}_s^a = 1 \\ & \underline{\mathfrak{h}_s^a} \leq \mathfrak{h}_s^a \leq \overline{\mathfrak{h}_s^a} \end{aligned}$$

The dual of the above problem is formulated as follows:

$$\begin{aligned} D(\mathbf{x}) := \max_{\gamma_{j,1}^{s,a}, \gamma_{j,2}^{s,a}, \gamma_{j,3}^{s,a}} \quad & \gamma_{j,1}^{s,a} + \underline{\mathfrak{h}_s^a}^T \gamma_{j,2}^{s,a} - \overline{\mathfrak{h}_s^a}^T \gamma_{j,3}^{s,a} \\ \text{subject to:} \quad & \mathbf{x} - \gamma_{j,2}^{s,a} + \gamma_{j,3}^{s,a} - \gamma_{j,1}^{s,a}\mathbf{1} = \mathbf{0} \\ & \gamma_{j,2}^{s,a} \geq 0, \gamma_{j,3}^{s,a} \geq 0 \end{aligned}$$

Since the latter inner optimization problem with fixed $\mathbf{x}$ is an LP, therefore due to the strong duality theorem 3.4, we have $P^*(\mathbf{x}) = D^*(\mathbf{x})$ where $P^*(\mathbf{x})$ and $D^*(\mathbf{x})$ are the primal and dual optimal values, respectively. Therefore, we can replace the original inner optimization problem with its dual LP to derive the ultimate LP formulation. Note that the inner optimization operator is removed as the outer optimization operator will find the least underestimate to maximize its objective function. Hence, maximizing the expected total reward for *IMDP* $\mathcal{M}$ with respect to the reward structure $\mathbf{w} \cdot \mathtt{r}$ is formulated as the following LP which can in turn be solved in polynomial time.

$$\begin{aligned} \max_{x,\gamma} \quad & -\mathbf{x}^T\mathbf{1} \\ \text{subject to:} \quad & x_s \geq \sum_{i=1}^{n} w_i \cdot \mathtt{r}_i(s,a) + \gamma_{j,1}^{s,a} + \underline{\mathfrak{h}_s^a}^T \gamma_{j,2}^{s,a} - \overline{\mathfrak{h}_s^a}^T \gamma_{j,3}^{s,a} && \forall s \in S, \forall a \in \mathscr{A}(s) \\ & \mathbf{x} - \gamma_{j,2}^{s,a} + \gamma_{j,3}^{s,a} - \gamma_{j,1}^{s,a}\mathbf{1} = \mathbf{0} && \forall s \in S, \forall a \in \mathscr{A}(s) \\ & \gamma_{j,2}^{s,a}, \gamma_{j,3}^{s,a} \geq 0 && \forall s \in S, \forall a \in \mathscr{A}(s) \end{aligned}$$

■

**Remark 8.3.** *It is worthwhile to mention that our robust controller synthesis approach can also be applied to MDPs with richer formalisms for uncertainties such as likelihood or ellipsoidal uncertainties while preserving the computational complexity. In particular, in every inner optimization*

*problem in Algorithm 10, the optimality of a Markovian deterministic controller and nature is guaranteed as long as the uncertainty set is convex, the set of actions is finite and the inner optimization problem which minimizes/maximizes the objective function over the choices of nature achieves its optimum (cf. [Pug14, Proposition 4.1]). Furthermore, due to the convexity of the generated optimization problems, the computational complexity of our approach remains intact.*

### 8.1.3 Multi-objective Robust Controller Synthesis: Other Queries

For the sake of completeness of our approach, in this section we discuss other types of multi-objective queries and present algorithms to solve them. In particular, we follow the same direction as [FKP12] and show how Algorithm 10 can be adapted to solve these types of queries.

We first start with the definition of quantitative and Pareto queries. Formally,

**Definition 8.10** (**Quantitative Queries**). *Given an IMDP $\mathcal{M}$ and a multi-objective predicate $\varphi$, a quantitative query is of the form $qnt([o]^{\leq k_1}_{\star}, (\varphi_2, \ldots, \varphi_n))$, consisting of a multi-objective predicate $(\varphi_2, \ldots, \varphi_n)$ of size $n-1$ and an objective $[o]^{\leq k_1}_{\star}$ where $o$ is a target set $T$ or a reward structure $\mathtt{r}$, $k_1 \in \mathbb{N} \cup \{\infty\}$ and $\star \in \{\min, \max\}$. We define:*

$$qnt([o]^{\leq k_1}_{\min}, (\varphi_2, \ldots, \varphi_n)) = \inf\{\, x \in \mathbb{R} \mid ([o]^{\leq k_1}_{\leq x}, \varphi_2, \ldots, \varphi_n) \text{ is satisfiable} \,\}$$
$$qnt([o]^{\leq k_1}_{\max}, (\varphi_2, \ldots, \varphi_n)) = \sup\{\, x \in \mathbb{R} \mid ([o]^{\leq k_1}_{\geq x}, \varphi_2, \ldots, \varphi_n) \text{ is satisfiable} \,\}.$$

**Definition 8.11** (**Pareto Queries**). *Given an IMDP $\mathcal{M}$ and a multi-objective predicate $\varphi$, a Pareto query is of the form $Pareto([o_1]^{\leq k_1}_{\star_1}, \ldots, [o_n]^{\leq k_n}_{\star_n})$, where each $[o_i]^{\leq k_i}_{\star_i}$ is an objective in which $o_i$ is either a target set $T$ or a reward structure $\mathtt{r}$, $k_i \in \mathbb{N} \cup \{\infty\}$ and $\star_i \in \{\min, \max\}$. We define the set of achievable values as $A = \{\, \mathbf{x} \in \mathbb{R}^n \mid ([o_1]^{\leq k_1}_{\sim_1 x_1}, \ldots, [o_n]^{\leq k_n}_{\sim_n x_n}) \text{ is satisfiable} \,\}$ where*

$$\sim_i = \begin{cases} \geq & \text{if } \star_i = \max \\ \leq & \text{if } \star_i = \min \end{cases}$$

*Then,*

$$Pareto([o_1]^{\leq k_1}_{\star_1}, \ldots, [o_n]^{\leq k_n}_{\star_n}) = \{\, \mathbf{x} \in A \mid \mathbf{x} \text{ is Pareto optimal} \,\}.$$

Note that the quantitative queries asks to maximize or minimize the reachability/reward objective over the set of controllers satisfying $\varphi$. The Pareto queries ask to determine the Pareto set for a given set of objectives.

#### 8.1.3.1 Algorithms for Robust Synthesis of Multi-objective Queries

We now discuss algorithmic solutions to solve quantitative and Pareto queries. These algorithms are in fact designed as an adaption of Algorithm 10 as detailed below and can also be considered as an extension of their counterparts in [FKP12] under presence of model uncertainty.

**Algorithm 12:** Algorithm for solving robust quantitative queries

**Input:** An *IMDP* $\mathcal{M}$, objective $[\mathtt{r}_1]^{\leq k_1}_{\max}$, multi-objective predicate $([\mathtt{r}_2]^{\leq k_2}_{\geq r_2}, \ldots, [\mathtt{r}_n]^{\leq k_n}_{\geq r_n})$
**Output:** value of $qnt([\mathtt{r}_1]^{\leq k_1}_{\max}, ([\mathtt{r}_2]^{\leq k_2}_{\geq r_2}, \ldots, [\mathtt{r}_n]^{\leq k_n}_{\geq r_n}))$

1 **begin**
2 $X = \emptyset$; $\mathtt{r} = (\mathtt{r}_1, \ldots, \mathtt{r}_n)$;
3 $\mathbf{k} = (k_1, \ldots, k_n)$; $\mathbf{r} = (\min_{\sigma \in \Sigma} ExpTot^{\sigma,\mathbf{k}}_{\mathcal{M}}[\mathtt{r}_1], r_2, \ldots, r_n)$;
4 **while** $\mathbf{r} \notin X{\downarrow}$ *or* $\mathbf{w} \cdot \mathbf{g} > \mathbf{w} \cdot \mathbf{r}$ **do**
5 Find $\mathbf{w}$ separating $\mathbf{r}$ from $X{\downarrow}$ such that $w_1 > 0$;
6 Find controller $\sigma$ maximizing $ExpTot^{\sigma,\mathbf{k}}_{\mathcal{M}}[\mathbf{w} \cdot \mathtt{r}]$;
7 $\mathbf{g} := (ExpTot^{\sigma,k_i}_{\mathcal{M}}[\mathtt{r}_i])_{1 \leq i \leq n}$;
8 **if** $\mathbf{w} \cdot \mathbf{g} < \mathbf{w} \cdot \mathbf{r}$ **then**
9 **return** $\perp$;
10 $X = X \cup \{\mathbf{g}\}$; $r_1 := \max\{r_1, \max\{\, r' \mid (r', r_2, \ldots, r_n) \in X{\downarrow}\,\}\}$;
11 **return** $r_1$;

**Quantitative queries.** Let us first focus on the quantitative queries. To this end, without loss of generality, consider the quantitative query $qnt([\mathtt{r}_1]^{\leq k_1}_{\max}, ([\mathtt{r}_2]^{\leq k_2}_{\geq r_2}, \ldots, [\mathtt{r}_n]^{\leq k_n}_{\geq r_n}))$. Algorithm 12, similarly to Algorithm 10, generates a sequence of points $\mathbf{g}$ on the Pareto curve from a sequence of weight vectors $\mathbf{w}$. In order to optimize the objective $\mathtt{r}_1$ as detailed in [FKP12], a sequence of lower bounds $r_1$ is generated which are used in the same manner as Algorithm 10. In particular, in the initial step we let $r_1$ be the minimum value for $\mathtt{r}_1$ that can be computed with an instance of value iteration [Pug14]. The sequence of non-decreasing values for $r_1$ are generated at the next steps based on the set of points $X$ specified so far. In each step, the computation in the lines 6-7 of Algorithm 12 can again be done using Algorithm 11.

At this point it is worthwhile to mention that our extended Algorithm 12 is different from its counterpart in [FKP12] (cf. Algorithm 3) especially in lines 3, 6-7. In fact, all computations in these lines are performed while considering the behaviour of an adversarial nature as detailed in Algorithm 11.

**Pareto queries.** We next discuss the Pareto queries. Our algorithm is depicted as Algorithm 13 which is in principle an extension of its counterpart in [FKP12] (cf. Algorithm 3). Likewise Algorithm 12, the key differences of this algorithm with its counterpart are in lines 3-4 and 7-8. Following the same direction as in [FKP12], we solely concentrate on two objectives while in theory this can be extended to an arbitrary number of objectives. Since the number of faces of the Pareto curve is exponentially large and also the result of the value iteration algorithm to compute the individual points is an approximation, Algorithm 13 only constructs an $\varepsilon$-approximation of the Pareto curve.

### 8.1.4 Generation of randomized controllers

At this point we provide a detailed explanation of the randomized controllers generation in Section 8.1.2 for the robust controller synthesis of multi-objective queries.

**Algorithm 13:** Algorithm for solving robust Pareto queries

**Input:** An *IMDP* $\mathcal{M}$, reward structures $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2)$, time bounds $(k_1, k_2)$, $\varepsilon \in \mathbb{R}_{\geq 0}$
**Output:** An $\varepsilon$-approximation of the Pareto curve

**begin**
  $X = \emptyset$; $Y : \mathbb{R}^2 \to 2^{\mathbb{R}^2}$ with initial $Y(x) = \emptyset$ for all $x$; $\mathbf{w} = (1, 0)$;
  Find controller $\sigma$ maximizing $ExpTot^{\sigma,\mathbf{k}}_{\mathcal{M}}[\mathbf{w} \cdot \mathbf{r}]$;
  $\mathbf{g} := (ExpTot^{\sigma,k_1}_{\mathcal{M}}[\mathbf{r}_1], ExpTot^{\sigma,k_2}_{\mathcal{M}}[\mathbf{r}_2])$;
  $X := X \cup \{\mathbf{g}\}$; $Y(\mathbf{g}) := Y(\mathbf{g}) \cup \{\mathbf{w}\}$; $\mathbf{w} := (0, 1)$;
  **while** $\mathbf{w} \neq \bot$ **do**
    Find controller $\sigma$ maximizing $ExpTot^{\sigma,\mathbf{k}}_{\mathcal{M}}[\mathbf{w} \cdot \mathbf{r}]$;
    $\mathbf{g} := (ExpTot^{\sigma,k_1}_{\mathcal{M}}[\mathbf{r}_1], ExpTot^{\sigma,k_2}_{\mathcal{M}}[\mathbf{r}_2])$;
    $X := X \cup \{\mathbf{g}\}$; $Y(\mathbf{g}) := Y(\mathbf{g}) \cup \{\mathbf{w}\}$; $\mathbf{w} := \bot$;
    Order $X$ to a sequence $\mathbf{x}^1, \ldots, \mathbf{x}^m$ such that $\forall i : x^i_1 \leq x^{i+1}_1$ and $x^i_2 \geq x^{i+1}_2$;
    **for** $i = 1$ **to** $m$ **do**
      Let $\mathbf{u}$ be the element of $Y(\mathbf{x}^i)$ with maximal $u_1$;
      Let $\mathbf{u}'$ be the element of $Y(\mathbf{x}^{i+1})$ with minimal $u'_1$;
      Find a point $\mathbf{p}$ such that $\mathbf{u} \cdot \mathbf{p} = \mathbf{u} \cdot \mathbf{x}^i$ and $\mathbf{u}' \cdot \mathbf{p} = \mathbf{u}' \cdot \mathbf{x}^{i+1}$;
      **if** *distance of* $\mathbf{p}$ *from* $X{\downarrow}$ *is* $\geq \varepsilon$ **then**
        Find $\mathbf{w}$ separating $X{\downarrow}$ from $\mathbf{p}$, maximising $\mathbf{w} \cdot \mathbf{p} - \max_{\mathbf{x} \in X{\downarrow}} \mathbf{w} \cdot \mathbf{x}$;
        **break**;
  **return** $X$;

We consider a fixed *IMDP* $\mathcal{M}$ and a basic multi-objective predicate $([\mathbf{r}_1]^{\leq k_1}_{\geq r_1}, \ldots, [\mathbf{r}_n]^{\leq k_n}_{\geq r_n})$. For clarity, we assume that all $k_i = \infty$; we discuss the extension to $k_i < \infty$ afterwards. In the following, we will describe how we can obtain a randomized algorithm from the results computed by Algorithms 10, 12, and 13. These algorithms compute a set $X = \{\mathbf{g}_1, \ldots, \mathbf{g}_m\}$ of reward vectors $\mathbf{g}_i = (g_{i,1}, \ldots, g_{i,n})$ and their corresponding set of controllers $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$, where controller $\sigma_i$ achieves the reward vector $\mathbf{g}_i$.

In the descriptions of the given algorithms, the controllers $\sigma_i$ are not explicitly stored and mapped to the reward they achieve, but they can be easily adapted. All used controllers are memoryless and deterministic; this means that we can treat them as functions of the form $\sigma_i : S \to \mathscr{A}$ or, equivalently, as functions $\sigma_i : S \times \mathscr{A} \to \{0, 1\}$ where $\sigma_i(s, a) = 1$ if $\sigma_i(s) = a$ and $\sigma_i(s, \cdot) = 0$ otherwise.

From the set $X$, we can compute a set $P = \{p_1, \ldots, p_m\}$ of the probabilities with which each of these controllers shall be executed. If we execute each $\sigma_i$ with its according probability $p_i$, the vector of total expected rewards is $\mathbf{g} = \sum_{i=1}^{m} p_i \mathbf{g}_i$. Let $\mathbf{r} = (r_1, \ldots, r_n)$ denote the vector of reward bounds of the multi-objective predicate. To obtain $P$ after having executed Algorithm 10, we can choose the values $p_i$ in $P$ such that they fulfill the constraints $\sum_{i=1}^{m} p_i \mathbf{g}_i \geq \mathbf{r}$, $\sum_{i=1}^{m} p_i = 1$ and $p_i \geq 0$ for each $1 \leq i \leq m$. For the other algorithms, $P$ can be computed accordingly.

To obtain a stochastic process with expected values $\mathbf{g}$, we initially randomly choose one of the memoryless deterministic controllers $\sigma_i$ according to their probabilities in $P$. Afterwards, we just keep executing the chosen $\sigma_i$. The initial choice of the controller to

execute is the only randomized choice to be made. We do *not* perform a random choice after the initial choice of $\sigma_i$.

This process of obtaining the expected rewards $\mathbf{g}$ indeed uses memory, because we have to remember the deterministic controller which was randomly chosen to be executed. On the other hand, we only need a very limited way of randomisation.

We like to emphasize that indeed we cannot just construct a memoryless randomized controller by choosing the controller $\sigma_i$ with probability $p_i$ in each step anew.

**Example 8.3.** *Consider the IMDP in Figure 8.4. We only have two possible actions, $a$ and $b$. The initial state is $s$ and all probability intervals are the interval $[1,1]$, which we omit for readability; thus, there is also only one possible nature $\pi$. There is only a single reward structure, indicated by the underlined numbers. If we choose $a$ in state $s$, we end up in $t$ in the next step and obtain a reward of $1$ with certainty, while if we choose $b$, we will be in $u$ in the next step and obtain a reward of $0$, and accordingly for the other states.*

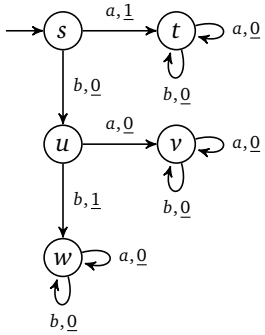



Figure 8.4: Computing randomised controllers.

*We consider the controllers $\sigma_a$ which chooses $a$ in each state and $\sigma_b$ which chooses $b$ in each state. With both controllers, we accumulate a reward of exactly $1$. Therefore, if we choose to execute $\sigma_a$ with probability $0.5$ and $\sigma_b$ with the same probability, this process will lead to a reward of $1$ as well.*

*Now, consider a controller which chooses the action selected by $\sigma_a$ in each state with probability $0.5$, and with the same probability chooses the action selected by $\sigma_b$. It is easy to see that this controller only obtains a reward of $0.5 \cdot 1 + 0.5 \cdot 0.5 \cdot 1 = 0.75$. As we see, this naive way of combining the two deterministic controllers into a memoryless randomized controller is not correct.* ♦

Thus, the way to construct a memoryless randomized controller is somewhat more involved. We will have to compute the *state-action frequencies*, that is the average number of times a given state-action pair is seen.

At first, we fix an arbitrary memoryless nature $\pi : Paths^{fin}_{\mathcal{M}} \times \mathscr{A} \to \mathrm{Disc}(S)$, that is, $\pi : S \times \mathscr{A} \to \mathrm{Disc}(S)$. The particular choice of $\pi$ is not important, which is due to the fact that our algorithms are robust against any choice of nature. We then let $x_i^\sigma(s)$ denote the probability to be in state $s$ at step $i$ when controller $\sigma$ is used (using nature $\pi$ and under the condition that we have started in $\bar{s}$).

For any $\sigma \in \Sigma$, we have $x_i^\sigma(s) = \sum_{\{\,\xi \in Paths^{fin}_{\mathcal{M}} | last(\xi)=s, |\xi|=i\,\}} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M}}[Paths^{\xi}_{\mathcal{M}}]$, which can be shown to be equivalent to the inductive form $x_0^\sigma(\bar{s}) = 1$ and $x_0^\sigma(s) = 0$ for $s \neq \bar{s}$, and $x_{i+1}^\sigma(s) = \sum_{s' \in S} \pi(s', \sigma(s'))(s) \cdot x_i^\sigma(s')$.

The state-action frequency $y^\sigma(s,a)$ is the number of times action $a$ is chosen in state $s$ when using controller $\sigma$. We then have that $y^\sigma(s,a) = \sum_{i=0}^{\infty} x_i^\sigma(s)\sigma(s,a)$. Thus, state-action frequencies can be approximated using a simple value iteration scheme. The *mixed state-action frequency* $y(s,a)$ is the average over all state action frequencies weighted by the probability with which a given controller is executed. Thus, $y(s,a) = \sum_{i=1}^{m} p_i y^{\sigma_i}(s,a)$ for all $s, a$. To construct a memoryless randomized controller $\sigma$, we normalize the probabilities to $\sigma(s,a) = \frac{y(s,a)}{\sum_{b \in \mathscr{A}} y(s,b)}$ for all $s, a$ (see also the description for the computation of strategies/adversaries below Proposition 4 of [FKN+11]).

**Example 8.4.** *In the model of Figure 8.4, we have* $y^{\sigma_a}(s,a) = 1$, $y^{\sigma_a}(s,b) = 0$, $y^{\sigma_a}(u,a) = 0$, $y^{\sigma_a}(u,b) = 0$, $y^{\sigma_b}(s,a) = 0$, $y^{\sigma_b}(s,b) = 1$, $y^{\sigma_b}(u,a) = 0$, *and* $y^{\sigma_b}(u,b) = 1$. *If we choose both* $\sigma_a$ *and* $\sigma_b$ *with probability* 0.5, *we obtain the mixed state-action frequencies* $y(s,a) = 0.5$, $y(s,b) = 0.5$, $y(u,a) = 0$, *and* $y(u,b) = 0.5$. *The memoryless randomized controller* $\sigma$ *we can construct is then* $\sigma(s,a) = 0.5$, $\sigma(s,b) = 0.5$, $\sigma(u,a) = 0$, $\sigma(u,b) = 1$, *which indeed achieves a reward of* 1. ⧫

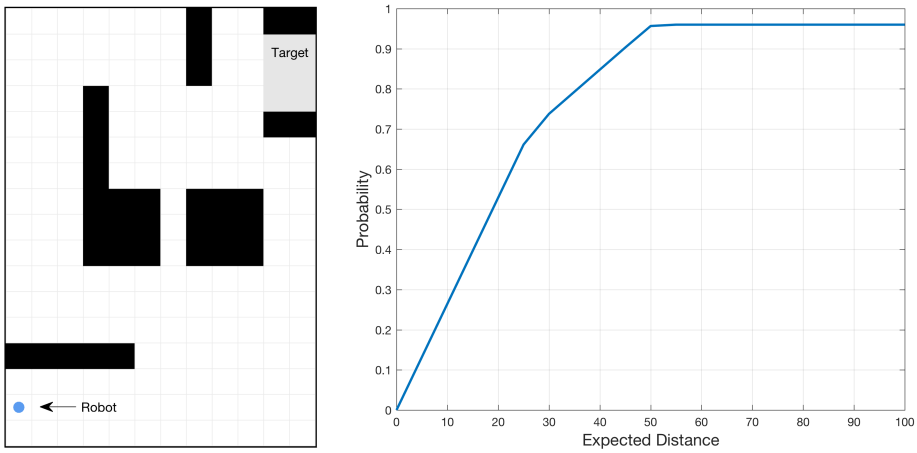
For the general case where $k_i < \infty$ for some $k$, we have to work with counting deterministic controllers and natures. Let $k_{\max}$ be the largest non-infinite step bound. The usage of memory is unavoidable here because it is required already in case of a single objective. To achieve optimal values, the computed controllers have to be able to make their decision dependent on how many steps are left before the step bound is reached. Thus, we have controllers of the form $\sigma_i \colon S \times \{0,\dots,k_{\max}\} \to \mathscr{A}$ or equivalently $\sigma_i \colon S \times \{0,\dots,k_{\max}\} \times \mathscr{A} \to \{0,1\}$ where $\sigma_i(s,j,a) = 1$ if $\sigma_i(s,j) = a$ and $\sigma_i(s,j,\cdot) = 0$ otherwise. For step $i$ with $i < k_{\max}$, a controller $\sigma$ chooses action $\sigma(s,i)$ for state $s$ whereas for all $i \geq k_{\max}$ the decision $\sigma(s,k_{\max})$ is used. Natures are of the form $\pi \colon S \times \mathscr{A} \times \{0,\dots,k_{\max}\} \to \mathrm{Disc}(S)$. The computation of the randomized controller changes accordingly: for any $\sigma \in \Sigma$, we have $x_0^\sigma(\bar{s}) = 1$ and $x_0^\sigma(s) = 0$ for $s \neq \bar{s}$, and $x_{i+1}^\sigma(s) = \sum_{s' \in S} \pi(s', \sigma(s',i'), i')(s) x_{i'}^\sigma(s')$ where $i' = \min\{i, k_{\max}\}$. Also the state-action frequencies are now defined as step-dependent. For $i \in \{0,\dots,k_{\max}-1\}$ we define $y^\sigma(s,i,a) = x_i^\sigma(s)\sigma(s,i,a)$ and $y^\sigma(s,k_{\max},a) = \sum_{i \geq k_{\max}} x_i^\sigma(s)\sigma(s,k_{\max},a)$.

