Diagnosis, Synthesis and Analysis of Probabilistic Models

Tingting Han

IPA Dissertation Series 2009-21.
CTIT Ph.D.-Thesis Series No. 09-149, ISSN 1381-3617.
ISBN: 978-90-365-2858-0.

The research reported in this dissertation has been carried out under the auspices of the Institute for Programming Research and Algorithmics (IPA) and within the context of the Center for Telematics and Information Technology (CTIT). The research funding was provided by the NWO Grant through the project: Verifying Quantitative Properties of Embedded Software (QUPES).

Translation of the Dutch abstract: Viet Yen Nguyen.
Translation of the German abstract: Martin R. Neuhäußer.
Typeset by LaTeX.
Cover design: Tingting Han. Picture from www.vladstudio.com.
Publisher: Wöhrmann Printing Service - www.wps.nl.

DIAGNOSIS, SYNTHESIS AND ANALYSIS OF PROBABILISTIC MODELS

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus
prof. dr. H. Brinksma,
volgens besluit van het College voor Promoties,
in het openbaar te verdedigen
op vrijdag 25 september 2009 om 13:15 uur

door

Tingting Han
geboren op 27 december 1980
te Hangzhou, Volksrepubliek China

Dit proefschrift is goedgekeurd door

de promotor, prof. dr. ir. Joost-Pieter Katoen.

DIAGNOSIS, SYNTHESIS AND ANALYSIS OF PROBABILISTIC MODELS

Der Fakultät für Mathematik, Informatik und Naturwissenschaften der Rheinisch-Westfälischen Technischen Hochschule Aachen vorgelegte Dissertation zur Erlangung des akademischen Grades einer Doktorin der Naturwissenschaften

vorgelegt von

Tingting Han, M. Eng
aus
Hangzhou, Volksrepublik China

Berichter: Prof. Dr. Ir. Joost-Pieter Katoen
Prof. Dr. Marta Z. Kwiatkowska

Tag der mündlichen Prüfung: 16. Oktober 2009

Acknowledgements

I still remember the summer back in 2005 when I took this PhD position. Four years were just like a snap of fingers, leaving me this booklet and many vivid pieces of snapshots deep and clear in my mind.

I was a “risk” four years ago when Joost-Pieter Katoen, my promotor and supervisor, decided to offer me this job. After all, to Joost-Pieter at that time, I was nothing more than a two-page CV and a short international call. I do appreciate this trust, which in these years keeps urging me to make progress and “make profits”:). His expertise, insights and far-reaching interests broadened my views and helped me find “shorter (if not the shortest:) paths” at each crossroad. I am grateful for his enduring guidance, his great support, understanding and flexibility. I am also thankful for his big effort and contribution going to China in 2008, visiting different universities and institutes, giving both tutorials and more advanced invited talks, attracting more Chinese students and researchers to the field of formal verification.

Many results presented in this thesis are a product of joint work. Apart from Joost-Pieter, I am grateful to Berteun Damman, Alexandru Mereacre and Taolue Chen, who shared ideas, had (usually long and fruitful) discussions and transferred their expertise to me. I am also thankful to Christel Baier, David N. Jansen and Jeremy Sproston for their useful remark and insightful discussion on my papers. The peer review and exchange of ideas inside the group, usually by/with Henrik Bohnenkamp, Daniel Klink and Martin Neuhäußer have provided me with their precious comments and enlightening thoughts. Besides, I would like to thank Prof. Marta Kwiatkowska for inviting me to visit her group in Birmingham. I also enjoyed the visit(s) from Husain Aljazzar, Miguel Andrés, Lars Grunske, Ralf Wimmer and Lijun Zhang for the interesting talks and discussions. Besides, the regular VOSS and QUASIMODO meetings made me feel like being in a big and happy family.

I would like also to thank my reading and defense committee in Twente: Prof. Pedro D’Argenio, Prof. Boudewijn Haverkort, Prof. Stefan Leue, Prof. Jaco van de Pol, Prof. Frits Vaandrager, Prof. Roel Wieringa and Prof. Leen van Wijngaarden as well as the examiners in Aachen: Prof. Gerhard Lakemeyer, Prof. Marta Kwiatkowska, Prof. Stefan Kowalewski and Prof. Wolfgang Thomas.

Although I already knew in the beginning that I would move with Joost-Pieter from Twente to Aachen, I had never anticipated what this “double identity” would mean to me. Actually I always feel proud when I can fill in two affiliations. I also feel lucky that I can get to know both top research groups. (Of course this acknowledgement will become twice as long as it would have been.:)

First, I really appreciate the liberty that Twente rendered me during my PhD study. The FMT group is so supportive, helpful and flexible that I do feel in debt to it. I would like to thank Ed Brinksma for introducing me to Joost-Pieter; and Jaco van de Pol for his understanding and support; also many thanks to Axel Belinfante, Rom Langerak, Arend Rensink, Theo Ruys, Mariëlle Stoelinga, Mark Timmer, Michael Weber and the former members Hichem Boudali and Ivan Zapreev. Special thanks go to Joke Lammerink for everything she has done for me. *Bedankt!*

On the other hand, I witnessed the growth and fast developments of the MOVES group, which offers me a relaxed and productive working atmosphere. As my roommate, Martin Neuhäußer is always there and ready to help in every respect. I owe you a lot! I am also very lucky to have Haidi Yue, Viet Yen Nguyen, Alexandru Mereacre and Henrik Bohnenkamp around who make my laughters loud and lasting. Martin and Viet Yen are specially thanked for translating the abstract. Also many thanks to Erika Ábráham, Xin Chen, Fabian Emmes, Carsten (Fuhs | Kern | Otto), Arnd Gehrmann, Jürgen Giesl, Jonathan Heinen, Nils Jansen, Daniel Klink, Ulrich Loup, Thomas Noll, Elke Ohlenforst and Stefan Rieger and the former members René Thiemann, Peter Schneider-Kamp and Stephan Swiderski. You made my time in the group more colorful! *Dank sehr!*

My life in Aachen would have not been as pleasant if there were no cheerful company and encouragements from friends in and outside Aachen. Just to name a few: Sebastian Bitzen, Andreas Brand, Yiwei Cao, Qingxia Gong, Olivier Guillard, Jianwei Han, Yingchun He, Xuan Li, Fang Liu, Leiqin Lu, Mikhail Pletyukhov, Kuangyu Shi, Leyi Shi, Hailin Wang, Wei Wang, Yan Wang, Yutian Wang, Shaochen Wei, Wei Wu, Ping Yu, Haidi Yue, Yuqi Zhang, Ziyun Zhang and many other friends.

I am forever indebted to my family, especially my parents in China. Without their unconditional love, encouragement and support, I won't reach this far. 感谢爸爸妈妈! Also many thanks go to my little grandpa and my cousins Bin, Jian and Xiaoyun, who share experience, give advices and send me many pictures of my lovely little nephew and nieces. 感谢所有支持关心我的家人和朋友!

Finally, to Taolue, well, due to the small space here, we can do this offline...:)

韓婷婷印

Tingting Han (韩婷婷) Amsterdam, May 2, 2009

Abstract

This dissertation considers three important aspects of model checking Markov models: *diagnosis* — generating counterexamples, *synthesis* — providing valid parameter values and *analysis* — verifying linear real-time properties. The three aspects are relatively independent while all contribute to developing new theory and algorithms in the research field of probabilistic model checking.

We start by introducing a formal definition of counterexamples in the setting of probabilistic model checking. We transform the problem of finding informative counterexamples to shortest path problems. A framework is explored and provided for generating such counterexamples. We then investigate a more compact representation of counterexamples by regular expressions. Heuristic based algorithms are applied to obtain short regular expression counterexamples. In the end of this part, we extend the definition and counterexample generation algorithms to various combinations of probabilistic models and logics.

We move on to the problem of synthesizing values for parametric continuous-time Markov chains (pCTMCs) wrt. time-bounded reachability specifications. The rates in the pCTMCs are expressed by polynomials over reals with parameters and the main question is to find all the parameter values (forming a synthesis region) with which the specification is satisfied. We first present a symbolic approach where the intersection points are computed by solving polynomial equations and then connected to approximate the synthesis region. An alternative non-symbolic approach based on interval arithmetic is investigated, where pCTMCs are instantiated. The error bound, time complexity as well as some experimental results have been provided, followed by a detailed comparison of the two approaches.

In the last part, we focus on verifying CTMCs against linear real-time properties specified by deterministic timed automata (DTAs). The model checking problem aims at computing the probability of the set of paths in CTMC $\mathcal{C}$ that can be accepted by DTA $\mathcal{A}$, denoted $Paths^{\mathcal{C}}(\mathcal{A})$. We consider DTAs with reachability (finite, DTA$^{\Diamond}$) and Muller (infinite, DTA$^{\omega}$) acceptance conditions, respectively. It is shown that $Paths^{\mathcal{C}}(\mathcal{A})$ is measurable and computing its probability for DTA$^{\Diamond}$ can be reduced to computing the reachability probability in a piecewise deterministic Markov process (PDP). The reachability probability is characterized as the least solution of a system of integral equations and is shown to be approximated by solving a system of PDEs. Furthermore, we show that the special case of single-clock DTA$^{\Diamond}$ can be simplified to solving a system of linear equations. We also deal with DTA$^{\omega}$ specifications, where the problem is proven to be reducible to the reachability problem as in the DTA$^{\Diamond}$ case.

Samenvatting

Dit proefschrift behandelt drie aspecten van het model checken van Markov modellen: *diagnose* — de generatie van tegenvoorbeelden, *synthese* — de berekening van valide waarden voor parameters en *analyse* — de verificatie van lineaire real-time eigenschappen. Hoewel deze drie aspecten onderling ongerelateerd lijken, dragen alle drie bij aan de theorie en ontwikkeling van algoritmen in het onderzoeksgebied van probabilistisch model checking.

We leiden in met een formele definitie van informatieve tegenvoorbeelden in de context van probabilistisch model checken. Vervolgens karakteriseren we dit probleem als een kortste pad probleem. Daaropvolgende presteren we een kader om tegenvoorbeelden te generen. Om de tegenvoorbeelden compact te representeren, laten we zien hoe deze uitgedrukt kunnen worden in reguliere expressies. We passen heuristische algoritmen toe om die expressies te verkrijgen. Ten slotte passen we ons kader van tegenvoorbeeldgeneratie toe voor verscheidene probabilistische modellen en logica's. Hiertoe breiden we onze definitie van informatieve tegenvoorbeelden enigszins uit.

In het tweede deel behandelen wij het synthese-probleem van parametrische Continuous-Time Markov Chains (pCTMC's) met betrekking tot tijdsbegrensde bereikbaarheidsspecificaties. Het doel is om de waarden van alle intensiteitsparameters (het gesynthetiseerde gebied) te bepalen die ervoor zorgen dat de gegeven CTMC aan de specificatie voldoet. De intensiteitsparameters worden dan beschouwd als polynomen over de reëele getallen. We presenteren eerst een methode die intersectiepunten bepaalt en vervolgens die verbindt om een benadering van het gesynthetiseerde gebied te verkrijgen. Een alternatieve methode met interval arithmetica wordt ook behandeld. Ten slotte worden de foutmarges, tijdscomplexiteiten en experimentele resultaten uiteengezet gevolgd door een gedetailleerde vergelijking tussen de twee methodes.

In het laatste deel richten we ons op de verificatie van CTMC's met lineaire real-time eigenschappen die gespecificeerd worden met deterministic timed automata (DTA's). In dit model checking probleem berekenen we de waarschijnlijkheid van de paden die door een CTMC $\mathcal{C}$ geaccepteerd wordt door DTA $\mathcal{A}$, beschreven als $\mathit{Paths}^{\mathcal{C}}(\mathcal{A})$. We behandelen DTA's met bereikbaarheid (eindig, DTA$^{\Diamond}$) en Muller (oneindig, DTA$^{\omega}$) acceptatie condities. We bewijzen dat $\mathit{Paths}^{\mathcal{C}}(\mathcal{A})$ meetbaar is en daardoor de berekening van de kans voor DTA$^{\Diamond}$ gereduceerd kan worden naar het berekenen van de bereikbaarheidskans van een Piecewise Deterministic Markov Process (PDP). De bereikbaarheidskans is gekarakteriseerd door de minimale oplossing van een stelsel van partiële differentiaalvergelijkingen. Daarnaast laten we zien dat DTA's met een enkele klok een speciaal geval zijn. Het stelsel van partiële differentiaalvergelijkingen kan dan worden gereduceerd naar een stelsel van lineaire vergelijkingen. We behandelen ook DTA$^{\omega}$ specificaties waar we aantonen dat dat probleem reduceerbaar is naar een bereikbaarheidsprobleem als dat van DTA$^{\Diamond}$.

Zusammenfassung

In dieser Dissertation werden drei wichtige Aspekte bei der Modellüberprüfung von Markov-Modellen betrachtet: Die *Diagnose* — das Generieren von Gegenbeispielen, die *Synthese* — das zur Verfügung stellen korrekter Parameterwerte und die *Analyse* — das Verifizieren von linearen Realzeiteigenschaften. Die drei Aspekte sind vergleichsweise unabhängig, obwohl sie alle dem Zweck dienen, neue Theorie und Algorithmen für das Forschungsfeld der probabilistischen Modellüberprüfung zu entwickeln.

Zu Beginn führen wir eine formale Definition von Gegenbeispielen im Bereich der probabilistische Modellüberprüfung ein. Wir transformieren das Problem, informative Gegenbeispiele zu finden, auf das Shortest-Path Problem. Es wird ein Framework untersucht und entwickelt um solche Gegenbeispiele zu erzeugen. Weiterhin untersuchen wir eine kompaktere Darstellung von Gegenbeispielen durch reguläre Ausdrücke. Algorithmen, die auf Heuristiken basieren, werden benutzt, um kurze reguläre Ausdrücke als Gegenbeispiele zu erhalten. Am Ende dieses Teils erweitern wir die Definition und die Algorithmen zum Generieren von Gegenbeispielen auf verschiedene Kombinationen von probabilistischen Modellen und Logiken.

Danach betrachten wir das Problem, Werte für parametrisierte zeitkontinuierliche Markovketten (pCTMCs) bezüglich zeitbeschränkter Erreichbarkeitseigenschaften zu synthetisieren. Das Ziel hierbei ist es, alle Werte für die Ratenparameter (die eine Syntheseregion bilden) zu finden, die die Spezifikation erfüllen können; hierbei sind die Ratenausdrücke Polynome über den reellen Zahlen. Zuerst stellen wir einen symbolischen Ansatz vor, in dem zunächst die Schnittpunkte durch das Lösen von Polynomgleichungen berechnet und dann miteinander verbunden werden, um die Syntheseregion zu approximieren. Ein anderer, nicht symbolischer Ansatz, der auf Intervallarithmetik beruht und für den pCTMCs instanziiert werden, wird ebenfalls untersucht. Die Fehlerschranke, die Zeitkomplexität sowie einige experimentelle Resultate werden dargestellt, gefolgt von einem detaillierten Vergleich der beiden Ansätze.

Im letzten Abschnitt steht das Verifizieren von linearen Realzeiteigenschaften auf CTMCs im Vordergrund, wobei die Eigenschaften als deterministische Zeitautomaten (DTA) gegeben sind. Das Modellüberprüfungsproblem zielt darauf ab, die Wahrscheinlichkeit der Menge aller Pfade einer CTMC $\mathcal{C}$, die von einem DTA $\mathcal{A}$ akzeptiert werden, zu bestimmen. Wir betrachten DTAs mit Erreichbarkeits- (endliche, DTA$^\diamond$) und Muller- (unendliche, DTA$^\omega$) Akzeptanzbedingungen. Es wird gezeigt, dass $\mathit{Paths}^{\mathcal{C}}(\mathcal{A})$ messbar ist und dass die Berechnung dieser Wahrscheinlichkeit im Falle von DTA$^\diamond$ auf die Berechnung der Erreichbarkeitswahrscheinlichkeit in einem Piecewise Deterministic Markov Process (PDP) reduziert werden kann. Die Erreichbarkeitswahrscheinlichkeit wird charakterisiert als die kleinste Lösung eines Systems von Integralgleichungen und es wird gezeigt, dass sie durch Lösen eines Systems von PDEs approximiert werden kann. Weiterhin zeigen wir, dass der Spezialfall eines DTA$^\diamond$, der auf eine Uhrenvariable beschränkt ist, zu einem linearen Gleichungssystem vereinfacht werden kann. Zusätzlich betrachten wir DTA$^\omega$ Spezifikationen und zeigen, dass das Problem hier wie im Fall von DTA$^\diamond$ auf das Erreichbarkeitsproblem reduziert werden kann.

摘 要

该论文讨论了概率模型检测领域的三个重要方面：*诊断* — 生成反例，*合成* — 生成有效的参数值与*分析* — 验证线性实时性质。这三个方面相对独立而又共同为概率模型检测提供理论和算法。

我们首先形式化地定义了概率模型检测中的反例。我们证明了寻找最有效反例的问题能被转化成图论中的最短路径问题。基于此，我们提出了一个能有效生成反例的算法框架。其次，我们研究了如何用正则表达式来紧凑表示反例的问题。我们采用启发式算法来获得相对短小的正则表达式反例。在此部分的最后，我们将反例的定义和算法扩充到了其他各种概率模型和逻辑。

我们然后研究了如何在连续时间马尔可夫链模型上检测时间受限的可达性性质时进行系统参数合成的问题。我们首考虑了一个速率上带有参数的马尔可夫链，这些速率表达式是在实域上的多项式。该问题在于寻找所有马尔可夫链中合适的速率参数的值（这些值形成了一个合成的区域），使之能够在得到的模型上检可达性性质成立。我们提供了两种解决方案：符号化方法和非符号化方法。在符号化方法中我们先计算所有合成区域的边界曲线和网格线的交点，然后将这些交点相连来近似待求的合成区域。在非符号化方法中模型的参数先被实例化，然后问题被归约到了无参数的模型之上。我们给出了两个方法的错误边界，时间复杂度和实验结果，并对它们进行了比较。

在最后一个部分中，我们研究了如何在连续时间马尔可夫链上验证由确定时间自动机描述的线性实时性质。该模型检测问题在于计算被确定时间自动机$\mathcal{A}$接受的马尔可夫链$\mathcal{C}$中路径集合（记作$Paths^{\mathcal{C}}(\mathcal{A})$）的概率。我们考虑确定时间自动机的两种接受条件：可达性接受条件（考虑有限路径）和Muller接受条件（考虑无限路径）。我们证明了$Paths^{\mathcal{C}}(\mathcal{A})$集合是可测度的。同时，对于可达性确定时间自动机，计算该概率可被归约为计算分段确定马尔可夫过程中的可达性问题的概率。而这个可达性概率可以被刻画为一个积分方程组的最小解，同时该概率也可以通过解决一个偏微分方程组来近似。我们还研究了单一时钟可达性确定时间自动机的特例。这个概率的计算可简化为解一个线性方程组。对于Muller确定时间自动机，我们证明了这个问题可以归约于可达性确定时间自动机中的可达性问题。

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

The increasing reliance on complex computer systems in diverse fields such as business, transport, and medicine has led to an increased interest in obtaining formal guarantees of system correctness. Consider a railway system. The most important questions one concerns are e.g., "Will it ever happen that trains collide with each other?" Or "is there a possible scenario in which two trains are mutually waiting for each other, thus effectively halting the system?" If such situations happen, this has far reaching, even catastrophic consequences.

Model Checking. A prominent discipline in computer science to assure the absence of errors (the correctness), or complementarily, to find errors (diagnosis) is *formal verification*. It mathematically proves the correctness of the design, provided in the form of a system model, with respect to a formal specification. Compared to other techniques in this spectrum (e.g., theorem proving), *model checking* is a highly automated model-based technique by a systematic, usually exhaustive, state-space exploration to check whether a system model satisfies some desired properties. Typically, those properties are expressed in some logical formalisms (e.g., temporal logic) or by automata, while the system behavior is captured by Kripke structures, i.e., finite state automata with labeled states.

Probabilistic Model Checking. Whereas model checking algorithms focus on the absolute guarantee of correctness — "it is impossible that the system fails" — in practice such rigid notions are hard, or even impossible, to guarantee. Instead, systems are subject to various phenomena of stochastic nature, such as message loss or garbling and the like. Correctness is thus of a less absolute nature.

Probabilistic model checking, based on conventional model checking, is a technique to verify system models in which transitions are equipped with or stochastic information. Popular models are *discrete-* and *continuous-time Markov chains* (DTMCs and CTMCs, respectively), and variants thereof which exhibit nondeterminism. Efficient model checking algorithms for these models have been developed, have been implemented in a variety of software tools, and have been applied to case studies from various application areas ranging from randomized distributed algorithms, computer systems and security protocols to biological systems and quantum computing. The crux of probabilistic model checking is to appropriately combine techniques from numerical mathematics and operations research with standard reachability analysis. In this way, properties such as "the (maximal) probability to reach a set of goal states by avoiding certain states is at most 0.6" can be automatically checked up to a user-defined precision. Markovian models comprising millions of states can be checked rather fast by dedicated tools such as PRISM [KNP04] and MRMC [KKZ05], as well as extensions to existing tools such as GreatSPN, SPIN, PEPA Workbench, and Statemate.

Model Checking Markov Chains. Let us zoom in and reflect on the history of verifying Markov chains against linear-time as well as branching-time properties. The goal of model checking of this kind is to compute the probability of a set of paths in the Markov chain that satisfy the property. As is summarized in the upper part of Table 1.1, for DTMCs, Hansson and Jonsson first introduced the *probabilistic computation tree logic* (PCTL) in [HJ94] and showed how the verification of a PCTL formula can be reduced to solving a system of linear equations. This can be done in polynomial time. Linear-time-wise,

- Vardi first proposed in [Var85] to check *linear temporal logic* (LTL) formulae by using the automata-based approach. The idea goes along similar lines as in the non-probabilistic setting, namely, the LTL formula φ is first transformed into a corresponding automaton (e.g., deterministic Rabin automaton, DRA for short); the product between the DTMC $\mathcal{D}$ and the DRA is then constructed; the property (adapted accordingly to a reachability property) is then checked (or to be exactly, computed) on the product DTMC. This shows that this model checking problem is in EXPSPACE.

- Courcoubetis and Yannakakis investigated in [CY95b] a tableau-based approach to solve the same problem. The algorithm transforms the LTL formula φ and the DTMC step-by-step, eliminating temporal modalities from φ, while preserving the probability of satisfaction of φ in the adapted DTMC. This reduced the upper-bound of the model checking problem to PSPACE, which matches the known lower-bound [Var85].

- Couvreur, Saheb and Sutre obtained the same PSPACE upper-bound using the

	branching time		linear time	
discrete-time (DTMC $\mathcal{D}$)	PCTL		LTL	
	linear equations [HJ94] ($\star$)		automata-based [Var85][CSS03] ($\star\star$)	tableau-based [CY95b]
	PTIME		PSPACE-C	
continuous-time (CTMC $\mathcal{C}$)	untimed PCTL	real-time CSL	untimed LTL	real-time ?
	$emb(\mathcal{C})$ cf. ($\star$)	integral equations [ASSB00][BHHK03]	$emb(\mathcal{C})$ cf. ($\star\star$)	?
	PTIME	PTIME	PSPACE-C	?

Table 1.1: An overview of verifying Markov chains

automata-based approach [CSS03]. The key of their approach is to exploit some nice characteristics of the obtained automaton (e.g., *separated*).