The mixed state-action frequency is then $y(s,i,a) = \sum_{j=1}^{m} p_j y^{\sigma_j}(s,i,a)$. Again using normalization we define the counting randomized controller $\sigma(s,i,a) = \frac{y(s,i,a)}{\sum_{b \in \mathscr{A}} y(s,i,b)}$. Here, for step $i$ with $i < k_{\max}$ we use decisions from $\sigma(\cdot,i,\cdot)$ while for $i \geq k_{\max}$ we use decisions from $\sigma(\cdot,k_{\max},\cdot)$.

The bounded step case can be derived from the unbounded step case in the following sense: we can transform the *MDP* and the predicate into an *unrolled MDP*. Here, we encode the step bounds in the state space as follows: we copy the state space $S$ a number of $k_{\max}+1$ times to a new state space $S_{\text{unrolled}} = \dot{\bigcup}_{i \in \{0,\dots,k_{\max}\}} S_i$. We call each set of states $S_i$ a *layer*. For each state $s \in S$ and $i \in \{0,\dots,k_{\max}\}$ we have $s_i \in S_i$. If we have a transition from a state $s$ to a state $s'$, in the unrolled *MDP* for all $i \in \{0,\dots,k_{\max}-1\}$ we have an according transition from $s_i$ to $s_{i+1}$ instead. We also have a transition from $s_{k_{\max}}$ to $s'_{k_{\max}}$. Formally, for $i < k_{\max}$ we have $I^{\text{unrolled}}(s_i,a,s'_{i+1}) = I(s,a,s')$ for some states $s,s'$ and some action $a$ and zero else, and then $I^{\text{unrolled}}(s_{k_{\max}},a,s'_{k_{\max}}) = I(s,a,s')$. Thus, there are only transitions from a one layer to the next layer, except for layer $k_{\max}$ which behaves like the original *MDP*.

Reward structures are defined as follows. We assume that each reward property uses a different reward structure. For unbounded reward properties using reward structure $\mathtt{r}$, we just let $\mathtt{r}^{\text{unrolled}}(s_i,a) = \mathtt{r}(s,a)$ for all $i$ and states $s$. For a step bounded reward property with bound $k$ we define a modified reward structure as follows: for layers 0 to $k-1$, the reward is obtained as usual, that is $\mathtt{r}^{\text{unrolled}}(s_i,a) = \mathtt{r}(s,a)$ for $i \in \{0,\dots,k-1\}$. However, to simulate the step bound, we let $\mathtt{r}(s_i,a) = 0$ for $i \geq k$.

By removing the step bound from predicate, we can now analyze the unrolled *MDP* and obtain the same result as in the original *MDP* using the original step bounded predicate. As we are considering only unbounded properties, we obtain a set of memoryless deterministic controllers. We can than construct a counting scheduler for the original model by mapping the layer number to the step number, that is $\sigma(s,i,a) = \sigma^{\text{unrolled}}(s_i,a)$. In this way, we can show the correctness of the above scheduler computation for the step




(a) Robot Environment

(b) Pareto Curve

Figure 8.5: Robotic Scenario. (a) Environment map, where obstacles and target are shown in black and gray, respectively. (b) Pareto curve for the property $([\mathtt{r}_p]^{\leq\infty}_{\max}, [\mathtt{r}_d]^{\leq\infty}_{\min})$.

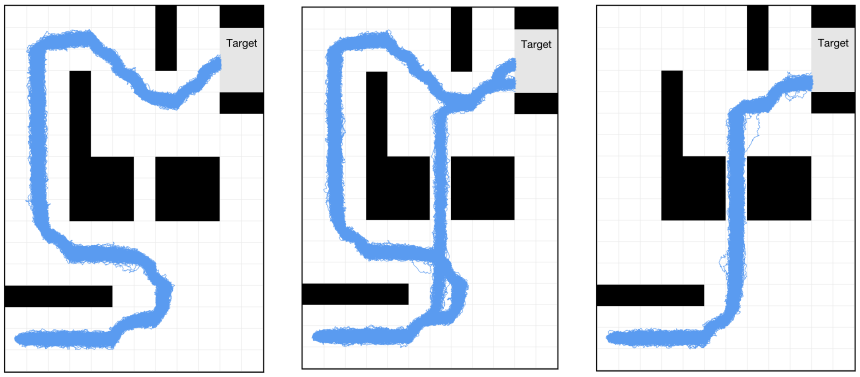
bounded case, because then also the values for the state action frequencies carry over, that is e.g. $y(s,i,a) = y^{\text{unrolled}}(s_i,a)$. Note that for $i < k_{\max}$ in $y^{\text{unrolled},\sigma}(s_i,a) = \sum_{j=0}^{\infty} x_j^{\sigma}(s_i)\sigma(s_i,a)$ only the summand for $j = i$ is relevant. This is the case because by construction of the unrolled *MDP* for the other $j$ with $j \neq i$ it is $x_j^{\sigma}(s_i) = 0$. Thus, $y^{\text{unrolled},\sigma}(s_i,a) = x_i^{\sigma}(s_i)\sigma(s_i,a)$. Accordingly, for $y^{\text{unrolled},\sigma}(s_{k_{\max}},a) = \sum_{j=0}^{\infty} x_j^{\sigma}(s_{k_{\max}})\sigma(s_{k_{\max}},a)$ only $j$ with $j \geq k_{\max}$ are relevant and thus $y^{\text{unrolled},\sigma}(s_{k_{\max}},a) = \sum_{j\geq k_{\max}}^{\infty} x_j^{\sigma}(s_{k_{\max}})\sigma(s_{k_{\max}},a)$.

## 8.2 Case Studies

We implemented the multi-objective robust controller synthesis algorithm and applied it to several case studies. The goal of these experiments is to quantitatively evaluate the performance of our proposed multi-objective robust controller synthesis approach on real world case studies. To this aim, we consider two case studies: (1) motion planning for a robot with noisy continuous dynamics and (2) autonomous nondeterministic tour guides drawn from [CRI07, HHS16a]. All experiments took a few seconds to complete on a standard laptop PC.

#### 8.2.0.1 Robot Motion Planning under Uncertainty

In robot motion planning, designers often seek a plan that simultaneously satisfies multiple objectives [LK16], e.g., *maximizing the chances of reaching the target while minimizing the energy consumption*. These objectives are usually in conflict with each other; hence, presenting the Pareto curve, i.e., the set of achievable points with optimal trade-off between the objectives, is helpful to the designers. They can then choose a point on the curve according to their desired guarantees and obtain the corresponding plan (controller) for the robot. In this case study, we considered such a motion planning problem for a noisy robot with




(a) $\varphi_1$ (b) $\varphi_2$ (c) $\varphi_3$

Figure 8.6: Robot sample paths under controllers for $\varphi_1$, $\varphi_2$, and $\varphi_3$

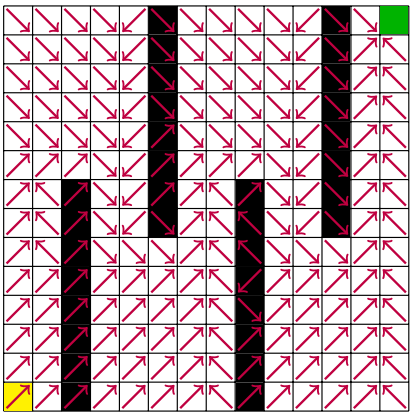
continuous dynamics in an environment with obstacles and a target region, as depicted in Figure 8.5(a). The robot's motion model was a single integrator with additive Gaussian noise. The initial state of the robot was on the bottom-left of the environment. The objectives were to reach the target safely while reducing the energy consumption, which is proportional to the travelled distance.

We approached this problem by first abstracting the motion of the noisy robot in the environment as an *IMDP* $\mathcal{M}$ and then computing controllers on $\mathcal{M}$ as in [LLMK14a, LLMK14c, LLMK14b]. The abstraction was achieved by partitioning the environment into a grid and computing local (continuous) controllers to allow transitions from every cell to each of its neighbours. The cells and the local controllers were then associated to the states and actions of the *IMDP*, respectively, resulting in 204 states (cells) and 4 actions per state. The boundaries of the environment were also associated with a state. Note that the transition probabilities between cells were raised by the noise in the dynamics and their ranges were due to variation of the possible initial robot (continuous) state within each cell.

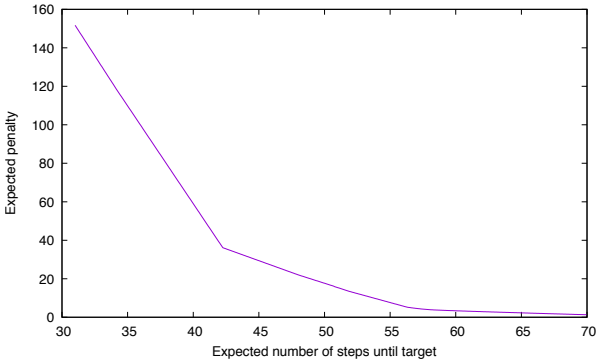
The *IMDP* states corresponding to obstacles (including boundaries) were given deterministic self-transitions, modelling robot termination as the result of a collision. To allow for the computation of the probability of reaching target, we included an extra state in the *IMDP* with a deterministic self-transition and then added incoming deterministic transitions to this state from the target states. A reward structure $\mathtt{r}_p$, which assigns a reward of 1 to these transitions and 0 to all the others, in fact, computes the probability of reaching the target. To capture the travelled distance, we defined a reward structure $\mathtt{r}_d$ assigning a reward of 0 to the state-action pairs with self-transitions and 1 to the rest.

The two robot objectives then can be expressed as: $([\mathtt{r}_p]^{\leq\infty}_{\max}, [\mathtt{r}_d]^{\leq\infty}_{\min})$. We first computed the Pareto curve for the property, which is shown in Figure 8.5(b), to find the set of all achievable values (optimal trade-offs) for the reachability probability and expected travelled distance. The Pareto curve shows that there is clearly a trade-off between the two objectives. To achieve high probability of reaching target safely, the robot needs to travel a longer distance, i.e., spend more energy, and vice versa. We chose three points on the curve and computed the corresponding robust controllers for

$$\varphi_1 = ([\mathtt{r}_p]^{\leq\infty}_{\geq 0.95}, [\mathtt{r}_d]^{\leq\infty}_{\leq 50}), \qquad \varphi_2 = ([\mathtt{r}_p]^{\leq\infty}_{\geq 0.90}, [\mathtt{r}_d]^{\leq\infty}_{\leq 45}), \qquad \varphi_3 = ([\mathtt{r}_p]^{\leq\infty}_{\geq 0.66}, [\mathtt{r}_d]^{\leq\infty}_{\leq 25}).$$



(a) The ANTG model for $n = 14$. The yellow, black and green cells represent the entrance, closed and exit parts of the museum, respectively. The purple arrows indicate an example controller.




(b) The Pareto Curve

Figure 8.7: The ANTG case study: model and analysis

We then simulated the robot under each controller 500 times. The statistical results of these simulations are consistent with the bounds in $\varphi_1$, $\varphi_2$, and $\varphi_3$. The collision-free robot trajectories are shown in Figure 8.6. These trajectories illustrate that the robot is conservative under $\varphi_1$ and takes a longer route with open spaces around it to go to target in order to be safe (Figure 8.6(a)), while it becomes reckless under $\varphi_3$ and tries to go through a narrow passage with the knowledge that its motion is noisy and could collide with the obstacles (Figure 8.6(c)). This risky behaviour, however, is required in order to meet the bound on the expected travelled distance in $\varphi_3$. The sample trajectories for $\varphi_2$ (Figure 8.6(b)) demonstrate the stochastic nature of the controller. That is, the robot probabilistically chooses between being safe and reckless in order to satisfy the bounds in $\varphi_2$.

#### 8.2.0.2 The Model of Autonomous Nondeterministic Tour Guides

Our second case study is inspired by "Autonomous Nondeterministic Tour Guides" (ANTG) in [CRI07, HHS16a], which models a complex museum with a variety of collections. We note that the model introduced in [CRI07] is an *MDP*. In this case study, we use an *IMDP* model by inserting uncertainties into the *MDP*.

Due to the popularity of the museum, there are many visitors at the same time. Different visitors may have different preferences of arts. We assume the museum divides all collections into different categories so that visitors can choose what they would like to visit and pay tickets according to their preferences. In order to obtain the best experience, a visitor can first assign certain weights to all categories denoting their preferences to the museum, and then design the best controller for a target. However, the preference of a sort of arts to a visitor may depend on many factors like price, weather, or the length of queue at that moment etc., hence it is hard to assign fixed values to these preferences. In our model we allow uncertainties of preferences such that their values may lie in an interval.

For simplicity we assume all collections are organized in an $n \times n$ square with $n \geq 10$,

with $(0,0)$ being the south-west corner of the museum and $(n-1,n-1)$ the north-east one. Let $c = \frac{n-1}{2}$; note that $(c,c)$ is at the center of the museum. We assume all collections at $(x,y)$ are assigned with a weight interval $[3,4]$ if $\max\{|x-c|,|y-c|\} \leq \frac{n}{10}$, with a weight 2 if $\frac{n}{10} < \max\{|x-c|,|y-c|\} \leq \frac{n}{5}$, and a weight 1 if $\max\{|x-c|,|y-c|\} > \frac{n}{5}$. In other words, we expect collections in the center to be more popular and subject to more uncertainties than others.

Furthermore, we assume that people at each location $(x,y)$ have four nondeterministic choices of moving to $(x',y')$ in the north east, south east, north west, and south west of $(x,y)$ (limited to the boundaries of the museum). The outcome of these choices, however, is not deterministic. That is, deciding to go to $(x',y')$ takes the visitor to either $(x,y')$ or $(x',y)$ depending on the weight intervals of $(x,y')$ and $(x',y)$. Thus, the actual outcome of the move is probabilistic to north, south, east or west. To obtain an *IMDP*, weights are normalized. For instance, if the visitor chooses to go to the north east and on $(x,y+1)$ there is a weight interval of $[3,4]$ and on $(x+1,y)$ there is a weight interval of $[2,2]$, it will go to $(x,y+1)$ with probability interval $[3/(3+2),4/(4+2)]$ and to $(x+1,y)$ with probability interval $[2/(2+4),2/(2+3)]$.

Therefore a model with parameter $n$ has $n^2$ states in total and roughly $4n^2$ transitions, a few of which are associated with uncertain transition probabilities. An instance of the museum model for $n = 14$ is depicted in Figure 8.7(a). In this instantiation, we assume that the visitor starts in the lower left corner (marked yellow) and wants to move to the upper right corner (marked green) with as few steps as possible. On the other hand, it wants to avoid moving to the black cells, because they correspond to exhibitions which are closed. For closed exhibitions located at $x = 2$, the visitor receive a penalty of 2, for those at $x = 5$ it receives a penalty of 4, for $x = 8$ one of 16 and for $x = 11$ one of 64. Therefore, there is a tradeoff between leaving the museum as fast as possible and minimizing the penalty received. With $\mathtt{r}_s$ being the reward structure for the number of steps and $\mathtt{r}_p$ denoting the penalty accumulated, $([\mathtt{r}_s]^{\leq\infty}_{\leq 40}, [\mathtt{r}_p]^{\leq\infty}_{\leq 70})$ requires that we leave the museum within 40 steps but with a penalty of no more than 70. The purple arrows indicate a controller which has been used when computing the Pareto curve by our tool. Here, the tourist mostly ignores closed exhibitions at $x = 2$ but avoids them later. We provide the Pareto curve for this situation in Figure 8.7(b). With an increasing step bound considered acceptable, the optimal accumulated penalty decreases. This is expected, because with an increasing step bound, the visitor has more time to walk around more of the closed exhibitions, thus facing a lower penalty.

In Figure 8.8, we provide controllers for different points on the Pareto curve in Figure 8.7(b). The lowest expected number of steps in which the museum can be left at all is 30.9665389. To achieve this number, there is a single optimal controller sketched in Figure 8.8(a). As we see, the tourist indeed leaves the museum as soon as possible, ignoring any closed exhibitions and receiving an expected penalty as high as 152.0609886.

In Figure 8.8(b) and Figure 8.8(c), we give the tourist somewhat more time, namely 31 steps, so that the penalty of 151.7077821 is somewhat lower. Here, with a high probability (0.9894174) the same controller as for the previous case is chosen. With a probability of 0.0105826 however, the less reckless controller of Figure 8.8(c) is used, which takes some efforts to avoid the last row of closed exhibitions at $x = 11$.

If we further increase the time bound to 40, as in Figure 8.8(d) and Figure 8.8(e), the controllers used become even less risky but more time consuming to execute. For a step

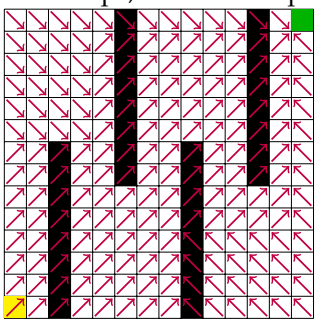
bound of 76.8658133 and larger, it is possible to avoid receiving any penalty by using the controller of Figure 8.8(f), which circumvents all of the closed exhibitions.

## 8.3 Concluding Remarks

In this chapter, we have analyzed interval Markov decision processes under controller (parameter) synthesis semantics in a dynamic setting. In particular, we discussed the problem of multi-objective robust controller synthesis for *IMDP*s, aiming for controllers that satisfy a given multi-objective predicate under all resolutions of the uncertainty in the transition probabilities. We first showed that this problem is **PSPACE**-hard and then introduced a value iteration-based decision algorithm to approximate the Pareto set of achievable points. Finally, we presented results obtained with a prototype tool on several real world case studies to show the effectiveness of the developed algorithms.

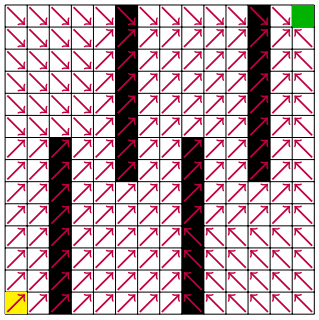
**Related work** Multi-objective model checking of probabilistic models with respect to various quantitative objectives has been recently investigated in a few works. The works in [FKN$^+$11, FKP12, KNPQ13, EKVY07] focused on multi-objective verification of exact *MDP*s. In [CFK$^+$13], these algorithms were extended to the more general models of 2-player stochastic games. These models, however, cannot capture the continuous uncertainty in the transition probabilities as *IMDP*s do. For the purposes of synthesis though, it is possible to transform an *IMDP* into a 2-player stochastic game; nevertheless, such a transformation raises an extra exponential factor to the complexity of the decision problem. This exponential blowup has been avoided in our setting.

30.9665389 steps, 152.0609886 penalty

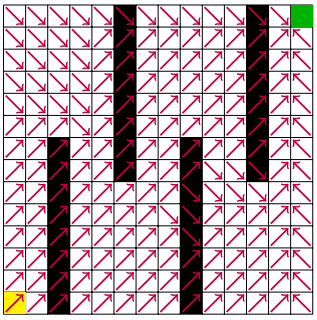


(a) Probability 1

31 steps, 151.7077821 penalty

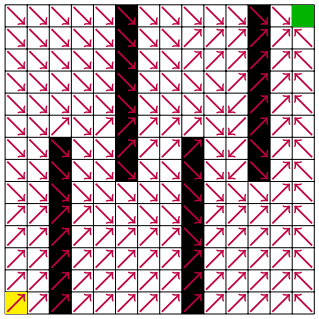


(b) Probability 0.9894174

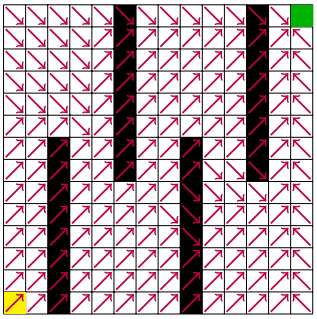


(c) Probability 0.0105826

40 steps, 59.0123994 penalty

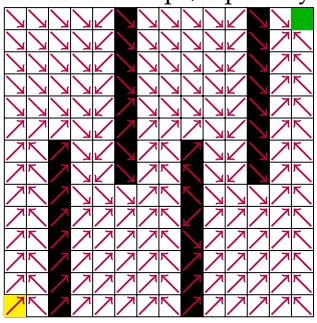


(d) Probability 0.7230247



(e) Probability 0.2769753

76.8658133 steps, 0 penalty



(f) Probability 1

Figure 8.8: Controllers for different points on the Pareto curve in Fig. 8.7(b).

CHAPTER 9

# Bisimulation Minimization for Model Checking of UwMDPs

In this chapter, we introduce *Uncertain weighted Markov Decision Process* (*UwMDP*) as a novel stochastic model to capture quantities like preferences or priorities in a nondeterministic scenario with uncertainties. The model is very close to the model of *IMDP*s but more convenient to model with when non-probability uncertainties like weights, preference, priority, etc. are involved. In particular, different from *IMDP*s, each transition in an *UwMDP* is associated with a weight interval such that any integer value in this interval is a feasible weight for that transition. Throughout this chapter, we only consider integer weights. The extension to rational weights is straightforward.

Weights in *UwMDP* models play a similar role as in GSPN [MCB84] or EMPA [BG96], namely, they will be used to induce a distribution over all transitions with the same label. For instance, if from a state $s$, there are transitions leading to $s_1$ and $s_2$ with weights 2 and 3, respectively, then this means that $s$ will evolve into $s_1$ and $s_2$ with probabilities $\frac{2}{5}$ and $\frac{3}{5}$, respectively. The model of *UwMDP*s has intervals of weights attached to each transition. One can think of this model as a game between a scheduler and nature. Different from ordinary *wMDP*s, nature might also be nondeterministic (besides being probabilistic). More precisely, nature selects a realization of weights from a set of feasible weights that is specified in terms of intervals. The selected weights by the nature then induces a probabilistic transition over states.

We examine modelling and verification in the context of *UwMDP*s and discuss our analysis of *UwMDP* models from two viewpoints. We first show that *UwMDP*s are equipped with an efficient theory on probabilistic bisimulation. This implies that probabilistic bisimulation minimization approaches [CGM$^+$96, BHH$^+$09, BCS10] can be applied to alleviate state space explosion problem. More concretely, we define bisimulation relations for *UwMDP*s and show that quotient of *UwMDP*s can be computed efficiently in polynomial time with respect to the size of *UwMDP*s.

As regards the model analysis, we next discuss extreme (maximal/minimal) reward-bounded reachability probabilities [AHK03] of *UwMDP*s. Thus each *UwMDP* is associated with a reward structure, which assigns to each weighted transition a reward. On the one hand, our work is an extension of the work in [AHK03] enriched with uncertainties,

while on the other hand, it can also be seen as an extension of the work in [PLSVS13,Pug14] to models with reward structures. Despite the fact that an *UwMDP* may represent an equivalent, but exponentially larger model without uncertainties, we propose an algorithm to compute extreme reward-bounded reachability probabilities in pseudo polynomial time – linear with respect to the size of the given *UwMDP* and quadratic with respect to the reward bound. Along the line of [PLSVS13], extreme reachability probabilities without reward bounds can be computed efficiently for *UwMDP*s as well.

We finally show promising results on a variety of case studies, obtained by a prototypical implementation of all algorithms to illustrate the effectiveness of our approaches.

The material presented in this chapter is an extended version of the results reported in [HHS16a].

**Organization of the chapter.** The rest of the chapter is organized as follows. We start in Section 9.1 by formally introducing the model of *UwMDP*s and provide its semantics. In Section 9.2, we give the definition of bisimulation for *UwMDP*s and afterwards, we present a tractable decision algorithm to compute it. In Section 9.3, we define the notion of maximal reward-bounded reachability probability for *UwMDP*s and show a least-fixed point characterization. In Section 9.4, we demonstrate our approach on some case studies and present promising experimental results. Finally we conclude this chapter in Section 9.5.

## 9.1 Uncertain weighted Markov Decision Processes

In Definition 4.4, all weights have to be given precisely, which sometimes is not possible, especially when all weights are estimated or based on experiments which are often affected by uncertainties due, for instance, to measurement errors. On the other hand, the model of *uncertain weighted MDPs* relax this condition such that it allows weights to vary as long as they are in certain intervals. Formally,

**Definition 9.1** (**Uncertain weighted MDP**). *An* Uncertain weighted Markov Decision Process *(UwMDP) is a tuple* $\mathcal{W} = (S, \bar{s}, \mathscr{A}, \mathit{AP}, L, W)$ *similar as in Definition 4.4 except that* $W : S \times \mathscr{A} \times S \mapsto [w_l, w_h]$ *defines a transition relation with uncertainties, where* $w_l, w_h \in \mathbb{N}_0$ *with* $w_l \leq w_h$*. We denote by* $[\mathcal{W}]$*, the class of all finite-state finite-transition UwMDPs.*

Let $\mathscr{A}(s)$ denote the set of available actions at state $s \in S$.

The only difference between Definition 4.4 and Definition 9.1 is that in a *wMDP* all transitions are labelled by a weight, while in an *UwMDP*, transitions are labelled by an interval specifying all allowed weights. We denote by $\mathfrak{w}_{st}^a$ a resolution of uncertainties corresponding to the transition from $s$ to $t$ with label $a$, i.e., $\mathfrak{w}_{st}^a \in W(s,a,t)$. We write $s \xrightarrow{a,w} \mu$ if and only if there exists $\mathfrak{w}_{st}^a \in W(s,a,t)$ for each $t \in S$ such that $w = \sum_{t \in S} \mathfrak{w}_{st}^a > 0$ and $\mu(t) = \frac{\mathfrak{w}_{st}^a}{w}$. Similarly, we can define the combined transitions for *UwMDP*s.

**Remark 9.1.** *The size of a given* $\mathcal{W}$ *is determined as follows. Let* $|S|$ *denote the number of states in* $\mathcal{W}$*. Then each state has* $\mathcal{O}(|\mathscr{A}|)$ *actions, while each action corresponds to at most* $\mathcal{O}(|S|^2)$ *transitions, each of which is associated with a weight interval. Therefore, the overall size of* $\mathcal{W}$ *i.e.,* $|\mathcal{W}|$ *is in* $\mathcal{O}(|S|^2 \, |\mathscr{A}|)$*.*

Let us formally state the semantics of an *UwMDP* $\mathcal{W}$. A transition initialized from state $s_i$ in $\mathcal{W}$ happens in three steps. First, an action $a \in \mathscr{A}(s_i)$ is chosen nondeterministically. Secondly, a resolution $\mathfrak{w}^a_{s_i t} \in W(s_i, a, t)$ is chosen for each $t \in S$. The selection of $\mathfrak{w}^a_{s_i t}$ models uncertainty in the transition. Lastly, a successor state $s_{i+1}$ is chosen randomly, according to the induced transition probability distribution $\mu$. It is not hard to see that each *UwMDP* $\mathcal{W}$ corresponds to a *wMDP*, which may be exponentially larger than $\mathcal{W}$.

A *path* in $\mathcal{W}$ is a finite or infinite sequence of the form $\xi = s_0 \mathfrak{w}^{a_0}_{s_0 s_1} s_1 \mathfrak{w}^{a_1}_{s_1 s_2} s_2 \cdots$ where $s_i \in S$, $a_i \in \mathscr{A}(s_i)$ and $0 < \mathfrak{w}^{a_i}_{s_i s_{i+1}} \in W(s_i, a_i, s_{i+1})$ for any $i \geq 0$. For a finite path $\xi$, we denote by $last(\xi)$ the last state of $\xi$. The $i$-th state (action) along a path $\xi$ is denoted by $\xi[i]$ ($\xi(i)$), if it exists. The set of all finite paths and the set of all infinite paths in the given $\mathcal{W}$ are denoted by $Paths^{fin}_{\mathcal{W}}$ and $Paths^{inf}_{\mathcal{W}}$, respectively. Furthermore, let $Paths^{\xi}_{\mathcal{W}} = \{\, \xi' \in Paths^{inf}_{\mathcal{W}} \mid \xi \text{ is a prefix of } \xi' \,\}$ denote the set of infinite paths with the prefix $\xi \in Paths^{fin}_{\mathcal{W}}$ which is also known as the *cylinder* set of $\xi$.

Due to the existing of nondeterminism, to resolve which, we need to introduce notions of *scheduler* and *nature* for *UwMDP*s. Formally,

**Definition 9.2** (**Scheduler and nature in *UwMDP*s**). *Given an UwMDP $\mathcal{W}$, a scheduler is a function $\lambda : Paths^{fin}_{\mathcal{W}} \to \mathrm{Disc}(\mathscr{A})$ that to each finite path $\xi$ assigns a distribution over the set of actions. A* nature *is a function $\gamma : Paths^{fin}_{\mathcal{W}} \times \mathscr{A} \to \mathrm{Disc}(S)$ that to each finite path $\xi$ and action $a$ assigns a* feasible distribution*, i.e. $\gamma(\xi, a) = \mu$ if and only if $last(\xi) \xrightarrow{a,w} \mu$ for some $w$. We denote by $\Lambda$ the set of all schedulers and by $\Gamma$ the set of all natures of $\mathcal{W}$.*

A scheduler $\lambda$ is said to be *deterministic* (**D**) if $\lambda(\xi) = \delta_a$ for all finite paths $\xi$ and some $a \in \mathscr{A}(last(\xi))$. Furthermore, a nature is said to be *deterministic* if $\gamma(\xi, a) = \delta_\mu$ for all finite paths $\xi$, for all $a \in \mathscr{A}(last(\xi))$ and some $\mu$ where $last(\xi) \xrightarrow{a,w} \mu$ for some $w$. Furthermore, a scheduler $\lambda$ (nature $\gamma$) is *Markovian* (**M**) if it depends only on $last(\xi)$.

For a scheduler $\lambda$ and a nature $\gamma$, let $\mathbf{Pr}^{\lambda,\gamma}_{\mathcal{W}}$ denote the unique probability measure over $(Paths^{inf}_{\mathcal{W}}, \mathscr{B})$ such that the probability $\mathbf{Pr}^{\lambda,\gamma}_{\mathcal{W}}[Paths^{s'}_{\mathcal{W}}]$ of starting in $s'$ equals 1 if $s' = \bar{s}$ and 0, otherwise; and the probability $\mathbf{Pr}^{\lambda,\gamma}_{\mathcal{W}}[Paths^{\xi \mathfrak{w}^a_{last(\xi)s'} s'}_{\mathcal{W}}]$ of traversing a finite path $\xi \mathfrak{w}^a_{last(\xi)s'} s'$ equals $\mathbf{Pr}^{\lambda,\gamma}_{\mathcal{W}}[Paths^{\xi \mathfrak{w}^a_{last(\xi)s'} s'}_{\mathcal{W}}] = \mathbf{Pr}^{\lambda,\gamma}_{\mathcal{W}}[Paths^{\xi}_{\mathcal{W}}] \cdot \lambda(\xi)(a) \cdot \gamma(\xi, a)(s')$. Here, $\mathscr{B}$ is the standard $\sigma$-algebra over $Paths^{inf}_{\mathcal{W}}$ generated from the set of all cylinder sets $\{Paths^{\xi}_{\mathcal{W}} \mid \xi \in Paths^{fin}_{\mathcal{W}}\}$. The unique probability measure is obtained by the application of the extension theorem (see, e.g. [Bil79]).

Following the same reasoning for *IMDP*s scheduler and nature, in *UwMDP*s a scheduler does not choose an action but a *distribution* over actions and a nature is not allowed to randomise over the set of feasible distributions.

In order to model other quantitative measures of an *UwMDP*, we associate a reward to each weighted transition of a state. This is done by introducing a *reward structure*:

**Definition 9.3** (***UwMDP*s reward structure**). *A* reward structure *for an UwMDP is a function $\mathtt{r} : S \times \mathscr{A} \times \mathbb{N}_0 \to \mathbb{N}_0$ that assigns to each state $s \in S$, action $a \in \mathscr{A}(s)$, and weight $w \in \mathbb{N}_0$ a reward $\mathtt{r}(s, a, w) > 0$.*

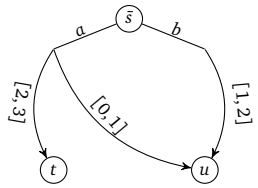



Figure 9.1: An example of *UwMDP*s: the *UwMDP* $\mathcal{W}$

Note that the definition of reward structure is quite flexible. We could easily define rewards independent of weights of transitions. Similarly to *IMDP* models and in order to avoid ambiguity, we sometimes describe the *UwMDP* $\mathcal{W}$ as a tuple $(S_{\mathcal{W}}, \bar{s}_{\mathcal{W}}, \mathscr{A}_{\mathcal{W}}, \mathtt{AP}_{\mathcal{W}}, L_{\mathcal{W}}, W)$ by adding the *UwMDP* model symbol as a subindex to its generic elements. In such cases, we also denote the *UwMDP* reward structure with $\mathtt{r}_{\mathcal{W}}$.

**Example 9.1.** *Figure 9.1 depicts an instance of UwMDPs. The set of states is* $S = \{\bar{s}, t, u\}$ *with* $\bar{s}$ *being the initial one. Let* $\mathtt{AP} = \{\mathfrak{u}, \mathfrak{v}\}$ *and* $L(\bar{s}) = \{\mathfrak{v}\}$, $L(t) = \{\mathfrak{u}\}$ *and* $L(u) = \{\mathfrak{u}, \mathfrak{v}\}$. *Moreover,* $W(\bar{s}, a, t) = [2, 3]$, $W(\bar{s}, a, u) = [0, 1]$, *and* $W(\bar{s}, b, u) = [1, 2]$. *A reward structure for the* $\mathcal{W}$ *can be as follows:* $\mathtt{r}(\bar{s}, a, 2) = 3$, $\mathtt{r}(\bar{s}, a, 3) = 1$, $\mathtt{r}(\bar{s}, a, 4) = 5$, $\mathtt{r}(\bar{s}, b, 1) = 4$, $\mathtt{r}(\bar{s}, b, 2) = 3$. ⧫

At this point, it is worthwhile to mention that weights in *UwMDP*s can be used to denote priorities or preferences, which are quantities used to generate probabilistic behaviours. For instance, a robot may choose to serve its clients stochastically relative to the preferences they expressed. In this respect, *UwMDP*s offer users more flexibility to model uncertainties than *IMDP*s. As we shall see later, *UwMDP*s provide a tractable bisimulation minimization approach which is essential to address state-space explosion problem in model checking large systems. The latter, however, is not the case for *IMDP*s in general as discussed in Chapter 6.

## 9.2 Bisimulation Minimization for UwMDPs

In this section we extend the notion of probabilistic bisimulation to *UwMDP*s and present an efficient decision algorithm to compute it. Afterwards we discuss the computational complexity of the decision algorithm.

### 9.2.1 Probabilistic Bisimulation

Below we define bisimulation relations over states of an *UwMDP*, which is an extension of the definition in [CS02]. For simplicity, we omit reward structures in this section, which can be integrated easily.

**Definition 9.4 (Probabilistic bisimulation for *UwMDP*s).** *Let* $\mathcal{W} := (S_{\mathcal{W}}, \mathscr{A}_{\mathcal{W}}, W, \bar{s}_{\mathcal{W}}, AP_{\mathcal{W}}, L_{\mathcal{W}})$ *be an UwMDP. Let* $\mathcal{R} \subseteq S_{\mathcal{W}} \times S_{\mathcal{W}}$. $\mathcal{R}$ *is a bisimulation if and only if* $s\ \mathcal{R}\ t$ *implies:*

- $L_{\mathcal{W}}(s) = L_{\mathcal{W}}(t)$;

- *whenever* $s \xrightarrow{a,w} \mu$, *there exists* $t \xrightarrow{a,w}_{\mathrm{c}} \nu$ *such that* $\mu \; \mathcal{L}(\mathcal{R}) \; \nu$;
- *symmetrically for* $t$.

*Two states* $s$ *and* $t$ *are bisimilar, written as* $s \sim t$, *if and only if there exists a bisimulation* $\mathcal{R}$ *such that* $s \; \mathcal{R} \; t$.

**Proposition 9.1.** $\sim$ *is an equivalence relation.*

*Proof.* The proof is straightforward from Definition 9.4. ∎

### 9.2.2 Decision Algorithm

We now propose a polynomial time decision algorithm to decide bisimulations defined in Definition 9.4. Our algorithm follows the classical partition-refinement approach [KS90, CS02, GHT14, HHT13].

Before presenting the algorithm we give an alternative definition of bisimulation and show that it is equivalent to Definition 9.4. Let $W^l(s,a,s') = w_l$ and $W^h(s,a,s') = w_h$, where $W(s,a,s') = [w_l, w_h]$.

**Lemma 9.1.** *Let* $\mathcal{W} := (S_{\mathcal{W}}, \mathscr{A}_{\mathcal{W}}, W, \bar{s}_{\mathcal{W}}, AP_{\mathcal{W}}, L_{\mathcal{W}})$ *be an UwMDP. Let* $\mathcal{R} \subseteq S_{\mathcal{W}} \times S_{\mathcal{W}}$ *be an equivalence relation.* $\mathcal{R}$ *is a bisimulation if and only if* $s \; \mathcal{R} \; t$ *implies*

- $L_{\mathcal{W}}(s) = L_{\mathcal{W}}(t)$;
- $W^l(s,a,\mathcal{C}) = W^l(t,a,\mathcal{C})$ *and* $W^h(s,a,\mathcal{C}) = W^h(t,a,\mathcal{C})$ *for each* $\mathcal{C} \in S_{\mathcal{W}}/\mathcal{R}$,

*where* $W^l(s,a,\mathcal{C}) = \sum_{s' \in \mathcal{C}} W^l(s,a,s')$ *and* $W^h(s,a,\mathcal{C}) = \sum_{s' \in \mathcal{C}} W^h(s,a,s')$.

*Proof.* Let $\sim'$ denote the new definition of bisimulation. We show that $\sim \equiv \sim'$.

- $\sim \subseteq \sim'$. Trivial.
- $\sim' \subseteq \sim$. Let $\mathcal{R} = \{(s_1, s_2) \mid s_1 \sim' s_2\}$. We show that $\mathcal{R}$ is a bisimulation by Definition 9.4. Let $s \; \mathcal{R} \; t$. Obviously, $L_{\mathcal{W}}(s) = L_{\mathcal{W}}(t)$. Let $s \xrightarrow{a,w} \mu$. We show that there exists $t \xrightarrow{a,w}_{\mathrm{c}} \nu$ such that $\mu \; \mathcal{L}(\mathcal{R}) \; \nu$. Note whenever $s \xrightarrow{a,w} \mu$, for each $\mathcal{C} \in S_{\mathcal{W}}/\mathcal{R}$,

$$W^l(s,a,\mathcal{C}) \leq \mu(\mathcal{C}) \leq W^l(s,a,\mathcal{C}).$$

  Since $s \sim' t$, $W^l(s,a,\mathcal{C}) = W^l(t,a,\mathcal{C})$ and $W^h(s,a,\mathcal{C}) = W^h(t,a,\mathcal{C})$ for each $\mathcal{C} \in S_{\mathcal{W}}/\mathcal{R}$. Therefore there exists $t \xrightarrow{a,w}_{\mathrm{c}} \nu$ such that $\mu \; \mathcal{L}(\mathcal{R}) \; \nu$.

∎

Because of Lemma 9.1, we can now present the algorithm to check whether two states are bisimilar. The key procedure is FindSplitter presented in Algorithm 15, which finds a pair $(\mathcal{C}', a)$ of block in the current partition and an action $a$ distinguishing two states in a block. The found splitter $(\mathcal{C}', a)$ is then used by Algorithm 14 to further refine the current partition. The algorithm terminates if no splitter can be found and the current partition stays stable.

Even though *UwMDP*s offer a compact manner to encode uncertainties appearing in a weighted *MDP*, which may be exponentially larger than its corresponding *UwMDP*, we

**Algorithm 14:** Deciding bisimulation

**Input:** An *UwMDP* $\mathcal{W} := (S_{\mathcal{W}}, \mathscr{A}_{\mathcal{W}}, W, \bar{s}_{\mathcal{W}}, \mathtt{AP}_{\mathcal{W}}, L_{\mathcal{W}})$, two states $s, t \in S_{\mathcal{W}}$.
**Output:** '*true*' if $s \sim t$, or '*false*' otherwise.

```
1  begin
2      Partition ← {{S_W}};
3      splitter ← FindSplitter(Partition);
4      while (splitter ≠ ∅) do
5          Partition ← Refine(W, Partition, splitter);
6          splitter ← FindSplitter(W, Partition);
7      if there exists C ∈ Partition such that s, t ∈ C then
8          return true;
9      else
10         return false;
```

**Algorithm 15:** Procedure FindSplitter

**Input:** An *UwMDP* $\mathcal{W} := (S_{\mathcal{W}}, \mathscr{A}_{\mathcal{W}}, W, \bar{s}_{\mathcal{W}}, \mathtt{AP}_{\mathcal{W}}, L_{\mathcal{W}})$ and `Partition`.
**Output:** A splitter of `Partition` with respect to $\sim$.

```
1  begin
2      for (C ∈ Partition and s, t ∈ C) do
3          for (C' ∈ Partition and a ∈ A_W) do
4              if W^l(s, a, C') ≠ W^l(t, a, C') or W^h(s, a, C') ≠ W^h(t, a, C') then
5                  return (C', a);
6      return ∅;
```

do not need to pay extra costs to compute bisimulation relations on *UwMDP*s. Formally, Algorithm 14 has the same complexity as Algorithm 1 in [CS02], where $n$ and $m$ are the numbers of states and (uncertain) transitions in an *UwMDP*, respectively.

**Theorem 9.1.** *Algorithm 14 terminates in time* $\mathcal{O}(n(n^2 + m))$ *in the worst case.*

*Proof.* Firstly, we show that one round of FindSplitter will terminate in $\mathcal{O}(n^2 + m)$ time. In the worst case, we need to compute $W^l(s, a, \mathcal{C}')$ and $W^h(s, a, \mathcal{C}')$ for each $s \in S_{\mathcal{W}}$ and $\mathcal{C}' \in \mathtt{Partition}$, which can be done in time $\mathcal{O}(n^2 + m)$. Secondly, the procedure Refine will terminate in $\mathcal{O}(n)$ time, given all $W^l(s, a, \mathcal{C}')$ and $W^h(s, a, \mathcal{C}')$ having been computed. Finally, Algorithm 14 will terminate in $\mathcal{O}(n(n^2 + m))$ time, since each state belongs to a distinct equivalence class in the worst case. ■

## 9.3 Reward-Bounded Reachability Probability for *UwMDP*s

In this section we shall first define maximal reward-bounded reachability probabilities for *UwMDP*s formally, and then show that they can be computed efficiently in pseudo-polynomial time. To this end, we first define the accumulated reward of a path before reaching a set of goal states in an *UwMDP*.