Model checking CTMCs (cf. the lower part of Table 1.1), on the other hand, has been focused more on branching-time logics, e.g., *continuous stochastic logic* (CSL) [ASSB00][BHHK03]. CSL model checking proceeds — like CTL model checking — by a recursive descent over the parse tree of the formula. One of the key ingredients is that the reachability probability for a time-bounded until-formula can be characterized as the least solution of a system of integral equations and approximated arbitrarily closely by a reduction to transient analysis in CTMCs. This results in a polynomial-time approximation algorithm. As a special case when the until operator is time-*unbounded* ($\star$ in the lower part of the table), the verification can be performed on the embedded DTMC by applying the same technique as in [HJ94]. Verifying LTL formulae on CTMCs ($\star\star$ in the lower part of the table) follows basically the same approach as in the discrete-time case, as same probability will be yielded in the CTMC and in the embedded DTMC.

This table, with a "hole" in it, sheds light on some problems that are still open, i.e., how to verify CTMCs against *linear real-time properties*? Those properties can either be expressed by linear real-time logics, e.g., *metric* (*interval*) *temporal logic* (M(I)TL) [Koy90][AFH96] or directly by a (timed) automaton [AD94]. We will partially answer this question by investigating the verification of a CTMC against a deterministic timed automaton (DTA) specification.

Counterexample Generation. A major strength of model checking is the possibility to generate diagnostic counterexamples in case a property is violated. This is nicely captured by Clarke in his reflections on 25 years of model checking [Cla08]: "*It is impossible to overestimate the importance of the counterexample feature. The counterexamples are invaluable in debugging complex systems. Some people use model checking just for this feature.*" Counterexamples are of utmost importance in model checking: first, and for all, they provide diagnostic feedback even in cases where only a fragment of the entire model can be searched. They also constitute the key to successful abstraction-refinement techniques [CGJ+00], and are at the core of obtaining feasible schedules in e.g., timed model checking [BLR05]. As a result, advanced counterexample generation and analysis techniques have intensively been investigated, see e.g., [JRS04][BNR03][dAHM00].

The shape of a counterexample depends on the checked formula and the temporal logic. For logics such as LTL, typically finite or infinite paths through the model are required. The violation of linear-time safety properties is indicated by finite paths that end in a "bad" state. The violation of liveness properties, instead, require infinite paths ending in a cyclic behavior indicating that something "good" will never happen. LTL model checkers usually incorporate breadth-first search algorithms to generate *shortest* counterexamples, i.e., paths of minimal length. For branching-time logics such as CTL, paths may act as counterexamples for a subclass of universally quantified formulae, i.e., those in ACTL$\cap$LTL. To cover a broader spectrum of formulae, though, more advanced structures such as trees of paths [CJLV02], proof-like counterexamples [GC03] (for ACTL$\setminus$LTL) or annotated paths [SG03] (for ECTL) are used.

The counterexample generation in the probabilistic model checking, however, only received scant attention dating back to 2005, when I started my doctor study. Due to the stochastic nature of probabilistic models, counterexamples in most cases cannot be simply captured by a single path which usually bears a low probability. Instead, we explore a set of paths as a counterexample, where the sum of the path probabilities shows the violation of the property. We studied the definition, compact representation as well as various counterexample generation algorithms to tackle different combinations of probabilistic models and logics.

Parameter Synthesis. A disadvantage of the traditional approaches to model checking, however, is that they can only check the validity of properties under the assumption that all parameter values are known. This means that concrete values of e.g., timing parameters, branching probabilities, costs, and so forth, need to be explicitly given. Although this might be appropriate for the a posteriori verification of concrete system realizations, for design models at a higher level of abstraction this is less adequate. In earlier design phases, such explicit information about model parameters is mostly absent, and instead, only the ranges of parameter values, or the relationship between parameters is known (if at all). For models that incorporate aspects of a random nature, the need for concrete parameter values is, in fact, a significant hurdle, as mostly precise information about the random variables is known after extensive experimentation and measurements only. This is, e.g., witnessed by the fact that fitting —roughly speaking, the attempt to find an appropriate and accurate distribution to actual measurements— is an active field of research in model-based performance analysis [TBT06].

In practical system design, one is not interested in checking a concrete instance, but rather, often in deriving parameter constraints that ensure the validity of the property under consideration. Typical examples are failure-repair systems such as multi-processor systems and modern distributed storage systems, in which components (such as memories or processors) may fail and where only lower- and upper-bounds on repair times are known. Rather than determining whether for a certain combination of failure and repair rates, a property holds, one would like to synthesize the set of pairs of rates for which the validity of the property is guaranteed.

To this end, we start with a CTMC with parameters on rates. Given a time-bounded reachability property, we answer the following question "With which parameter values can it be guaranteed that the property holds on the CTMC"? This, compared to model checking problem, gives a more "constructive" way in the modeling phase.

1.2 Outline of the Dissertation

As the title of the dissertation suggests, three aspects of probabilistic models will be addressed — *diagnosis* (counterexample generation for probabilistic model checking), *synthesis* (synthesizing system parameters for probabilistic models) and *analysis* (verifying linear real-time properties for probabilistic models). Prior to presenting the main results, **Chapter 2** presents some preliminaries for the models and logics that are referred intensively and extensively throughout this dissertation.

Diagnosis

This part considers the generation of counterexamples in probabilistic model checking. It consists of three chapters:

- **Chapter 3** establishes the theoretical underpinnings of counterexample generation in the setting of checking a fragment of PCTL (of the form $\mathcal{P}_{\leqslant p}(\Phi \, \mathsf{U}^I \, \Psi)$) on DTMCs. We formally define the concept of a (strongest) evidence and a (smallest) counterexample and propose algorithms to generate such evidences and counterexamples by reducing the problems to (variants of) shortest path(s) problems in graph theory. Correctness and complexity results are provided as well.

- **Chapter 4** proposes an alternative and more compact way of representing a counterexample. This is motivated by the experimental results — partially substantiated with combinatorial arguments — showing that the cardinality of such sets may be excessive. We use *regular expressions* to compactly represent counterexamples for reachability properties. An algorithm is represented to generate minimal regular expressions and a recursive scheme is adapted to determine the likelihood of a counterexample. The state space minimization on DTMCs prior to counterexample generation may yield even shorter regular expressions. The feasibility of the approach is illustrated by means of two protocols: leader election and the Crowds protocol.

- **Chapter 5** focuses on the applicability of the established approaches in Chapter 3 and 4 to different probabilistic models and logics. We show that those approaches can be extended to full PCTL, in particular probability thresholds with lower-bounds as well as qualitative fragment of PCTL; to Markov reward models; also to various combinations of the models DTMC and Markov decision processes (MDPs) and the logics PCTL, LTL and PCTL*. Besides the discrete-time settings, the approaches can also be utilized in CTMC model checking of CSL.

Synthesis

Chapter 6 considers the problem of synthesizing parametric rate values in CTMCs that can ensure the validity of time-bounded reachability properties. Rate expressions over variables indicate the average speed of state changes and are expressed using the polynomials over reals. A symbolic and a non-symbolic approach are proposed to approximate the set of parameter values which can guarantee the validity of the given property. Both approaches are based on discretizing parameter ranges together with a refinement technique. We compare the two approaches, analyze the respective time complexity and show some experimental results on a case study — a real-time storage system with probabilistic error checking facilities.

Analysis

Chapter 7 considers the problem of quantitative verification of a CTMC against a linear real-time property specified by a deterministic timed automaton (DTA) $\mathcal{A}$. Specifically, what is the probability of the set of paths of $\mathcal{C}$ that are accepted by $\mathcal{A}$ ($\mathcal{C}$ satisfies $\mathcal{A}$)? It is shown that this set of paths is measurable. We consider two kinds of acceptance conditions: the reachability condition (in DTA$^\Diamond$) and the Muller acceptance condition (in DTA$^\omega$). The former accepts (finite) paths which reach some final states and the latter accepts (infinite) paths that infinitely often visit some set of final states. For DTA$^\Diamond$, we prove that computing this probability can be reduced to computing the reachability probability in a piecewise deterministic Markov process (PDP). The reachability probability is characterized as the least solution of a system of integral equations and is shown to be approximated by solving a system of partial differential equations. For the special case of single-clock DTA, the system of integral equations can be transformed into a system of linear equations where the coefficients are solutions of ordinary differential equations. For DTA$^\omega$, by finding the accepting BSCCs in the region graph, the ω-regular acceptance condition is proven to be reducible to the finite paths case, i.e., the reachability problem.

Chapter 8 concludes each part and discusses the future work.

1.3 Origins of the Chapters and Credits

- Chapter 3 and 4 are an extension of [HK07a] and [DHK08], and the recent journal version [HKD09]. Chapter 5 has partially appeared in [HKD09], where the CTMC part was originally in [HK07b].

  1. [HK07a] Tingting Han and Joost-Pieter Katoen. Counterexamples in probabilistic model checking. In *TACAS*, LNCS 4424, pages 72–86, 2007.
  2. [HK07b] Tingting Han and Joost-Pieter Katoen. Providing evidence of likely being on time: Counterexample generation for CTMC model checking. In *ATVA*, LNCS 4762, pages 331–346, 2007.
  3. [DHK08] Berteun Damman, Tingting Han, and Joost-Pieter Katoen. Regular expressions for PCTL counterexamples. In *QEST*, pages 179-188, IEEE CS Press, 2008.
  4. [HKD09] Tingting Han, Joost-Pieter Katoen, and Berteun Damman. Counterexample generation in probabilistic model checking. *IEEE Trans. Software Eng.*, 35(2):241–257, 2009.

- Chapter 6 is an extension of [HKM08a], where together with Alexandru Mereacre, I worked out the symbolic algorithm for generating the synthesis region. The non-symbolic algorithm in Chapter 6 is new and not part of this paper.

  5. [HKM08a] Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Approximate parameter synthesis for probabilistic time-bounded reachability. In *RTSS*, IEEE CS Press, pages 173–182, 2008.

- Chapter 7 is an extension of [CHKM09a], where together with Taolue Chen and Alexandru Mereacre, I defined the product model as well as the region construction of the product and I also worked actively on the recursive equations for computing the reachability probability. The part regarding Muller acceptance condition in Chapter 7 is new and is not part of this paper.

6. [CHKM09a] Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Quantitative model checking of continuous-time Markov chains against timed automata specification. In *LICS*, pages 309–318, IEEE CS Press, 2009.

- The following of my publications are not included in this dissertation:

7. [CHK08] Taolue Chen, Tingting Han, and Joost-Pieter Katoen. Time-abstracting bisimulation for probabilistic timed automata. In *TASE*, pages 177–184, IEEE CS Press, 2008.
8. [HKM08b] Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Compositional modeling and minimization of time-inhomogeneous Markov chains. In *HSCC*, LNCS 4981, pages 244–258, 2008.
9. [CHKM09b] Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. LTL model checking of time-inhomogeneous Markov chains. In *ATVA*, LNCS, to appear, 2009.

Suggested Way of Reading. Due to the diversity of the topics and heavy usage of symbols in the dissertation, it is difficult to unify the notations while obeying the conventions for all the chapters. In other words, overloading is unavoidable. However, for each part, the notation is consistent and unambiguous, so readers are kindly requested to take a local instead of a global view of notations throughout this dissertation. Moreover, since most of the results presented here are originated from the publications that I have worked as a coauthor, I shall use "we" instead of "I" in this dissertation.

Chapter 2

Preliminary

2.1 Probabilistic Models

2.1.1 Discrete-Time Markov Chains

Let AP be a fixed, finite set of atomic propositions ranged over by $a, b, c, \ldots$.

Definition 2.1 (FPS) *A* fully probabilistic system *is a triple* $(S, \mathbf{P}, L)$*, where:*

- S *is a finite set of states;*
- $\mathbf{P} : S \times S \to [0,1]$ *is a* sub-stochastic *matrix, i.e.,* $\forall s \in S.\ \sum_{s' \in S} \mathbf{P}(s,s') \in [0,1]$*;*
- $L : S \to 2^{\mathrm{AP}}$ *is a labeling function which assigns to each state* $s \in S$ *the set* $L(s)$ *of atomic propositions that are valid in* s*.*

Definition 2.2 (DTMC) *A* (*labeled*) discrete-time Markov chain $\mathcal{D}$ *is a* FPS $(S, \mathbf{P}, L)$ *where* $\mathbf{P}$ *is a* stochastic *matrix, i.e.,* $\sum_{s' \in S} \mathbf{P}(s,s') = 1$ *for any* $s \in S$*.*

Intuitively, a DTMC is a Kripke structure in which all transitions are equipped with discrete probabilities such that the sum of outgoing transitions of each state equals one. A state s in $\mathcal{D}$ is called *absorbing* (resp. *sinking*) if $\mathbf{P}(s,s) = 1$ (resp. $\mathbf{P}(s,s') = 0$ for $s' \in S$). Note that if a state in a DTMC $\mathcal{D}$ is made sinking, then $\mathcal{D}$ becomes an FPS. W.l.o.g., we assume a DTMC (or FPS) to have a unique initial state.

Definition 2.3 (Paths) *Let* $\mathcal{D} = (S, \mathbf{P}, L)$ *be a* DTMC*. An* infinite path ρ *in* $\mathcal{D}$ *is an infinite sequence* $s_0 {\cdot} s_1 {\cdot} s_2 \cdots$ *of states such that* $\forall i \geqslant 0.\ \mathbf{P}(s_i, s_{i+1}) > 0$*. A* finite path σ *is a finite prefix of an infinite path.*

Let $Paths^{\omega}_{\mathcal{D}}(s)$ and $Paths^{\star}_{\mathcal{D}}(s)$ denote the set of infinite and finite paths in $\mathcal{D}$ that start in state s, respectively. The subscript $\mathcal{D}$ is omitted when it is clear from the

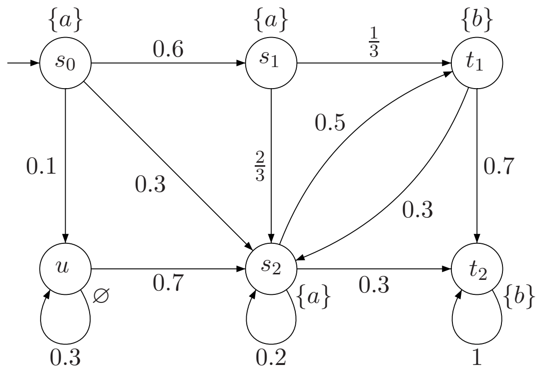


Figure 2.1: An example DTMC $\mathcal{D}$

context. For state s and finite path $\sigma = s_0 \cdots s_n$ with $\mathbf{P}(s_n, s) > 0$, let $\sigma{\cdot}s$ denote the path obtained by extending σ by s.

Let τ denote either a finite or infinite path. Let $|\tau|$ denote the *length* (or *hop count*) of τ, i.e., $|s_0{\cdot}s_1 \cdots s_n| = n$, $|s_0| = 0$ and $|\tau| = \infty$ for infinite τ. For $0 \leqslant i \leqslant |\tau|$, $\tau[i] = s_i$ denotes the $(i{+}1)$-th state in τ. We use $\tau[..i]$ to denote the *prefix* of τ truncated at length i (thus ending in s_i), formally, $\tau[..i] = \tau[0]{\cdot}\tau[1] \cdots \tau[i]$. We use $\mathit{Pref}(\tau)$ to denote the set of prefixes of τ, i.e., $\mathit{Pref}(\tau) = \{\tau[..i] \mid 0 \leqslant i \leqslant |\tau|\}$. Similarly, $\tau[i..]$ and $\tau[i..j]$ denote the *suffix* starting from $\tau[i]$ and the *infix* between $\tau[i]$ and $\tau[j]$, respectively.

Probability Measure on Paths. A DTMC $\mathcal{D}$ induces a probability space. The underlying σ-algebra is defined over the basic cylinder set induced by the finite paths starting in the initial state s_0. The *probability measure* $\Pr^{\mathcal{D}}_{s_0}$ (briefly Pr) induced by $(\mathcal{D}, s_0)$ is the unique measure on this σ-algebra where:

$$\Pr\{\underbrace{\rho \in \mathit{Paths}^{\omega}_{\mathcal{D}}(s_0) \mid \rho[..n] = s_0 \cdots s_n}_{\mathit{Cyl}(s_0 \cdots s_n)}\} = \prod_{0 \leqslant i < n} \mathbf{P}(s_i, s_{i+1}).$$

The *probability of finite path* $\sigma = s_0 \cdots s_n$ is defined as $\mathbb{P}(\sigma) = \prod_{0 \leqslant i < n} \mathbf{P}(s_i, s_{i+1})$. Note that although $\Pr(\mathit{Cyl}(\sigma)) = \mathbb{P}(\sigma)$, they have different meanings: Pr is a measure on sets of infinite paths whereas $\mathbb{P}$ refers to finite ones. A set C of finite paths is *prefix containment free* if for any $\sigma, \sigma' \in C$ with $\sigma \neq \sigma'$, it holds that $\sigma \notin \mathit{Pref}(\sigma')$. The probability of a prefix containment free set C is $\mathbb{P}(C) = \sum_{\sigma \in C} \mathbb{P}(\sigma)$. Note that paths in C induce disjoint cylinder sets.

Example 2.4 *Fig. 2.1 illustrates a* DTMC *with initial state* s_0. AP $= \{a, b\}$ *and* L *is given as* $L(s_i) = \{a\}$, *for* $0 \leqslant i \leqslant 2$; $L(t_1) = L(t_2) = \{b\}$ *and* $L(u) = \varnothing$. t_2 *is an absorbing state.* $\sigma_1 = s_0{\cdot}u{\cdot}s_2{\cdot}t_1{\cdot}t_2$ *is a finite path with* $\mathbb{P}(\sigma_1) = 0.1 \times 0.7 \times 0.5 \times 0.7$ *and* $|\sigma_1| = 4$, $\sigma_1[3] = t_1$. $\rho_1 = s_0{\cdot}(s_2{\cdot}t_1)^{\omega}$ *is an infinite path.* ♦

2.1.2 Continuous-Time Markov Chains

Definition 2.5 (CTMC) *A* continuous-time Markov chain *is a triple* $\mathcal{C} = (S, \mathbf{R}, L)$ *with* S *and* L *the same as in* DTMC*s and* $\mathbf{R} : S \times S \to \mathbb{R}_{\geqslant 0}$ *is the rate matrix.*

W.l.o.g., we assume a CTMC to have a unique initial state s_0. Given n the cardinality of S, $\vec{E} = [E(s_0), \ldots, E(s_{n-1})]$ is the vector of *exit rates*, where $E : S \to \mathbb{R}_{\geqslant 0}$ is the exit rate function with $E(s_i) = \sum_{s' \in S} \mathbf{R}(s_i, s')$. Intuitively, $E(s)$ is the speed of firing a transition from s, or the average delay in s. More precisely, with probability $(1 - e^{-E(s) \cdot t})$ a transition is enabled within the next t time units provided that the current state is s. The density function for this is $den(s, t) = E(s) \cdot e^{-E(s) \cdot t}$, where $\int_0^t den(s, x) dx = 1 - e^{-E(s) \cdot t}$.

Definition 2.6 (Alternative definition of CTMC) *A* CTMC $\mathcal{C} = (S, \mathbf{R}, L)$ *can equivalently be represented as* $\mathcal{C} = (S, \mathbf{P}, E, L)$*, where* $(S, \mathbf{P}, L)$ *is the embedded* DTMC *of* $\mathcal{C}$ *and* $\mathbf{P}(s, s') = \frac{\mathbf{R}(s,s')}{E(s)}$*, if* $E(s) > 0$ *and* $\mathbf{P}(s, s) = 1$*, if* $E(s) = 0$*.*

In the following, we will use the two CTMC definitions interchangeably. If $\mathbf{P}(s, s') > 0$ for more than one state s', a *race* between the outgoing transitions from s exists. The probability of transition $s \to s'$ winning this race in time interval $[0, t]$ is given by:

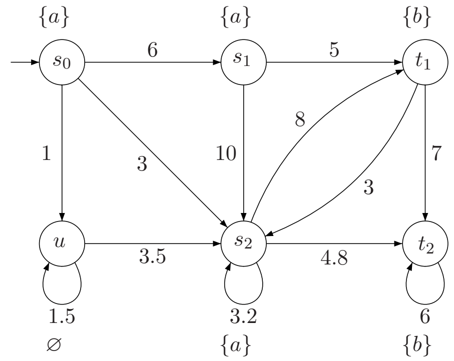
$$\mathbf{P}(s, s', t) = \mathbf{P}(s, s') \cdot \big(1 - e^{-E(s) \cdot t}\big).$$

We define $den(s, s', t) = \mathbf{P}(s, s') \cdot E(s) \cdot e^{-E(s) \cdot t}$. Note that $\mathbf{P}(s, s', t)$ and $den(s, s', t_1)$ have the following relation:

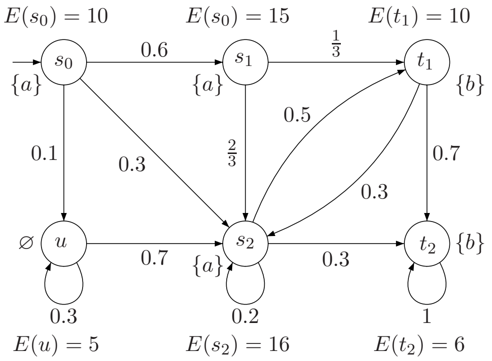
$$\mathbf{P}(s, s', t) = \mathbf{P}(s, s') \cdot \int_0^t den(s, t_1)\, dt_1 = \int_0^t den(s, s', t_1)\, dt_1.$$

Definition 2.7 (Timed paths) *Let* $\mathcal{C}$ *be a* CTMC*. An infinite timed path* ρ *is of the form* $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \cdots$ *with* $s_i \in S$ *and* $t_i \in \mathbb{R}_{\geqslant 0}$ *such that* $\mathbf{P}(s_i, s_{i+1}) > 0$ *for* $i \geqslant 0$*. Let* $Paths^n_{\mathcal{C}} \subseteq S \times (\mathbb{R}_{\geqslant 0} \times S)^n$ *be the set of paths of length* n *in* $\mathcal{C}$*; the set of finite paths in* $\mathcal{C}$ *is defined by* $Paths^{\star}_{\mathcal{C}} = \bigcup_{n \in \mathbb{N}} Paths^n_{\mathcal{C}}$ *and* $Paths^{\omega}_{\mathcal{C}} \subseteq (S \times \mathbb{R}_{\geqslant 0})^{\omega}$ *is the set of infinite paths in* $\mathcal{C}$*.* $Paths_{\mathcal{C}} = Paths^{\star}_{\mathcal{C}} \cup Paths^{\omega}_{\mathcal{C}}$ *denotes the set of all paths in* $\mathcal{C}$*.*

Note that t_i are delays instead of absolute time stamps. All the definitions on paths in DTMCs can be adopted here. Let τ denote a finite or infinite path in CTMC. $\tau[i] = s_i$ and $\tau\langle i \rangle = t_i$ denote the i-th state s_i and the time spent in s_i, respectively. We use $\tau@t$ to denote the state occupied in τ at $t \in \mathbb{R}_{\geqslant 0}$, i.e., $\tau@t = \tau[i]$ where i is the smallest index such that $\sum_{j=0}^{i} \tau\langle j \rangle > t$. For finite path σ and $\ell = |\sigma|$, $\sigma\langle \ell \rangle = \infty$; and for $t \geqslant \sum_{j=0}^{\ell-1} t_j$, $\sigma@t = s_\ell$.



(a) $\mathcal{C} = (S, \mathbf{R}, L)$



(b) $\mathcal{C} = (S, \mathbf{P}, E, L)$

Figure 2.2: An example CTMC $\mathcal{C}$

Probability Measure on Paths. The definition of a Borel space on paths through CTMCs follows [Var85][BHHK03]. A CTMC $\mathcal{C}$ with initial state s_0 yields a probability measure $\Pr^{\mathcal{C}}$ on paths as follows: Let $s_0, \ldots, s_k \in S$ with $\mathbf{P}(s_i, s_{i+1}) > 0$ for $0 \leqslant i < k$ and $I_0, \ldots, I_{k-1}$ nonempty intervals in $\mathbb{R}_{\geqslant 0}$, $Cyl(s_0, I_0, \ldots, I_{k-1}, s_k)$ denotes the *cylinder set* consisting of all paths $\rho \in Paths^{\omega}_{\mathcal{C}}(s_0)$ such that $\rho[i] = s_i$ $(i \leqslant k)$, and $\rho\langle i \rangle \in I_i$ $(i < k)$. $\mathcal{F}(Paths^{\omega}_{\mathcal{C}}(s_0))$ is the smallest σ-algebra on $Paths^{\omega}_{\mathcal{C}}(s_0)$ which contains all sets $Cyl(s_0, I_0, \ldots, I_{k-1}, s_k)$ for all state sequences $(s_0, \ldots, s_k) \in S^{k+1}$ with $\mathbf{P}(s_i, s_{i+1}) > 0$ $(0 \leqslant i < k)$ and $I_0, \ldots, I_{k-1}$ range over all sequences of nonempty intervals in $\mathbb{R}_{\geqslant 0}$. The probability measure $\Pr^{\mathcal{C}}$ on $\mathcal{F}(Paths^{\omega}_{\mathcal{C}}(s_0))$ is the unique measure defined by induction on k by $\Pr^{\mathcal{C}}(Cyl(s_0)) = 1$ and $\Pr^{\mathcal{C}}(Cyl(s)) = 0$ if $s \neq s_0$ and for $k > 0$:

$$\begin{aligned} \Pr^{\mathcal{C}}\big(Cyl(s_0, I_0, \ldots, I_{k-1}, s_k)\big) &= \Pr^{\mathcal{C}}\big(Cyl(s_0, I_0, \ldots, I_{k-2}, s_{k-1})\big) \\ &\quad \cdot \int_{I_{k-1}} \mathbf{P}(s_{k-1}, s_k) E(s_{k-1}) \cdot e^{-E(s_{k-1})\tau} d\tau. \end{aligned} \tag{2.1}$$

The vector $\vec{\wp}(t) = (\wp_0(t), \ldots, \wp_{n-1}(t))$ gives the *transient probability* of the CTMC, i.e., the probability of being in state s_i $(0 \leqslant i < n)$ at time t. The Chapman-Kolmogorov equations describe the evolution of the transient probability distribution over time:

$$\frac{d\vec{\wp}(t)}{dt} = \vec{\wp}(t)\mathbf{Q}, \qquad \sum_{i=0}^{n-1} \wp_i(t_0) = 1, \tag{2.2}$$

where $t_0 = 0$ and $\vec{\wp}(t_0)$ is the initial condition. Note that $\vec{\wp}(t_0) = (1, 0, \ldots, 0)$ if s_0 is the unique initial state. The matrix $\mathbf{Q} = \mathbf{R} - diag(\vec{E})$ is the *infinitesimal generator* of CTMC $\mathcal{C}$ and $diag(\vec{E})$ is the diagonal matrix constructed from $\vec{E}$.

Example 2.8 *Fig. 2.2 illustrates a* CTMC $\mathcal{C}$ *in two equivalent forms. The embedded* DTMC *is in Fig. 2.1. A finite path in* $\mathcal{C}$ *is* $\sigma = s_0 \xrightarrow{2} s_1 \xrightarrow{\sqrt{2}} s_2 \xrightarrow{0.3} t_2$, *where* $\sigma[2] =$

s_2, $\sigma\langle 1\rangle = \sqrt{2}$ and $\sigma@3.41 = s_1$. An infinite path is $\rho = s_0 \xrightarrow{3.9} (u \xrightarrow{0.2})^\omega$. The probability to take the transition $s_0 \to s_1$ within 5 time units is $\mathbf{P}(s_0, s_1, 5) = 0.6 * (1 - e^{-10*5})$ and the corresponding function $den(s_0, s_1, 5) = 6 * 10 * e^{-10*5}$. ♦

Uniformization. It is a well-known method (a.k.a. *Jensen's method* or *randomization* [Jen53]) for computing transient probabilities of a CTMC at specific time t. This method reduces the evolution of a CTMC to the evolution of a DTMC subordinated to a Poisson process. Intuitively, we pick the rate of the fastest state (or greater) as the uniformization rate q and force (or normalize) all the other states to evolve with this rate. Since now all the states take the same "rhythm" of evolution (as in a DTMC), we can thus reduce the original CTMC to a DTMC, called *uniformized* DTMC. The Poisson process relates the uniformized DTMC to the original CTMC in the way that it captures with which probability a certain number of epochs evolve in the DTMC in time $[0, t]$. Uniformization is attractive because of its excellent numerical stability and the fact that the computational error is well-controlled and can be specified in advance.

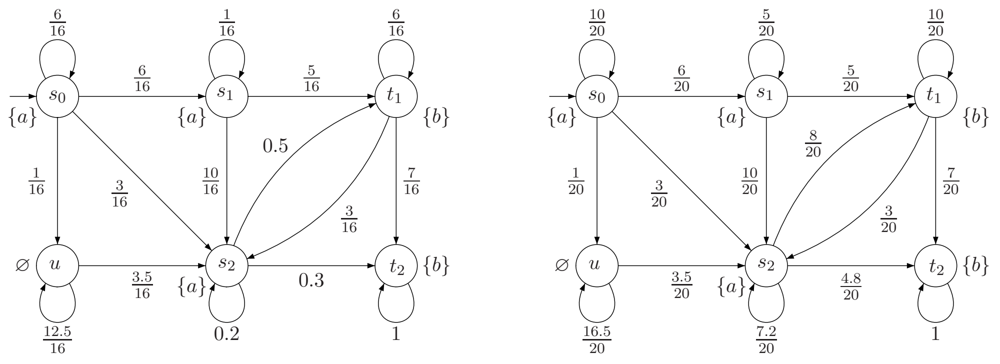
Definition 2.9 (Uniformized DTMC) *For* CTMC $\mathcal{C} = (S, \mathbf{P}, E, L)$, *the* uniformized DTMC *is $\mathcal{U} = unif(\mathcal{C}) = (S, \mathbf{U}, L)$, where $\mathbf{U}$ is defined by $\mathbf{U} = I + \frac{\mathbf{Q}}{q}$ with $q \geqslant \max_i\{E(s_i)\}$ and $\mathbf{Q} = \mathbf{R} - diag(\vec{E})$. For the case $q = 0$, $\mathbf{U}(s, s) = 1$ for any $s \in S$.*

In the rest of the dissertation, we always use $\mathcal{U}$ to denote $unif(\mathcal{C})$. The uniformization rate q can be any value no less than the shortest mean residence time. All rates in the CTMC are normalized with respect to q. For each state s with $E(s) = q$, one epoch in the uniformized DTMC corresponds to a single exponentially distributed delay with rate q, after which one of its successor states is selected probabilistically. As a result, such states have no additional self-loop in the DTMC. If $E(s) < q$, i.e., state s has, on average, a longer state residence time than $\frac{1}{q}$, one epoch in the DTMC might not be "long enough"; hence, in the next epoch, these states might be revisited with some positive probability. This is represented by equipping these states with a self-loop with probability $1 - \frac{E(s)}{q} + \frac{\mathbf{R}(s,s)}{q}$.

The transient probability vector $\vec{\wp}^{\mathcal{C}}(t) = \big(\wp_0^{\mathcal{C}}(t), \ldots, \wp_{n-1}^{\mathcal{C}}(t)\big)$ at time t is computed in the uniformized DTMC $\mathcal{U}$ as:

$$\vec{\wp}^{\mathcal{C}}(t) = (1, 0, \ldots, 0) \cdot \sum_{i=0}^{\infty} PP(i, qt)\mathbf{U}^i = \sum_{i=0}^{\infty} PP(i, qt)\vec{\wp}^{\mathcal{U}}(i), \tag{2.3}$$

where $(1, 0, \ldots, 0)$ is the initial distribution. Note that $\wp_j^{\mathcal{U}}(i)$ characterizes the probability to be in state s_j at i-th hop in the DTMC $\mathcal{U}$, given the same initial distribution as in the CTMC. $\wp_j^{\mathcal{U}}(i)$ is determined recursively by $\vec{\wp}^{\mathcal{U}}(i) = \vec{\wp}^{\mathcal{U}}(i-1) \cdot \mathbf{U}$. $PP(i, qt) = e^{-qt}\frac{(qt)^i}{i!}$ is the i-th Poisson probability that i epochs occur in $[0, t]$ when the average rate is $\frac{1}{qt}$. The Poisson probabilities can be computed in a stable way



s_0 s_1 t_1 u s_2 t_2
$\{a\}$ $\{b\}$ $\varnothing$
$\frac{6}{16}$ $\frac{1}{16}$ $\frac{5}{16}$ $\frac{3}{16}$ $\frac{10}{16}$ $\frac{7}{16}$ $\frac{3.5}{16}$ $\frac{12.5}{16}$ 0.5 0.3 0.2 1
$\frac{10}{20}$ $\frac{5}{20}$ $\frac{6}{20}$ $\frac{8}{20}$ $\frac{1}{20}$ $\frac{3}{20}$ $\frac{10}{20}$ $\frac{7}{20}$ $\frac{3.5}{20}$ $\frac{4.8}{20}$ $\frac{16.5}{20}$ $\frac{7.2}{20}$ 1

(a) $q = 16$ (b) $q = 20$

Figure 2.3: The uniformized DTMC $\mathcal{U}$

with the Fox-Glynn algorithm [FG88], thus avoiding numerical instability. The infinite summation problem is solved by introducing a required accuracy ε, such that $\|\vec{\wp}^{\mathcal{C}}(t) - \tilde{\vec{\wp}}^{\mathcal{C}}(t)\| \leqslant \varepsilon$, where $\tilde{\vec{\wp}}^{\mathcal{C}}(t) = \sum_{i=0}^{k_\varepsilon} PP(i, qt)\cdot\vec{\wp}^{\mathcal{U}}(i)$ is the approximation of $\vec{\wp}^{\mathcal{C}}(t)$ and k_ε is the number of terms to be taken in (2.3), which is the smallest value satisfying:

$$\sum_{i=0}^{k_\varepsilon} \frac{(qt)^i}{i!} \geqslant \frac{1-\varepsilon}{e^{-qt}} = (1-\varepsilon)\cdot e^{qt}. \tag{2.4}$$

If qt is larger, k_ε tends to be of the order $\mathcal{O}(qt)$.

Example 2.10 *For the* CTMC $\mathcal{C}$ *in Fig. 2.2, the uniformized* DTMC $\mathcal{U}$ *is shown in Fig. 2.3 for different uniformization rates* $q = 16$ *and* 20. ♦

We note that the larger q is, the shorter one epoch is. This indicates that the discretization step is finer and thus it would take more rounds (a larger k_ε) to reach the same error bound ε. In this dissertation, we take the uniformization rate $q = \max_i\{E(s_i)\}$.

2.1.3 Markov Decision Processes

Definition 2.11 (Probability distribution) *For a finite set* S*, a* distribution *is a function* $\zeta : S \to [0, 1]$ *such that* $\sum_{s\in S} \zeta(s) = 1$*. With* $Distr(S)$ *we denote the set of all probability distributions on* S*.*

Definition 2.12 (MDP) *A* Markov decision process *is a triple* $\mathcal{M} = (S, Steps, L)$ *where* S *and* L *are as in* DTMCs *and* $Steps : S \to 2^{Distr(S)}$ *assigns to each state a set of distributions on* S*.*

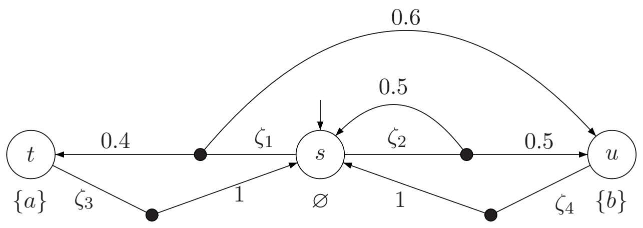


Figure 2.4: An example MDP $\mathcal{M}$

An MDP exhibits a two-phase behavior: whenever the system is in state s, first a probability distribution $\zeta \in \mathit{Steps}(s)$ is nondeterministically selected and then the successor state is probabilistically chosen according to ζ.

Definition 2.13 (Paths in MDPs) *Let $\mathcal{M} = (S, \mathit{Steps}, L)$ be an* MDP. *An* infinite path *in $\mathcal{M}$ is a sequence $\rho = s_0 \xrightarrow{\zeta_1} s_1 \xrightarrow{\zeta_2} s_2 \cdots$ s.t. for all i, $s_i \in S$, $\zeta_{i+1} \in \mathit{Steps}(s_i)$ and $\zeta_{i+1}(s_{i+1}) > 0$. A* finite path *σ in $\mathcal{M}$ is a finite prefix of an infinite path.*

Given a finite path $\sigma = s_0 \xrightarrow{\zeta_1} \cdots \xrightarrow{\zeta_n} s_n$, let $\mathit{first}(\sigma) = s_0$ and $\mathit{last}(\sigma) = s_n$. The notations for set of finite and infinite paths are similar as those in DTMCs.

Example 2.14 *An* MDP *$\mathcal{M}$ is illustrated in Fig. 2.4. There are two distributions ζ_1, ζ_2 in state s, one of which is nondeterministically selected in s. $\sigma = s \xrightarrow{\zeta_2} u \xrightarrow{\zeta_4} s \xrightarrow{\zeta_2} u$ is a finite path.* ♦

Definition 2.15 (Schedulers) *Let $\mathcal{M} = (S, \mathit{Steps}, L)$ be an* MDP. *A* scheduler *of $\mathcal{M}$ is a function $\mathfrak{G} : \mathit{Paths}^{\star}_{\mathcal{M}} \to \mathit{Distr}(S)$ mapping every finite path $\sigma \in \mathit{Paths}^{\star}_{\mathcal{M}}$ to a distribution $\mathfrak{G}(\sigma)$ on S such that $\mathfrak{G}(\sigma) \in \mathit{Steps}(\mathit{last}(\sigma))$.*

A scheduler resolves the nondeterminism by choosing a probability distribution based on the process executed so far. Formally, if an MDP is guided by scheduler $\mathfrak{G}$ and has the following path $\sigma = s_0 \xrightarrow{\zeta_1} \cdots \xrightarrow{\zeta_n} s_n$ as its *history* at the moment, then it will be in state s in the next step with probability $\mathfrak{G}(\sigma)(s)$.

In this dissertation, two types of schedulers will be mentioned: *simple* schedulers and *finite-memory* (fm-) schedulers. We briefly introduce them here. A simple scheduler always selects the same distribution in a given state. The choice only depends on the current state and is independent of what happen in the history, i.e., which path led to the current state. Differently, an fm-scheduler formulates its behavior by a deterministic finite automaton (DFA). The selection of the distribution to be taken in $\mathcal{M}$ depends on the current state (as before) and the current state (called *mode*) of the scheduler, i.e., the DFA. Simple schedulers can be considered as finite-memory schedulers with just a single mode. The formal definitions can be found in [BK08] (Chapter 10).

The basic cylinder and probability space of an MDP are constructed in a standard way [BdA95]. For MDP $\mathcal{M}$ and a scheduler $\mathfrak{G}$, a Markov chain $\mathcal{M}^{\mathfrak{G}}$ can be derived [BdA95]. The state space of $\mathcal{M}^{\mathfrak{G}}$ is in general infinite, however, if a scheduler is simple or finite-memory, the resulting DTMC is finite. It suffices to consider simple schedulers for model checking PCTL formulae without hop-bounded until operators. For the PCTL formulae with hop-bounded until operators, fm-schedulers are required [BK98]. The problem of model checking ω-regular properties can be solved by an automata-based approach [BK08].

As an important aspect for nondeterministic systems, *fairness* can be also taken into consideration. The fairness assumptions (e.g. specified by an LTL formula) on the resolution of the nondeterministic choices are constraints on the schedulers. Instead of ranging over all schedulers, only the schedulers that generate fair paths (i.e., paths satisfying the fairness assumption) are considered and taken into account for the analysis. A scheduler is *fair* if it almost surely generates fair paths. It has been proven that it suffices to only consider the finite-memory fair schedulers to model check the PCTL and ω-regular properties.

2.2 Probabilistic Logics

2.2.1 Probabilistic Computation Tree Logic

Probabilistic computation tree logic (PCTL) [HJ94] is an extension of CTL in which state-formulae are interpreted over states of a DTMC and path-formulae are interpreted over infinite paths in a DTMC. The syntax of PCTL is:

$$\Phi ::= \mathtt{tt} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\bowtie p}(\phi)$$

where $p \in [0,1]$ is a probability, $\bowtie \in \{<, \leqslant, >, \geqslant\}$ and ϕ is a path formula defined according to the following grammar:

$$\phi ::= \Phi \, \mathsf{U}^I \, \Phi \mid \Phi \, \mathsf{W}^I \, \Phi$$

where $I \subseteq \mathbb{N}_{\geqslant 0}$. The path formula $\Phi \, \mathsf{U}^I \, \Psi$ asserts that Ψ is satisfied within $h \in I$ transitions and that all preceding states satisfy Φ. For $I = \mathbb{N}_{\geqslant 0}$ such path-formulae are standard (*unbounded*) until-formulae, whereas in other cases, these are *bounded* until-formulae $\mathsf{U}^{\leqslant h}$, *point-interval* until-formulae $\mathsf{U}^{=h}$, *lower-bounded* until formulae $\mathsf{U}^{\geqslant h}$ and *interval* until-formulae $\mathsf{U}^{[h_l,h_u]}$, for $h, h_l, h_u \in \mathbb{N}_{\geqslant 0}$. W^I is the weak counterpart of U^I which does not require Ψ to eventually become true. In this dissertation we do not consider the next-operator. The temporal operators $\Diamond^I$ and $\Box^I$ are obtained as follows, where $\mathtt{ff} = \neg\mathtt{tt}$:

$$\Diamond^I \Phi = \mathtt{tt} \, \mathsf{U}^I \, \Phi \qquad \text{and} \qquad \Box^I \Phi = \Phi \, \mathsf{W}^I \, \mathtt{ff} \tag{2.5}$$

Example 2.16 *The formula $\mathcal{P}_{\leqslant 0.5}(a \,\mathsf{U}\, b)$ asserts that the probability of reaching a b-state via an a-path is at most 0.5, and $\mathcal{P}_{>0.001}(\Diamond^{\leqslant 50} error)$ states that the probability for a system error within 50 steps exceeds 0.001. Dually, $\mathcal{P}_{<0.999}(\Box^{\leqslant 50} \neg error)$ states that the probability for no error in the next 50 steps is less than 0.999.* ♦

Semantics over DTMCs. Let DTMC $\mathcal{D} = (S, \mathbf{P}, L)$. The semantics of PCTL is defined by a satisfaction relation, denoted $\models$, which is characterized as the least relation over the states in S (infinite paths in $\mathcal{D}$, respectively) and the state formulae (path formulae) satisfying:

$$\begin{array}{lll} s \models \mathtt{tt} & & \\ s \models a & \text{iff} & a \in L(s) \\ s \models \neg\Phi & \text{iff} & \text{not } (s \models \Phi) \\ s \models \Phi \wedge \Psi & \text{iff} & s \models \Phi \text{ and } s \models \Psi \\ s \models \mathcal{P}_{\bowtie p}(\phi) & \text{iff} & Prob(s, \phi) \bowtie p \end{array}$$

Let $Paths^{\omega}(s, \phi)$ denote the set of infinite paths that start in state s and satisfy ϕ. Formally, $Paths^{\omega}(s, \phi) = \{\rho \in Paths^{\omega}(s) \mid \rho \models \phi\}$. Then, $Prob(s, \phi) = \Pr\{\rho \mid \rho \in Paths^{\omega}(s, \phi)\}$. Let ρ be an infinite path in $\mathcal{D}$. The semantics of PCTL path formulae is defined as:

$$\rho \models \Phi \,\mathsf{U}^I\, \Psi \quad \text{iff} \quad \exists\, i \in I.\, \big(\rho[i] \models \Psi \,\wedge\, \forall\, 0 \leqslant j < i.\, \rho[j] \models \Phi\big) \tag{2.6}$$

$$\rho \models \Phi \,\mathsf{W}^I\, \Psi \quad \text{iff} \quad \text{either } \rho \models \Phi \,\mathsf{U}^I\, \Psi \text{ or } \forall\, i \leqslant \sup I.\, \rho[i] \models \Phi \tag{2.7}$$

For finite path σ, the semantics of path formulae is defined in a similar way by changing the range of variable i to $i \leqslant \min\{\sup I, |\sigma|\}$.

Definition 2.17 (Semantic equivalence) *Let $\equiv$ denote the semantic equivalence of two PCTL[1] formulae. For state formulae Φ_1, Φ_2 and path formulae ϕ_1, ϕ_2,*

$$\begin{array}{rll} \Phi_1 \equiv \Phi_2 & \text{iff} & \forall s \in S.\ s \models \Phi_1 \iff s \models \Phi_2 \\ \phi_1 \equiv \phi_2 & \text{iff} & \forall \rho \in Paths^{\omega}.\ \rho \models \phi_1 \iff \rho \models \phi_2 \end{array}$$

There is a close relationship between until and weak until. More precisely, for any state s and PCTL-formulae Φ and Ψ:

$$\mathcal{P}_{\geqslant p}(\Phi \,\mathsf{W}^I\, \Psi) \;\equiv\; \mathcal{P}_{\leqslant 1-p}\big((\Phi \wedge \neg\Psi) \,\mathsf{U}^I\, (\neg\Phi \wedge \neg\Psi)\big) \tag{2.8}$$

$$\mathcal{P}_{\geqslant p}(\Phi \,\mathsf{U}^I \Psi) \;\equiv\; \mathcal{P}_{\leqslant 1-p}\big((\Phi \wedge \neg\Psi) \,\mathsf{W}^I\, (\neg\Phi \wedge \neg\Psi)\big) \tag{2.9}$$

This relationship is used later on to show that counterexamples for formulae with probability lower-bounds can be obtained using algorithms for formulae with upper-bounds.

[1]This equivalence is also defined for PCTL* and CSL.

Let $\phi = \Phi \, \mathsf{U}^I \Psi$ be an until-formula. Let $Paths^\star(s, \phi)$ denote the set of finite paths starting in s that fulfill ϕ. For finite path σ, the relation $\models_{\min}$ denotes the minimal satisfaction of a PCTL path formula. Formally, $\sigma \models_{\min} \phi$ iff $\sigma \models \phi$ and $\sigma' \not\models \phi$ for any $\sigma' \in Pref(\sigma) \backslash \{\sigma\}$.

Example 2.18 *For the* PCTL *state formula* $\mathcal{P}_{\leqslant 0.95}(a \, \mathsf{U} \, b)$ *and the* DTMC $\mathcal{D}$ *in Fig. 2.1, let path* $\sigma = s_0{\cdot}s_2{\cdot}t_1{\cdot}t_2$. $\sigma \models a \, \mathsf{U} \, b$ *but* $\sigma \not\models_{\min} a \, \mathsf{U} \, b$. $s_0 \models \mathcal{P}_{\leqslant 0.95}(a \, \mathsf{U} \, b)$ *since* $Prob(s_0, a \, \mathsf{U} \, b) = 0.9$. ♦

Let $Paths^\star_{\min}(s, \phi) = \{\sigma \in Paths^\star(s) \mid \sigma \models_{\min} \phi\}$.