**Definition 9.5 (Accumulated reward in *UwMDP*s).** *Given an UwMDP $\mathcal{W}$, a reward structure $\mathtt{r}_{\mathcal{W}}$, a path $\xi = s_0 \mathfrak{w}^{a_0}_{s_0 s_1} s_1 \mathfrak{w}^{a_1}_{s_1 s_2} s_2 \cdots \in Paths^{inf}_{\mathcal{W}}$ and a set $G_{\mathcal{W}} \subseteq S_{\mathcal{W}}$ of states, we denote by $rew(\xi, G_{\mathcal{W}})$ the accumulated reward along $\xi$ until $G_{\mathcal{W}}$ is reached; Formally, if $\xi[t] \in G_{\mathcal{W}}$ for some $t \geq 0$ then $rew(\xi, G_{\mathcal{W}}) = \sum_{i=0}^{n-1} \mathtt{r}_{\mathcal{W}}(s_i, a_i, \mathfrak{w}^{a_i}_{s_i s_{i+1}})$ where $n \in \mathbb{N}_0$ is the smallest integer such that $\xi[n] \in G_{\mathcal{W}}$; otherwise $\infty$ if $\xi[t] \notin G_{\mathcal{W}}$ for every $t \geq 0$.*

Below defines *maximal reward-bounded reachability probabilities for UwMDP*s:

**Definition 9.6 (Maximal reward-bounded reachability probabilities for *UwMDP*s).** *Let $\lambda \in \Lambda$ and $\gamma \in \Gamma$ be a scheduler and a nature of a given UwMDP $\mathcal{W}$. Let $G_{\mathcal{W}}$ be a set of goal states. Define $\mathbf{Pr}^{\lambda,\gamma}_{\mathcal{W},G_{\mathcal{W}}} : S_{\mathcal{W}} \times \mathbb{N}_0 \to [0,1]$ by: $\mathbf{Pr}^{\lambda,\gamma}_{\mathcal{W},G_{\mathcal{W}}}(s, \mathrm{R}) := \mathbf{Pr}^{\lambda,\gamma}_{\mathcal{W},s}(\Omega^{\mathrm{R}}_{G_{\mathcal{W}}})$ where $\Omega^{\mathrm{R}}_{G_{\mathcal{W}}} := \{\xi \in Paths^{s}_{\mathcal{W}} \mid rew(\xi, G_{\mathcal{W}}) \leq \mathrm{R}\}$. Define $\mathbf{Pr}^{max}_{\mathcal{W},G_{\mathcal{W}}}(s, \mathrm{R}) : S_{\mathcal{W}} \times \mathbb{N}_0 \to [0,1]$ by:*

$$\mathbf{Pr}^{max}_{\mathcal{W},G_{\mathcal{W}}}(s, \mathrm{R}) := \sup_{\lambda \in \Lambda} \sup_{\gamma \in \Gamma} \mathbf{Pr}^{\lambda,\gamma}_{\mathcal{W},G_{\mathcal{W}}}(s, \mathrm{R}). \tag{9.1}$$

From the definition, we can see that $\Omega^{\mathrm{R}}_{G_{\mathcal{W}}}$ is the set of paths which can reach $G_{\mathcal{W}}$ within the reward bound R. Therefore, $\mathbf{Pr}^{max}_{\mathcal{W},G_{\mathcal{W}}}(s, \mathrm{R})$ denotes the maximal probability of reaching states in $G_{\mathcal{W}}$ from $s$ within R rewards. It is easy to see that $\Omega^{\mathrm{R}}_{G_{\mathcal{W}}}$ is measurable and all functions in Definition 9.6 are well-defined.

**Remark 9.2.** *Here we follow the convention of model checking MDPs by considering all possible resolutions of nondeterminism in an UwMDP. Differently, in UwMDPs, there are two levels of nondeterminism resolved by schedulers and natures, respectively. Therefore, the maximal reachability probability is achieved by maximizing over all possible schedulers and natures as in Eq. (9.1). We mention that a distinction between schedulers and natures is not necessary. However, such a distinction increases readability of proofs and arguments discussed in the chapter.*

In order to compute maximal reward-bounded reachability probabilities, we could have transformed each *UwMDP* to its equivalent *wMDP*, for which existing algorithms [AHK03] can be applied. However, as we mentioned before, the transformation may result in a *wMDP* exponentially larger than the original *UwMDP*, which makes the computation tedious. Instead, we show that each *UwMDP* $\mathcal{W}$ can be alternatively transformed into an *IMDP* $\mathcal{M}$, where maximal reward-bounded reachability probabilities are preserved and can be computed efficiently. Notably, the size of the resulting $\mathcal{M}$ is in $\mathcal{O}(|\mathcal{W}|W)$, i.e., pseudo polynomial, where

$$W = \max\{w \mid \exists s \xrightarrow{a,w} \mu \wedge \mathtt{r}_{\mathcal{W}}(s,a,w) \leq \mathrm{R}\} \tag{9.2}$$

for a given reward structure $\mathtt{r}_{\mathcal{W}}$ and a reward bound R. Clearly, $W \leq \max\{w \mid \exists s \xrightarrow{a,w} \mu\}$.

Now we describe in details how an *UwMDP* can be transformed into an *IMDP*.

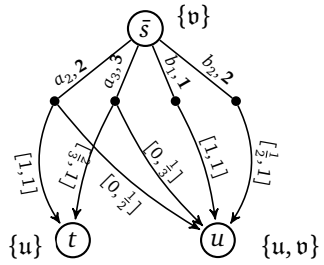



Figure 9.2: The resultant *IMDP* $\mathcal{M}$ generated from the $\mathcal{W}$ in Figure 9.1.

**Definition 9.7 (Model transformation).** *Given an UwMDP* $\mathcal{W} = (S_{\mathcal{W}}, \mathscr{A}_{\mathcal{W}}, W, \bar{s}_{\mathcal{W}}, AP_{\mathcal{W}}, L_{\mathcal{W}})$*, a given set of goal states* $G_{\mathcal{W}} \subseteq S_{\mathcal{W}}$*, a reward structure* $\mathtt{r}_{\mathcal{W}}$*, and a reward bound* $\mathrm{R} \in \mathbb{N}_0$*, the corresponding IMDP* $\mathcal{M} = (S_{\mathcal{M}}, \bar{s}_{\mathcal{M}}, \mathscr{A}_{\mathcal{M}}, AP_{\mathcal{M}}, L_{\mathcal{M}}, I)$ *is defined as follows:* $S_{\mathcal{M}} = S_{\mathcal{W}}$*,* $\bar{s}_{\mathcal{M}} = \bar{s}_{\mathcal{W}}$*,* $AP_{\mathcal{M}} = AP_{\mathcal{W}}$*,* $L_{\mathcal{M}} = L_{\mathcal{W}}$*,* $\mathscr{A}_{\mathcal{M}} = \{a_w | a \in \mathscr{A}_{\mathcal{W}}, w \in \{1, \cdots, W\}\}$*, where* $W$ *is defined as in Eq. (9.2). The reward structure* $\mathtt{r}_{\mathcal{M}}$ *of* $\mathcal{M}$ *is defined by:* $\mathtt{r}_{\mathcal{M}}(s, a_w) = \mathtt{r}_{\mathcal{W}}(s, a, w)$ *for each* $s \in S_{\mathcal{M}}$ *and* $a_w \in \mathscr{A}_{\mathcal{M}}(s)$*. Moreover,* $I(s, a_w, t) := (\frac{1}{w} \cdot W(s, a, t)) \cap [0, 1]$ *for each* $s, t \in S_{\mathcal{M}}$ *and* $a_w \in \mathscr{A}_{\mathcal{M}}(s)$*, provided* $I(s, a_w, t) \neq \emptyset$*. Finally, in the resulting IMDP* $\mathcal{M}$*, the set of goal states is* $G_{\mathcal{M}} = G_{\mathcal{W}}$ *and the initial state and the reward bound are the same as the ones in UwMDP* $\mathcal{W}$*.*

**Example 9.2.** *Consider the UwMDP* $\mathcal{W}$ *depicted in Figure 9.1. Assume the reward bound is* $\mathrm{R} = 4$*. The value of* $W$ *is computed as*

$$W = \max\{w \mid \exists s \xrightarrow{a,w} \mu \wedge \mathtt{r}_{\mathcal{W}}(s, a, w) \leq 4\} = 3$$

*According to the transformation described in Definition 9.7, we obtain an IMDP* $\mathcal{M}$*, which is depicted as in Figure 9.2. The bold numbers indicated besides the actions are the reward structure for the generated IMDP* $\mathcal{M}$*.* ⧫

Directly from Definition 9.7, we can see that the transformation can be done in time linear in both the size of the original *UwMDP* and $W$ defined as in Eq. (9.2). Therefore, the size of the resultant *IMDP* is in $\mathcal{O}(|\mathcal{W}|\, W)$. Without loss of generality, in the sequel we assume $\mathtt{r}(s, a, w) = w$, hence $W = \mathrm{R}$.

**Remark 9.3.** *In Definition 9.7, we show a procedure to transform an UwMDP to an IMDP preserving all maximal reward-bounded reachability probabilities with reward less than a given value. By setting* $W$ *to be the largest possible reward associated to transitions in an UwMDP in Definition 9.7, we can obtain an equivalent IMDP, where all maximal reward-bounded reachability probabilities coincide with the original UwMDP. Indeed, despite that an UwMDP represents a discrete set of wMDPs, while an IMDP corresponds to a continuous set of MDPs, UwMDPs and IMDPs are closely related with respect to properties considered in this chapter:*

- *Any UwMDP can be transformed into an IMDP by associating its transitions with proper rewards. However, in order to preserve all maximal reward-bounded reachability probabilities, the size of the IMDP may blow up, as its size depends on the largest reward in the UwMDP, which can be any positive integer in principle.*

- *Conversely, any IMDP essentially corresponds to an MDP, whose size may be exponentially larger than the original IMDP. It is obvious that any MDP with rational transition probabilities can be transformed into a wMDP, which in turn can be seen as a special case of UwMDP. Thus, the transformation from rational IMDPs to UwMDPs may also cause an exponentially blow-up.*

**Definition 9.8** (**Maximal reward-bounded reachability probabilities for *IMDP*s**). *Let $\sigma \in \Sigma_{\mathcal{M}}$ and $\pi \in \Pi_{\mathcal{M}}$ be a scheduler and nature of a given IMDP $\mathcal{M}$. Define the function $\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},G_{\mathcal{M}}} : S_{\mathcal{M}} \times \mathbb{N}_0 \to [0,1]$ by: $\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) := \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{G_{\mathcal{M}}})$ where $\Omega^{\mathrm{R}}_{G_{\mathcal{M}}} := \{\xi \in Paths^{s}_{\mathcal{M}} \mid rew(\xi, G_{\mathcal{M}}) \leq \mathrm{R}\}$. Define $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) : S_{\mathcal{M}} \times \mathbb{N}_0 \to [0,1]$ by:*

$$\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) := \sup_{\sigma\in\Sigma_{\mathcal{M}}} \sup_{\pi\in\Pi_{\mathcal{M}}} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) \tag{9.3}$$

*for each $s \in S_{\mathcal{M}}$ and $\mathrm{R} \in \mathbb{N}_0$.*

In the following proposition, we show that our *model transformation* preserves maximal reward-bounded reachability probabilities. More precisely, our transformation guarantees that all optimal resolutions in the *IMDP* can be projected back to the original given *UwMDP*. Formally,

**Proposition 9.2.** *Assume we are given an UwMDP $\mathcal{W} = (S_{\mathcal{W}}, \mathscr{A}_{\mathcal{W}}, W, \bar{s}_{\mathcal{W}}, AP_{\mathcal{W}}, L_{\mathcal{W}})$, a state $s \in S_{\mathcal{W}}$, a set of goal states $G_{\mathcal{W}} \subseteq S_{\mathcal{W}}$, and a reward bound $\mathrm{R} \in \mathbb{N}_0$. Let $\mathcal{M} = (S_{\mathcal{M}}, \bar{s}_{\mathcal{M}}, \mathscr{A}_{\mathcal{M}}, AP_{\mathcal{M}}, L_{\mathcal{M}}, I)$ be the corresponding IMDP obtained according to Definition 9.7. Then:*

$$\mathbf{Pr}^{max}_{\mathcal{W},G_{\mathcal{W}}}(s,\mathrm{R}) = \mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}). \tag{9.4}$$

*Proof.* For any scheduler $\lambda \in \Lambda$ and nature $\gamma \in \Gamma$ at state $s$ of the given *UwMDP* $\mathcal{W}$, a (simple) scheduler $\sigma \in \Sigma_{\mathcal{M}}$ and nature $\pi \in \Pi_{\mathcal{M}}$ can be constructed in the corresponding *IMDP* $\mathcal{M}$ as follows. For any $a_w \in \mathscr{A}_{\mathcal{M}}(s)$,

$$\sigma(s)(a_w) = \begin{cases} \lambda(s)(a) & \text{if } w = \sum_{s'\in S_{\mathcal{W}}} \gamma(s,a)(s') \\ 0 & \text{otherwise} \end{cases}$$

and if $\sigma(s)(a_w) \neq 0$ then $\pi(s,a_w)(s') = \frac{\gamma(s,a)(s')}{w}$. We hence, based on Definitions 9.6 and 9.8, obtain that

$$\mathbf{Pr}^{max}_{\mathcal{W},G_{\mathcal{W}}}(s,\mathrm{R}) = \sup_{\lambda\in\Lambda}\sup_{\gamma\in\Gamma} \mathbf{Pr}^{\lambda,\gamma}_{\mathcal{W},G_{\mathcal{W}}}(s,\mathrm{R}) \leq \sup_{\sigma\in\Sigma_{\mathcal{M}}} \sup_{\pi\in\Pi_{\mathcal{M}}} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) = \mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}).$$

On the other hand, suppose that $\sigma^* \in \Sigma_{\mathcal{M}}$ and nature $\pi^* \in \Pi_{\mathcal{M}}$ are optimal scheduler and nature which together achieve the maximum reachability probability in the corresponding *IMDP* $\mathcal{M}$. As we shall see later, based on Theorem 9.2, we can assume that the optimal scheduler and nature are deterministic reward-positional. Hence, we can construct scheduler $\lambda \in \Lambda$ and nature $\gamma \in \Gamma$ in the given *UwMDP* $\mathcal{W}$ as follows. For any $a \in \mathscr{A}_{\mathcal{W}}(s)$,

$$\lambda(s)(a) = \begin{cases} 1 & \text{if } \exists\ w \text{ such that } \sigma^*(s)(a_w) > 0 \\ 0 & \text{otherwise} \end{cases}$$

and if $\lambda(s)(a) > 0$ then $\gamma(s,a)(s') = \lfloor w.\pi^*(s,a_w)(s') \rfloor$ for all $s' \in S_{\mathcal{W}}$. Thus,

$$\mathbf{Pr}^{max}_{\mathcal{W},G_{\mathcal{W}}}(s,\mathrm{R}) = \sup_{\lambda\in\Lambda}\sup_{\gamma\in\Gamma}\mathbf{Pr}^{\lambda,\gamma}_{\mathcal{W},G_{\mathcal{W}}}(s,\mathrm{R}) \geq \sup_{\sigma\in\Sigma_{\mathcal{M}}}\sup_{\pi\in\Pi_{\mathcal{M}}}\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) = \mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}).$$

The conclusion then follows. ■

Due to this result, in the rest of this section, we turn our attention to *IMDP*s to compute maximal reward-bounded reachability probabilities. Given an *IMDP* $\mathcal{M}$, a state $s \in S_{\mathcal{M}}$, a set of goal states $G_{\mathcal{M}} \subseteq S_{\mathcal{M}}$, and a reward bound $\mathrm{R} \in \mathbb{N}_0$, we shall present a routine to compute $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R})$.

We first define reward-positional schedulers and natures and then show that deterministic reward-positional schedulers and natures suffice to obtain the maximal reward-bounded reachability probabilities in an *IMDP*.

**Definition 9.9 (Reward-positional schedulers and natures in *IMDP*s).** *Suppose that $\mathfrak{R}[\xi]$ is total accumulated reward along a finite path $\xi$. A scheduler $\sigma$ is reward-positional if and only if $\sigma(\xi) = \sigma(\xi')$ whenever $last(\xi) = last(\xi')$ and $\mathfrak{R}[\xi] = \mathfrak{R}[\xi']$. Similarly, we can define reward-positional natures.*

In an intuitive description, reward-positional schedulers and natures make their decision entirely on the current state and the reward accumulated so far. Below we show that a deterministic reward-positional scheduler and nature suffices to achieve maximal reward-bounded reachability probability in an *IMDP*.

**Theorem 9.2.** *Given an IMDP $\mathcal{M}$, a set of goal states $G_{\mathcal{M}} \subseteq S_{\mathcal{M}}$, there exist a deterministic reward-positional scheduler $\boldsymbol{\sigma}$ and nature $\boldsymbol{\pi}$ such that $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) = \mathbf{Pr}^{\boldsymbol{\sigma},\boldsymbol{\pi}}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{G_{\mathcal{M}}})$.*

*Proof.* Let $G_{\mathcal{M}} \subseteq S_{\mathcal{M}}$. Let $\preceq$ be a fixed arbitrary linear order on $\mathscr{A}_{\mathcal{M}}$. We prove the theorem in two steps by getting inspiration from [Fu14] as follows.

**Step 1.** We construct a measurable scheduler $\boldsymbol{\sigma}$ and nature $\boldsymbol{\pi}$ for each $\mathrm{R} \in \mathbb{N}_0$. Let us fix $\mathrm{R} \in \mathbb{N}_0$. Define the function

$$\mathfrak{L}(s,a_w,\mathfrak{h}^{a_w}_s,x) := \sum_{s'\in S_{\mathcal{M}}} \mathfrak{h}^{a_w}_{ss'} \,.\, \mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s',\mathrm{R}-x-\mathfrak{r}_{\mathcal{M}}(s,a_w))$$

where $\mathfrak{h}^{a_w}_s$ is a feasible probability distribution resolved by the nature from the uncertainty set $\mathscr{H}^{a_w}_s$. Clearly, if $s \notin G_{\mathcal{M}}$ then $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}-x) = \max_{a_w\in\mathscr{A}_{\mathcal{M}}(s)} \max_{\mathfrak{h}^{a_w}_s\in\mathscr{H}^{a_w}_s} \mathfrak{L}(s,a_w,\mathfrak{h}^{a_w}_s,x)$. Consider a finite path $\xi \in Paths^{fin}_{\mathcal{M}}$. Define

$$S^{\xi,1}_I := \Big\{ s \in S_{\mathcal{M}} \Big| \exists\, a^*_w \in \mathscr{A}_{\mathcal{M}}(s), \exists\, \mathfrak{h}^{a_w*}_s \in \mathscr{H}^{a_w}_s . \Big[ \mathfrak{L}(s,a^*_w,\mathfrak{h}^{a_w*}_s,\mathfrak{R}[\xi]) = \max_{a_w\in\mathscr{A}_{\mathcal{M}}(s)} \max_{\mathfrak{h}^{a_w}_s\in\mathscr{H}^{a_w}_s} \mathfrak{L}(s,a_w,\mathfrak{h}^{a_w}_s,\mathfrak{R}[\xi]) \Big] \wedge \mathfrak{r}_{\mathcal{M}}(s,a^*_w) \neq 0 \Big\}$$

and $S^{\xi,2}_I := \Big\{ s \in S_{\mathcal{M}} \Big| \mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}-\mathfrak{R}[\xi]) = 0 \Big\}$. The probability distributions $\boldsymbol{\sigma}(\xi,\boldsymbol{.})$ and $\boldsymbol{\pi}(s,a_w)(\boldsymbol{.})$ are determined by the following routine:

1. If $last(\xi) \in S^{\xi,2}_I$, then $\boldsymbol{\sigma}(\xi,\boldsymbol{.}) = \mathbf{1}_{\{a^*_w\}}$ and $\boldsymbol{\pi}(s,a_w)(\boldsymbol{.}) = \mathbf{1}_{\{\mathfrak{h}^{a_w*}_s\}}$ where $a^*_w \in \mathscr{A}_{\mathcal{M}}(s)$ and $\mathfrak{h}^{a_w*}_s \in \mathscr{H}^{a_w}_s$ are arbitrarily fixed.

2. If $last(\xi) \in S_I^{\xi,1} \setminus S_I^{\xi,2}$, then $\boldsymbol{\sigma}(\xi, \boldsymbol{.}) = \mathbf{1}_{\{a_w^*\}}$ and $\boldsymbol{\pi}(s, a_w)(\boldsymbol{.}) = \mathbf{1}_{\{\mathfrak{h}_s^{a_w *}\}}$where $a_w^* \in \mathscr{A}_{\mathcal{M}}(s)$ and $\mathfrak{h}_s^{a_w *} \in \mathscr{H}_s^{a_w}$ satisfy the following two conditions:
   i. $\mathfrak{L}(last(\xi), a_w^*, \mathfrak{h}_s^{a_w *}, \mathfrak{R}[\xi]) := \max_{a_w \in \mathscr{A}_{\mathcal{M}}(s)} \max_{\mathfrak{h}_s^{a_w} \in \mathscr{H}_s^{a_w}} \mathfrak{L}(last(\xi), a_w, \mathfrak{h}_s^{a_w}, \mathfrak{R}[\xi])$
   ii. $\mathfrak{r}_{\mathcal{M}}(s, a_w^*) \neq 0$
3. If $last(\xi) \in S_{\mathcal{M}} \setminus (S_I^{\xi,1} \cup S_I^{\xi,2})$, then $\boldsymbol{\sigma}(\xi, \boldsymbol{.}) = \mathbf{1}_{\{a_w^*\}}$ and $\boldsymbol{\pi}(s, a_w)(\boldsymbol{.}) = \mathbf{1}_{\{\mathfrak{h}_s^{a_w *}\}}$where $a_w^* \in \mathscr{A}_{\mathcal{M}}(s)$ and $\mathfrak{h}_s^{a_w *} \in \mathscr{H}_s^{a_w}$ satisfy the following two conditions:
   i. $\mathfrak{L}(last(\xi), a_w^*, \mathfrak{h}_s^{a_w *}, \mathfrak{R}[\xi]) := \max_{a_w \in \mathscr{A}_{\mathcal{M}}(s)} \max_{\mathfrak{h}_s^{a_w} \in \mathscr{H}_s^{a_w}} \mathfrak{L}(last(\xi), a_w, \mathfrak{h}_s^{a_w}, \mathfrak{R}[\xi])$
   ii. there exists $s \in S_{\mathcal{M}}$ such that $\mathfrak{h}^{a_w^*}_{last(\xi)s'} > 0$ and the distance from $s$ to $S_I^{\xi,1} \cup S_I^{\xi,2}$ is one step smaller than that from $last(\xi)$.

By definition, $\boldsymbol{\sigma}(\xi, \boldsymbol{.})$ and $\boldsymbol{\pi}(s, a_w)(\boldsymbol{.})$ are reward-positional.

**Step 2.** We show that $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s, \mathrm{R}) = \mathbf{Pr}^{\boldsymbol{\sigma},\boldsymbol{\pi}}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{G_{\mathcal{M}}})$ for all $s \in S_{\mathcal{M}}$ and $\mathrm{R} \in \mathbb{N}_0$. The proof is done by induction on the remaining value of reward as follows. Let us fix $\mathrm{R} \in \mathbb{N}_0$. Assume that $x \in [0, \mathrm{R}] \cap \mathbb{N}_0$ is the value of residual reward. The base case when $x = 0$ can be easily seen. Since no transition can be performed, thus the class of scheduler and nature has no influence. As regards the inductive step, assume $x = k$. Let $\boldsymbol{\sigma}$ and $\boldsymbol{\pi}$ be reward-positional scheduler and nature that are optimal for residual reward $x$, i.e., $\mathbf{Pr}^{\boldsymbol{\sigma},\boldsymbol{\pi}}_{\mathcal{M},s}(\Omega^{x}_{G_{\mathcal{M}}}) = \mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s, x)$. To find the optimal strategy and nature from state $s$ under the reward bound $x = k + 1$, we choose an action and nature that lead to the highest probability under scheduler $\boldsymbol{\sigma}$ and nature $\boldsymbol{\pi}$. The latter is formalized later in Theorem 9.3.