Lemma 2.19 *For any state s and until path formula ϕ, $Paths^\star_{\min}(s, \phi)$ is prefix containment free.*

Proof: By contradiction. Assume $Paths^\star_{\min}(s, \phi)$ is not prefix contain free, i.e., there exists $\sigma, \sigma' \in Paths^\star_{\min}(s, \phi)$ such that $\sigma' \in Pref(\sigma)$. Since $\sigma \in Paths^\star_{\min}(s, \phi)$, $\sigma \models_{\min} \phi$. Due to the definition of $\models_{\min}$, for $\sigma' \in Pref(\sigma)$, $\sigma' \not\models \phi$, which contradicts $\sigma' \models_{\min} \phi$. ■

Lemma 2.20 *For any state s and until path formula ϕ, $Prob(s, \phi) = \mathbb{P}\left(Paths^\star_{\min}(s, \phi)\right)$.*

Proof: Recall that for any finite path σ, it holds that $\Pr(Cyl(\sigma)) = \mathbb{P}(\sigma)$. $(\star)$

$$
\begin{aligned}
Prob(s, \phi) &\overset{\text{def.}}{=} \Pr\{\, \rho \in Paths^\omega(s) \mid \rho \models \phi \,\} \\
&= \Pr\left\{ \bigcup_{i=0}^{\infty} \{\, \rho \in Paths^\omega(s) \mid \sigma \in Pref(\rho),\ \sigma \models_{\min} \phi,\ |\sigma| = i \,\} \right\} \\
&= \Pr\left\{ \bigcup_{i=0}^{\infty} \{\, Cyl(\sigma) \mid \sigma \in Paths^\star(s),\ \sigma \models_{\min} \phi,\ |\sigma| = i \,\} \right\} \\
&= \sum_{i=0}^{\infty} \{\, \Pr(Cyl(\sigma)) \mid \sigma \in Paths^\star(s),\ \sigma \models_{\min} \phi,\ |\sigma| = i \,\} \\
&\overset{(\star)}{=} \sum_{i=0}^{\infty} \{\, \mathbb{P}(\sigma) \mid \sigma \in Paths^\star(s),\ \sigma \models_{\min} \phi,\ |\sigma| = i \,\} \\
&= \sum_{\sigma \in Paths^\star_{\min}(s,\phi)} \mathbb{P}(\sigma) \\
&\overset{\text{def.}}{=} \mathbb{P}\left(Paths^\star_{\min}(s, \phi)\right)
\end{aligned}
$$

■

Note that Lemma 2.19 and 2.20 only apply for until, but not for weak until formulae. This is because the validity of a weak until formula might be witnessed by infinite paths (e.g., $\Box\,\Phi$), which does not fit into the definition of minimal satisfaction (that requires finite paths). In Chapter 3, we explore counterexamples for formulae of the form $\mathcal{P}_{\leqslant p}(\Phi \,\mathsf{U}^I\, \Psi)$ with $p \neq 0, 1$. In Chapter 5, we will extend the results to $\mathcal{P}_{\geqslant p}(\Phi \,\mathsf{U}^I\, \Psi)$ (i.e., the weak until operators) and other models and logics.

Semantics over MDPs. The syntax of PCTL as a logic for MDPs is the same as for Markov chains. The satisfaction relation of PCTL state- and path- formulae over MDP $\mathcal{M} = (S, \mathit{Steps}, L)$ is the same as in the DTMCs except for:

$$s \models \mathcal{P}_{\bowtie p}(\phi) \quad \text{iff} \quad \text{for all schedulers } \mathfrak{G} \text{ for } \mathcal{M}.\ \mathit{Prob}^{\mathfrak{G}}(s, \phi) \bowtie p$$

where $\mathit{Prob}^{\mathfrak{G}}(s, \phi)$ is the probability of the set of ϕ-paths in DTMC $\mathcal{M}^{\mathfrak{G}}$ that start in s.

It has been proven that for finite MDPs and any PCTL path formula ϕ, there exists a finite-memory scheduler[1] that maximizes or minimizes the probabilities for ϕ. Thus for finite MDPs,

$$\Pr^{\max}(s, \phi) = \max_{\mathfrak{G}} \Pr^{\mathfrak{G}}(s, \phi) \qquad \text{and} \qquad \Pr^{\min}(s, \phi) = \min_{\mathfrak{G}} \Pr^{\mathfrak{G}}(s, \phi).$$

The PCTL model-checking problem on DTMCs [HJ94] and on MDPs [BdA95] can be solved by a recursive descent procedure over the parse tree of the state-formula to be checked. This also applies to the PCTL* and CSL (see the following subsections) model checking problems.

2.2.2 PCTL*

Like standard CTL we can extend PCTL to a richer logic PCTL* which allows the negation and conjunction of path formulae and also the combination of temporal modalities. In addition, every state formula is a path formula.

Definition 2.21 (PCTL* syntax) *The syntax of* PCTL* *is as follows:*

$$\Phi ::= \mathtt{tt} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\bowtie p}(\varphi)$$

where $p \in [0, 1]$ is a probability, $\bowtie \in \{<, \leqslant, >, \geqslant\}$ and φ is a path formula defined according to the following grammar:

$$\varphi ::= \Phi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\,\varphi \mid \varphi \,\mathsf{U}\, \varphi$$

Other boolean and temporal operators ff, $\vee$, $\Box$, $\Diamond$ and W can be derived as in CTL*. The hop-bounded until operator is omitted here for simplicity. $\mathsf{U}^{\leqslant h}$ can be recovered

[1]For unbounded-until-only path formulae, simple schedulers suffice.

by the aid of the next operator. We may interpret PCTL* state- and path- formulae on DTMCs or MDPs. For state-formulae, the satisfaction relation is the same as PCTL for DTMC and MDP, respectively. The satisfaction relation for path formulae is the same as CTL*, cf. [BK08] (Chapter 10). We won't elaborate them here.

Note that PCTL is a sub-logic of PCTL*. The semantic equivalence $\equiv$ can be defined in a similar way as in PCTL. If all the maximal state subformulae in the PCTL* path formula are replaced by fresh atomic propositions, then we obtain an LTL formula. For the syntax of LTL logic, please refer to [Pnu77]. The interpretation of LTL formulae over DTMCs and MDPs is similar to that for PCTL* path formulae. The model checking problem is to compute the probability of the set of paths in DTMC or MDP satisfying the LTL formula.

2.2.3 Continuous Stochastic Logic

Continuous Stochastic Logic (CSL) [BHHK03] is a variant of the logic originally proposed by Aziz et al. [ASSB00] and extends PCTL by path operators that reflect the real-time nature of CTMCs: in particular, a time-bounded until operator.

Syntax and Semantics. The syntax of CSL state-formulae is defined as:

$$\Phi ::= \mathtt{tt} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\bowtie p}(\phi),$$

where $p \in [0,1]$ is a probability, $\bowtie \in \{<, \leqslant, >, \geqslant\}$. For t a non-negative real number or $t = \infty$, ϕ is a path-formula defined according to the following grammar:

$$\phi ::= \Phi \,\mathsf{U}^{\leqslant t}\, \Psi \mid \Phi \,\mathsf{W}^{\leqslant t}\, \Psi.$$

The newly introduced path formula $\Phi \,\mathsf{U}^{\leqslant t}\, \Psi$ asserts that Ψ is satisfied within t time units and that at all preceding time instants Φ holds. We write a path in a CTMC $\rho \in Paths^{\omega}_{\mathcal{C}}$ satisfies $\Phi \,\mathsf{U}^{\leqslant t}\, \Psi$, denoted $\rho \models \Phi \,\mathsf{U}^{\leqslant t}\, \Psi$, iff $\rho@x \models \Psi$ for some $x \leqslant t$ and $\rho@y \models \Phi$ for all $y < x$. $\Phi \,\mathsf{W}^{\leqslant t}\, \Psi$ is the weak counterpart which does not require Ψ to eventually become true. In this dissertation we don't consider the next-operator and the steady-state operator [BHHK03]. The operators $\mathtt{ff}$, $\Diamond^{\leqslant t}$ and $\Box^{\leqslant t}$ are defined in a similar way as in PCTL, so is the relationship between $\mathsf{U}^{\leqslant t}$ and $\mathsf{W}^{\leqslant t}$.

Note that for $t = \infty$, $\Phi \,\mathsf{U}^{\leqslant t}\, \Psi$ denotes the *time-unbounded* until operator. As it can be verified on the embedded DTMC (cf. Def. 2.6, page 11), counterexamples can be obtained as for DTMCs. In the sequel, we therefore only consider $t \neq \infty$.

2.2.4 Linear Time Logic

The set of linear temporal logic (LTL) formulae over a set of atomic propositions AP is defined as follows:

Definition 2.22 (LTL syntax) *Given a set of atomic propositions* AP *which is ranged over by* $a, b, \ldots$, *the syntax of* LTL *formulae is defined by:*

$$\varphi ::= \mathrm{tt} \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi.$$

The semantics of LTL over DTMCs (resp. MDPs) is interpreted by paths in DTMCs (resp. MDPs). We omit the formal definition here and the reader are referred to [Var85][CY95b] for details. The quantitative model checking problem is to compute the probability of a set of paths that satisfy the LTL formula.

Chapter 3

Counterexample Generation

This chapter considers the generation of counterexamples in probabilistic model checking, in which it has already been established that a certain state refutes a given property. In this chapter we concentrate on PCTL formulae of the form $\mathcal{P}_{\leqslant p}(\Phi \, \mathsf{U}^{I} \, \Psi)$ for DTMCs. In this setting, typically there is no single path but rather a *set* of paths that indicates why a given property is refuted. We consider two problems that are aimed to provide useful diagnostic feedback for this violation: generating *strongest evidences* and *smallest counterexamples*. The problems are transformed into shortest paths problems in graph theory. The algorithms as well as the time complexity results are presented.

3.1 Evidences and Counterexamples

Let us first consider what a counterexample in our setting actually is. To that end, consider the PCTL formula $\mathcal{P}_{\leqslant p}(\phi)$, where $p \in (0,1)$ and let $\phi = \Phi \, \mathsf{U}^{I} \, \Psi$ for the remainder of this subsection. We have:

$$\begin{aligned} & s \not\models \mathcal{P}_{\leqslant p}(\phi) \\ \overset{\text{sem.}}{\Longleftrightarrow} \quad & \text{not } (\mathit{Prob}(s,\phi) \leqslant p) \\ \Longleftrightarrow \quad & \mathit{Prob}(s,\phi) > p \\ \overset{\text{Lem.2.20}}{\Longleftrightarrow} \quad & \mathbb{P}\left(\mathit{Paths}^{\star}_{\min}(s,\phi)\right) > p. \end{aligned}$$

So, $\mathcal{P}_{\leqslant p}(\phi)$ is refuted by state s whenever the total probability mass of all ϕ-paths that start in s exceeds p. Even for unbounded until formulae, the validity can be shown by finite paths as only paths that end in a Ψ-state contribute to $\mathit{Paths}^{\star}_{\min}(s,\phi)$. This indicates that a counterexample for $s \not\models \mathcal{P}_{\leqslant p}(\phi)$ is a *set of finite paths* starting in s and minimally satisfying ϕ. Any finite path that contributes to the violation is called an *evidence*. Note that for weak until formulae, the last step does not always hold. We will discuss this in Chapter 5.

Definition 3.1 (Evidence) *An* evidence *for violating $\mathcal{P}_{\leqslant p}(\phi)$ in state s is a path $\sigma \in Paths^{\star}_{\min}(s, \phi)$.*

The contribution of each evidence is characterized by its probability. Thus, an evidence with the largest contribution is defined:

Definition 3.2 (Strongest evidence) *An evidence $\sigma \in Paths^{\star}_{\min}(s, \phi)$ is* strongest *if $\mathbb{P}(\sigma) \geqslant \mathbb{P}(\sigma')$ for any evidence $\sigma' \in Paths^{\star}_{\min}(s, \phi)$.*

A strongest evidence always exists and may not be unique. Dually, a strongest evidence for violating $\mathcal{P}_{\leqslant p}(\phi)$ is a strongest witness for fulfilling $\mathcal{P}_{>p}(\phi)$. Evidently, a strongest evidence is not necessarily a counterexample as its probability mass may be (far) below p. We thus define a counterexample as follows:

Definition 3.3 (Counterexample) *A* counterexample *for violating $\mathcal{P}_{\leqslant p}(\phi)$ in state s is a set C of evidences for $\mathcal{P}_{\leqslant p}(\phi)$ such that $\mathbb{P}(C) > p$.*

A counterexample for state s is a subset of $Paths^{\star}_{\min}(s, \phi)$. We will, at the moment, not dwell further upon how to represent this set and assume that an abstract representation as a set suffices; a compact representation will be proposed in Section 4.2. Note that the measurability of counterexamples is ensured by the fact that $C \subseteq Paths^{\star}_{\min}(s, \phi)$ is prefix containment free (cf. page 10); hence, $\mathbb{P}(C)$ is well-defined. Let $CX_p(s, \phi)$ denote the set of all counterexamples for $\mathcal{P}_{\leqslant p}(\phi)$ in state s. For $C \in CX_p(s, \phi)$ and C's superset C': $C \subseteq C' \subseteq Paths^{\star}_{\min}(s, \phi)$, it follows that $C' \in CX_p(s, \phi)$, since $\mathbb{P}(C') \geqslant \mathbb{P}(C) > p$. That is to say, any extension of a counterexample C with paths in $Paths^{\star}_{\min}(s, \phi)$ is a counterexample. This motivates the notion of minimality.

Definition 3.4 (Minimal counterexample) *$C \in CX_p(s, \phi)$ is a* minimal counterexample *if $|C| \leqslant |C'|$, for any $C' \in CX_p(s, \phi)$.*

As in conventional model checking, we are not interested in generating arbitrary counterexamples, but those that are easy to comprehend, and provide a clear evidence of the refutation of the formula. So, akin to shortest counterexamples for linear-time logics, we consider the notion of a smallest counterexample. Such counterexamples are required to be succinct, i.e., minimal, allowing easier analysis of the cause of refutation, and most distinctive, i.e., their probability should exceed p more than all other minimal counterexamples. This motivates the following definition:

Definition 3.5 (Smallest counterexample) *$C \in CX_p(s, \phi)$ is a* smallest counterexample *if it is minimal and $\mathbb{P}(C) \geqslant \mathbb{P}(C')$ for any minimal counterexample $C' \in CX_p(s, \phi)$.*

The intuition is that a smallest counterexample is the one that deviates most from the required probability bound given that it contains a minimal number of paths. Thus, there does not exist an equally sized counterexample that deviates more from p. Strongest evidences, minimal or smallest counterexamples may not be unique, as different paths may have equal probability. As a result, not every strongest evidence is contained in a minimal (or smallest) counterexample. Whereas minimal counterexamples may not contain any strongest evidence, any smallest counterexample contains at least one strongest evidence. Using standard mathematical results we obtain:

Lemma 3.6 *A finite counterexample for $s \not\models \mathcal{P}_{\leqslant p}(\phi)$ exists.*

Proof: By contradiction. Assume there are only infinite counterexamples for $s \not\models \mathcal{P}_{\leqslant p}(\phi)$. Let $C = \{\sigma_1, \sigma_2, \ldots\}$ be one such counterexample, i.e., $\sum_{i=1}^{\infty} \mathbb{P}(\sigma_i) > p$. Since $\mathbb{P}(\sigma_i)$ is positive for any i, it follows by standard arguments that

$$\underbrace{\sum_{i=1}^{\infty} \mathbb{P}(\sigma_i)}_{=d} \;=\; \lim_{j \to \infty} \underbrace{\sum_{i=1}^{j} \mathbb{P}(\sigma_i)}_{d_j}.$$

By the definition of limit, this means that

$$\forall \epsilon > 0.\ \exists N_\epsilon \in \mathbb{N}.\ \forall n \geqslant N_\epsilon.\ |d_n - d| \;<\; \epsilon \tag{3.1}$$

Take ϵ such that $0 < \epsilon < d - p$. By (3.1), for some $n \geqslant N_\epsilon$, $|d_n - d| < d - p$, i.e., $d_n > p$. But then, the finite set $C' = \{\sigma_1, \ldots, \sigma_n\}$ is also a counterexample as $\mathbb{P}(C') > p$. Contradiction. ■

This lemma indicates that a *smallest* counterexample for $s \not\models \mathcal{P}_{\leqslant p}(\phi)$ is finite.

Remark 3.7 (Finiteness) *For until-formulae with* strict *upper bounds on the probability, i.e., $\mathcal{P}_{<p}(\phi)$, a finite counterexample may not exist. This occurs when, e.g., the only counterexample is an infinite set C of finite paths with $\mathbb{P}(C) = p$. The limit of the sum of the path probabilities (obeying a geometric distribution) equals p, but infinitely many paths are needed to reach p. For instance, consider the* DTMC *in Fig. 3.1. The violation of $\mathcal{P}_{<\frac{1}{2}}(\Diamond\, a)$ in state s can only be shown by an infinite set of paths, viz. all paths that traverse the self-loop at state s arbitrarily often and reach t.*

Example 3.8 *Consider the* DTMC *in Fig. 2.1 (page 10), for which s violates $\mathcal{P}_{\leqslant\frac{1}{2}}(a \,\mathsf{U}\, b)$. Evidences are, amongst others, $\sigma_1 = s_0{\cdot}s_1{\cdot}t_1$, $\sigma_2 = s_0{\cdot}s_1{\cdot}s_2{\cdot}t_1$, $\sigma_3 = s_0{\cdot}s_2{\cdot}t_1$, $\sigma_4 = s_0{\cdot}s_1{\cdot}s_2{\cdot}t_2$, and $\sigma_5 = s_0{\cdot}s_2{\cdot}t_2$. Their respective probabilities are* 0.2, 0.2, 0.15, 0.12 *and* 0.09*. $\sigma = s_0{\cdot}s_1{\cdot}t_1{\cdot}t_2$ is not an evidence as it contains a proper prefix, $s_0{\cdot}s_1{\cdot}t_1$,*

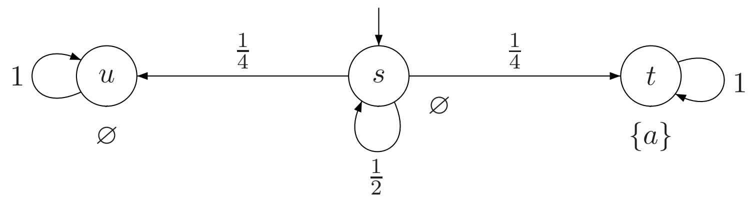


Figure 3.1: A DTMC with infinite counterexample for $\mathcal{P}_{<\frac{1}{2}}(\Diamond\, a)$

that satisfies $a \,\mathsf{U}\, b$. The strongest evidences are σ_1 and σ_2, which are not unique. The set $C_1 = \{\sigma_1, \ldots, \sigma_5\}$ with $\mathbb{P}(C_1) = 0.76$ is a counterexample, but not a minimal one, as the removal of either σ_1 or σ_2 also yields a counterexample. $C_2 = \{\sigma_1, \sigma_2, \sigma_4\}$ is a minimal but not a smallest counterexample, as $C_3 = \{\sigma_1, \sigma_2, \sigma_3\}$ is minimal too with $\mathbb{P}(C_3) = 0.56 > 0.52 = \mathbb{P}(C_2)$. C_3 is a smallest counterexample. ♦

In the remainder of the chapter, we consider the *strongest evidence* problem (SE), that determines the strongest evidence for violation $s \not\models \mathcal{P}_{\leqslant p}(\phi)$. Subsequently, we consider the corresponding *smallest counterexample* problem (SC).

3.2 Model Transformation

Prior to finding strongest evidences or smallest counterexamples, we modify the DTMC and turn it into a *weighted digraph*. This enables us, as we will show, to exploit well-known efficient graph algorithms to the SE and SC problems. Let $Sat(\Phi) = \{s \in S \mid s \models \Phi\}$ for any Φ. Due to the bottom-up traversal of the model-checking algorithm over the formula $\phi = \Phi \,\mathsf{U}^{\leqslant h}\, \Psi$, we may assume that $Sat(\Phi)$ and $Sat(\Psi)$ are known.

3.2.1 Step 1: Adapting the DTMC

We first define two PCTL state-formula-driven transformation on DTMCs.

Definition 3.9 *For* DTMC $\mathcal{D} = (S, \mathbf{P}, L)$ *and* PCTL *state formula* Ψ,

- *let $\mathcal{D}[\Psi]$ result from $\mathcal{D}$ by removing all the outgoing transitions from all the Ψ-states in $\mathcal{D}$, i.e., $\mathcal{D}[\Psi] = (S, \tilde{\mathbf{P}}, L)$, where $\tilde{\mathbf{P}}(s, s') = \mathbf{P}(s, s')$ for any $s' \in S$, if $s \notin Sat(\Psi)$ and 0 otherwise;*
- *let $\mathcal{D}\langle t_\Psi \rangle$ result from $\mathcal{D}$ by adding an extra state $t \notin S$ such that all outgoing transitions from a Ψ-state are replaced by a transition to t with probability 1, i.e., $\mathcal{D}\langle t_\Psi \rangle = (S \cup \{t\}, \hat{\mathbf{P}}, \hat{L})$, where*

$$\hat{\mathbf{P}}(s,t) = \begin{cases} 1 & \text{if } s \in Sat(\Psi) \text{ or } s = t \\ 0 & \text{otherwise} \end{cases}$$

and otherwise $\hat{\mathbf{P}}(s,s') = \mathbf{P}(s,s')$. $\hat{L}(t) = \{at_t\}$, *where* $at_t \notin L(s')$ *for any* $s' \in S$, *i.e.,* at_t *uniquely identifies being at state* t *and* $\hat{L}(s) = L(s)$ *otherwise.*

To differentiate an absorbing state s where $\mathbf{P}(s,s) = 1$, recall that a state is "sinking" or a "sink" if $\mathbf{P}(s,s') = 0$ for any $s' \in S$ (page 9). Here $\mathcal{D}[\Phi]$ is an FPS (cf. page 9). Later we abuse the name DTMC also for FPS, because by adding an absorbing state an FPS can be easily transformed into a DTMC. All the definitions and computation for DTMCs can be applied to FPSs. Note that $\mathcal{D}[\Phi][\Psi] = \mathcal{D}[\Phi \vee \Psi]$ and in $\mathcal{D}\langle t_\Psi \rangle$ state t can only be reached via a Ψ-state.

We adapt the DTMC $\mathcal{D}$ into $\mathcal{D}[\neg\Phi \wedge \neg\Psi]\langle t_\Psi \rangle$ (denoted $\mathcal{D}'$ in the following). The $(\neg\Phi \wedge \neg\Psi)$-states are made sinking as they will refute the path formula $\Phi \, \mathsf{U}^{\leqslant h} \, \Psi$. This sub-step is standard when model checking DTMC against PCTL. All the $(\neg\Phi \wedge \neg\Psi)$-states could be collapsed into a single state, but this is not further explored here. State t is added to identify the goal states Ψ. The time complexity of this transformation is $\mathcal{O}(n)$ where $n = |S|$. The following lemma shows that the validity of $\Phi \, \mathsf{U}^{\leqslant h} \, \Psi$ is not affected by this amendment of the DTMC, and that computing the probability of $\Phi \, \mathsf{U}^{\leqslant h} \, \Psi$ can be reduced to computing the reachability probability (with only one goal state t).

Lemma 3.10 $\sigma \models^{\mathcal{D}}_{\min} \Phi \, \mathsf{U}^{\leqslant h} \, \Psi$ iff $\sigma{\cdot}t \models^{\mathcal{D}'}_{\min} \Diamond^{\leqslant h+1} at_t$, *remark that* $\mathbb{P}^{\mathcal{D}}(\sigma) = \mathbb{P}^{\mathcal{D}'}(\sigma{\cdot}t)$.

Proof: We prove it from two directions:

- $\Longrightarrow$: If $\sigma \models^{\mathcal{D}}_{\min} \Phi \, \mathsf{U}^{\leqslant h} \, \Psi$ and $\sigma' = \sigma{\cdot}t$, then due to the construction, σ' is of the following form:

$$\sigma' = \underbrace{\underbrace{s_0 \cdots s_{\ell-1}}_{\models \Phi \wedge \neg at_t} \cdot \underbrace{s_\ell}_{\models \Psi \wedge \neg at_t}}_{\sigma} \cdot \underbrace{t}_{\models at_t},$$

where $0 \leqslant \ell = |\sigma| \leqslant h$ and $1 \leqslant \ell + 1 = |\sigma'| \leqslant h+1$. Thus $\sigma' \models^{\mathcal{D}'}_{\min} \Diamond^{[1,h+1]} at_t$. Note that since at_t cannot be reached at position 0, the hop constraint $[1, h+1]$ can be relaxed to $\leqslant h+1$. Thus, (i) $\sigma' \models^{\mathcal{D}'}_{\min} \Diamond^{\leqslant h+1} at_t$ holds. For (ii),

$$\mathbb{P}(\sigma') = \prod_{0 \leqslant i < \ell} \mathbf{P}'(s_i, s_{i+1}) \cdot \mathbf{P}'(s_\ell, t) = \prod_{0 \leqslant i < \ell} \mathbf{P}(s_i, s_{i+1}) \cdot 1 = \mathbb{P}(\sigma) \cdot 1 = \mathbb{P}(\sigma).$$

- $\Longleftarrow$: Similar. ■

Proposition 3.11 (Correctness of Step 1) *For any* DTMC $\mathcal{D}$, *state* s, *and* PCTL *formula* $\Phi \, \mathsf{U}^{\leqslant h} \, \Psi$ $(h \in \mathbb{N}_{\geqslant 0} \cup \{\infty\})$,

$$Prob^{\mathcal{D}}(s, \Phi \, \mathsf{U}^{\leqslant h} \, \Psi) = Prob^{\mathcal{D}'}(s, \Diamond^{\leqslant h+1} at_t).$$

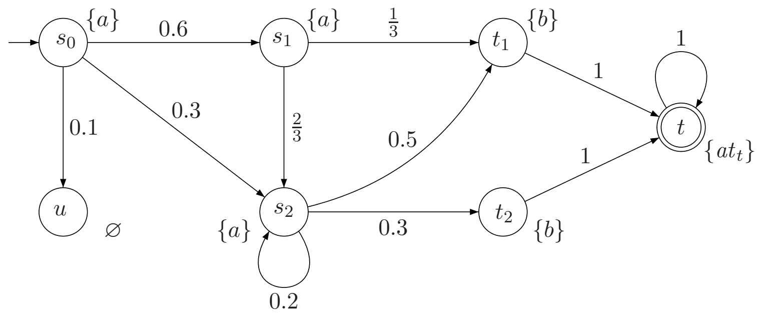


Figure 3.2: Model transformation (Step 1): from $\mathcal{D}$ to $\mathcal{D}[\neg a \wedge \neg b]\langle t_b\rangle$

Proof: Due to Lemma 3.10, it holds that the paths in $Paths^{\star}_{\min}\left(s, \Phi \,\mathsf{U}^{\leqslant h}\, \Psi\right)$ and $Paths^{\star}_{\min}\left(s, \Diamond^{\leqslant h+1} at_t\right)$ have a one-to-one correspondence and that

$$\mathbb{P}^{\mathcal{D}}\left(Paths^{\star}_{\min}\left(s, \Phi \,\mathsf{U}^{\leqslant h}\, \Psi\right)\right) = \mathbb{P}^{\mathcal{D}'}\left(Paths^{\star}_{\min}\left(s, \Diamond^{\leqslant h+1} at_t\right)\right).$$

And due to Lemma 2.20, the following holds

$$Prob^{\mathcal{D}}(s, \Phi \,\mathsf{U}^{\leqslant h}\, \Psi) = Prob^{\mathcal{D}'}(s, \Diamond^{\leqslant h+1} at_t). \qquad \blacksquare$$

Example 3.12 *Given path formula $a \,\mathsf{U}\, b$, applying the above transformation to the* DTMC $\mathcal{D}$ *in Fig.* 2.1 (*page* 10) *yields the* DTMC $\mathcal{D}' = \mathcal{D}[\neg a \wedge \neg b]\langle t_b\rangle$ *illustrated in Fig.* 3.2. *The* $(\neg a \wedge \neg b)$*-state u is made sinking and both b-states* (*i.e.*, t_1 *and* t_2) *are equipped with a transition with probability* 1 *to the new absorbing state t* (*indicated by a double circle*). ♦

3.2.2 Step 2: Conversion into a Weighted Digraph

As a second preprocessing step, the DTMC obtained in the first step is transformed into a weighted digraph, i.e., a triple $\mathcal{G} = (V, E, w)$ where V is a finite set of vertices, $E \subseteq V \times V$ is a set of edges, and $w : E \to \mathbb{R}_{\geqslant 0}$ is a weight function.

Definition 3.13 (Weighted digraph of a DTMC) *For* DTMC $\mathcal{D} = (S, \mathbf{P}, L)$, *the weighted digraph of* $\mathcal{D}$ *is* $\mathcal{G}_{\mathcal{D}} = (V, E, w)$ *where* $V = S$, $(v, v') \in E$ *iff* $\mathbf{P}(v, v') > 0$, *and* $w(v, v') = -\log \mathbf{P}(v, v')$.

The edge weights are obtained by taking the negation of the logarithm of the corresponding transition probabilities. Note that $w(s, s') \in [0, \infty)$ if $\mathbf{P}(s, s') > 0$. Thus, we indeed obtain a digraph with nonnegative weights. This transformation can be done in $\mathcal{O}(m \log \frac{1}{z})$, where m is the number of non-zero elements in $\mathbf{P}$ and $z = \min_{s,s' \in S} \mathbf{P}(s, s')$ is the smallest probability appearing in the DTMC. We often omit the self-loop on vertex t in $\mathcal{G}_{\mathcal{D}}$, as it has weight 0.

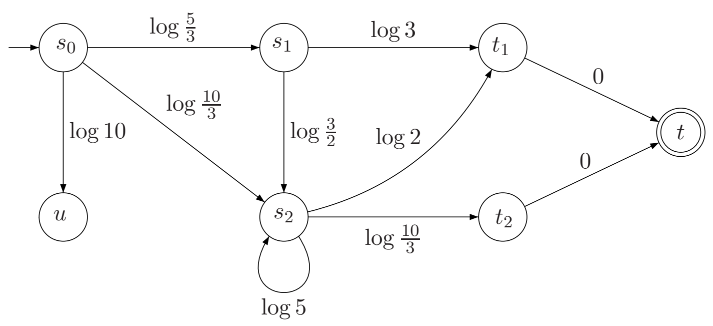


Figure 3.3: Model transformation (Step 2): from $\mathcal{D}[\neg a \wedge \neg b]\langle t_b \rangle$ to $\mathcal{G}_{\mathcal{D}[\neg a \wedge \neg b]\langle t_b \rangle}$

Example 3.14 (Continuing Example 3.12) *Applying this transformation to* DTMC $\mathcal{D}'$ *in Fig. 3.2 yields the weighted digraph in Fig. 3.3. Note the fact that* $-\log \mathbf{P}(v, v') = \log \mathbf{P}(v, v')^{-1}$. *Thus,* $-\log \mathbf{P}(s_0, s_1) = -\log \frac{3}{5} = \log(\frac{3}{5})^{-1} = \log \frac{5}{3}$. ♦

A path σ from s to t in the digraph $\mathcal{G}$ is a sequence $\sigma = v_0{\cdot}v_1 \cdots v_j \in V^+$, where $v_0 = s, v_j = t$ and $(v_i, v_{i+1}) \in E$, for $0 \leqslant i < |\sigma|$ ($|\sigma|$ denotes the length of σ). The *weight* of finite path $\sigma = v_0{\cdot}v_1 \cdots v_j$ in graph $\mathcal{G}$ is $w(\sigma) = \sum_{i=0}^{j-1} w(v_i, v_{i+1})$. Path weights in $\mathcal{G}_{\mathcal{D}}$ and path probabilities in DTMC $\mathcal{D}$ are related as follows:

$$\begin{aligned} w(\sigma) &= \sum_{i=0}^{j-1} w(v_i, v_{i+1}) = \sum_{i=0}^{j-1} -\log \mathbf{P}(v_i, v_{i+1}) \\ &= -\sum_{i=0}^{j-1} \log \mathbf{P}(v_i, v_{i+1}) = -\log \prod_{i=0}^{j-1} \mathbf{P}(v_i, v_{i+1}) \\ &= -\log \mathbb{P}(\sigma) \end{aligned}$$

Now the multiplication of probabilities in $\mathcal{D}$ corresponds to addition of weights in $\mathcal{G}_{\mathcal{D}}$ and the next two lemmata directly follow:

Lemma 3.15 *Let σ and σ' be finite paths in* DTMC $\mathcal{D}$ *and its graph* $\mathcal{G}_{\mathcal{D}}$. *Then:* $\mathbb{P}(\sigma') \geqslant \mathbb{P}(\sigma)$ *iff* $w(\sigma') \leqslant w(\sigma)$.

Proof: Since $w(\sigma) = -\log \mathbb{P}(\sigma)$, it holds that

$$\begin{aligned} & w(\sigma') \leqslant w(\sigma) \quad \text{iff} \quad -\log \mathbb{P}(\sigma') \leqslant -\log \mathbb{P}(\sigma) \\ \text{iff} \quad & \log \mathbb{P}(\sigma') \geqslant \log \mathbb{P}(\sigma) \quad \text{iff} \quad \mathbb{P}(\sigma') \geqslant \mathbb{P}(\sigma). \end{aligned}$$ ■

This result implies that the most probable path between two states in DTMC $\mathcal{D}$ equals the shortest path (i.e., the path with the least weight) between these states in the weighted digraph $\mathcal{G}_{\mathcal{D}}$.

Let $Paths_{\mathcal{G}}(s,t)$ denote the set of finite paths in $\mathcal{G}$ that start in s and end in t. The paths in $Paths_{\mathcal{G}}(s,t)$ form a *total order* under $\leqslant$ wrt. the path weight function w. In other words, paths in $Paths_{\mathcal{G}}(s,t)$ can be ordered as $\sigma^{(1)}$, $\sigma^{(2)}$,... such that for any $0 \leqslant i < j$, $w(\sigma^{(i)}) \leqslant w(\sigma^{(j)})$. A *$k$-th shortest path* is $\sigma^{(k)}$. Notice that such a total order is not necessarily unique (as paths may have equal weights), so a k-th shortest path may also not be unique. However, for a fixed total order, a k-th shortest path is unique. The k-th most probable path is defined in a similar way. Due to the non-uniqueness, we aim at deriving *arbitrary* one smallest counterexample and won't explore further the priority between different smallest counterexamples.

Lemma 3.15 can be generalized to paths of a certain length (or, equivalently number of *hops*), and to the second, third, etc. most probable paths. This yields:

Lemma 3.16 *For any path σ from s to t in* DTMC $\mathcal{D}$ *with $|\sigma| = h$ and $k \in \mathbb{N}_{>0}$, it holds that σ is a k-th most probable path of h hops in $\mathcal{D}$ iff σ is a k-th shortest path of h hops in $\mathcal{G}_{\mathcal{D}}$.*

Proof: Let $Paths^{\leqslant h}(s,t)$ be the set of paths in $\mathcal{D}$ and $\mathcal{G}_{\mathcal{D}}$ that start in s and end in t with at most h hops.

- $\Longleftarrow$: Assume $\sigma^{(1)}, \ldots, \sigma^{(k)}$ are the 1-st, $\ldots$, k-th shortest paths in $Paths^{\leqslant h}(s,t)$. Applying Lemma 3.15, the following holds:

$$w(\sigma^{(1)}) \leqslant w(\sigma^{(2)}) \leqslant \cdots \leqslant w(\sigma^{(k)}) \text{ iff } \mathbb{P}(\sigma^{(1)}) \geqslant \mathbb{P}(\sigma^{(2)}) \geqslant \cdots \geqslant \mathbb{P}(\sigma^{(k)}).$$

  Thus, $\sigma^{(k)}$ is the k-th most probable paths in $Paths^{\leqslant h}(s,t)$.
- $\Longrightarrow$: Similar. ■

This lemma provides the basis for the algorithms in the following sections.

3.3 Finding Strongest Evidences

3.3.1 Unbounded Until — U

Based on Lemma 3.16 with $k = 1$ and $h = \infty$, we consider the well-known shortest path problem. Recall that $Paths^{\mathcal{G}}(s,t)$ is the set of finite paths in $\mathcal{G}$ between s and t.

Definition 3.17 (SP problem) *Given a weighted digraph $\mathcal{G} = (V, E, w)$ and $s, t \in V$, the* shortest path *problem is to determine a path $\sigma \in Paths^{\mathcal{G}}(s,t)$ such that $w(\sigma) \leqslant w(\sigma')$ for any path $\sigma' \in Paths^{\mathcal{G}}(s,t)$.*

From Lemma 3.16 together with the transformation of a DTMC into a weighted digraph, it follows that there is a polynomial reduction from the SE problem for unbounded until to the SP problem. As the SP problem is in PTIME, it follows:

Theorem 3.18 *The SE problem for unbounded until is in* PTIME.

Various efficient algorithms [Dij59][Bel58][CLRS01a] exist for the SP problem, e.g., when using Dijkstra's algorithm, the SE problem for unbounded until can be solved in time $\mathcal{O}(m + n \log n)$, where $m = |E|$ and $n = |V|$, provided appropriate data structures such as Fibonacci heaps are used.

3.3.2 Bounded Until — $\mathsf{U}^{\leqslant h}$

Lemma 3.16 for $k = 1$ and $h \in \mathbb{N}_{\geqslant 0}$ suggests to consider the hop-constrained SP problem. Recall that $Paths^{\mathcal{G}}_{\leqslant h}(s,t)$ is the set of paths in $Paths^{\mathcal{G}}(s,t)$ with at most h hops.

Definition 3.19 (HSP problem) *Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$ and $h \in \mathbb{N}_{\geqslant 0}$, the* hop-constrained SP *problem is to determine a path $\sigma \in Paths^{\mathcal{G}}_{\leqslant h}(s,t)$ such that $w(\sigma) \leqslant w(\sigma')$ for any path $\sigma' \in Paths^{\mathcal{G}}_{\leqslant h}(s,t)$.*

The HSP problem is a special case of the (*resource*) *constrained shortest path* (CSP) problem [MZ00][AMO93], where the only constraint is the hop count. Besides the weight w on each edge, it may consume other resources $w_1, \ldots, w_c$ and the sum of each resource should be bounded by the resource constraints $\lambda_1, \ldots, \lambda_c$, where c is the number of resources. Weighted digraphs with multiple resources are obtained by allowing multiple weights to edges.

Definition 3.20 (Multi-weighted digraph) $\mathcal{G} = (V, E, \{w\} \cup \{w_1, \ldots, w_c\})$ *is a* multi-weighted digraph *where V and E are set of vertices and edges, respectively; w is the main weight function and w_i is the (secondary) weight function for resource i $(1 \leqslant i \leqslant c)$.*

Edge $e \in E$ uses $w_i(e) \geqslant 0$ units of resource i. The main weight (resp. weight i) of a path σ is $w(\sigma) = \sum_{e \in \sigma} w(e)$ (resp. $w_i(\sigma) = \sum_{e \in \sigma} w_i(e)$).

Definition 3.21 (CSP problem) *Let $\mathcal{G} = (V, E, \{w\} \cup \{w_1, \ldots, w_c\})$ be a multi-weighted digraph with $s, t \in V$ and λ_i the resource constraints, for $1 \leqslant i \leqslant c$. The* (resource) constrained SP *problem is to determine a shortest path $\sigma \in Paths^{\mathcal{G}}(s,t)$ wrt. the weight $w(\sigma)$ such that $w_i(\sigma) \leqslant \lambda_i$ for $1 \leqslant i \leqslant c$.*

Intuitively, the CSP problem is to optimize the main objective function $w(\sigma)$ and at the same time guarantee that the secondary objective functions $w_i(\sigma)$ meet the upper bounds λ_i. The CSP problem is NP-complete, even for a single resource constraint (i.e., $c = 1$) [AMO93]. However, if each edge uses the same constant unit of that resource (such as the hop count), the CSP problem can be solved in polynomial time, cf. [GJ79], problem [ND30].

Theorem 3.22 *The SE problem for bounded until is in* PTIME.

For $h \geqslant n-1$, it is possible to use Dijkstra's SP algorithm (as for unbounded until), as a shortest path does not contain cycles. If $h < n-1$, however, Dijkstra's algorithm does not guarantee to obtain a shortest path of at most h hops. We, therefore, adopt the Bellman-Ford (BF) algorithm [Bel58][CLRS01a] which fits well to our problem as it proceeds by increasing hop count. It can be readily modified to generate a shortest path within a given hop count. In the remainder of the chapter, this algorithm is generalized for computing smallest counterexamples. The BF algorithm is based on a set of recursive equations; we extend it with the hop count h. We call this algorithm $\mathrm{BF}^{\leqslant h}$ to differentiate $\mathrm{BF}^{=h}$ introduced later in Section 3.3.3. For $v \in V$, let $\pi_{\leqslant h}(s,v)$ denote the shortest path from s to v of at most h hops (if it exists). Then:

$$\pi_{\leqslant h}(s,v) = \begin{cases} s & \text{if } v = s \wedge h \geqslant 0 \qquad (3.2) \\ \perp & \text{if } v \neq s \wedge h = 0 \qquad (3.3) \\ \arg\min\limits_{u \in S} \Big\{ w\Big(\pi_{\leqslant h-1}(s,u)\cdot v\Big) \mid (u,v) \in E \Big\} & \text{otherwise} \qquad (3.4) \end{cases}$$

where $\perp$ denotes the non-existence of such path. Note that $\perp\cdot v = v$ for any v, $\{\perp\} = \varnothing$, $\arg\min \varnothing = \perp$, $\mathbb{P}(\perp) = 0$, and $w(\perp) = \infty$. The first two clauses are self-explanatory. The last clause states that $\pi_{\leqslant h}(s,v)$ consists of the shortest path to v's direct predecessor u, i.e., $\pi_{\leqslant h-1}(s,u)$, extended with the edge (u,v). Note that $\min_{u\in S}\{w\big(\pi_{\leqslant h-1}(s,u)\cdot v\big) \mid (u,v) \in E\}$ is the weight of a shortest path; by means of arg, such shortest path is obtained. It follows (cf. [Law76]) that $\pi_{\leqslant h}(s,v)$ characterizes the shortest path from s to v in at most h hops, and can be obtained in time $\mathcal{O}(hm)$. As $h < n-1$, this is indeed in PTIME. Recall that for $h \geqslant n-1$, Dijkstra's algorithm has a favorable time complexity.

Example 3.23 *To illustrate the* $\mathrm{BF}^{\leqslant h}$ *algorithm, we compute the shortest path* $\pi_{\leqslant 4}(s_0,t)$ *in the digraph in Fig. 3.3. The invocation relation of function* π *is shown in Fig. 3.4. Note that we use probabilities instead of weights in the graph for clarity. The transition probabilities are labeled on the edges. In order to compute* $\pi_{\leqslant 4}(s_0,t)$*, the two predecessors of* t *are considered such that* $\pi_{\leqslant 3}(s_0,t_2)$ *and* $\pi_{\leqslant 3}(s_0,t_1)$ *are invoked and the corresponding probabilities* $\mathbb{P}(\pi_{\leqslant 3}(s_0,t_2))\cdot 1$ *and* $\mathbb{P}(\pi_{\leqslant 3}(s_0,t_1))\cdot 1$ *are stored in the boxes to its right. Those values are known only when the results are returned from the next level. Continuing the invocation, in order to compute* $\pi_{\leqslant 3}(s_0,t_1)$*, the two predecessors of* t_1 *are considered such that* $\pi_{\leqslant 2}(s_0,s_1)$ *and* $\pi_{\leqslant 2}(s_0,s_2)$ *are invoked. The invocation won't terminate till the end of the computation chains derived by* (3.2)(3.3)*. Note that* $\pi_{\leqslant 2}(s_0,s_2)$*,* $\pi_{\leqslant 1}(s_0,s_0)$ *and* $\pi_{\leqslant 0}(s_0,s_0)$ *are invoked more than once, but are computed only once using dynamic programming. To choose the minimal weight (maximal probability) of a path we select the minimal value from the boxes attached to the path,*

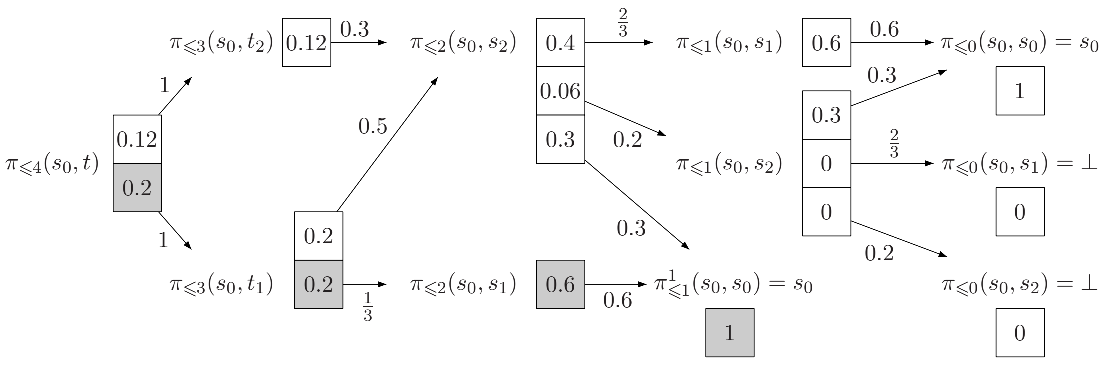

$\pi_{\leqslant 4}(s_0,t)$
0.12
0.2
1
1
$\pi_{\leqslant 3}(s_0,t_2)$
0.12
0.3
$\pi_{\leqslant 3}(s_0,t_1)$
0.2
0.2
0.5
$\frac{1}{3}$
$\pi_{\leqslant 2}(s_0,s_2)$
0.4
0.06
0.3
$\pi_{\leqslant 2}(s_0,s_1)$
0.6
$\frac{2}{3}$
0.2
0.3
0.6
$\pi_{\leqslant 1}(s_0,s_1)$
$\pi_{\leqslant 1}(s_0,s_2)$
$\pi^1_{\leqslant 1}(s_0,s_0)=s_0$
1
0.6
0.3
0
0
0.6
0.3
$\frac{2}{3}$
0.2
$\pi_{\leqslant 0}(s_0,s_0)=s_0$
1
$\pi_{\leqslant 0}(s_0,s_1)=\perp$
0
$\pi_{\leqslant 0}(s_0,s_2)=\perp$
0

Figure 3.4: An example run of the Bellman-Ford algorithm

which is labeled gray. Note that the two candidate paths for $\pi_{\leqslant 3}(s_0,t_1)$ *have the same minimal value. We choose one nondeterministically. The shortest path is obtained by connecting the gray-labeled vertices from backwards. In this example, it is* $s_0{\cdot}s_1{\cdot}t_1{\cdot}t$. ♦

Exploiting the Viterbi Algorithm. An alternative to the $\mathrm{BF}^{\leqslant h}$ algorithm is to adopt the *Viterbi algorithm* [For73][Vit67][VTdlH+05]. In fact, to apply this algorithm the transformation into a weighted digraph is not needed. The Viterbi algorithm is based on dynamic programming and aims to find the most likely sequence of hidden states (i.e., a finite path) that result in a sequence of observed events (a trace). It is used in the context of hidden Markov models, which are used in, e.g., speech recognition and bioinformatics.