■

Because of Theorem 9.2, we shall assume all schedulers and natures are deterministic reward-positional in the sequel.

**Corollary 9.1.** *Given an IMDP $\mathcal{M}$ equivalent to some UwMDP $\mathcal{W}$, let $\sigma \in \Sigma_{\mathcal{M}}$ and $\pi \in \Pi_{\mathcal{M}}$ be a scheduler and nature of the given $\mathcal{M}$. The function $\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},G_{\mathcal{M}}}(s, \mathrm{R})$ satisfies the following conditions:*

1. *If $s \in G_{\mathcal{M}}$, then $\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},G_{\mathcal{M}}}(s, \mathrm{R}) = 1$;*
2. *If $s \notin G_{\mathcal{M}}$, then*

$$\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},G_{\mathcal{M}}}(s, \mathrm{R}) = \sum_{a_w \in \mathscr{A}_{\mathcal{M}}(s) \wedge \mathfrak{r}_{\mathcal{M}}(s,a_w) \leq \mathrm{R}} \sigma(s)(a_w). \left\{ \sum_{s' \in S_{\mathcal{M}}} \mathfrak{h}^{a_w}_{ss'} \ . \ \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},G_{\mathcal{M}}}(s', \mathrm{R} - \mathfrak{r}_{\mathcal{M}}(s, a_w)) \right\}$$

*where $\mathfrak{h}^{a_w}_{ss'}$ is a feasible transition probability resolved by $\pi(s, a_w)$, i.e, $\mathfrak{h}^{a_w}_{ss'} = \pi(s, a_w)(s')$.*

*Proof.* The result directly follows from Definitions 4.11 and 9.8. ■

In the following theorem, we present the fixed-point characterization for $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s, \mathrm{R})$.

**Theorem 9.3 (Fixed-point characterization for $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s, \mathrm{R})$).** *Given an IMDP $\mathcal{M}$ equivalent to some UwMDP $\mathcal{W}$, the function $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(\boldsymbol{.}, \boldsymbol{.})$ is the least fixed-point (w.r.t $\leq$) of the high-order operator $\mathfrak{F}_{G_{\mathcal{M}}} : [S_{\mathcal{M}} \times \mathbb{N}_0 \to [0, 1]] \to [S_{\mathcal{M}} \times \mathbb{N}_0 \to [0, 1]]$ defined as follows:*

- $\mathfrak{F}_{G_{\mathcal{M}}}(h)(s,\mathrm{R}) = 1$ *for all* $s \in G_{\mathcal{M}}$ *and* $\mathrm{R} \in \mathbb{N}_0$ ;
- *For any given* $s \notin G_{\mathcal{M}}$ ,

$$\mathfrak{F}_{G_{\mathcal{M}}}(h)(s,\mathrm{R}) = \max_{a_w \in \mathscr{A}_{\mathcal{M}}(s) \wedge \mathfrak{r}_{\mathcal{M}}(s,a_w) \leq \mathrm{R}} \max_{\mathfrak{h}_s^{a_w} \in \mathscr{H}_s^{a_w}} \sum_{s' \in S_{\mathcal{M}}} \mathfrak{h}_{ss'}^{a_w} \,.\, h(s', \mathrm{R} - \mathfrak{r}_{\mathcal{M}}(s,a_w))$$

*for each* $h : S_{\mathcal{M}} \times \mathbb{N}_0 \to [0,1]$.

*Proof.* Suppose that $\mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}}$ is the maximum reward-bounded reachability probability function within $n$ steps. Formally, the function $\mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}} : [S_{\mathcal{M}} \times \mathbb{N}_0 \to [0,1]]$ is defined as:

$$\mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) := \sup_{\sigma \in \Sigma_{\mathcal{M}}} \sup_{\pi \in \Pi_{\mathcal{M}}} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{n,G_{\mathcal{M}}})$$

where $\Omega^{\mathrm{R}}_{n,G_{\mathcal{M}}} := \{\xi \in Paths^s_{\mathcal{M}} \big|\ rew(\xi, G_{\mathcal{M}}) \leq \mathrm{R} \text{ and } \xi[m] \in G_{\mathcal{M}} \text{ for some } 0 \leq m \leq n\}$. We prove the theorem in three steps by getting inspiration from [Fu14] as follows.

**Step 1.** We show by induction on $n$ that the following three statements hold.

(a) $\mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) = 1$ if $s \in G_{\mathcal{M}}$;

(b) $\mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) = 0$ if $s \notin G_{\mathcal{M}}$ and $\mathrm{R} \leq 0$;

(c) If $n > 0$ and $s \notin G_{\mathcal{M}}$ then $\mathbf{Pr}^{max,n+1}_{\mathcal{M},G_{\mathcal{M}}} = \mathfrak{F}_{G_{\mathcal{M}}}(\mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}})$.

The base step when $n = 0$ can be easily seen. Since

$$\mathbf{Pr}^{max,0}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) := \sup_{\sigma \in \Sigma_{\mathcal{M}}} \sup_{\pi \in \Pi_{\mathcal{M}}} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{0,G_{\mathcal{M}}}),$$

it is obvious that $\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{0,G_{\mathcal{M}}}) = \mathbf{1}_{G_{\mathcal{M}}}(s).1 = \mathbf{1}_{G_{\mathcal{M}}}(s)$ for all measurable schedulers $\sigma$ and natures $\pi$. Therefore, (a) and (b) holds true and (c) is vacuum true. For the inductive step, let $n = k+1$ with $k \geq 0$. It is easy to see that (a) and (b) holds true. In the following, we show that (c) also holds true. Let $n > 0$ and $s \notin G_{\mathcal{M}}$. By Corollary 9.1, for all measurable schedulers $\sigma$ and natures $\pi$,

$$\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{k+1,G_{\mathcal{M}}}) = \sum_{a_w \in \mathscr{A}_{\mathcal{M}}(s)} \sigma(s)(a_w).\left\{ \sum_{s' \in S_{\mathcal{M}}} \mathfrak{h}_{ss'}^{a_w} \ .\ \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s'}(\Omega^{\mathrm{R}-\mathfrak{r}_{\mathcal{M}}(s,a_w)}_{k,G_{\mathcal{M}}}) \right\}.$$

If we modify $\pi$ to $\pi'$ by setting $\pi'(s,a_w)(s') = \widehat{\mathfrak{h}_{ss'}^{a_w}}$ where

$$\widehat{\mathfrak{h}_s^{a_w}} = \operatorname*{arg\,max}_{\mathfrak{h}_s^{a_w} \in \mathscr{H}_s^{a_w}} \sum_{s' \in S_{\mathcal{M}}} \mathfrak{h}_{ss'}^{a_w} \ .\ \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s'}(\Omega^{\mathrm{R}-\mathfrak{r}_{\mathcal{M}}(s,a_w)}_{k,G_{\mathcal{M}}})$$

and modify $\sigma$ to $\sigma'$ by setting $\sigma'(s,\boldsymbol{.}) = \sigma(s,\widehat{a})$ where

$$\widehat{a} = \operatorname*{arg\,max}_{a_w \in \mathscr{A}_{\mathcal{M}}(s)} \sum_{s' \in S_{\mathcal{M}}} \widehat{\mathfrak{h}_{ss'}^{a_w}} \ .\ \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s'}(\Omega^{\mathrm{R}-\mathfrak{r}_{\mathcal{M}}(s,a_w)}_{k,G_{\mathcal{M}}})$$

and $\sigma'(\xi,.) = \sigma(\xi,.)$ and $\pi'(\xi,a_w)(.) = \pi(\xi,a_w)(.)$ if $\xi \neq s$; then $\sigma'$ and $\pi'$ are two measurable scheduler and nature which satisfy

$$\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{k+1,G_{\mathcal{M}}}) \leq \mathbf{Pr}^{\sigma',\pi'}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{k+1,G_{\mathcal{M}}}).$$

Thus we have

$$\mathbf{Pr}^{max,k+1}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) := \sup_{\sigma\in\Sigma_{\mathcal{M}}}\sup_{\pi\in\Pi_{\mathcal{M}}}\max_{a_w\in\mathscr{A}_{\mathcal{M}}(s)}\max_{\mathfrak{h}^{a_w}_s\in\mathscr{H}^{a_w}_s}\sum_{s'\in S_{\mathcal{M}}}\mathfrak{h}^{a_w}_{ss'}\,.\,\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s'}(\Omega^{\mathrm{R}-\mathfrak{r}_{\mathcal{M}}(s,a_w)}_{k,G_{\mathcal{M}}}).$$

We now prove the inductive step for (c). In particular, we show that the value of

$$\mathbf{Pr}^{max,k+1}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) := \sup_{\sigma\in\Sigma_{\mathcal{M}}}\sup_{\pi\in\Pi_{\mathcal{M}}}\max_{a_w\in\mathscr{A}_{\mathcal{M}}(s)}\max_{\mathfrak{h}^{a_w}_s\in\mathscr{H}^{a_w}_s}\mathfrak{J}(s,a_w,\mathfrak{h}^{a_w}_s,\mathrm{R})$$

where $\mathfrak{J}(s,a_w,\mathfrak{h}^{a_w}_s,\mathrm{R}) := \sum_{s'\in S_{\mathcal{M}}}\mathfrak{h}^{a_w}_{ss'}\,.\,\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s'}(\Omega^{\mathrm{R}-\mathfrak{r}_{\mathcal{M}}(s,a_w)}_{k,G_{\mathcal{M}}})$ equals

$$\max_{a_w\in\mathscr{A}_{\mathcal{M}}(s)}\mathfrak{L}(s,a_w,\mathrm{R})$$

where

$$\mathfrak{L}(s,a_w,\mathrm{R}) := \max_{\mathfrak{h}^{a_w}_s\in\mathscr{H}^{a_w}_s}\mathfrak{L}(s,a_w,\mathfrak{h}^{a_w}_s,\mathrm{R})$$

and

$$\mathfrak{L}(s,a_w,\mathfrak{h}^{a_w}_s,\mathrm{R}) := \sum_{s'\in S_{\mathcal{M}}}\mathfrak{h}^{a_w}_{ss'}\,.\,\mathbf{Pr}^{max,k}_{\mathcal{M},G_{\mathcal{M}}}(s',\mathrm{R}-\mathfrak{r}_{\mathcal{M}}(s,a_w)).$$

It is not difficult to see that the latter is greater than the former. In order to prove the reverse side, assume that $\overline{a_w} = \arg\max_{a_w\in\mathscr{A}_{\mathcal{M}}(s)}\mathfrak{L}(s,a_w,\mathrm{R})$ and $\overline{\mathfrak{h}^{a_w}_s} = \arg\max_{\mathfrak{h}^{a_w}_s\in\mathscr{H}^{a_w}_s}\mathfrak{L}(s,a_w,\mathfrak{h}^{a_w}_s,\mathrm{R})$. We consider two cases:

**Case 1.** $\mathfrak{r}_{\mathcal{M}}(s,\overline{a_w}) = 0$. For all $\varepsilon > 0$, we can choose scheduler $\sigma^\varepsilon$ and nature $\pi^\varepsilon$ such that

- $\sigma^\varepsilon(s,.) = \sigma(s,\overline{a_w})$;
- $\pi^\varepsilon(s,a_w)(s') = \pi(s,\overline{a_w})(s') = \overline{\mathfrak{h}^{\overline{a_w}}_{ss'}}$

and

- $\sigma^\varepsilon(s\overline{\mathfrak{h}^{\overline{a_w}}_{ss'}}s'\xi,.) = \sigma^{k,\varepsilon}(s'\xi,.)$;
- $\pi^\varepsilon(s',\sigma^{k,\varepsilon}(s'\xi,.))(.) = \pi^{k,\varepsilon}(s',\sigma^{k,\varepsilon}(s'\xi,.))(.)$

where $\sigma^{k,\varepsilon}(s'\xi,.)$ and $\pi^{k,\varepsilon}(s',\sigma^{k,\varepsilon}(s'\xi,.))(.)$ are measurable scheduler and nature for such that

$$\mathbf{Pr}^{\sigma^{k,\varepsilon},\pi^{k,\varepsilon}}_{\mathcal{M},s'}(\Omega^{\mathrm{R}}_{k,G_{\mathcal{M}}}) \geq \mathbf{Pr}^{max,k}_{\mathcal{M},G_{\mathcal{M}}}(s',\mathrm{R}) - \varepsilon.$$

It is not hard to see that for $\sigma^\varepsilon$ and $\pi^\varepsilon$,

$$\max_{a_w\in\mathscr{A}_{\mathcal{M}}(s)}\max_{\mathfrak{h}^{a_w}_s\in\mathscr{H}^{a_w}_s}\sum_{s'\in S_{\mathcal{M}}}\mathfrak{h}^{a_w}_{ss'}\,.\,\mathbf{Pr}^{\sigma^\varepsilon,\pi^\varepsilon}_{\mathcal{M},s'}(\Omega^{\mathrm{R}-\mathfrak{r}_{\mathcal{M}}(s,a_w)}_{k,G_{\mathcal{M}}}) \geq \mathfrak{L}(s,\overline{a_w},\overline{\mathfrak{h}^{\overline{a_w}}_s},\mathrm{R}) - \varepsilon.$$

Thus, $\mathbf{Pr}^{max,k+1}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) = \mathfrak{L}(s,\overline{a_w},\overline{\mathfrak{h}^{\overline{a_w}}_s},\mathrm{R})$ by the arbitrary choice of $\varepsilon$.

**Case 2.** $\mathtt{r}_{\mathcal{M}}(s, \overline{a_w}) \neq 0$. The proof follows the same direction as case 1.

**Step 2.** We show that $\lim_{n\to+\infty} \mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) = \mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R})$. Suppose that $s \in S_{\mathcal{M}}$. Be definition, it is easy to see that $\mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}} \leq \mathbf{Pr}^{max,n+1}_{\mathcal{M},G_{\mathcal{M}}}$ and $\mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}} \leq \mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R})$ for all $n \geq 0$. Therefore, $\lim_{n\to+\infty} \mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R})$ exists and it is at most $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R})$. As regards the reverse direction, let us fix an arbitrary $\varepsilon > 0$. Suppose that $\sigma$ and $\pi$ are measurable scheduler and nature such that $\mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{G_{\mathcal{M}}}) \geq \mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) - \varepsilon$. By definition, $\mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) \geq \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{n,G_{\mathcal{M}}})$. It thus follows that $\lim_{n\to+\infty} \mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}} \geq \lim_{n\to+\infty} \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{n,G_{\mathcal{M}}}) = \mathbf{Pr}^{\sigma,\pi}_{\mathcal{M},s}(\Omega^{\mathrm{R}}_{G_{\mathcal{M}}}) \geq \mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) - \varepsilon$. Since $\varepsilon$ was arbitrary chosen, thus $\lim_{n\to+\infty} \mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) = \mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R})$.

**Step 3.** In the last step, we prove that $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}$ is actually the least fixed-point of the high-order operator $\mathfrak{F}_{G_{\mathcal{M}}}$. If $s \in G_{\mathcal{M}}$ then clearly $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) = 1$. By using Monotone Convergence Theorem [Hal50] on (c), we conclude that $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R}) = \mathfrak{F}_{G_{\mathcal{M}}}(\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}})(s,\mathrm{R})$. Thus, $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}$ is actually the fixed-point of the high-order operator $\mathfrak{F}_{G_{\mathcal{M}}}(\centerdot)$. To see that it is the least fixed-point of $\mathfrak{F}_{G_{\mathcal{M}}}$, we can continue by induction on $n \geq 0$ and show that for any given fixed-point $l$, $\mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}} \leq l$ for all $n \geq 0$. Based on the facts that $\mathbf{Pr}^{max,0}_{\mathcal{M},G_{\mathcal{M}}} \leq l$ and $\mathbf{Pr}^{max,n+1}_{\mathcal{M},G_{\mathcal{M}}} = \mathfrak{F}_{G_{\mathcal{M}}}(\mathbf{Pr}^{max,n}_{\mathcal{M},G_{\mathcal{M}}})$, it follows that $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}} \leq l$ for any fixed-point $l$.

■

We show that the problem of computing the maximal reward-bounded reachability probability in an *IMDP* can be reduced to solving a sequence of linear programming problems. The algorithm is shown in Algorithm 16. Intuitively, we let *probs* store all computed reachability probabilities such that $probs[i][r]$ is the maximal probability of reaching $G_{\mathcal{M}}$ from $s_i$ within the reward bound $r$. We compute *probs* inductively starting from $r = 1$ until $r = \mathrm{R}$. At each step all probabilities $probs[i][r]$ are computed using values $probs[i][r']$ with $r' < r$ which have been computed before. From the moment we fix a bound $r$. Let $x_i$ denote the probability $probs[i][r]$. Let $y_{i,j,a,w}$ be the probability of going to $s_j$ by choosing the transition with label $a_w$ when at state $s_i$. Hence the constraint in line 8 has to be satisfied. Line 9 guarantees the probability mass from $s_i$ sums up to 1. The variables $y_{i,j,a,w}$ are not arbitrary but have to be within their corresponding given bounds. This is guaranteed in line 10.

In the next lemma, we discuss the time complexity of the proposed routine to compute $\mathbf{Pr}^{max}_{\mathcal{M},G_{\mathcal{M}}}(s,\mathrm{R})$. Formally,

**Lemma 9.2.** *Algorithm 16 is sound and complete, and it is guaranteed to terminate in time polynomial with respect to the size of IMDP $\mathcal{M}$ and the reward bound* R.

*Proof.* The proof of Lemma 9.2 makes use of the following key result in linear programming:

**Proposition 9.3** (cf. [Kha79, Kar84])**.** *Given the linear program:*

$$\begin{array}{rl} \min & \mathbf{c}^T\mathbf{x} \\ \textit{subject to:} & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{array}$$

**Algorithm 16:** Computing maximal reward-bounded reachability

**Input:** An *IMDP* $\mathcal{M}$, a state $s_0$, a set of goal states $G_{\mathcal{M}}$, and a reward bound R.
**Output:** The maximal probability of reaching $G_{\mathcal{M}}$ from $s_0$ within the bound R.

1 **begin**
2 $n \leftarrow |S_{\mathcal{M}}|$;
3 $\forall 0 \leq i < n, 0 \leq r \leq \mathrm{R}$. $probs[i][r] = (1$ if $s_i \in G_{\mathcal{M}}$ else $0)$;
4 **for** $(r = 1$ *to* $\mathrm{R})$ **do**
5 $\forall 0 \leq i < n$. $probs[i][r] = x_i$, where $x_i$ is determined by the following LP problem;
6 $\min \sum_{0 \leq i < n} x_i$;
7 **for** $(0 \leq i < n$ *and* $a_w \in \mathscr{A}_{\mathcal{M}}(s_i)$ *with* $\mathtt{r}_{\mathcal{M}}(s_i, a_w) \leq r)$ **do**
8 $x_i \geq \sum_{0 \leq j < n} probs[j][r - \mathtt{r}_{\mathcal{M}}(s_i, a_w)] \cdot y_{i,j,a,w}$;
9 $\sum_{0 \leq j < n} y_{i,j,a,w} = 1$;
10 $\forall 0 \leq j < n. y_{i,j,a,w} \in I(s_i, a_w, s_j)$;
11 **return** $probs[0][\mathrm{R}]$;

*in which* **x** *represents the vector of variables (to be determined),* **c** *and* **b** *are vectors of (known) coefficients, A is an (known) $m \times n$ matrix of coefficients, and $(\cdot)^{\mathrm{T}}$ is the matrix transpose; the optimal solution can be found to within error $\pm\varepsilon$ in time complexity that is polynomial in the size of problem $(n, m)$ and $\log(1/\varepsilon)$.*

Algorithm 16 has one *inner* linear program for each $r \in \{1, \cdots, \mathrm{R}\}$, totally $\mathcal{O}(\mathrm{R})$ linear programming problems. Each linear program has $\mathcal{O}(|S_{\mathcal{M}}|)$ unknowns, representing the maximal probability of reaching goal states $G_{\mathcal{M}}$ within the given reward bound. It is not difficult to see that the number of constraints is linear in the size $|\mathcal{M}|$ of the *IMDP* $\mathcal{M}$. Each inner LP problem can be solved in time complexity that is polynomial in the size $|\mathcal{M}|$ (cf. Proposition 9.3). Therefore, the overall time complexity of algorithm turns to be $\mathcal{O}(poly(|\mathcal{M}|).\mathrm{R})$ which is polynomial in the size of $\mathcal{M}$ and pseudo-polynomial in R. The soundness and completeness of the Algorithm 16 follows directly by polynomial algorithms for LP problems and a simple observation that the accuracy error at each iteration of the outer **for** loop linearly increases due to the linear constraints in line 8. More precisely, while each inner problem is solved with accuracy $\pm\varepsilon_{inn}$ in time linear in $\log(1/\varepsilon_{inn})$ by Proposition 9.3, linear constraints in line 8 guarantee that the accuracy error cannot nonlinearly get amplified. To see this, suppose that $\varepsilon_i^r$ is the error accumulated at step $r$ for state $i$, $x_i^r = x_{i,nep}^r + \varepsilon_i^r$, where $x_{i,nep}^r$ is the solution with *no error propagation*, and $\varepsilon_i^{\mathrm{R}}$ the error in the final solution. Also, assume that $x_i^r$ and $y_{i,j,a,w}^r$ are the optimal solutions of the inner linear programming at step $r$ of the Algorithm 16. Due to linear constraints in line 8, the optimal value $x_i^r$ of the inner problem is computed through $\sum_{0 \leq j < n} x_j^{r - \mathtt{r}_{\mathcal{M}}(s_i, a_w)} \cdot y_{i,j,a,w}^r$, with $\sum_{0 \leq j < n} y_{i,j,a,w}^r = 1$ and $y_{i,j,a,w}^r \in I(s_i, a_w, s_j)$ for all $0 \leq j < n$. At the first iteration,

$$x_i^1 = x_{i,nep}^1 + \varepsilon_i^1 \geq \sum_{0 \leq j < n} x_j^0 \cdot y_{i,j,a,w}^1 + \varepsilon_{inn}$$

and at the second iteration:

$$x_i^2 \geq \sum_{0\leq j<n} x_j^{2-\mathbf{r}_{\mathcal{M}}(s_i,a_w)} \cdot y_{i,j,a,w}^2 + \varepsilon_{inn} = \sum_{\{0\leq j<n|2-\mathbf{r}_{\mathcal{M}}(s_i,a_w)=0\}} x_j^0 \cdot y_{i,j,a,w}^2 + \sum_{\{0\leq j<n|2-\mathbf{r}_{\mathcal{M}}(s_i,a_w)=1\}} x_j^1 \cdot y_{i,j,a,w}^2 + \varepsilon_{inn} \geq \sum_{\{0\leq j<n|2-\mathbf{r}_{\mathcal{M}}(s_i,a_w)=0\}} x_j^0 \cdot y_{i,j,a,w}^2 + \sum_{\{0\leq j<n|2-\mathbf{r}_{\mathcal{M}}(s_i,a_w)=1\}} \{\sum_{0\leq s<n} x_s^0 \cdot y_{i,s,a,w}^1 + \varepsilon_{inn}\} \cdot y_{i,j,a,w}^2 + \varepsilon_{inn} = \sum_{\{0\leq j<n|2-\mathbf{r}_{\mathcal{M}}(s_i,a_w)=0\}} x_j^0 \cdot y_{i,j,a,w}^2 + \sum_{\{0\leq j<n|2-\mathbf{r}_{\mathcal{M}}(s_i,a_w)=1\}} \sum_{0\leq s<n} x_s^0 \cdot y_{i,s,a,w}^1 \cdot y_{i,j,a,w}^2 + \sum_{\{0\leq j<n|2-\mathbf{r}_{\mathcal{M}}(s_i,a_w)=1\}} \varepsilon_{inn} \cdot y_{i,j,a,w}^2 + \varepsilon_{inn} = \sum_{\{0\leq j<n|2-\mathbf{r}_{\mathcal{M}}(s_i,a_w)=0\}} x_j^0 \cdot y_{i,j,a,w}^2 + \sum_{\{0\leq j<n|2-\mathbf{r}_{\mathcal{M}}(s_i,a_w)=1\}} \sum_{0\leq s<n} x_s^0 \cdot y_{i,s,a,w}^1 \cdot y_{i,j,a,w}^2 + 2\varepsilon_{inn}.$$

Following the above inductive procedure, we arrive at a conclusion that the accuracy error $\varepsilon_i^r$ increases linearly with respect to the iteration $r$. The desired accuracy error $\varepsilon_d$ of the final solution can thus be guaranteed by solving each inner linear programming problem with accuracy $\varepsilon_{inn} = \mathcal{O}(\frac{\varepsilon_d}{\mathrm{R}})$. ■

As the transformation in Definition 9.7 is also pseudo polynomial, we obtain the main result of this section.