Let DTMC $\mathcal{D}$ be obtained after the first step described in Section 3.2.1, and suppose that $L(s)$ is extended with all subformulae of the formula under consideration that hold in s. Note that these labels are known due to the recursive descent nature of the PCTL model-checking algorithm. Let $tr(\sigma)$ denote the projection of a path $\sigma = s_0{\cdot}s_1 \cdots s_h$ on its trace, i.e., $tr(\sigma) = L(s_0)L(s_1)\cdots L(s_h)$. Recall that $\sigma[..i]$ denotes the prefix of path σ truncated at length i (thus ending in s_i), thus $tr(\sigma[..i]) = L(s_0)L(s_1)\cdots L(s_i)$. $\gamma[..i]$ denotes the prefix of trace γ with length $i{+}1$. Note that the length of a trace is one more than the length of the corresponding path, i.e., $|tr(\sigma)| = |\sigma|+1$. Let $\varrho(\gamma,i,v)$ denote the probability of the most probable path $\sigma[..i]$ whose trace equals $\gamma[..i]$ and reaches state v. Formally,

$$\varrho(\gamma,i,v) = \max_{tr(\sigma[..i])=\gamma[..i]\wedge\sigma\in Paths^\star(s_0)} \prod_{j=0}^{i-1} \mathbf{P}(s_j,s_{j+1})\cdot\mathbf{1}_v(s_i),$$

where $\mathbf{1}_v(s_i)$ is the characteristic function of v, i.e., $\mathbf{1}_v(s_i)=1$ iff $s_i = v$. The Viterbi algorithm provides an algorithmic solution to compute $\varrho(\gamma, i, v)$:

$$\varrho(\gamma, i, v) = \begin{cases} 1 & \text{if } s = v \text{ and } i = 0 \\ 0 & \text{if } s \neq v \text{ and } i = 0 \\ \max_{u \in S}\Big\{\varrho(\gamma, i{-}1, u) \cdot \mathbf{P}(u, v)\Big\} & \text{otherwise} \end{cases}$$

By computing $\varrho(\Phi^h\Psi, h, s_h)$, the Viterbi algorithm determines the most probable path $\sigma = s_0{\cdot}s_1 \cdots s_h$ that generates the trace $\gamma = L'(s_0)L'(s_1)\cdots L'(s_h) = \Phi^h\Psi$ with length $h+1$. Here, $L'(s) = L(s) \cap \{\Phi, \Psi\}$, i.e., L' is the labeling restricted to the subformulae Φ and Ψ. For the SE problem for bounded until, the trace of the most probable hop-constrained path from s to t is among $\{\Psi at_t, \Phi\Psi at_t, \ldots, \Phi^h\Psi at_t\}$. The self-loop at vertex t with probability one ensures that all these paths have length $h+1$ while not changing their probabilities. For instance, the path with trace $\Phi^i\Psi at_t$ can be extended so that the trace becomes $\Phi^i\Psi {at_t}^{h+1-i}$, where $i \leqslant h$. Since the DTMC is already transformed as in Step 1 (cf. page 26), we can obtain the most probable path for $\Phi \, \mathsf{U}^{\leqslant h} \, \Psi$ by computing $\varrho\big((\Phi \vee \Psi \vee at_t)^{h+1} at_t, h+1, t\big)$ using the Viterbi algorithm. The time complexity is $\mathcal{O}(hm)$, as for the $\mathrm{BF}^{\leqslant h}$ algorithm.

Viterbi algorithm provides more possibilities to implement the algorithm of computing the strongest evidences. However, it has certain restrictions. For instance, it can only deal with until formulae that have an upper-bound, i.e., U and $\mathsf{U}^{\geqslant h}$ cannot be handled by Viterbi algorithm (see Table 3.1, page 40). Another drawback is that when some of the transition probabilities are very small or when the valid paths are very long, the rounding-off error of doing successive multiplications may be much higher than first having log and doing addition and finally transforming the log back to a probability. In this sense, our algorithm may give a more precise result than the Viterbi algorithm.

3.3.3 Point Interval Until — $\mathsf{U}^{=h}$

We now deal with the until operator with point intervals, i.e., $\mathsf{U}^{=h}$. Together with the previous two cases, it forms the basis for the more general until form $\mathsf{U}^{[h_l,h_u]}$ with $h_u > h_l > 0$ in Section 3.3.4.

As the bounded until $\Phi \, \mathsf{U}^{\leqslant h} \, \Psi$ allows a path to reach a Ψ-state in less than h hops, this justifies making all Ψ-states "absorbing" (connecting to t). However, in the $\Phi \, \mathsf{U}^{=h} \, \Psi$ case, since the hop count is exact some $(\Phi \wedge \Psi)$-states might be counted as a Φ-state when the hop number is less than h and as a Ψ-state when it is exactly h. Let us explain this by an example.

Example 3.24 *Consider the* DTMC $\mathcal{D}$ *in Fig. 3.5(a). Note that the missing probabilities from each state are omitted. The states labeled with different formulae are labeled*

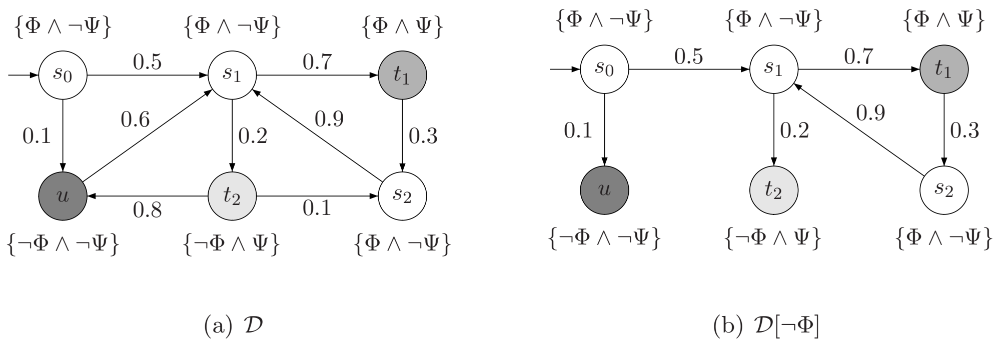

$\{\Phi \wedge \neg\Psi\}$
s_0
0.5
s_1
0.7
t_1
0.1
0.6
0.2
0.9
0.3
u
t_2
s_2
0.8
0.1
$\{\neg\Phi \wedge \neg\Psi\}$
$\{\neg\Phi \wedge \Psi\}$
$\{\Phi \wedge \Psi\}$
(a) $\mathcal{D}$
(b) $\mathcal{D}[\neg\Phi]$

Figure 3.5: Model transformation for $\mathsf{U}^{=h}$

with different gray shadings. Let formula $\phi = \Phi\,\mathsf{U}^{=5}\,\Psi$*. Consider the path*

$$\sigma = s_0{\cdot}s_1{\cdot}\underbrace{t_1}_{\models \underline{\Phi}\wedge\Psi}{\cdot}s_2{\cdot}s_1{\cdot}\underbrace{t_1}_{\models \Phi\wedge\underline{\Psi}}.$$

Due to the semantics of $\mathsf{U}^{=h}$, $\sigma \models \phi$*, where the first appearance of* t_1 *is regarded as a* Φ*-state and the second as a* Ψ*-state.* ♦

Model Transformation for $\mathsf{U}^{=h}$. We turn the DTMC $\mathcal{D}$ into $\mathcal{D}[\neg\Phi \wedge \neg\Psi][\neg\Phi \wedge \Psi] = \mathcal{D}[\neg\Phi]$. This transformation is based on the following observation: *once the* $(\neg\Phi \wedge \neg\Psi)$*- and* $(\neg\Phi \wedge \Psi)$*-states are reached in less than* h *hops, any continuation of the path will falsify the property.* This is the main difference with the $(\Phi \wedge \Psi)$-states.

Example 3.25 (Continuing Example 3.24) *For the* DTMC $\mathcal{D}$ *in Fig. 3.5(a),* $\mathcal{D}[\neg\Phi]$ *is shown in Fig. 3.5(b).* ♦

On the transformed model, for each Ψ-state s_Ψ (including both $(\neg\Phi \wedge \Psi)$- and $(\Phi \wedge \Psi)$-states), the validity of $\Diamond^{=h} at_{s_\Psi}$ is to be checked. This can be justified by the following proposition:

Proposition 3.26 *For any* DTMC $\mathcal{D}$*, state* s*, and* PCTL *formula* $\Phi\,\mathsf{U}^{=h}\,\Psi$*,*

$$Prob^{\mathcal{D}}(s, \Phi\,\mathsf{U}^{=h}\,\Psi) = \sum_{v \models \Psi} Prob^{\mathcal{D}[\neg\Phi]}(s, \Diamond^{=h} at_v).$$

Proof: Due to the model transformation, all paths satisfying $\Phi\,\mathsf{U}^{=h}\,\Psi$ can only be of the following form in $\mathcal{D}[\neg\Phi]$:

$$\sigma = \underbrace{s_0 \cdot s_1 \cdots s_{h-1}}_{\models \Phi}\cdot \underbrace{s_h}_{\models \Phi\wedge\Psi} \qquad \text{or} \qquad \sigma' = \underbrace{s_0 \cdot s_1 \cdots s_{h-1}}_{\models \Phi}\cdot \underbrace{s_h}_{\models \neg\Phi\wedge\Psi}.$$

On the other hand, any path in $\mathcal{D}$ satisfying $\Phi\,\mathsf{U}^{=h}\,\Psi$, is of the above two forms and is a $(\Diamond^{=h} at_v)$-path in $\mathcal{D}[\neg\Phi]$. Also because the set of paths $Paths^{\star}_{\mathcal{D}[\neg\Phi]}(s, \Diamond^{=h} at_v)$ for $v \models \Psi$ are pairwise disjoint, it holds that

$$Paths^{\star}_{\mathcal{D}}(s, \Phi\,\mathsf{U}^{=h}\,\Psi) = \bigcup_{v \models \Psi} Paths^{\star}_{\mathcal{D}[\neg\Phi]}(s, \Diamond^{=h} at_v).$$

It directly follows that

$$Prob^{\mathcal{D}}(s, \Phi\,\mathsf{U}^{=h}\,\Psi) = \sum_{v \models \Psi} Prob^{\mathcal{D}[\neg\Phi]}(s, \Diamond^{=h} at_v). \qquad \blacksquare$$

Proposition 3.26 together with Lemma 3.16 suggests that the strongest evidences for point-interval until can be reduced to computing *several* shortest paths with h hops, which we define as the *fixed-hop shortest path* problem. Let $Paths^{\mathcal{G}}_{=h}(s,t)$ be the set of paths in $\mathcal{G}$ from s to t with h hops.

Definition 3.27 (FSP) *Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$ and $h \in \mathbb{N}_{\geqslant 0}$, the* fixed-hop SP *problem is to determine a path $\sigma \in Paths^{\mathcal{G}}_{=h}(s,t)$ such that $w(\sigma) \leqslant w(\sigma')$ for any path $\sigma' \in Paths^{\mathcal{G}}_{=h}(s,t)$.*

Let $\pi_{=h}(s,v)$ denote the FSP of h hops from s to v. In the following, we discuss how $\pi_{=h}(s,v)$ can be computed.

$$\pi_{=h}(s,v) = \begin{cases} s & \text{if } v = s \wedge h = 0 \quad (3.5) \\ \perp & \text{if } v \neq s \wedge h = 0 \quad (3.6) \\ \arg\min\limits_{u \in S} \Big\{ w\Big(\pi_{=h-1}(s,u)\cdot v\Big) \mid (u,v) \in E \Big\} & \text{if } v \neq s \wedge h > 0 \quad (3.7) \end{cases}$$

Notice that the only difference with (3.2)-(3.4) (the BF$^{\leqslant h}$ algorithm) is that the termination condition in (3.2) is changed from "$h \geqslant 0$" to "$h = 0$" in (3.5) here. In other words, those paths that reach s early ($h > 0$ when computed backwards) are now ruled out by the condition $h = 0$. Only the paths that start in s with exactly h hops are accepted. We call this algorithm BF$^{=h}$. The complexity remains $\mathcal{O}(hm)$.

Proposition 3.28 *The strongest evidence for $s \models \Phi\,\mathsf{U}^{=h}\,\Psi$ in $\mathcal{D}$ can be computed by* $\arg\min_{v \models \Psi} \Big\{ w\big(\pi_{=h}(s,v)\big) \Big\}$ *in the graph $\mathcal{G}_{\mathcal{D}[\neg\Phi]}$.*

The time complexity of obtaining the strongest evidence is $\mathcal{O}(hmn)$, where for each $v \models \Psi$, it takes $\mathcal{O}(hm)$ and in the worst case there are n Ψ-states, which justifies $\mathcal{O}(hmn)$.

Example 3.29 (Continuing Example 3.24) *For $\Phi\,\mathsf{U}^{=5}\,\Psi$ in $\mathcal{G}_{\mathcal{D}[\neg\Phi]}$, there are two Ψ-states t_1 and t_2. The shortest paths from s_0 to t_1, t_2 are*

$$\begin{aligned}\pi_{=5}(s_0,t_1) &= s_0{\cdot}s_1{\cdot}t_1{\cdot}s_2{\cdot}s_1{\cdot}t_1 \quad \textit{with probability } 0.06615\\ \pi_{=5}(s_0,t_2) &= s_0{\cdot}s_1{\cdot}t_1{\cdot}s_2{\cdot}s_1{\cdot}t_2 \quad \textit{with probability } 0.0189.\end{aligned}$$

The strongest evidence is the one with a larger probability, i.e., $\pi_{=5}(s_0,t_1)$. ♦

Remark 3.30 *The Viterbi algorithm can also be applied here. By computing the path $\arg\max_{v\models\Psi}\left\{\mathbb{P}\Big(\varrho\big((\Phi\vee\Psi)^h\Psi,h,v\big)\Big)\right\}$, we obtain the strongest evidence for the point-interval until formula $\Phi\,\mathsf{U}^{=h}\,\Psi$. The time complexity for coincides with that using* BF$^{=h}$*, i.e., $\mathcal{O}(hmn)$.*

3.3.4 Interval Until — $\mathsf{U}^{[h_l,h_u]}$, $\mathsf{U}^{\geqslant h_l}$

We now deal with the until operator with general bounds $\mathsf{U}^{[h_l,h_u]}$, where $h_u > h_l > 0$. Since the cases $\mathsf{U}^{[h_l,h_u]}$ and $\mathsf{U}^{\geqslant h_l}$ are similar, we discuss them both in this section.

Model Transformation. Let $h = h_u - h_l$ if $h_u \neq \infty$ and $h=\infty$ otherwise. Note that $\mathsf{U}^{\leqslant\infty}$ is U and $\infty+1=\infty$. The DTMC $\mathcal{D}$ is transformed into two different DTMCs: $\mathcal{D}[\neg\Phi]$ and $\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle$. This transformation is justified by the observation that for each $\sigma\models\Phi\,\mathsf{U}^{[h_l,h_u]}\,\Psi$, σ can be divided into two parts: the prefix $\sigma[..h_l]$ and the suffix $\sigma[h_l..]$, where $\sigma[..(h_l{-}1)]\models\Box^{=h_l-1}\,\Phi$, $\sigma[h_l]\models\Phi\vee\Psi$ and $\sigma[h_l..]\models\Phi\,\mathsf{U}^{\leqslant h}\,\Psi$ or $\Phi\,\mathsf{U}\,\Psi$, respectively, depending on h_u. We first define the concatenation of two paths:

Definition 3.31 *Given paths σ_1 and σ_2 where $\sigma_1[|\sigma_1|]=\sigma_2[0]$, the concatenation $\circ$ is defined as $\sigma_1\circ\sigma_2=\sigma_1\cdot\sigma_2[1..]$. We sometimes write $\sigma_1\circ_v\sigma_2$ if $\sigma_2[0]=v$.*

For instance, if $\sigma_1=s_0{\cdot}s_1$ and $\sigma_2=s_1{\cdot}s_2$, then $\sigma_1\circ\sigma_2=\sigma_1\circ_{s_1}\sigma_2=s_0{\cdot}s_1{\cdot}s_2$. We have the following proposition:

Proposition 3.32 *For any* DTMC *$\mathcal{D}$, state s, and* PCTL *formulae $\Phi\,\mathsf{U}^{[h_l,h_u]}\,\Psi$, let $h=h_u-h_l$. The following holds:*

$$\mathit{Prob}^{\mathcal{D}}(s,\Phi\,\mathsf{U}^{[h_l,h_u]}\,\Psi) \;=\; \sum_{s'\models\Phi\vee\Psi}\mathit{Prob}^{\mathcal{D}[\neg\Phi]}(s,\Diamond^{=h_l}\,at_{s'})\cdot\mathit{Prob}^{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}(s',\Diamond^{\leqslant h+1}\,at_t).$$

Proof: Let $\mathcal{D}_1=\mathcal{D}[\neg\Phi]$ and $\mathcal{D}_2=\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle$. For any path $\sigma\models\Phi\,\mathsf{U}^{[h_l,h_u]}\,\Psi$, due to the semantics, σ is of the following form:

$$\sigma=\underbrace{s_0\cdot s_1\cdots s_{h_l}}_{\sigma[..h_l]}\!\!\!\!\!\!\!\overbrace{\phantom{s_{h_l}}\cdot s_{h_l+1}\cdots s_j}^{\sigma[h_l..]}=\sigma[..h_l]\circ\sigma[h_l..],$$

where $j \in [h_l, h_u]$, and the prefix $\sigma[..(h_l-1)] \models \Box^{=h_l-1}\Phi$ and $s_{h_l} \models \Phi \vee \Psi$. Due to the definition of $\mathcal{D}_1$, it holds that $\sigma[..h_l] \models \Diamond^{=h_l} at_v$ for $v \models \Phi \vee \Psi$ in $\mathcal{D}_1$. Similarly, for the suffix $\sigma[h_l..] \models \Phi \,\mathsf{U}^{\leqslant h}\, \Psi$, for $h = h_u - h_l$. Due to the construction of $\mathcal{D}_2$, $\sigma[h_l..]{\cdot}t \models \Diamond^{[1,h+1]}\, at_t$. Also because t cannot be reached at the 0-th position on path $\sigma[h_l..]{\cdot}t$, the interval can be relaxed to $[0, h+1]$, i.e., $\sigma[h_l..]{\cdot}t \models \Diamond^{\leqslant h+1}\, at_t$. It is also clear that

$$\mathbb{P}(\sigma) = \mathbb{P}(\sigma[..h_l]) \cdot \mathbb{P}(\sigma[h_l..]) = \mathbb{P}(\sigma[..h_l]) \cdot \mathbb{P}(\sigma[h_l..]{\cdot}t).$$

Let

$$\Pi^{\mathcal{D}}(v) = \left\{\sigma \in \mathit{Paths}^{\star}_{\mathcal{D}}(s, \Phi\,\mathsf{U}^{[h_l,h_u]}\,\Psi) \mid \sigma[h_l] = v\right\}$$

be the set of $\Phi\,\mathsf{U}^{[h_l,h_u]}\,\Psi$ paths in $\mathcal{D}$ whose h_l-th position is v. Then,

$$\mathbb{P}\left(\Pi^{\mathcal{D}}(v)\right) = \mathbb{P}\left(\mathit{Paths}^{\mathcal{D}_1}(s, \Diamond^{=h_l} at_v)\right) \cdot \mathbb{P}\left(\mathit{Paths}^{\mathcal{D}_2}(v, \Diamond^{\leqslant h+1} at_t)\right).$$

As the sets $\Pi^{\mathcal{D}}(v)$ for $v \models \Phi \vee \Psi$ are pairwise disjoint, we obtain:

$$\mathbb{P}\left(\mathit{Paths}^{\mathcal{D}}(s, \Phi\,\mathsf{U}^{[h_l,h_u]}\,\Psi)\right) = \sum_{v \models \Phi\vee\Psi} \mathbb{P}\left(\mathit{Paths}^{\mathcal{D}_1}(s, \Diamond^{=h_l} at_v)\right) \cdot \mathbb{P}\left(\mathit{Paths}^{\mathcal{D}_2}(v, \Diamond^{\leqslant h+1} at_t)\right).$$

Applying Lemma 2.20 results in:

$$\mathit{Prob}^{\mathcal{D}}(s, \Phi\,\mathsf{U}^{[h_l,h_u]}\,\Psi) = \sum_{v \models \Phi\vee\Psi} \mathit{Prob}^{\mathcal{D}_1}(s, \Diamond^{=h_l} at_v) \cdot \mathit{Prob}^{\mathcal{D}_2}(v, \Diamond^{\leqslant h+1} at_t). \qquad \blacksquare$$

Note that Proposition 3.26 (for $\mathsf{U}^{=h}$) is a simplified version of Proposition 3.32 (for $\mathsf{U}^{[h_l,h_u]}$). The latter, together with Lemma 3.16, suggests that the strongest evidences for interval until can be reduced to computing several shortest paths in different graphs. Recall the shortest path notations $\pi^{\mathcal{G}}(s,u)$, $\pi^{\mathcal{G}}_{=h}(s,u)$ and $\pi^{\mathcal{G}}_{\leqslant h}(s,u)$. Let $\pi^{\mathcal{G}}_{[h_l,h_u]}(s,u)$ denote the shortest path of hops in $[h_l, h_u]$ from s to u in graph $\mathcal{G}$. In the following, we show how $\pi^{\mathcal{G}}_{[h_l,h_u]}(s,u)$ can be computed:

$$\pi^{\mathcal{G}}_{[h_l,h_u]}(s,u) = \arg\min_{\sigma_1\circ_v\sigma_2} \left\{ w\big(\underbrace{\pi^{\mathcal{G}}_{=h_l}(s,v)}_{\sigma_1}\big) + w\big(\underbrace{\pi^{\mathcal{G}}_{\leqslant h}(v,u)}_{\sigma_2}\big) \;\Big|\; v \in M \right\}, \tag{3.8}$$

where $h = h_u - h_l$ and $M = \{\sigma[h_l] \mid \sigma \in \mathit{Paths}^{\mathcal{G}}(s)\}$ is the set of intermediate states that can be visited at the h_l-th hop when starting from s.

Note that (3.8) computes *any* paths with length $[h_l, h_u]$ in $\mathcal{G}$. If those paths are further required to satisfy $\Phi\,\mathsf{U}\,\Psi$, then the graphs should be adapted accordingly:

Proposition 3.33 *The strongest evidence for $\Phi\,\mathsf{U}^{[h_l,h_u]}\,\Psi$ at state s in $\mathcal{D}$ is*

$$\arg\min_{\sigma_1\circ_v\sigma_2} \left\{ w\,\big(\underbrace{\pi^{\mathcal{G}_1}_{=h_l}(s,v)}_{\sigma_1}\big) + w\,\big(\underbrace{\pi^{\mathcal{G}_2}_{\leqslant h+1}(v,t)}_{\sigma_2}\big) \;\Big|\; v \in M \right\},$$

where $M = \{\sigma[h_l] \mid \sigma \in \mathit{Paths}^{\star}_{\mathcal{G}_1}(s)\}$, $\mathcal{G}_1 = \mathcal{G}_{\mathcal{D}[\neg\Phi]}$, and $\mathcal{G}_2 = \mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$.

Algorithm and Time Complexity. Now we discuss the algorithm and time complexity of computing this strongest evidence.

1. Transforming $\mathcal{D}$ into $\mathcal{G}_1$ and $\mathcal{G}_2$ takes $\mathcal{O}(n)$ time, where $n = |S|$.
2. The set M can be obtained in a "breadth-first" manner by using a queue in $\mathcal{G}_1$. The elements in the queue are of the form (state s, hop i), meaning "state s can be visited in exactly i hops". At first, $(s_0, 0)$ is put in the queue, after all its direct successors, say $(u_1, 1), ..., (u_j, 1)$ are put in the queue, $(s_0, 0)$ is removed from the queue. This process is repeated until the head of the queue is $(\cdot, h)$. Note that if a pair has already existed in the queue, we do not insert it twice.

   For hop count i, each successor of the $(\cdot, i)$ should be checked, that is at most m times, among which at most n vertices will be inserted, and before each insertion, at most n vertices will be checked whether they have already been in the queue, which comes to $\mathcal{O}(mn)$. There are h_l hops in total, so the time complexity of this step is $\mathcal{O}(h_l mn)$.
3. Determine the shortest path $\pi^{\mathcal{G}_1}_{=h_l}(s, v)$ from s to each vertex $v \in M$ of exactly h_l hops. As the result in Section 3.3.3, the complexity of computing one such shortest path is $\mathcal{O}(h_l m)$. Since each vertex in M should be computed, and in the worst case, $|M| = n$; thus the time complexity of this step is $\mathcal{O}(h_l mn)$.
4. In graph $\mathcal{G}_2$, determine the shortest path $\pi^{\mathcal{G}_2}_{\leqslant h+1}(v, t)$ from each vertex $v \in M$ to t within $h + 1$ hops. We apply Bellman-Ford algorithm (for $\mathsf{U}^{\leqslant h}$) or Dijkstra's algorithm (for U) as mentioned in Section 3.3.2 and 3.3.1. The time complexity of this step is $\mathcal{O}(hmn)$ and $\mathcal{O}(mn + n^2 \log n)$, respectively.
5. Determine the shortest path among $|M|$ candidates, i.e.,

$$\arg\min_{v\in M} \Big\{ w\Big(\pi_{=h_l}(s, v)\Big) + w\Big(\pi_{\leqslant h+1}(v, t)\Big) \Big\}.$$

   It takes $\mathcal{O}(\log n)$ to choose the minimal one when heaps are used.

Since $m > n$ and $h_u > h_l$, the total time complexity is $\mathcal{O}(h_u mn)$ for $\mathsf{U}^{[h_l, h_u]}$ and $\mathcal{O}(h_l mn + n^2 \log n)$ for $\mathsf{U}^{\geqslant h_l}$.

Remark 3.34 *The Viterbi algorithm can be applied for $\mathsf{U}^{[h_l, h_u]}$, but not to $\mathsf{U}^{\geqslant h_l}$. For similar reasons, the model transformation should be done in two phases and for each intermediate state $s \in M$, the two most probable paths are computed in each phase, where the most probable path among all the concatenations of those two paths is the strongest evidence.*

$$\sigma = \arg\max_{v\in M} \Big\{ \varrho_{s_0}\Big((\Phi \vee \Psi)^{h_l+1}, h_l, v \Big) \cdot \varrho_v \Big((\Phi \vee \Psi \vee at_t)^h at_t, h+1, t \Big) \Big\}.$$

Until	Model Transformation	Strongest Evidence		
		Path Problem	Algorithm	Time Complexity
$\Phi \,\mathsf{U}\, \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$	SP	Dijkstra	$\mathcal{O}(m + n\log n)$
$\Phi \,\mathsf{U}^{\leqslant h}\, \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$	HSP	$\text{BF}^{\leqslant h}$/ Viterbi	$\mathcal{O}(hm)$
$\Phi \,\mathsf{U}^{=h}\, \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi]}$	FSP	$\text{BF}^{=h}$/ Viterbi	$\mathcal{O}(hmn)$
$\Phi \,\mathsf{U}^{[h_l,h_u]}\, \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi]}$ $\mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$	FSP + HSP	$\text{BF}^{=h}$ + $\text{BF}^{\leqslant h}$/ Viterbi	$\mathcal{O}(h_u mn)$
$\Phi \,\mathsf{U}^{\geqslant h_l}\, \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi]}$ $\mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$	FSP + SP	$\text{BF}^{=h}$ + Dijkstra	$\mathcal{O}(h_l mn + n^2\log n)$

Table 3.1: Overview of finding strongest evidence problems

Note that s in ϱ_s indicates the starting state of the path. The time complexity is $\mathcal{O}(h_u mn)$. The Viterbi algorithm can neither be applied on $\mathsf{U}^{\geqslant h_l}$, nor on U, as the number of hops to reach a Ψ-state is unbounded.

3.3.5 Summary

So far we have discussed the algorithms for finding strongest evidences, which are summarized in Table 3.1. In the remainder of the chapter, we dwell on finding smallest counterexamples.

3.4 Finding Smallest Counterexamples

Recall that a smallest counterexample is a minimal counterexample, whose probability — among all minimal counterexamples — deviates maximally from the required probability bound. In this section, we investigate algorithms and their time complexity for computing smallest counterexamples.

3.4.1 Unbounded Until — U

Lemma 3.16 is applicable here for $k > 1$ and $h = \infty$. This suggests to consider the k shortest paths problem.

Definition 3.35 (KSP problem) *Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$, and $k \in \mathbb{N}_{>0}$, the k* shortest paths *problem is to find k distinct paths $\sigma^{(1)}, \ldots, \sigma^{(k)} \in Paths^{\mathcal{G}}(s,t)$ (if such paths exist) such that* 1) *for $1 \leqslant i < j \leqslant k$, $w(\sigma^{(i)}) \leqslant w(\sigma^{(j)})$ and* 2) *for every $\sigma \in Paths^{\mathcal{G}}(s,t)$, if $\sigma \notin \{\sigma^{(1)}, \ldots, \sigma^{(k)}\}$, then $w(\sigma) \geqslant w(\sigma^{(k)})$.*

Note that $\sigma^{(i)}$ denotes the i-th shortest path and for $i \neq j$, it is possible that $w(\sigma^{(i)}) = w(\sigma^{(j)})$. Stated in words, the i-th shortest path is not necessarily "strictly shorter" than the j-th one, for $i < j$.

Theorem 3.36 *The SC problem for unbounded until reduces to a* KSP *problem.*

Proof: We prove by contraposition that a smallest counterexample of size k, contains k most probable paths. Let C be a smallest counterexample for ϕ with $|C| = k$, and assume C does not contain the k most probable paths satisfying ϕ. Then there is a path $\sigma \notin C$ satisfying ϕ such that $\mathbb{P}(\sigma) > \mathbb{P}(\sigma')$ for some $\sigma' \in C$. Let $C' = C \setminus \{\sigma'\} \cup \{\sigma\}$. Then C' is a counterexample for ϕ, $|C| = |C'|$ and $\mathbb{P}(C) > \mathbb{P}(C')$. This contradicts C being a smallest counterexample. ■

The question remains how to obtain k. Various algorithms for the KSP problem require k to be known a priori. This is inapplicable in our setting, as the number of paths in a smallest counterexample is not known in advance. We therefore consider algorithms that allow to determine k *on the fly*, i.e., that can halt at any k and resume if necessary. A good candidate is Eppstein's algorithm [Epp98]. Although this algorithm has the best known asymptotic time complexity, viz. $\mathcal{O}(m+n\log n+k)$, in practice the *recursive enumeration algorithm* (REA) by Jiménez and Marzal [JM99] prevails. This algorithm has a time complexity in $\mathcal{O}(m + kn\log\frac{m}{n})$ and is based on a generalization of the recursive equations for the BF algorithm. Besides, although REA itself cannot deal with the k shortest *hop-bounded* shortest paths problem, it is readily adaptable to the case for bounded h, as we demonstrate in the next subsection. Note that the time complexity of all known KSP algorithms depends on k, and as k can be exponential in the size of the digraph, their complexity is *pseudo-polynomial*.

3.4.2 Bounded Until — $\mathsf{U}^{\leqslant h}$

Similar to strongest evidences for bounded until, we now consider the KSP problem with constrained path lengths.

Definition 3.37 (HKSP problem) *Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$, $h \in \mathbb{N}_{\geqslant 0}$ and $k \in \mathbb{N}_{>0}$, the* hop-constrained KSP *problem is to determine k shortest paths in $Paths^{\mathcal{G}}_{\leqslant h}(s,t)$.*

Theorem 3.38 *The SC problem for bounded until reduces to a* HKSP *problem.*

To our knowledge, algorithms for the HKSP problem do not exist in the literature. In order to solve the HKSP problem, we propose to adapt Jiménez and Marzal's REA algorithm [JM99]. The advantage of this algorithm is that k can be determined on the fly, an essential characteristic for our setting. For $v \in V$, let $\pi^k_{\leqslant h}(s,v)$ denote the k-th shortest path in $Paths^{\mathcal{G}}_{\leqslant h}(s,v)$ (if it exists). As before, we use $\perp$ to denote the non-existence of a path. We establish:

$$\pi^k_{\leqslant h}(s,v) = \begin{cases} s & \text{if } k=1,\ v=s \text{ and } h \geqslant 0 & (3.9)\\ \perp & \text{if } h=0 \text{ and } \big(v \neq s \text{ or } (v=s \wedge k>1)\big) & (3.10)\\ \arg\min_\sigma \Big\{ w(\sigma) \mid \sigma \in Q^k_{\leqslant h}(s,v) \Big\} & \text{otherwise} & (3.11) \end{cases}$$

where $Q^k_{\leqslant h}(s,v)$ is defined by:

$$Q^k_{\leqslant h}(s,v) = \begin{cases} \big\{\pi^1_{\leqslant h-1}(s,u')\cdot v \mid (u',v) \in E\big\} & \\ \qquad \text{if } k=1, v \neq s, h>0 \text{ or } k=2, v=s, h>0 & (3.12)\\ \big(Q^{k-1}_{\leqslant h}(s,v) - \{\pi^{k'}_{\leqslant h-1}(s,u)\cdot v\}\big) \cup \big\{\pi^{k'+1}_{\leqslant h-1}(s,u)\cdot v\big\} & \\ \qquad \text{if } k>1, h>0, \text{ and } \exists u,k'.\big(\pi^{k-1}_{\leqslant h}(s,v) = \pi^{k'}_{\leqslant h-1}(s,u)\cdot v\big) & (3.13)\\ \varnothing \qquad \text{otherwise} & (3.14) \end{cases}$$

Let us explain these equations. The k-th shortest path of length h is chosen from a set $Q^k_{\leqslant h}(s,v)$ of "candidate" paths. This principle is identical to that in the Bellman-Ford equations given earlier. In particular, if this set contains several shortest paths, a nondeterministic selection is made. The main difference with the BF equations is the more complex definition of the set of candidate paths. Eq. (3.12) of $Q^k_{\leqslant h}(s,v)$ are the base cases and will become clear later. Let $k>1$, $h>0$ and $v \neq s$. By the inductive nature, the set $Q^{k-1}_{\leqslant h}(s,v)$ is at our disposal. Assume that the path $\pi^{k-1}_{\leqslant h}(s,v)$ has the form $s \cdots u \cdot v$ where prefix $s \cdots u$ is the k'-th shortest path between s and u (for some k') of at most $h-1$ hops, i.e., $s \cdots u$ equals $\pi^{k'}_{\leqslant h-1}(s,u)$. Then $Q^k_{\leqslant h}(s,v)$ is obtained from $Q^{k-1}_{\leqslant h}(s,v)$ by replacing the path $s \cdots u \cdot v$ (as it just has been selected) by the path $\pi^{k'+1}_{\leqslant h-1}(s,u)\cdot v$, if this exists. Thus, as a result of the removal of a $(k-1)$-st shortest path which reaches v via u, say, the set of candidate paths is updated with the next shortest path from s to v that goes via u. If such path does not exist (i.e., equals $\perp$), then the candidate set is not extended (as $\{\perp\} = \varnothing$). In case there is no k' such that $\pi^{k-1}_{\leqslant h}(s,v)$ can be decomposed into a k'-th shortest path between s and some direct predecessor u of v, it means that $Q^{k-1}_{\leqslant h}(s,v)$ is empty, and we return the empty set (3.14). Let's go back to (3.12). For $k=1, v \neq s, h>0$, we need to define the candidate path set explicitly, since we cannot apply (3.13) where the path for $k=0$ does not exist. Likewise for $k=2, v=s, h>0$, the shortest path from s to s is s, i.e. $\pi^1_{\leqslant h}(s,s) = s$ cannot be rewritten in the form of $\pi^1_{\leqslant k-1}(s,u)\cdot s$ as $\pi^1_{\leqslant k-1}(s,u)$ does not exist. This justifies the two conditions for (3.12).

Lemma 3.39 *Eq. (3.9)-(3.14) characterize the hop-constrained k shortest paths from s to v in at most h hops.*

Proof: This proof goes along similar lines as [JM99]. Let $\mathcal{X}^k_{\leqslant h}(s,v)$ denote the set of k shortest paths from s to v in at most h hops. Each path in $\mathcal{X}^k_{\leqslant h}(s,v)$ reaches v from some vertex $u \in \mathit{Pred}(v) = \{w \in V \mid (w,v) \in E\}$. In order to compute $\pi^k_{\leqslant h}(s,v)$, we should consider for every $u \in \mathit{Pred}(v)$, *all* paths from s to u that do not yield a path in $\mathcal{X}^{k-1}_{\leqslant h}(s,v)$. However, since $k_1 < k_2$ implies that $w\big(\pi^{k_1}_{\leqslant h-1}(s,u)\big) + w(u,v) \leqslant w\big(\pi^{k_2}_{\leqslant h-1}(s,u)\big) + w(u,v)$, only *the shortest* of these paths needs to be taken into account when computing $\pi^k_{\leqslant h}(s,v)$. Thus we can associate to (v,h) a set of candidate paths $Q^k_{\leqslant h}(s,v)$ among which $\pi^k_{\leqslant h}(s,v)$ can be chosen, that contains at most one path for each predecessor $u \in \mathit{Pred}(v)$. This set $Q^k_{\leqslant h}(s,v)$ is recursively defined by (3.12)-(3.14). ■

Adapted Recursive Enumeration Algorithm (aREA$^{\leqslant h}$). Eq. (3.9)-(3.14) provide the basis for the adapted REA for the HKSP problem (denoted as aREA$^{\leqslant h}$). In the main program (Alg. 1), first the shortest path from s to t is determined using, e.g., BF$^{\leqslant h}$. Then, the k shortest paths are determined iteratively using the subroutine *NextPath* (Alg. 2). The computation terminates when the total probability mass of the k shortest paths so far exceeds the bound p (Alg. 1, line 4). Recall that p is the upper probability bound of the PCTL formula to be checked. Note that $Q[v,h,k]$ in the algorithm corresponds to $Q^k_{\leqslant h}(s,v)$. The paths in the priority queue $Q[v,h,k]$ are ordered wrt. their weights. When $k=1$, $Q[v,h,k-1]$ and $\pi^{k-1}_{\leqslant h}(s,v)$ do not exist and are $\varnothing$ and $\perp$, respectively. $Q[v,h,k]$ is constructed explicitly in two cases (Alg. 2, lines 4-5) and inherits from $Q[v,h,k-1]$ for the remaining cases (line 12). In the latter case, σ' is the path $\sigma = \pi^{k-1}_{\leqslant h}(s,v)$ without the last state v, i.e., $\sigma = \sigma'{\cdot}v$; u is the last state on σ', or equivalently the predecessor state of v on σ with $\sigma = s \cdots u{\cdot}v$ and σ' is the k'-th shortest path from s to u within $h-1$ hops, i.e., $\sigma' = \pi^{k'}_{\leqslant h-1}(s,u)$. In other words, the function $\mathit{index}(s \cdots u, h{-}1)$ returns k' where $s \cdots u$ is the k'-th shortest s-u path within $h{-}1$ hops. The set $Q^k_{\leqslant h}(s,v)$ is updated according to (3.12)-(3.14) (Alg. 2, lines 6-13). In line 14, $\pi^k_{\leqslant h}(s,v)$ is selected from $Q^k_{\leqslant h}(s,v)$ according to (3.11).

Time complexity. Before we analyze the time complexity of the algorithm, we first prove that the recursive calls to *NextPath* to compute $\pi^k_{\leqslant h}(s,t)$ visit, in the worst case, all the vertices in $\pi^{k-1}_{\leqslant h}(s,t)$, which is at most h.

Lemma 3.40 *Let $k > 1$ and $v \in V$. If NextPath(v,h,k) calls NextPath$(u,h{-}1,j)$ then vertex u occurs in $\pi^{k-1}_{\leqslant h}(s,v)$.*

Proof: Consider *NextPath*(v,h,k) and let $\pi^{k-1}_{\leqslant h}(s,v) = u_1 \cdots u_\ell$ with $u_1 = s$ and $u_\ell = v$. Let k_i be the index such that $\pi^{k_i}_{\leqslant h-1}(s,u_i) = u_1 \cdots u_i$, for $0 < i \leqslant \ell$. As $\pi^{k-1}_{\leqslant h}(s,v) = \pi^{k_{\ell-1}}_{\leqslant h-1}(s,u_{\ell-1}){\cdot}v$, *NextPath*$(v,h,k)$ needs to recursively invoke *NextPath*$(u_{\ell-1}, h{-}1, k_{\ell-1}{+}1)$ in case the path $\pi^{k_{\ell-1}+1}_{\leqslant h-1}(s,u_{\ell-1})$ has not been computed

Algorithm 1 Hop-constrained k shortest paths

```
Require: weighted digraph $\mathcal{G}$, states $s, t$, $h \in \mathbb{N}_{\geqslant 0}$, $p \in [0,1]$
Ensure: $C = \{\pi^1_{\leqslant h}(s,t), \ldots, \pi^k_{\leqslant h}(s,t)\}$ with $\mathbb{P}(C) > p$
 1: compute $\pi^1_{\leqslant h}(s,t)$ by $\mathrm{BF}^{\leqslant h}$;
 2: $k := 1$;
 3: $pr := \mathbb{P}(\pi^1_{\leqslant h}(s,t))$;
 4: while $pr \leqslant p$ do
 5:     $k := k+1$;
 6:     $\pi^k_{\leqslant h}(s,t) := \mathit{NextPath}(t,h,k)$;
 7:     $pr := pr + \mathbb{P}(\pi^k_{\leqslant h}(s,t))$;
 8: end while
 9: return $\pi^1_{\leqslant h}(s,t), \ldots, \pi^k_{\leqslant h}(s,t)$
```

Algorithm 2 $\mathit{NextPath}(v,h,k)$

```
Require: weighted digraph $\mathcal{G}$, $\pi^{k-1}_{\leqslant h}(s,v)$ (if it exists),
         and candidate path set $Q[v,h,k{-}1]$ (if it exists)
Ensure: $\pi^k_{\leqslant h}(s,v)$
 1: PriorityQueue $Q[v,h,k]$;
 2: if $k = 1, v = s, h \geqslant 0$ then return $s$;
 3: if $(h = 0) \wedge ((k > 1 \wedge v = s) \vee (v \neq s))$ then return $\perp$;
 4: if $(k = 1, v \neq s, h > 0) \vee (k = 2, v = s, h > 0)$ then
 5:     $Q[v,h,k] := \{\pi^1_{\leqslant h-1}(s,u')\cdot v \mid (u',v) \in E\}$;
 6: else
 7:     Path $\sigma' := \pi^{k-1}_{\leqslant h}(s,v) \setminus \{v\}$;
 8:     State $u := \mathit{last}(\sigma')$;
 9:     Int $k' := \mathit{index}(\sigma', h-1)$;
10:     if $\pi^{k'+1}_{\leqslant h-1}(s,u)$ is not computed yet then
11:         $\pi^{k'+1}_{\leqslant h-1}(s,u) := \mathit{NextPath}(u, h{-}1, k'{+}1)$;
12:     $Q[v,h,k] := Q[v,h,k{-}1]$;
13:     $Q[v,h,k].\mathit{enqueue}\big(\pi^{k'+1}_{\leqslant h-1}(s,u)\cdot v\big)$;
14: return $Q[v,h,k].\mathit{dequeue}()$;
```

yet. By a similar reasoning, the path $\pi^{k_{\ell-1}}_{\leqslant h-1}(s, u_{\ell-1})$ is of the form $\pi^{k_{\ell-2}}_{\leqslant h-2}(s, u_{\ell-2}) \cdot u_{\ell-1}$, and $\mathit{NextPath}(u_{\ell-1}, h{-}1, k_{\ell-1}{+}1)$ may need to invoke $\mathit{NextPath}(u_{\ell-2}, h{-}2, k_{\ell-2}{+}1)$, and so on. In the worst case, this sequence of recursive calls covers the vertices $u_\ell, u_{\ell-1}, \ldots, u_1$ and ends when it either reaches $\pi^1_{\leqslant h'}(s, s)$ for some $0 < h' \leqslant h$, or a hop bound zero. This conforms to the termination conditions (3.9)(3.10) or Alg. 2 lines 2-3 hold. ■

To determine the computational complexity of the algorithm, we assume the candidate sets to be implemented by heaps (as in [JM99]). The k shortest paths to a vertex v can be stored in a linked list, where each path $\pi^k_{\leqslant h}(s, v) = \pi^{k'}_{\leqslant h-1}(s, u) \cdot v$ is compactly represented by its length and a back pointer to $\pi^{k'}_{\leqslant h-1}(s, u)$. Using these data structures, we obtain:

Theorem 3.41 *The time complexity of the* aREA$^{\leqslant h}$ *is* $\mathcal{O}(hm + hk \log(\frac{m}{n}))$.

Proof: The computation of the first step takes $\mathcal{O}(hm)$ using the BF$^{\leqslant h}$ algorithm. Due to Lemma 3.40, the number of recursive invocations to *NextPath* is bounded by h, the maximum length of $\pi^{k-1}_{\leqslant h}(s, t)$. At any given time, the set $Q^k_{\leqslant h}(s, v)$ contains at most $|\mathit{Pred}(v)|$ paths where $\mathit{Pred}(v) = \{u \in V \mid (u, v) \in E\}$, i.e., one path for each predecessor vertex of v. By using heaps to store the candidate sets, a minimal element can be determined and deleted (cf. Alg. 2, line 14) in $\mathcal{O}(\log |\mathit{Pred}(v)|)$ time. Insertion of a path (as in Alg. 2, line 5 and 13) takes the same time complexity. Since $\sum_{v \in V} |\mathit{Pred}(v)| = m$, $\sum_{v \in V} \log |\mathit{Pred}(v)|$ is maximized when all vertices have an equal number of predecessors, i.e., $|\mathit{Pred}(v)| = \frac{m}{n}$. Hence, it takes $\mathcal{O}(h \log(\frac{m}{n}))$ to compute $\pi^k_{\leqslant h}(s, v)$. We have k such paths to compute, yielding $\mathcal{O}(hm{+}hk \log(\frac{m}{n}))$. ■

Note that the time complexity is pseudo-polynomial due to the dependence on k which may be exponential in n. As in our setting, k is not known in advance, this cannot be reduced to a polynomial time complexity.