**Theorem 9.4.** *Maximal reward-bounded reachability probabilities for an UwMDP $\mathcal{W}$ can be computed in pseudo polynomial time $\mathcal{O}(|\mathcal{W}|\,\mathrm{R}^2)$.*

*Proof.* In order to compute maximal reward-bounded reachability probabilities for an *UwMDP* $\mathcal{W}$, we first generate the corresponding *IMDP* $\mathcal{M}$ using the *model transformation* procedure in Definition 9.7 that has size $\mathcal{O}(|\mathcal{W}|.\mathrm{R})$ in the worst case. Afterwards, due to the preservation of reachability probabilities followed by the Proposition 9.2, maximal reward-bounded reachability probabilities on the generated $\mathcal{M}$ is computed in time $\mathcal{O}(|\mathcal{M}|.\mathrm{R})$ (cf. Lemma 9.2). As a result, computing maximal reward-bounded reachability probabilities for an *UwMDP* $\mathcal{W}$ can be done in time $\mathcal{O}(|\mathcal{W}|.\mathrm{R}^2)$ that is linear in the size of $\mathcal{W}$ and pseudo-polynomial in R. ■

**Remark 9.4.** *The extension of Algorithm 16 to deal with minimal reward-bounded reachability is straightforward. We note that extreme reachability probabilities without reward bounds in UwMDPs can also be computed efficiently using the technique presented in [PLSVS13]. The only change we need to make is Definition 9.7, where a reward bound is necessary for the transformation. However, if the reward bound is not available, we can simply let* R *be the maximal weight appearing in the given UwMDP. After that, the algorithm in [PLSVS13] can be applied directly. Along the same routine in [AHK03], the algorithm can be extended to deal with full PRCTL.*

## 9.4 Case Studies

In order to show the effectiveness of our approaches, we developed a prototype tool and applied it to some case studies. In particular, the goal of this experiment is two-fold: 1)

quantitatively evaluate the impact of uncertainty on the results of verification of reward-bounded reachability probabilities of *UwMDP*s; 2) assess the impact of compositional minimization, as a pre-processing step, on speeding up the run time of the model checking algorithm. Our prototype is built upon the tool presented in [PLSVS13], which is able to model check PCTL properties over *IMDP*s. The tool is implemented in Python and relies on MOSEK [MOS] to solve all linear programming problems. All experiments were obtained on a laptop with an Intel i7-4600U 2.1GHz CPU and 4GB RAM running Ubuntu.

### 9.4.1 Autonomous Nondeterministic Tour Guides (ANTG)

Our first case study is inspired by "Autonomous Nondeterministic Tour Guides" (ANTG) in [CRI07], which models a complex museum with a variety of collections. Models in [CRI07] are *MDP*s. In our experiment, we will insert some uncertainties in a way that we will describe in details soon. Due to the popularity of the museum, there are many visitors at the same time. Different visitors may have different preferences of arts. We assume the museum divides all collections into different categories so that visitors can choose what they would like to visit and pay tickets according to their preferences. In order to obtain the best experience, a visitor can first assign certain weights to all categories denoting their preferences to the museum, and then design the best strategy for a target. However, the preference of a sort of arts to a visitor may depend on many factors like price, weather, or the length of queue at that moment etc., hence it is hard to assign fixed values to these preferences. In our model we allow uncertainties of preferences such that their values may lie in an interval.

For simplicity we assume all collections are organized in an $n \times n$ square with $n \geq 10$. Let $m = \frac{n-1}{2}$. We assume all collections at $(i, j)$ are assigned with a weight 1 if $|i-m| > \frac{n}{5}$ or $|j-m| > \frac{n}{5}$, with a weight 2 if $|i-m| \in (\frac{n}{10}, \frac{n}{5}]$ or $|j-m| \in (\frac{n}{10}, \frac{n}{5}]$; otherwise they are assigned with a weight interval $[2, 4]$. In other words, we expect collections in the middle will be more popular and subject to more uncertainties than others. Furthermore, we assume that people at each location $(i, j)$ have two nondeterministic choices: either move to the north and west, or to the north and east if $i \geq j$, while if $i \leq j$, they can move either to the south and west, or to the south and east. Therefore, for a model with parameter $n$, it has $n^2$ states in total and roughly $2n^2$ transitions, 2% of which are associated with uncertain weights. Notice that a transition with uncertain weights essentially corresponds to several transitions with concrete weights. In each ANTG model, the 2% transitions with uncertain transitions contribute to about 20% of transitions in the resultant *wMDP*.

We define a reward structure denoting the reward one can obtain by visiting each collection. For simplicity, we let the reward be the same as the weight of a collection. Let the point $(0, 0)$ be the entrance and $(n-1, n-1)$ the exit. We can ask questions like "Whether it is possible to go through the museum, i.e., from the entrance to the exit, with probability greater than 0.9, while the accumulated reward is not greater than R, i.e., $\mathbf{Pr}_{\geq 0.9}(\mathsf{F}^{\leq \mathrm{R}} \mathit{exit})$".

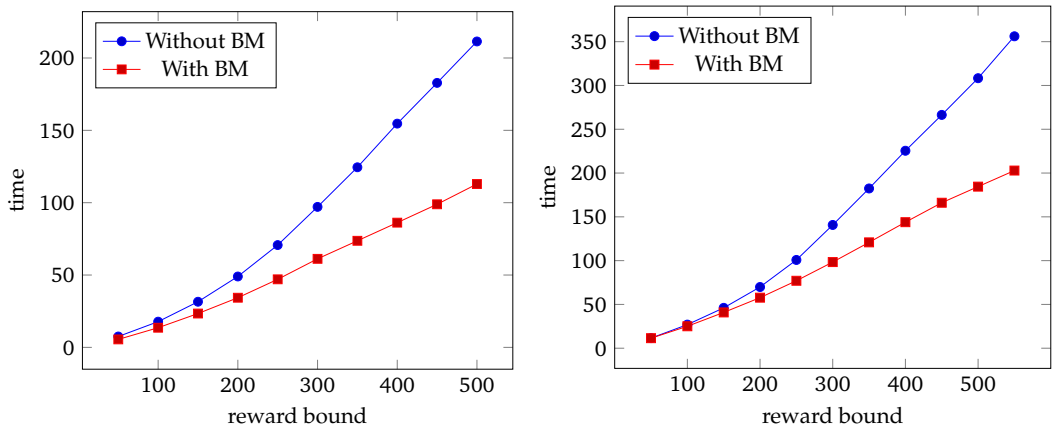
Our experiment results without computing bisimulation quotients are shown in Table 9.1, which presents the time (in minute) taken to compute $\mathbf{Pr}_{\geq 0.9}(\mathsf{F}^{\leq \mathrm{R}} \mathit{exit})$ with corresponding reward bounds and models of different sizes. For each case, we keep increasing the bound until the probability is greater than 0.9. All cells marked with '-' denote cases that we did not reach. We also implemented the algorithm to compute bisimulation relations. Table 9.2 presents the experiment results, where bisimulation minimization was

| n \ R | 50 | 100 | 150 | 200 | 250 | 300 | 350 | 400 | 450 | 500 | 550 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0.08 | - | - | - | - | - | - | - | - | - | - |
| 20 | 0.17 | 0.42 | - | - | - | - | - | - | - | - | - |
| 30 | 0.32 | 0.97 | 1.68 | - | - | - | - | - | - | - | - |
| 40 | 0.56 | 1.62 | 3.37 | 5.04 | - | - | - | - | - | - | - |
| 50 | 1.11 | 2.85 | 5.68 | 8.98 | 12.33 | - | - | - | - | - | - |
| 60 | 1.85 | 4.58 | 8.79 | 14.47 | 20.63 | 26.71 | - | - | - | - | - |
| 70 | 2.95 | 7.13 | 12.95 | 21.63 | 31.73 | 42.30 | 52.69 | - | - | - | - |
| 80 | 5.81 | 12.59 | 22.62 | 37.08 | 56.42 | 74.72 | 94.01 | 114.19 | - | - | - |
| 90 | 7.52 | 17.78 | 31.54 | 48.99 | 70.71 | 97.10 | 124.47 | 154.64 | 182.80 | 211.43 | - |
| 100 | 11.55 | 26.94 | 46.08 | 69.88 | 100.78 | 140.73 | 182.40 | 225.46 | 266.39 | 308.30 | 356.15 |

Table 9.1: Experiment Results without Bisimulation Minimization(in minute)

| n \ R | 50 | 100 | 150 | 200 | 250 | 300 | 350 | 400 | 450 | 500 | 550 | BisimMin | Ratio |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 0.05 | - | - | - | - | - | - | - | - | - | - | 0.00 | 0.63 |
| 20 | 0.15 | 0.30 | - | - | - | - | - | - | - | - | - | 0.0 | 0.71 |
| 30 | 0.13 | 0.41 | 0.72 | - | - | - | - | - | - | - | - | 0.14 | 0.43 |
| 40 | 0.60 | 1.55 | 2.49 | 3.45 | - | - | - | - | - | - | - | 0.48 | 0.68 |
| 50 | 0.45 | 1.53 | 3.95 | 6.78 | 9.20 | - | - | - | - | - | - | 1.03 | 0.75 |
| 60 | 1.43 | 4.02 | 7.41 | 11.68 | 16.09 | 20.21 | - | - | - | - | - | 2.40 | 0.76 |
| 70 | 3.81 | 8.55 | 14.51 | 21.06 | 29.05 | 37.27 | 45.16 | - | - | - | - | 4.21 | 0.86 |
| 80 | 6.74 | 13.09 | 19.39 | 27.10 | 36.07 | 44.88 | 52.48 | 60.01 | - | - | - | 5.27 | 0.53 |
| 90 | 5.51 | 13.59 | 23.39 | 34.32 | 47.03 | 61.20 | 73.67 | 86.21 | 98.89 | 112.90 | - | 6.50 | 0.53 |
| 100 | 11.45 | 25.04 | 40.85 | 57.72 | 77.02 | 98.34 | 120.86 | 143.89 | 166.11 | 184.50 | 202.85 | 9.66 | 0.60 |

Table 9.2: Experiment Results with Bisimulation Minimization (in minute)




(a) $n = 90$ (b) $n = 100$

Figure 9.3: Performance Difference with and without Bisimulation Minimization (in minute)

conducted before performing verification. In this table, reported time is sum of the time

spent to conduct the minimization and the time spent to check the quotient systems. The column "BisimMin" of Table 9.2 denotes the time spent to conduct the minimization, while the column "Ratio" shows ratios between time to compute reachability probabilities with and without bisimulation minimization. All values in column "Ratio" are obtained by comparing time corresponding to the maximal reached reward for each case in Table 9.1 and 9.2. For instance, when $n = 80$, we divide 60.01 by 114.19 ($\mathrm{R} = 400$), hence 0.53 is obtained. Similarly, for the case with $n = 100$ and $\mathrm{R} = 550$, bisimulation minimization accelerated the verification for more than 40%. The time of computing reachability probabilities without/with computing bisimulation quotients is visualized in Figure 9.3(a) for $n = 90$ where "BM" denotes "bisimulation minimization". The counterpart for $n = 100$ is depicted in Figure 9.3(b). We shall see that the larger of the model and the reward bound, the more time we will save by applying bisimulation minimization.

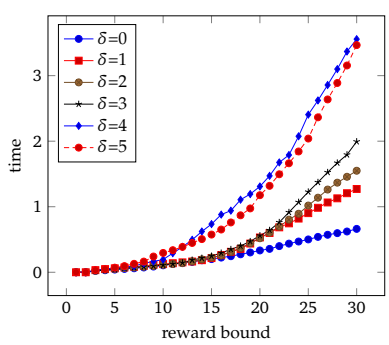
To the best of our knowledge, there is no algorithm or tool, which can deal with *UwMDP*s directly. However, we can reduce the model checking of *UwMDP*s to checking their equivalent *wMDP*s. For instance if $W(s,a,s_1) = [1,2]$ and $W(s,a,s_2) = [2,3]$, then essentially $s$ has four nondeterministic transitions: $s \xrightarrow{a,3} \{\frac{1}{3} : s_1, \frac{2}{3} : s_2\}$, $s \xrightarrow{a,4} \{\frac{1}{4} : s_1, \frac{3}{4} : s_2\}$, $s \xrightarrow{a,4} \{\frac{1}{2} : s_1, \frac{1}{2} : s_2\}$, and $s \xrightarrow{a,5} \{\frac{2}{5} : s_1, \frac{3}{5} : s_2\}$. After resolving all uncertainties, we can apply the existing algorithm [AHK03] to compute maximal reward-bounded reachability probabilities. Obviously, this step may cause an exponential blow-up. Indeed, our experiment showed that when the proportion of states with uncertain weights in an *UwMDP* is not trivial, such enumeration is very time consuming. For instance when $n = 10$ and $\mathrm{R} = 100$, our algorithm took around 135 seconds, while the naïve approach took more than 10 minutes, provided that all transitions are associated with a weight interval $[1,6]$.

### 9.4.2 Randomized Consensus Protocol

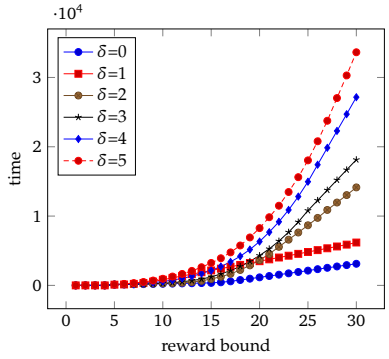
In the second case study we consider *randomized consensus protocol* [AH90,KNS01]. Models are obtained from PRISM benchmarks by adding some uncertainties to the transition probabilities. In order to evaluate how the algorithm scales with respect to weight intervals of different sizes, i.e., their lower and upper bounds and lengths, we performed experiments on models after inserting weight intervals of various sizes. Specifically, instead of using a fair coin, we adopt an unfair coin with uncertainties: After each coin tossing, head and tail will occur with probabilities according to weights in $[6-\delta, 7-\delta]$ and $[8-\delta, 10-\delta]$, respectively, where $\delta$ is an integer in $[0,5]$. We consider the minimal probability of reaching a set of goal states within a given reward bound, where the goal states are those labeled by atomic propositions "finished" and "all_coins_equal_0". The experiment results are shown in Figure 9.4, for which we can see that the amount of uncertainties has an important impact on the verification time, especially when the reward bound is large. This is as expected since the size of the underlying *wMDP* of an *UwMDP* increases exponentially with respect to the amount of uncertainties in the *UwMDP*.

### 9.4.3 Randomized Dining Philosophers

The above observation is further verified by the third case study – *randomized dining philosophers* [LR81]. Same as in the second case study, we modify the original PRISM models by

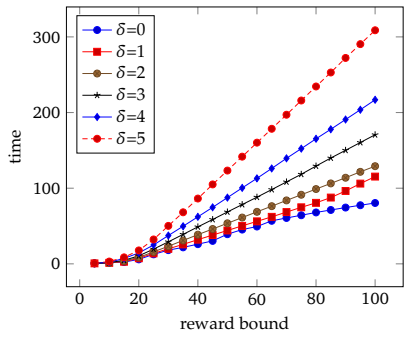



(a) 2 processes, 1296 states, 2412 transitions

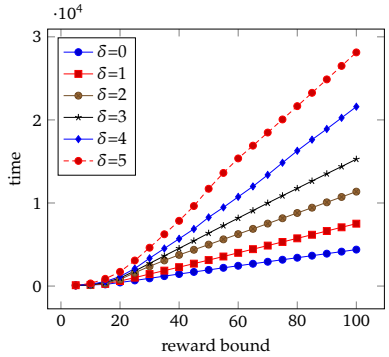



(b) 4 processes, 104576 states, 351712 transitions

Figure 9.4: Experiment results of randomized consensus protocols (in seconds)




(a) 3 philosophers, 956 states, 3696 transitions




(b) 4 philosophers, 9440 states, 48656 transitions

Figure 9.5: Experiment results of randomized dining philosophers (in seconds)

injecting some uncertainties of various sizes. Figure 9.5 shows the time to compute the maximal probabilities of reaching states labeled by "eat" within given reward bounds.

## 9.5 Concluding Remarks

In this chapter, we established a fixed-point characterization for maximal reward-bounded reachability probabilities in *UwMDP*s as well as a pseudo polynomial algorithm to compute these probabilities. In particular, for the sake of verification we assumed *cooperative* resolution of nondeterminisms in which both scheduler and nature are playing *together* to maximize the model performance, i.e., reachability probabilities. We proposed a notion of bisimulation relations for *UwMDP*s and showed that they can be computed efficiently in polynomial time. We have also demonstrated feasibility of our theory via some case studies. All results proposed in this chapter can be extended to model check the

Probabilistic Reward Computation Tree Logic (PRCTL) [AHK03] in a standard way [BK08].

**Related work** Related work falls into two main categories: Verification of PCTL [HJ94] specifications and compositional minimization for uncertain *MDP*s. Probabilistic modeling formalisms with uncertainties have attracted much attention recently. Interval Markov chains [JL91, KU02] or abstract Markov chains [FLW06] extend classical discrete-time Markov chains (*MC*) with uncertainties; however, they do not reflect nondeterminism in transitions. In uncertain *MDP*s [NG05, PLSVS13, WTM12] both nondeterministic and probabilistic choices coexist and more expressive uncertainty sets are allowed to model transition probabilities. Over the last few years, several new verification algorithms for uncertain Markovian models have been proposed in the literature. The problems of computing reachability probabilities and expected total reward were studied for interval Markov chains [CHK13] and interval *MDP*s [WK08]. Model checking of PCTL and LTL has been investigated in [BLW13, CSH08, CHK13] for interval Markov chains and also in [WTM12, PLSVS13] for *IMDP*s. Strategy synthesis for *MDP*s with respect to PCTL properties was first studied in [BGL$^+$04], which was then extended to parametric *MDP*s [HHZ11a] and to *MDP*s with ellipsoidal uncertainty [NG05]. Uncertain Markovian models were also extensively studied in the control community [GLD00, NG05, WK08], with the aim to maximize expected finite-horizon (un)discounted rewards. We are not aware of any existing result related to reward-bounded reachability for uncertain *MDP*s. However, our algorithm is inspired by [AHK03], which deals with the model checking of reachability properties on *MC*s with rewards.

From the point of view of compositional minimization, interval Markov chains [KKN09] and abstract probabilistic automata [DKL$^+$11a, DKL$^+$11b] offer extensive specification theories for Markov chains and probabilistic automata. These theories support both satisfaction and various refinement relations [DLL$^+$11, DKL$^+$11a, JL91]. In [HHK14] probabilistic bisimulation relations were introduced in order to reduce the size of interval MDPs while preserving PCTL properties. Moreover, an algorithm was given to compute the quotients induced by these bisimulations in time polynomial in the size of the model and exponential in the uncertain branching. Notably, we show that for *UwMDP*s, bisimulation quotients can be computed in polynomial time even with respect to the uncertain branching. Furthermore, bisimulation relations are proved to be compositional with respect to a parallel operator. This enables compositional minimization to enhance the model checking of *UwMDP*s.

# Part III

# Conclusion

CHAPTER 10

# Conclusion

Soon after the emergence of embedded systems, their ever increasing complexity has been the primary concern of system designers. To account for the latter, formal verification techniques help to design and verify correctness of these systems through an exhaustive exploration of all possible execution paths of their abstract mathematical model. The interdependency of formal verification methods on the modelling formalisms which provide abstractions of the underlying systems results in guarantees as good as the models. In the real world applications, such modelling formalisms are almost always affected by uncertainties due to measurement errors, statistical estimates, or mathematical approximations. It is thus essential to provide mathematically rigorous techniques for modelling and formal analysis of such systems.

In this dissertation, we have established a framework for modelling, formal verification and optimal control of probabilistic systems. In particular, we provided a thorough complexity analysis and if applicable, efficient decision algorithm to minimize, verify and optimally control these systems. We additionally evaluated the effectiveness of our proposed approaches by applying them on several real world case studies.

## 10.1 Summary

To conclude the dissertation, we first summarize our main achievements.

We started in Chapter 5 with analysis of deciding weak probabilistic bisimulation for probabilistic automata, a discrete time model for systems exhibiting probabilistic and nondeterministic behaviour. This problem was already investigated by Cattani and Segala, who developed an exponential time algorithm in 2002. Some years later, Hermanns and Turrini [HT12] developed a polynomial time algorithm based on a partition refinement procedure where the core part is to decide bisimilairty of a pair of states which is modeled as a linear program that can be solvable in polynomial time. We further refined the complexity analysis to obtain precise polynomial bounds and discussed several optimizations to improve computation speed in theory and in practice.

We first provided a thorough complexity analysis to decide whether two probabilistic automata are weakly bisimilar. To this aim, we first derived equivalent but smaller in size LP problems compared to that of [HT12] by exploiting the underlying network structure of the original LP. For these LP problems we performed a complexity analysis to compute the exponent of the polynomial and subsequently the theoretical upper bound on the worst case complexity of the problem.

We then presented an implementation of the decision algorithm which can use either linear programming solvers or an SMT solver and can be used to minimize probabilistic automata under weak probabilistic bisimulation. Such a minimization has clear implications for reducing the state space explosion problem when model checking such automata. To further mitigate this problem we investigated how to use this approach in a compositional manner when systems are expressed as the parallel composition of a number of sub-automata. The implementation is tested on a number of case studies both to analyze different optimizations and the advantages of using a compositional approach.

Modelling formalisms like Markov decision processes and probabilistic automata are used to represent systems featuring nondeterminism and probabilistic behaviour. However, assigning fixed probability distributions to transitions is not realistic in modelling real life applications. In fact, measurement errors, statistical estimates, or mathematical approximations all lead to intervals instead of fixed probabilities.

We considered in Chapters 6 and 7, interval *MDP*s (also called Bounded-parameter *MDP*s) to address this need by bounding the probabilities of each successor state by an interval instead of a fixed number. In such modelling formalisms, the transition probabilities are not fully specified and this uncertainty again needs to be resolved nondeterministically. In order to describe the properties of such systems, we focused on probabilistic CTL (PCTL), a probabilistic logic which extends CTL by including a probabilistic operator. The semantics of the PCTL logic varies for different applications. We elucidated different PCTL semantics in the course of these chapters.

In Chapter 6, we started with cooperative semantics in which both the choice of transitions (resolved by a scheduler) and their probability distributions (resolved by a nature) is adversarial. This semantics is the case in the verification settings where our goal is to verify whether the *IMDP* model satisfies a given PCTL property under all resolutions of the two sources of nondeterminism. In such cases, we assume that the scheduler and nature are playing together against the model, i.e., their goal is to make the *IMDP* fail the PCTL property. Verification of PCTL properties of *MDP*s with convex uncertainties has been investigated recently by Puggelli et al [PLSVS13]. However, model checking algorithms typically suffer from state space explosion. In this chapter, we address probabilistic bisimulation to reduce the size of such an *IMDP* while preserving PCTL properties it satisfies. We defined the first probabilistic bisimulation for PCTL model checking of interval *MDP*s (that are also the first bisimulations for *MDP*s with uncertain transitions in general). We showed the worst case time complexity of deciding probabilistic bisimulation for model checking *IMDP*s is **coNP**-complete and then provided an algorithm based on comparing polytopes of probability distributions associated with each transition. The algorithm is fixed parameter tractable with respect to the maximal dimension of the polytopes. With the aim of designing an efficient approximation algorithm, we exploited the robust optimization and established a novel modelling of the probabilistic bisimulation problem for *IMDP*s as an instance of an uncertain LP problem. Afterwards, we showed that by using affine decision rules, the probabilistic bisimulation problem for *IMDP*s can be approximately decided in

polynomial time. We finally applied a prototypical implementation of our algorithms on a number of case studies to show the practical effectiveness of our developed approaches.

In Chapter 7, we extended previous results by considering the other way of resolving two sources of nondeterminism, namely the competitive semantics. There are indeed applications where it is natural to interpret the two sources of nondeterminism in a competitive way. As discussed earlier, in control synthesis for systems with uncertain probabilities, the transitions correspond to various control actions. We search for a choice of transitions that is optimal against an adversarial choice of probability distributions satisfying the interval bounds. Furthermore, in parameter synthesis for parallel systems, the transition probabilities are underspecified to allow freedom in implementation of such models. We search for a choice of probability distributions that is optimal for adversarial choice of transitions. In both applications controllers and natures are playing a game against one another.

In the context of competitive semantics, we defined two novel alternating probabilistic bisimulations for different interpretations of the nondeterminisms, i.e. controller synthesis and parameter synthesis, to reduce the size of *IMDP* models while preserving PCTL properties they satisfy. We first showed that the two alternating probabilistic bisimulations coincide and then designed a polynomial time decision algorithm to compute it. We then proved the effectiveness of our algorithm by applying it on various case studies and argued that, if uncertainty stems from a small number of different phenomena such as node failure or loss of a message, the same shape of polytopes will repeat many times over the state space. We demonstrated that the redundancy in this case may result in a massive state space minimization.

For the sake of understanding better the ways how large models with interval uncertainties can be composed, we additionally discussed in Chapters 6 and 7 the key ingredients to build up the operations of parallel composition for composing *IMDP* components at run-time. More precisely, we investigate how the parallel composition operator for *IMDP*s can be defined so as to arrive at a congruence closure. As a result, we proved that all defined probabilistic bisimulation for *IMDP*s are congruence with respect to two facets of parallelism, namely synchronous product and interleaving.

In Chapter 8, we further considered the problem of multi-objective robust strategy synthesis for *IMDP*s, where the aim is to find a robust strategy that guarantees the satisfaction of multiple properties at the same time in face of the transition probability uncertainty. We first proved that this problem is **PSPACE**-hard. Then, we presented a value iteration-based decision algorithm to approximate the Pareto set of achievable points. We finally demonstrated the practical effectiveness of our proposed approaches by applying them on several case studies.

Our minimization theory for *IMDP* models reveals the hardness of defining (cooperative) bisimulation relations that can be computed efficiently. To account for the latter, we introduced in Chapter 9 the *UwMDP*s as a novel stochastic model to capture quantities like preferences or priorities in a nondeterministic scenario with uncertainties. The model is very close to the *IMDP*s but more convenient to model with when uncertainties like weights, preference, priority, etc. are involved. Apart from the latter, we also showed that this modelling formalism admits an efficient bisimulation minimization algorithm.

We also considered the problem of computing maximal/minimal reward-bounded reachability probabilities on *UwMDP*s, for which we presented an efficient algorithm run-

ning in pseudo polynomial time. We finally showed promising results on a variety of case studies, obtained by a prototypical implementation of all algorithms.

## 10.2 Future works

The work we have presented in this dissertation gives path to several directions for future works. We briefly summarize them as follows.

- From modelling formalism aspect, it would be interesting to address richer formalisms for uncertainties such as polynomial constraints or even parameters appearing in multiple states/actions [AD16, DLP16] as well as other convex models of uncertainties such as likelihood or ellipsoidal uncertainties to capture a less conservative analysis.
- From logical characterization viewpoint, in this dissertation we have considered PCTL to express a wide variety of quantitative properties of systems. Although this logic is very useful in practice, it suffers from some expressiveness limitations for instance, arbitrary nested path formulas. It would therefore be interesting to extend the proposed approaches in this dissertation to more expressive logics such as $\omega$-PCTL [CSH08] or PCTL* [BDA95].
- In Chapter 5, we discussed efficiency of deciding probabilistic automata weak bisimulation from two different perspectives, namely theoretical and practical efficiency. From theoretical side, the network simplex algorithm specialized for the minimum cost flow problem with additional side constraints can be seen itself as the foremost next step. In fact, designing a new data structure to be able to deal with a large number of additional side constraints is not only a very significant contribution in the combinatorial optimization setting but also it improves the practical efficiency of the decision problem under our consideration. From practical side, it is interesting to improve the efficiency of proposed heuristics as well as to optimize the code in order to speed up the response time. Moreover, it would be important to investigate heuristics that allow us to optimize the sequence of the parallel compositions in order to take advantage from the compositional minimization approach, as done in [CL11, CH10]. For both facets of efficiency, it would be interesting to see whether the proposed results in this chapter can be exploited for efficiently deciding weak bisimilarity over richer modelling formalisms such as Markov automata [EHZ10b] as well as for the weak bisimilarity over probabilistic timed automata [KNSS02].
- In Chapters 6 and 7, we showed that our novel probabilistic bisimulations for *IMDP*s are congruence with respect to two facets of parallelism, namely synchronous product and interleaving. That would be an important research direction to explore the possibility of developing a probabilistic bisimulation minimization approach which not only preserves the PCTL properties but also is a congruence with respect to the asynchronous parallelism with synchronization. In the context of compositional reasoning for *IMDP*s, an interesting research direction would be to explore the possibility of mixing synchronous product and interleaving parallel operators as well as to investigate partially synchronous product operator.
- Even though we focused in Chapter 8 on robust controller synthesis of *IMDP*s with multi-objective reachability and reward properties, the proposed robust synthesis al-

gorithm can also handle *MDP*s with convex uncertain sets and any $\omega$-regular properties such as LTL. This, however, is not so clear for the proposed probabilistic bisimulation algorithm. The core part of this algorithm relies on verifying strictly minimal polytopes in polynomial time, which depends on the special structure of the uncertainty polytopes. As a future work, it would be worthwhile to explore the possibility of preserving this computational efficiency for *MDP*s with richer formalisms. Furthermore, the upper bound of the time complexity of the multi-objective robust strategy synthesis problem for *IMDP*s is left open in this thesis which needs further exploration.

- Finally, in Chapter 9 we introduced a new model of *UwMDP*s to express quantities in nondeterministic scenarios with uncertainty. The interval model of uncertainty in *UwMDP*s can be extended with other convex uncertainty models such as ellipsoidal model to capture a less conservative analysis. It is also interesting to discuss optimal control as well as multi-objective model checking of *UwMDP*s and support these theoretical developments by more realistic case studies where the priorities are introduced more naturally. Furthermore, compositional reasoning for *UwMDP*s deserves a more systematic treatment. Understanding better the ways how large *UwMDP*s with interval uncertainties can be composed, may bring further ideas for efficient analysis of these models.

In conclusion, we believe the research area of quantitative verification and synthesis of probabilistic systems with discrete or continuous parameters has attracted a lot of attention in recent years. In order to achieve desired goals such as synthesizing suitable parameters to ensure favorable behavior of a system further research are needed to develop efficient algorithms, tools and applications in different areas of computer science, bioinformatics and control engineering.

# Bibliography


[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[AD16] Étienne André and Benoît Delahaye. Consistency in parametric interval probabilistic timed automata. In *23rd International Symposium on Temporal Representation and Reasoning, TIME 2016, Kongens Lyngby, Denmark, October 17-19, 2016*, pages 110–119, 2016.

[ADD00] Robert B. Ash and Catherine Doléans-Dade. *Probability and Measure Theory*. Academic Press, 2000.

[AEG04] Alessandro Abate and Laurent El Ghaoui. Robust model predictive control through adjustable variables: an application to path planning. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 3, pages 2485–2490. IEEE, 2004.

[AH90] James Aspnes and Maurice Herlihy. Fast randomized consensus using shared memory. *J. Algorithms*, 11(3):441–461, 1990.

[AHK98] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Kupferman, Ornaand Vardi. Alternating refinement relations. In *CONCUR*, volume 1466 of *LNCS*, pages 163–178. Springer, 1998.

[AHK03] Suzana Andova, Holger Hermanns, and Joost-Pieter Katoen. Discrete-time rewards model-checked. In *FORMATS*, volume 2791 of *LNCS*, pages 88–104, 2003.

[AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: A meta-algorithm and applications. *Theory of Computing*, 8:121–164, 2012.

[AMO93] Ravindra K. Ahuja, Thomas J. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[Ans99] Kurt M. Anstreicher. Linear programming in $\mathcal{O}(\frac{n^3}{\ln n}L)$ operations. *SIAM J. on Optimization*, 9(4):803–812, 1999.

[ASSB00] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert K. Brayton. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.

[BBC11] Dimitris Bertsimas, David B. Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

[BCS10] Hichem Boudali, Pepijn Crouzen, and Mariëlle Stoelinga. A rigorous, compositional, and extensible framework for dynamic fault tree analysis. *IEEE TDSC*, 7(2):128–143, 2010.

[BDA95] Andrea Bianco and Luca De Alfaro. Model checking of probabilistic and nondeterministic systems. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 499–513. Springer, 1995.

[BDH+12] Marius Bozga, Alexandre David, Arnd Hartmanns, Holger Hermanns, Kim G. Larsen, Axel Legay, and Jan Tretmans. State-of-the-art tools and techniques for quantitative modeling and analysis of embedded systems. In *DATE*, pages 370–375. IEEE, March 2012.

[BDH+17] Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. JANI: Quantitative model and tool interaction. In *TACAS*, LNCS, pages 151–168, 2017.

[BDL+17] Anicet Bart, Benoît Delahaye, Didier Lime, Eric Monfroy, and Charlotte Truchet. Reachability in parametric interval Markov chains using constraints. In *QEST*, volume 10503 of *Lecture Notes in Computer Science*, pages 173–189. Springer, 2017.

[Bel57] Richard Bellman. A Markovian decision process. *Indiana University Mathematics Journal*, 6:679–684, 1957.

[Bel01] Peter A. Beling. Exact algorithms for linear programming over algebraic extensions. *Algorithmica*, 31(4):459–478, 2001.

[Ber95] Dimitri P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[Ber05] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

[BF12] Ufuk Bahçeci and Orhan Feyzioğlu. A network simplex based algorithm for the minimum cost proportional flow problem with disconnected subnetworks. *Optimization Letters*, 6:1173–1184, 2012.

[BG96] Marco Bernardo and Roberto Gorrieri. Extended Markovian process algebra. In *CONCUR*, volume 1119 of *LNCS*, pages 315–330. Springer, 1996.

[BGL+04] Christel Baier, Marcus Größer, Martin Leucker, Benedikt Bollig, and Frank Ciesinski. Controller synthesis for probabilistic systems. In *Exploring New Frontiers of Theoretical Informatics*, pages 493–506. Springer, 2004.

[BHH+09] Eckard Böde, Marc Herbstritt, Holger Hermanns, Sven Johr, Thomas Peikenkamp, Reza Pulungan, Jan Rakow, Ralf Wimmer, and Bernd Becker. Compositional dependability evaluation for STATEMATE. *ITSE*, 35(2):274–292, 2009.

[BHHK03] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

[BHKH05] Christel Baier, Holger Hermanns, Joost-Pieter Katoen, and Boudewijn R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *TCS*, 345(1):2–26, 2005.

[Bil79] Patrick Billingsley. *Probability and Measure*. John Wiley and Sons, New York, Toronto, London, 1979.

[BJS11] Mokhtar S. Bazaraa, John J. Jarvis, and Hanif D. Sherali. *Linear programming and network flows*. John Wiley & Sons, 2011.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[BKW14] Nicolas Basset, Marta Kwiatkowska, and Clemens Wiltsche. Compositional controller synthesis for stochastic games. In *CONCUR*, pages 173–187. Springer, 2014.

[BLW13] Michael Benedikt, Rastislav Lenhardt, and James Worrell. LTL model checking of interval Markov chains. In *TACAS*, volume 7795 of *LNCS*, pages 32–46. Springer, 2013.

[BMN00] Pierfrancesco Bellini, Riccardo Mattolini, and Paolo Nesi. Temporal logics for real-time system specification. *ACM Computing Surveys (CSUR)*, 32(1):12–42, 2000.

[Bra13] Willem K Brauers. *Optimization methods for a stakeholder society: a revolution in economic thinking by multi-objective optimization*, volume 73. Springer Science & Business Media, 2013.

[BSS89] Lenore Blum, Mike Shub, and Steve Smale. On a theory of computation and complexity over the real numbers; *NP*-completeness, recursive functions and universal machines. *Bullettin of the American Mathematical Society*, 21(1):1–46, 1989.

[BST10] Clark Barrett, Aaron Stump, and Cesare Tinelli. The SMT-LIB standard: Version 2.0. In *SMT*, 2010.

[BT97] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[BTEGN09] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[BTGGN04] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[BTN99] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of uncertain linear programs. *Operations research letters*, 25:1–13, 1999.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[Cal02] Herminia I. Calvete. Network simplex algorithm for the general equal flow problem. *European J. Operational Research*, 150(3):585–600, 2002.

[CE81] Edmund M Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logic of Programs*, pages 52–71. Springer, 1981.

[CFK+13] Taolue Chen, Vojtěch Forejt, Marta Kwiatkowska, Aistis Simaitis, and Clemens Wiltsche. On stochastic games with multiple objectives. In *International Symposium on Mathematical Foundations of Computer Science*, pages 266–277. Springer, 2013.

[CGM+96] Ghassan Chehaibar, Hubert Garavel, Laurent Mounier, Nadia Tawbi, and Ferruccio Zulian. Specification and verification of the PowerScale® bus arbitration protocol: An industrial experiment with LOTOS. In *FORTE*, pages 435–450, 1996.

[CH10] Pepijn Crouzen and Holger Hermanns. Aggregation ordering for massively compositional models. In *ACSD*, pages 171–180, 2010.

[Chi06] John W Chinneck. Practical optimization: a gentle introduction. *Systems and Computer Engineering, Carleton University, Ottawa.*, 2006. http://www.sce.carleton.ca/faculty/chinneck/po.html/.

[CHK13] Taolue Chen, Tingting Han, and Marta Z. Kwiatkowska. On the complexity of model checking interval-valued discrete time Markov chains. *Inf. Process. Lett.*, 113(7):210–216, 2013.

[CHLS09] Nicolas Coste, Holger Hermanns, Etienne Lantreibecq, and Wendelin Serwe. Towards performance prediction of compositional models in industrial GALS designs. In *CAV*, volume 5643 of *LNCS*, pages 204–218, 2009.

[CK70] Donald R. Chand and Sham S. Kapur. An algorithm for convex polytopes. *J. ACM*, 17(1):78–86, January 1970.

[CKMS11] Paul Christiano, Jonathan A. Kelner, Aleksander Mądry, and Daniel Spielman. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *STOC*, pages 273–282, 2011.

[CL11] Pepijn Crouzen and Frédéric Lang. Smart reduction. In *FASE*, volume 6603 of *LNCS*, pages 111–126, 2011.

[CMH06] Krishnendu Chatterjee, Rupak Majumdar, and Thomas A. Henzinger. Markov decision processes with multiple objectives. In *STACS*, volume 3884 of *LNCS*, pages 325–336, 2006.

[cpl] IBM ILOG CPLEX Optimizer. http://www.ibm.com/software/commerce/optimization/cplex-optimizer/.

[CRI07] Andrew S. Cantino, David L. Roberts, and Charles L. Isbell. Autonomous nondeterministic tour guides: improving quality of experience with TTD-MDPs. In *AAMAS*, page 22. IFAAMAS, 2007.

[CS02] Stefano Cattani and Roberto Segala. Decision algorithms for probabilistic bisimulation. In *CONCUR*, volume 2421 of *LNCS*, pages 371–385, 2002.

[CSH08] Krishnendu Chatterjee, Koushik Sen, and Thomas A. Henzinger. Model-checking $\omega$-regular properties of interval Markov chains. In *FoSSaCS*, pages 302–317, 2008.

[CSKN05] Stefano Cattani, Roberto Segala, Marta Kwiatkowska, and Gethin Norman. Stochastic transition systems for continuous state spaces and non-determinism. In *FoSSaCS*, volume 3441 of *LNCS*, pages 125–139, 2005.

[CSV07] Ling Cheung, Mariëlle Stoelinga, and Frits W. Vaandrager. A testing scenario for probabilistic processes. *JACM*, 54(6), 2007.

[CY95] Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *Journal of the ACM (JACM)*, 42(4):857–907, 1995.

[Den05] Yuxin Deng. *Axiomatisations and Types for Probabilistic and Mobile Processes*. PhD thesis, École des Mines de Paris, 2005.

[DH12] Yuxin Deng and Matthew Hennessy. On the semantics of Markov automata. *I&C*, 222:139–168, 2012.

[DJJ+15] Christian Dehnert, Sebastian Junges, Nils Jansen, Florian Corzilius, Matthias Volk, Harold Bruintjes, Joost-Pieter Katoen, and Erika Abraham. Prophesy: A probabilistic parameter synthesis tool. In *International Conference on Computer Aided Verification*, pages 214–231. Springer, 2015.

[DJKV17] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A STORM is coming: A modern probabilistic model checker. In *CAV*, pages 592–600, 2017.

[DKL+11a] Benoît Delahaye, Joost-Pieter Katoen, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, Falak Sher, and Andrzej Wasowski. Abstract probabilistic automata. In *VMCAI*, pages 324–339, 2011.

[DKL+11b] Benoît Delahaye, Joost-Pieter Katoen, Kim Guldstrand Larsen, Axel Legay, Mikkel Larsen Pedersen, Falak Sher, and Andrzej Wasowski. New results on abstract probabilistic automata. In *ACSD*, pages 118–127. IEEE, 2011.

[DLL+11] Benoît Delahaye, Kim Guldstrand Larsen, Axel Legay, Mikkel Larsen Pedersen, and Andrzej Wasowski. Decision problems for interval Markov chains. In *LATA*, volume 6638 of *LNCS*, pages 274–285. Springer, 2011.

[DLP16] Benoît Delahaye, Didier Lime, and Laure Petrucci. Parameter synthesis for parametric interval Markov chains. In *Verification, Model Checking, and Abstract Interpretation - 17th International Conference, VMCAI 2016, St. Petersburg, FL, USA, January 17-19, 2016. Proceedings*, pages 372–390, 2016.

[DM61] George B. Dantzig and Albert Madansky. On the solution of two-stage linear programs under uncertainty. In *Proc. Fourth Berkeley Symp. on Math. Statist. and Prob., Vol. 1*, pages 165–176, 1961.

[dMB08] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340, 2008.

[DSH16] Kalyanmoy Deb, Karthik Sindhya, and Jussi Hakanen. Multi-objective optimization. In *Decision Sciences: Theory and Practice*, pages 145–184. CRC Press, 2016.

[EH86] E. Allen Emerson and Joseph Y Halpern. “sometimes" and “not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM (JACM)*, 33(1):151–178, 1986.

[EHKZ13] Christian Eisentraut, Holger Hermanns, Joost-Pieter Katoen, and Lijun Zhang. A semantics for every GSPN. In *PETRI NETS*, volume 7927 of *Lecture Notes in Computer Science*, pages 90–109, 2013.

[Ehr06] Matthias Ehrgott. *Multicriteria optimization*. Springer Science & Business Media, 2006.

[EHS+13] Christian Eisentraut, Holger Hermanns, Johann Schuster, Andrea Turrini, and Lijun Zhang. The quest for minimal quotients for probabilistic automata. In *TACAS*, volume 7795 of *LNCS*, pages 16–31, 2013.

[EHZ10a] Christian Eisentraut, Holger Hermanns, and Lijun Zhang. Concurrency and composition in a stochastic world. In *CONCUR*, volume 6269 of *LNCS*, pages 21–39, 2010.

[EHZ10b] Christian Eisentraut, Holger Hermanns, and Lijun Zhang. On probabilistic automata in continuous time. In *LICS*, pages 342–351, 2010.

[Eis17] Christian Eisentraut. *Principles of Markov Automata*. PhD thesis, Saarland University, 2017.

[EKN+12] Marie-Aude Esteve, Joost-Pieter Katoen, Viet Yen Nguyen, Bart Postma, and Yuri Yushtein. Formal correctness, safety, dependability and performance analysis of a satellite. In *ICSE*, pages 1022–1031, 2012.

[EKVY07] Kousha Etessami, Marta Kwiatkowska, Moshe Y Vardi, and Mihalis Yannakakis. Multi-objective model checking of Markov decision processes. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 50–65. Springer, 2007.

[Eme90] E. Allen Emerson. Temporal and modal logic. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, 995(1072):5, 1990.

[ERSW14] Pavlos Eirinakis, Salvatore Ruggieri, K Subramani, and Piotr Wojciechowski. On quantified linear implications. *AMAI*, 71(4):301–325, 2014.

[FFHHT16] Luis María Ferrer Fioriti, Vahid Hashemi, Holger Hermanns, and Andrea Turrini. Deciding probabilistic automata weak bisimulation: Theory and practice. *FAoC*, 28(1):109–143, 2016.

[FKN+11] Vojtěch Forejt, Marta Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Quantitative multi-objective verification for probabilistic systems. In *TACAS*, volume 6605 of *LNCS*, pages 112–127, 2011.

[FKNP11] Vojtěch Forejt, Marta Kwiatkowska, Gethin Norman, and David Parker. Automated verification techniques for probabilistic systems. In *Formal Methods for Eternal Networked Software Systems (SFM'11)*, volume 6659, pages 53–113, 2011.

[FKP12] Vojtěch Forejt, Marta Kwiatkowska, and David Parker. Pareto curves for probabilistic model checking. In *International Symposium on Automated Technology for Verification and Analysis*, pages 317–332. Springer, 2012.

[FL81] Aviezri S Fraenkel and David Lichtenstein. Computing a perfect strategy for n× n chess requires time exponential in n. In *International Colloquium on Automata, Languages, and Programming*, pages 278–293. Springer, 1981.

[FLW06] Harald Fecher, Martin Leucker, and Verena Wolf. Don't know in probabilistic systems. In *SPIN*, volume 3925 of *LNCS*, pages 71–88. Springer, 2006.

[Fre05] Goran Frehse. Phaver: Algorithmic verification of hybrid systems past hytech. In *International conference on hybrid systems: computation and control*, pages 258–273. Springer, 2005.

[Fu14] Hongfei Fu. Maximal cost-bounded reachability probability on continuous-time Markov decision processes. In *FoSSaCS*, volume 8412, pages 73–87. LNCS, 2014.

[Fur80] Nagata Furukawa. Characterization of optimal policies in vector-valued Markovian decision processes. *Mathematics of operations research*, 5(2):271–279, 1980.

[Gho90] Mrinal K Ghosh. Markov decision processes with multiple costs. *Operations Research Letters*, 9(4):257–260, 1990.

[GHT14] Daniel Gebler, Vahid Hashemi, and Andrea Turrini. Computing behavioral relations for probabilistic concurrent systems. In *Stochastic Model Checking. Rigorous Dependability Analysis Using Model Checking Techniques for Stochastic Systems*, volume 8453 of *LNCS*, pages 117–155. Springer Berlin Heidelberg, 2014.

[GJ90] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[GLD00] Robert Givan, Sonia M. Leach, and Thomas L. Dean. Bounded-parameter Markov decision processes. *Artif. Intell.*, 122(1-2):71–109, 2000.

[GLP] GLPK (GNU Linear Programming Kit). http://www.gnu.org/software/glpk/.

[GLS81] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[Goe14] Marc Goerigk. ROPI–a robust optimization programming interface for C++. *Optimization Methods and Software*, 29(6):1261–1280, 2014.

[GOR00] Jacob E. Goodman, Joseph O'Rourke, and Kenneth H. Rosen. *Handbook of discrete and computational geometry*. cRc Press LLc, 2000.

[GSL96] Susanne Graf, Bernhard Steffen, and Gerald Lüttgen. Compositional minimisation of finite state systems using interface specifications. *Formal Aspects of Computing*, 8(5):607–616, 1996.

[GTT89] Andrew V Goldberg, Éva Tardos, and ROBERTE Tarjan. Network flow algorithm. Technical report, Cornell University Operations Research and Industrial Engineering, 1989.

[GUR] Gurobi 4.0.2. http://www.gurobi.com/.

[Gus02] Elana Guslitser. *Uncertainty-immunized solutions in linear programmin*. PhD thesis, Technion-Israel Institute of Technology, 2002.

[Hag14] Willem Hagemann. Reachability analysis of hybrid systems using symbolic orthogonal projections. In *International Conference on Computer Aided Verification*, pages 407–423. Springer, 2014.

[Hal50] Paul Richard Halmos. *Measure Theory*. Springer, 1950.

[Han91] Hans A. Hansson. *Time and Probability in Formal Design of Distributed Systems*. PhD thesis, Uppsala University, 1991.

[Has16] Vahid Hashemi. Towards a combinatorial approach for undiscounted MDPs: student research abstract. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016*, pages 1708–1709, 2016.

[Has17] Vahid Hashemi. Reformulation of the linear program for completely ergodic MDPs with average cost criteria. *Optimization Letters*, 11(7):1477–1487, 2017.

[Hen83] Mordechai I Henig. Vector-valued dynamic programming. *SIAM Journal on Control and Optimization*, 21(3):490–499, 1983.

[Her02] Holger Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, volume 2428 of *LNCS*. Springer, 2002.

[HHH+17a] Ernst Moritz Hahn, Vahid Hashemi, Holger Hermanns, Morteza Lahijanian, and Andrea Turrini. Multi-objective robust strategy synthesis for interval Markov decision processes. *CoRR*, abs/1706.06875, 2017.

[HHH+17b] Ernst Moritz Hahn, Vahid Hashemi, Holger Hermanns, Morteza Lahijanian, and Andrea Turrini. Multi-objective robust strategy synthesis for interval MDPs. In *QEST*, volume 10503 of *LNCS*, pages 207–223, 2017.

[HHHT16] Ernst Moritz Hahn, Vahid Hashemi, Holger Hermanns, and Andrea Turrini. Exploiting robust optimization for interval probabilistic bisimulation. In *QEST*, pages 55–71, 2016.

[HHK14] Vahid Hashemi, Hassan Hatefi, and Jan Krčál. Probabilistic bisimulations for PCTL model checking of interval MDPs (extended version). In *SynCoP*, volume 145 of *EPTCS*, pages 19–33, 2014.

[HHS16a] Vahid Hashemi, Holger Hermanns, and Lei Song. Reward-bounded reachability probability for uncertain weighted MDPs. In *VMCAI*, pages 351–371, 2016.

[HHS+16b] Vahid Hashemi, Holger Hermanns, Lei Song, K. Subramani, Andrea Turrini, and Piotr Wojciechowski. Compositional bisimulation minimization for interval Markov decision processes. In *LATA'16*, volume 9618 of *LNCS*, pages 114–126, 2016.

[HHT13] Vahid Hashemi, Holger Hermanns, and Andrea Turrini. On the efficiency of deciding probabilistic automata weak bisimulation. *ECEASST*, 66, 2013.

[HHT16] Vahid Hashemi, Holger Hermanns, and Andrea Turrini. Compositional reasoning for interval Markov decision processes. Available at http://arxiv.org/abs/1607.08484, 2016.

[HHWZ10] Ernst Moritz Hahn, Holger Hermanns, Björn Wachter, and Lijun Zhang. Param: A model checker for parametric Markov models. In *International Conference on Computer Aided Verification*, pages 660–664. Springer, 2010.

[HHZ11a] Ernst Moritz Hahn, Tingting Han, and Lijun Zhang. Synthesis for PCTL in parametric Markov decision processes. In *NASA Formal Methods*, volume 6617 of *LNCS*, pages 146–161, 2011.

[HHZ11b] Ernst Moritz Hahn, Holger Hermanns, and Lijun Zhang. Probabilistic reachability for parametric Markov chains. *STTT*, 13(1):3–19, 2011.

[HJ90] Hans Hansson and Bengt Jonsson. A calculus for communicating systems with time and probabilities. In *RTSS*, pages 278–287, 1990.

[HJ94] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994.

[HK95] Richard V. Helgason and Jeffery L. Kennington. Primal simplex algorithms for minimum cost network flows. In *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, chapter 2, pages 85–113. Elsevier, 1995.

[HK00] Holger Hermanns and Joost-Pieter Katoen. Automated compositional Markov chain generation for a plain-old telephone system. *Science of Computer Programming*, 36(1):97–127, 2000.

[HKNP06] Andrew Hinton, Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM: A tool for automatic verification of probabilistic systems. In *TACAS*, volume 3920 of *LNCS*, pages 441–444, 2006.

[HLS+14] Ernst Moritz Hahn, Yi Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. IscasMC: A web-based probabilistic model checker. In *Nineteenth international symposium of the Formal Methods Europe association (FM)*, volume 8442 of *Lecture Notes in Computer Science*, pages 312–317. Springer, 2014.

[HN94] Dorit S. Hochbaum and Joseph Seffi Naor. Simple and fast algorithms for linear and integer programs with two variables per inequality. *SIAM J. on Computing*, 23(6):1179–1192, 1994.

[How60] Ronald A. Howard. *Dynamic Programming and Markov Processes*. John Wiley and Sons, Inc., 1960.

[How71] Ronald A. Howard. *Dynamic Probabilistic Systems*. John Wiley & Sons, 1971.

[How07] Ronald A. Howard. *Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes*. Dover Publications, 2007.

[HS65] Juris Hartmanis and Richard E Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[HT12] Holger Hermanns and Andrea Turrini. Deciding probabilistic automata weak bisimulation in polynomial time. In *FSTTCS*, pages 435–447, 2012.

[HTH+17] Vahid Hashemi, Andrea Turrini, Ernst Moritz Hahn, Holger Hermanns, and Khaled Elbassioni. Polynomial-time alternating probabilistic bisimulation for interval MDPs. In *SETTA*, volume 10606 of *LNCS*, pages 25–41, 2017.

[Iye05] Garud N. Iyengar. Robust dynamic programming. *Math. Oper. Res.*, 30(2):257–280, 2005.

[JL91] Bengt Jonsson and Kim Guldstrand Larsen. Specification and refinement of probabilistic processes. In *LICS*, pages 266–277. IEEE Computer Society, 1991.

[JT80] William B. Jones and Wolfgang Joseph Thron. *Continued Fractions: Analytic Theory and Applications*. Encyclopedia of Mathematics and its Applications. Addison-Wesley, 1980.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[Kat16] Joost-Pieter Katoen. The probabilistic model checking landscape. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 31–45. ACM, 2016.

[Kha79] Leonid Genrikhovich Khachyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20(1):191–194, 1979.

[KKN09] Joost-Pieter Katoen, Daniel Klink, and Martin R. Neuhäußer. Compositional abstraction for stochastic systems. In *FORMATS*, volume 5813 of *LNCS*, pages 195–211. Springer, 2009.

[KKZJ07] Joost-Pieter Katoen, Tim Kemna, Ivan S. Zapreev, and David N. Jansen. Bisimulation minimisation mostly speeds up probabilistic model checking. In *TACAS*, volume 4424 of *LNCS*, pages 76–92, 2007.

[KM72] Victor Klee and George J. Minty. How good is the simplex algorithm? In *Inequalities*, volume III, pages 159–175. Defense Technical Information Center, 1972.

[KM00] Jean-Pierre Krimm and Laurent Mounier. Compositional state space generation with partial order reductions for asynchronous communicating systems. In *TACAS*, volume 1785 of *LNCS*, pages 266–282, 2000.

[KNP07] Marta Kwiatkowska, Gethin Norman, and David Parker. Stochastic model checking. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pages 220–270. Springer, 2007.

[KNP11] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591, 2011.

[KNPQ13] Marta Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Compositional probabilistic verification through multi-objective model checking. *Information and Computation*, 232:38–65, 2013.

[KNS01] Marta Z. Kwiatkowska, Gethin Norman, and Roberto Segala. Automated verification of a randomized distributed consensus protocol using cadence SMV and PRISM. In *CAV*, volume 2102 of *Lecture Notes in Computer Science*, pages 194–206. Springer, 2001.

[KNSS02] Marta Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. *TCS*, 282:101–150, 2002.

[Kol12] Andrey Kolobov. Planning with Markov decision processes: An AI perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–210, 2012.

[KS76] John G. Kemeny and James Laurie Snell. *Finite Markov Chains*. Springer-Verlang, 1976.

[KS90] Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *I&C*, 86(1):43–68, 1990.

[KS05] Antonín Kučera and Oldřich Stražovskỳ. On the controller synthesis for finite-state Markov decision processes. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 541–552. Springer, 2005.

[KSK66] John G. Kemeny, James Laurie Snell, and Anthony W. Knapp. *Denumerable Markov Chains*. Van Nostrand Company, 1966.

[KU02] Igor O Kozine and Lev V Utkin. Interval-valued finite Markov chains. *Reliable computing*, 8(2):97–113, 2002.

[Kul96] Vidyadhar G Kulkarni. *Modeling and analysis of stochastic systems*. CRC Press, 1996.

[Lam83] Leslie Lamport. What good is temporal logic? In *IFIP congress*, volume 83, pages 657–668, 1983.

[Law76] Eugene L Lawler. *Combinatorial optimization: networks and matroids*. Courier Corporation, 1976.

[LK16] Morteza Lahijanian and Marta Kwiatkowska. Specification revision for Markov decision processes with optimal trade-off. In *Conf. on Decision and Control*, pages 7411–7418. IEEE, Dec. 2016.

[LLMK14a] Ryan Luna, Morteza Lahijanian, Mark Moll, and Lydia E. Kavraki. Asymptotically optimal stochastic motion planning with temporal goals. In *The Eleventh International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 335–352, Istanbul, Turkey, Aug. 2014.

[LLMK14b] Ryan Luna, Morteza Lahijanian, Mark Moll, and Lydia E. Kavraki. Fast stochastic motion planning with optimality guarantees using local policy reconfiguration. In *IEEE Conference on Robotics and Automation*, pages 3013–3019, Hong Kong, China, May 2014.

[LLMK14c] Ryan Luna, Morteza Lahijanian, Mark Moll, and Lydia E. Kavraki. Optimal and efficient stochastic motion planning in partially-known environments. In *The Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2549–2555, Quebec City, Canada, 2014.

[Lof04] Johan Lofberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pages 284–289. IEEE, 2004.

[Lof12] Johan Lofberg. Automatic robust convex programming. *Optimization methods and software*, 17(1):115–129, 2012.

[LpS] LpSolve mixed integer linear programming solver. `http://lpsolve.sourceforge.net`.

[LR81] Daniel J. Lehmann and Michael O. Rabin. On the advantages of free choice: A symmetric and fully distributed solution to the dining philosophers problem. In *POPL*, pages 133–138. ACM Press, 1981.

[LWAB10] Morteza Lahijanian, Joseph Wasniewski, Sean B Andersson, and Calin Belta. Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3227–3232. IEEE, 2010.

[LY84] David G Luenberger and Yinyu Ye. *Linear and nonlinear programming*, volume 2. Springer, 1984.

[MA04] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[Mao03] Wenbo Mao. *Modern cryptography: theory and practice*. Prentice Hall Professional Technical Reference, 2003.

[Mar15] Stephen Marsland. *Machine learning: an algorithmic perspective*. CRC press, 2015.

[MCB84] Marco Ajmone Marsan, Gianni Conte, and Gianfranco Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst.*, 2(2):93–122, 1984.

[Mil89] Robin Milner. *Communication and concurrency*. PHI Series in computer science. Prentice Hall, 1989.

[MOS] MOSEK. `http://www.mosek.com/`.

[Mou04] Abdel-Illah Mouaddib. Multi-objective decision-theoretic plan problem. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2814–2819, 2004.

[MSJ11] David R. Morrison, Jason J. Sauppe, and Sheldon H. Jacobson. A network simplex algorithm for the equal flow problem on a generalized network. *INFORMS J. on Computing*, 25(1):2–12, 2011.

[MSJ13] David R. Morrison, Jason J. Sauppe, and Sheldon H. Jacobson. An algorithm to solve the proportional network flow problem. *Optimization Letters*, 8(3):801–809, 2013.

[NG05] Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[OPW13] Wlodzimierz Ogryczak, Patrice Perny, and Paul Weng. A compromise programming approach to multiobjective Markov decision processes. *International Journal of Information Technology and Decision Making*, 12(5):1021–1054, 2013.

[Pap03] Christos H. Papadimitriou. Computational complexity. In *Encyclopedia of Computer Science*, pages 260–265. John Wiley and Sons Ltd., Chichester, UK, 2003.

[PIC] PICOS: A Python interface for conic optimization solvers. http://picos.zib.de/.

[PLS00] Anna Philippou, Insup Lee, and Oleg Sokolsky. Weak bisimulation for probabilistic systems. In *CONCUR*, volume 1877 of *LNCS*, pages 334–349, 2000.

[PLSVS13] Alberto Puggelli, Wenchao Li, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In *CAV*, pages 527–542, 2013.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57. IEEE, 1977.

[PRI] PRISM model checker. http://www.prismmodelchecker.org/.

[PS04] Augusto Parma and Roberto Segala. Axiomatization of trace semantics for stochastic nondeterministic processes. In *QEST*, pages 294–303, 2004.

[PSVS14] Alberto Puggelli, Alberto L Sangiovanni-Vincentelli, and Sanjit A Seshia. Robust strategy synthesis for probabilistic systems applied to risk-limiting renewable-energy pricing. In *Proceedings of the 14th International Conference on Embedded Software*, page 13. ACM, 2014.

[PT87] Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM J. on Computing*, 16(6):973–989, 1987.

[Pug14] Alberto Puggelli. *Formal Techniques for the Verification and Optimal Control of Probabilistic Systems in the Presence of Modeling Uncertainties*. PhD thesis, EECS Department, University of California, Berkeley, Aug 2014.

[pul] PuLP. https://pythonhosted.org/PuLP/.

[Pul89] P. Simin Pulat. A decomposition algorithm to determine the maximum flow in a generalized network. *Computers & Operations Research*, 16:161–172, 1989.

[Put05] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Number 594 in Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., 2005.

[PWGH13] Patrice Perny, Paul Weng, Judy Goldsmith, and Josiah P. Hanna. Approximation of Lorenz-optimal solutions in multiobjective Markov decision processes. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 92–94, 2013.

[QS82] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in cesar. In *International Symposium on programming*, pages 337–351. Springer, 1982.

[Rin09] Matthias Ringwald. *Reducing Uncertainty in Wireless Sensor Networks - Network Inspection and Collision-Free Medium Access*. PhD thesis, ETH Zurich, Zurich, Switzerland, March 2009.

[RRS15] Mickael Randour, Jean-François Raskin, and Ocan Sankur. Percentile queries in multi-dimensional Markov decision processes. In *Computer Aided Verification*, pages 123–139. Springer, 2015.

[SBHH17] Dimitri Scheftelowitsch, Peter Buchholz, Vahid Hashemi, and Holger Hermanns. Multi-objective approaches to Markov decision processes with uncertain transition parameters. In *ValueTools*, 2017. to appear.

[Sch98] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[Sch03] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

[SDV04] Ana Sokolova and Erik P De Vink. Probabilistic automata: system types, parallel composition and comparison. In *Validation of Stochastic Systems*, pages 1–43. Springer, 2004.

[Seg95] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, MIT, 1995.

[Seg06] Roberto Segala. Probability and nondeterminism in operational models of concurrency. In *CONCUR*, volume 4137 of *LNCS*, pages 64–78, 2006.

[Sha87] Ron Shamir. The efficiency of the simplex method: A survey. *Management Science*, 33(3):301–334, 1987.

[SL73] Jay K. Satia and Roy E. Lave. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21(3):728–740, 1973.

[SL94] Roberto Segala and Nancy A. Lynch. Probabilistic simulations for probabilistic processes. In *CONCUR*, volume 836 of *LNCS*, pages 481–496, 1994.

[SL95] Roberto Segala and Nancy A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic J. Computing*, 2(2):250–273, 1995.

[SM73] Larry J Stockmeyer and Albert R Meyer. Word problems requiring exponential time (preliminary report). In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 1–9. ACM, 1973.

[Ste94] William J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[Sub09] K. Subramani. On the complexities of selected satisfiability and equivalence queries over boolean formulas and inclusion queries over hulls. *JAMDS*, 2009, 2009.

[SVA06] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Model-checking Markov chains in the presence of uncertainties. In *TACAS*, volume 3920 of *LNCS*, pages 394–410. Springer, 2006.

[TH14] Andrea Turrini and Holger Hermanns. Cost preserving bisimulations for probabilistic automata. *Logical Methods in Computer Science*, 4(11):1–58, 2014.

[TH15] Andrea Turrini and Holger Hermanns. Polynomial time decision algorithms for probabilistic automata. *Information and Computation*, 244:134–171, 2015.

[Tiw08] Hans Raj Tiwary. On computing the shadows and slices of polytopes. *arXiv preprint arXiv:0804.4150*, 2008.

[Tre02] Luca Trevisan. Lecture notes on computational complexity. http://people.eecs.berkeley.edu/ luca/notes/complexitynotes02.pdf, 2002.

[Var85] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS*, pages 327–338, 1985.

[Vaz04] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2004.

[WE94] Chelsea C. White and Hany K. Eldeib. Markov decision processes with imprecise transition probabilities. *Operations Research*, 42(4):739–749, 1994.

[Whi82] DJ White. Multi-objective infinite-horizon discounted Markov decision processes. *Journal of mathematical analysis and applications*, 89(2):639–647, 1982.

[WJ06] Nicolás Wolovick and Sven Johr. A characterization of meaningful schedulers for continuous-time Markov decision processes. In *Formal Modeling and Analysis of Timed Systems*, volume 4202 of *LNCS*, pages 352–367, 2006.

[WK08] Di Wu and Xenofon D. Koutsoukos. Reachability analysis of uncertain systems using bounded-parameter Markov decision processes. *Artif. Intell.*, 172(8-9):945–954, 2008.

[WT98] K Wakuta and K Togawa. Solution procedures for multi-objective Markov decision processes. *Optimization*, 43(1):29–46, 1998.

[WTM12] Eric M. Wolff, Ufuk Topcu, and Richard M. Murray. Robust control of uncertain Markov decision processes with temporal logic specifications. In *CDC*, pages 3372–3379. IEEE, 2012.

[Ye11] Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011.

[Yi94] Wang Yi. Algebraic reasoning for real-time probabilistic processes with uncertain information. In *FTRTFT*, volume 863 of *LNCS*, pages 680–693. Springer, 1994.

[Zie95] Günter M Ziegler. *Lectures on polytopes*, volume 152. Springer Science & Business Media, 1995.