Example 3.42 *To illustrate the* aREA$^{\leqslant h}$ *algorithm, we compute the shortest paths of at most 4 hops from* s_0 *to* t *in the digraph in Fig. 3.3. The process of computing* $\pi^1_{\leqslant 4}(s_0, t)$ *(Fig. 3.6(a)) coincides with that for* $\pi_{\leqslant 4}(s_0, t)$ *(Fig. 3.4, Example 3.23, page 32). The set of boxes attached to each path is actually the candidate path set of the path. The values in the gray boxes in Fig. 3.6(a) are the shortest paths for the respective pairs of vertices.*

To compute the second shortest path $\pi^2_{\leqslant 4}(s_0, t)$*, since the values in the gray boxes in Fig. 3.6(a) are already used, they should be removed and replaced by the values of* $\pi^2_{\leqslant 1}(s_0, s_0)$*,* $\pi^2_{\leqslant 1}(s_0, s_0) \cdot s_1$*,* $\pi^2_{\leqslant 1}(s_0, s_0) \cdot s_1 \cdot t_1$*,* $\pi^2_{\leqslant 1}(s_0, s_0) \cdot s_1 \cdot t_1 \cdot t$*, which are marked with* $*$ *in Fig. 3.6(b). Note that the* $*$ *marked positions are just the gray labeled position* *in the last round (Fig. 3.6(a)). In this example, both $\pi^1_{\leqslant 1}(s_0, s_0)$ and $\pi^2_{\leqslant 1}(s_0, s_0)$ are needed for computing two different candidate paths. We then pick the minimal weights (maximal probability) from each candidate path set and the result is labeled gray.*

The same applies to $\pi^3_{\leqslant 4}(s_0, t)$ as in Fig. 3.6(c). The shortest paths can be recovered by tracing back the minimal valued boxes. For instance, $\pi^1_{\leqslant 4}(s_0, t) = s_0{\cdot}s_1{\cdot}t_1{\cdot}t$, $\pi^2_{\leqslant 4}(s_0, t) = s_0{\cdot}s_1{\cdot}s_2{\cdot}t_1{\cdot}t$ and $\pi^3_{\leqslant 4}(s_0, t) = s_0{\cdot}s_2{\cdot}t_1{\cdot}t$. ♦

3.4.3 Point Interval Until — $\mathsf{U}^{=h}$

Based on Proposition 3.28 in Section 3.3.3, there are two levels in choosing a strongest evidence in a DTMC $\mathcal{D}$ for $\Phi\,\mathsf{U}^{=h}\,\Psi$. The inner level is to decide for a fixed state $v \models \Psi$ the shortest path, and the outer level is to decide the shortest path among all the s-v paths for all $v \models \Psi$. When generating the smallest counterexample for $\Phi\,\mathsf{U}^{=h}\,\Psi$, the two-level scheme remains the same, and for the inner level a KSP algorithm for exactly h hops is applied.

For the inner level, we define the k fixed-hop shortest paths problem as follows:

Definition 3.43 (FKSP problem) *Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$, $h \in \mathbb{N}_{\geqslant 0}$ and $k \in \mathbb{N}_{>0}$, the* fixed-hop KSP *problem is to determine k shortest paths in $Paths^{\mathcal{G}}_{=h}(s, t)$.*

Let $\pi^k_{=h}(s, v)$ denote the k-th shortest path in $Paths^{\mathcal{G}_{\mathcal{D}[\neg\Phi]}}_{=h}(s, v)$, which is computed by the following equations:

$$\pi^k_{=h}(s,v) = \begin{cases} s & \text{if } k = 1 \wedge v = s \wedge h = 0 \quad (3.15) \\ \perp & \text{if } k = 1 \wedge v \neq s \wedge h = 0 \quad (3.16) \\ \arg\min\limits_{\sigma} \left\{ w(\sigma) \,\middle|\, \sigma \in Q^k_{=h}(s,v) \right\} & \text{if } v \neq s \wedge Q^k_{=h}(s,v) \neq \varnothing \quad (3.17) \end{cases}$$

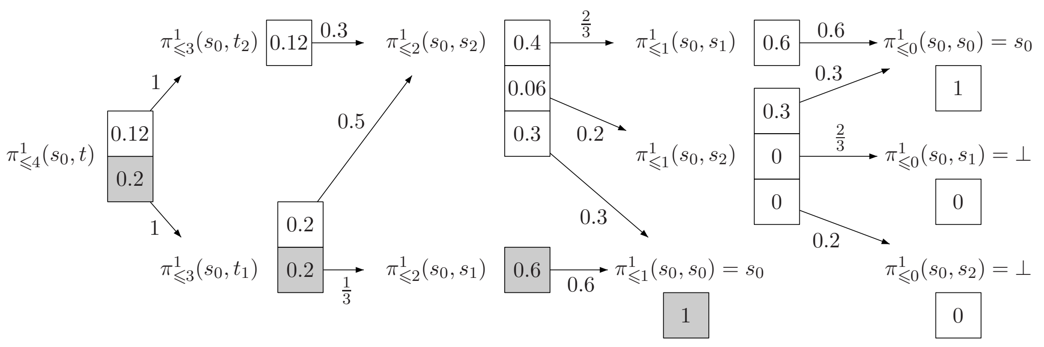
where $Q^k_{=h}(s, v)$ is a set of candidate paths among which $\pi^k_{=h}(s, v)$ is chosen. The candidate sets are defined by:

$$Q^k_{=h}(s,v) = \begin{cases} \left\{ \pi^1_{=h-1}(s,u){\cdot}v \,\middle|\, (u,v) \in E \right\} \\ \qquad \text{if } k = 1 \wedge v \neq s \text{ or } k = 2 \wedge v = s \quad (3.18) \\ \left(Q^{k-1}_{=h}(s,v) - \{\pi^{k'}_{=h-1}(s,u){\cdot}v\} \right) \cup \left\{ \pi^{k'+1}_{=h-1}(s,u){\cdot}v \right\} \\ \qquad \text{if } k > 1 \text{ and } \exists u, k'.\ \pi^{k-1}_{=h}(s,v) = \pi^{k'}_{=h-1}(s,u){\cdot}v \quad (3.19) \end{cases}$$

The above equations are a variant of the aREA$^{\leqslant h}$ algorithm in Section 3.4.2 and we denote it as aREA$^{=h}$.

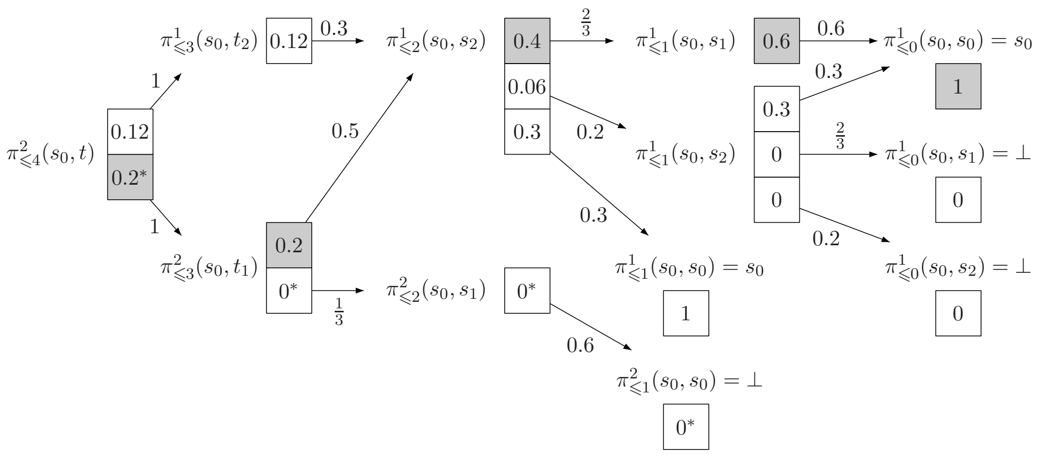
For the outer level, we define ι^k to be the k-th strongest evidence for the formula $\Phi\,\mathsf{U}^{=h}\,\Psi$ as follows:

$$\iota^k = \arg\min_{\sigma} \left\{ w(\sigma) \,\middle|\, \sigma \in Z^k \right\},$$



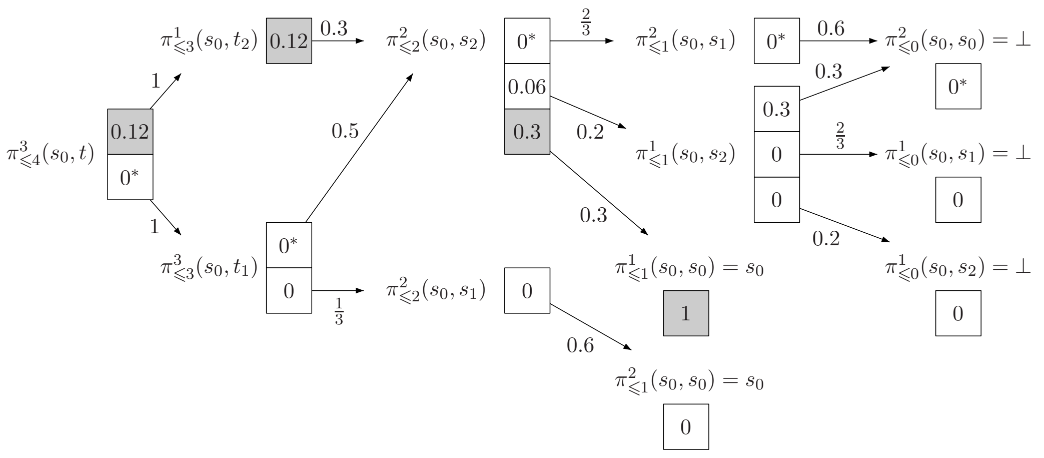
$\pi^1_{\leqslant 4}(s_0,t)$
$\pi^1_{\leqslant 3}(s_0,t_2)$
$\pi^1_{\leqslant 3}(s_0,t_1)$
$\pi^1_{\leqslant 2}(s_0,s_2)$
$\pi^1_{\leqslant 2}(s_0,s_1)$
$\pi^1_{\leqslant 1}(s_0,s_1)$
$\pi^1_{\leqslant 1}(s_0,s_2)$
$\pi^1_{\leqslant 1}(s_0,s_0)=s_0$
$\pi^1_{\leqslant 0}(s_0,s_0)=s_0$
$\pi^1_{\leqslant 0}(s_0,s_1)=\bot$
$\pi^1_{\leqslant 0}(s_0,s_2)=\bot$

(a) $\pi^1_{\leqslant 4}(s_0,t)$



$\pi^2_{\leqslant 4}(s_0,t)$
$\pi^1_{\leqslant 3}(s_0,t_2)$
$\pi^2_{\leqslant 3}(s_0,t_1)$
$\pi^1_{\leqslant 2}(s_0,s_2)$
$\pi^2_{\leqslant 2}(s_0,s_1)$
$\pi^1_{\leqslant 1}(s_0,s_1)$
$\pi^1_{\leqslant 1}(s_0,s_2)$
$\pi^1_{\leqslant 1}(s_0,s_0)=s_0$
$\pi^2_{\leqslant 1}(s_0,s_0)=\bot$
$\pi^1_{\leqslant 0}(s_0,s_0)=s_0$
$\pi^1_{\leqslant 0}(s_0,s_1)=\bot$
$\pi^1_{\leqslant 0}(s_0,s_2)=\bot$

(b) $\pi^2_{\leqslant 4}(s_0,t)$



$\pi^3_{\leqslant 4}(s_0,t)$
$\pi^1_{\leqslant 3}(s_0,t_2)$
$\pi^3_{\leqslant 3}(s_0,t_1)$
$\pi^2_{\leqslant 2}(s_0,s_2)$
$\pi^2_{\leqslant 2}(s_0,s_1)$
$\pi^2_{\leqslant 1}(s_0,s_1)$
$\pi^1_{\leqslant 1}(s_0,s_2)$
$\pi^1_{\leqslant 1}(s_0,s_0)=s_0$
$\pi^2_{\leqslant 1}(s_0,s_0)=s_0$
$\pi^2_{\leqslant 0}(s_0,s_0)=\bot$
$\pi^1_{\leqslant 0}(s_0,s_1)=\bot$
$\pi^1_{\leqslant 0}(s_0,s_2)=\bot$

(c) $\pi^3_{\leqslant 4}(s_0,t)$

Figure 3.6: An example run of the aREA$^{\leqslant h}$ algorithm

where Z^k is the candidate path set for the k-th strongest evidence defined as:

$$Z^k = \begin{cases} \left\{ \pi^1_{=h}(s,s') \,\middle|\, s' \models \Psi \right\} & \text{if } k=1 \quad (3.20) \\ \left(Z^{k-1} - \pi^{k'}_{=h}(s,u) \right) \cup \left\{ \pi^{k'+1}_{=h}(s,u) \right\} & \text{if } k>1 \text{ and } \exists k', u, \text{ s.t. } \iota^{k-1} = \pi^{k'}_{=h}(s,u) \quad (3.21) \end{cases}$$

The size of Z^k is the size of the set M of intermediate states. In the worst case, $|Z^k| = n$. The time complexity for computing Z^1 is $\mathcal{O}(hmn)$, and for computing the successive $k-1$ paths, it needs $\mathcal{O}(hk \log n)$, where n is the cardinality of Z^k. For detailed explanation, refer to Theorem 3.41 (cf. page 45). The total time complexity of computing the strongest evidence is $\mathcal{O}\Big(hmn + hk \log n\Big)$.

3.4.4 Interval Until — $\mathsf{U}^{[h_l,h_u]}$, $\mathsf{U}^{\geqslant h_l}$

According to Section 3.3.4, the strongest evidence for the formula $\Phi \, \mathsf{U}^{[h_l,h_u]} \, \Psi$ (resp. $s \models \Phi \, \mathsf{U}^{\geqslant h_l} \, \Psi$) is computed by two phases. The first phase is to obtain a shortest s-v path σ_1 of exactly h_l hops in graph $\mathcal{G}_1 = \mathcal{G}_{\mathcal{D}[\Phi]}$ and the second phase looks for the shortest v-t path σ_2 of at most $(h_u - h_l + 1)$ hops (resp. with no hop constraints) in graph $\mathcal{G}_2 = \mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi \rangle}$. A strongest evidence is $\sigma_1 \circ_v \sigma_2$ for all v in the set of intermediate states M (cf. Section 3.3.4 for more details).

A smallest counterexample consists of the first k strongest evidences, which is reduced to computing the k shortest such concatenated paths where each path $\sigma = \sigma_1 \circ \sigma_2$ has the following properties: $|\sigma| \in [h_l, h_u]$; $\sigma[0] = s$ and $\sigma[|\sigma|] = t$; $\sigma_1 \in \mathit{Paths}^\star_{\mathcal{G}_1}(s)$ and $\sigma_2 \in \mathit{Paths}^\star_{\mathcal{G}_2}(v)$, where $v \in M$. Let $\pi^k_{[h_l,h_u]}(s,t)$ denote the k-th shortest such path and recall that $h = h_u - h_l$. We establish the following equations:

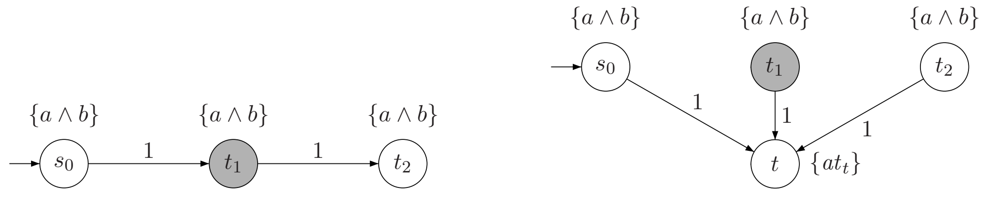
$$\pi^k_{[h_l,h_u]}(s,t) = \arg\min_{\sigma_1\circ\sigma_2} \left\{ w(\sigma_1) + w(\sigma_2) \,\middle|\, \sigma_1 \circ \sigma_2 \in Q^k_{[h_l,h_u]}(s,t) \right\},$$

where $Q^k_{[h_l,h_u]}(s,t)$ is a set of candidate paths among which $\pi^k_{[h_l,h_u]}(s,t)$ is chosen. As is shown, the candidate paths in $Q^k_{[h_l,h_u]}(s,t)$ are of the form $\sigma_1 \circ \sigma_2$. The candidate sets are defined by:

$$Q^k_{[h_l,h_u]}(s,t) = \begin{cases} \left\{ \pi^1_{=h_l-1}(s,u) \circ \pi^1_{\leqslant h+1}(u,t) \mid u \in M \right\} & \text{if } k=1 \quad (3.22) \\ \left(Q^{k-1}_{[h_l,h_u]}(s,t) - \left\{ \pi^{k_1}_{=h_l}(s,u) \circ \pi^{k_2}_{\leqslant h+1}(u,t) \right\} \right) \cup \\ \qquad \left\{ \pi^{k_1+1}_{=h_l}(s,u) \circ \pi^{k_2}_{\leqslant h+1}(u,t),\ \pi^{k_1}_{=h_l}(s,u) \circ \pi^{k_2+1}_{\leqslant h+1}(u,t) \right\} \\ \qquad \text{if } k>1 \text{ and } \exists u, k_1, k_2 \text{ such that} \\ \qquad \pi^{k-1}_{[h_l,h_u]}(s,t) = \pi^{k_1}_{=h_l}(s,u) \circ \pi^{k_2}_{\leqslant h+1}(u,t) \quad (3.23) \end{cases}$$

If $k=1$, $\pi^1_{[h_l,h_u]}(s,t)$ coincides with $\pi_{[h_l,h_u]}(s,t)$ in Section 3.3.4. If $h_l = h_u$, $\pi^k_{[h_l,h_u]}(s,t)$ coincides with $\pi^k_{=h_l}(s,t)$ in Section 3.4.3.

Remark 3.44 *In the second phase, making the Ψ-states absorbing and connecting to t is required. Otherwise, we might not get a prefix containment free set of paths. This can be seen by the following example. Given the formula $\phi = a\,\mathsf{U}^{[1,2]}\,b$ and the* DTMC *in Fig. 3.7(a). Consider two paths $\sigma_1 = s_0{\cdot}t_1$ and $\sigma_2 = s_0{\cdot}t_1{\cdot}t_2$. According to the semantics, $\sigma_1, \sigma_2 \models \phi$. However, $\mathbb{P}(\sigma_1) + \mathbb{P}(\sigma_2) = 2$. It is because $\sigma_2 \in \mathit{Pref}(\sigma_1)$, i.e., the set $\{\sigma_1, \sigma_2\}$ is not prefix containment free. This can be fixed by adding a new state t. Note that in this example $\mathcal{D}$ and $\mathcal{D}[\neg a]$ are the same. In $\mathcal{D}[\neg a]$, the only state that can be reached in $h_l = 1$ step is t_1. In $\mathcal{D}[\neg a \wedge \neg b]\langle t_b\rangle$, from t_1, the only possible path to t is $t_1{\cdot}t$. Thus, we only obtain path σ_1.*



(a) $\mathcal{D}$ and $\mathcal{D}[\neg a]$ (b) $\mathcal{D}[\neg a \wedge \neg b]\langle t_b\rangle$

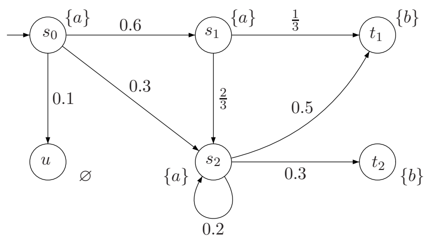
Figure 3.7: An example showing the necessity of adding t in the second phase

This indicates that the minimal satisfaction also applies in the $\mathsf{U}^{[h_l,h_u]}$ case. Now we explain how the algorithm works. Suppose the $(k{-}1)$-th shortest path is $\pi^{k_1}_{=h_l}(s,v) \circ \pi^{k_2}_{\leqslant h+1}(v,t)$, in order to compute the k-th ($k > 1$) shortest path $\pi^{k}_{[h_l,h_u]}(s,t)$, two new paths $\pi^{k_1+1}_{=h_l}(s,v)$ in graph $\mathcal{G}_1$ and $\pi^{k_2+1}_{\leqslant h+1}(v,t)$ in graph $\mathcal{G}_2$ will be computed. To show that two (instead of one) candidate paths should be added, we use the following lattice in Fig. 3.8. It shows that the possible replacement of $\pi^{k_1} \circ_v \pi^{k_2}$ (short for $\pi^{k_1}_{=h_l}(s,v) \circ \pi^{k_2}_{\leqslant h+1}(v,t)$) is either $\pi^{k_1+1} \circ_v \pi^{k_2}$ or $\pi^{k_1} \circ_v \pi^{k_2+1}$, but definitely not $\pi^{k_1+1} \circ_v \pi^{k_2+1}$. Since $w\big(\pi^{k_1+1} \circ_v \pi^{k_2}\big)$ and $w\big(\pi^{k_1} \circ_v \pi^{k_2+1}\big)$ are "incomparable", they both should be added into $Q^{k}_{[h_l,h_u]}(s,t)$ for $\pi^{k}_{[h_l,h_u]}(s,t)$.

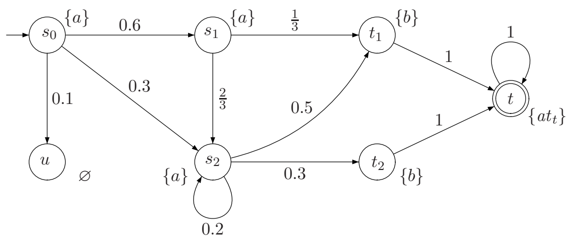
$$\begin{tikzcd}
 & w(\pi^{k_1} \circ_v \pi^{k_2}) \arrow[dl, "\leqslant"'] \arrow[dr, "\leqslant"] & \\
\underline{w(\pi^{k_1+1} \circ_v \pi^{k_2})} \arrow[dr, "\leqslant"'] & & \underline{w(\pi^{k_1} \circ_v \pi^{k_2+1})} \arrow[dl, "\leqslant"] \\
 & w(\pi^{k_1+1} \circ_v \pi^{k_2+1}) &
\end{tikzcd}$$

Figure 3.8: Adding two new candidate paths

Example 3.45 *To illustrate this most general case, we show how to compute the smallest counterexample for $s_0 \not\models \mathcal{P}_{\leqslant 0.38}(a\,\mathsf{U}^{[3,4]}\,b)$ for the* DTMC *in Fig. 2.1. The trans-*



(a) $\mathcal{D}_1 = \mathcal{D}[\neg a]$



(b) $\mathcal{D}_2 = \mathcal{D}[\neg a \wedge \neg b]\langle t_b \rangle$

Figure 3.9: Computing smallest counterexample for $s \not\models \mathcal{P}_{\leqslant 0.38}(a \,\mathsf{U}^{[3,4]}\, b)$

formed DTMC*s are as in Fig. 3.9. It is to compute paths* $\pi^1_{[3,4]}(s_0,t),\ldots,\pi^k_{[3,4]}(s_0,t)$ *s.t.* $\sum_{i=1}^k \mathbb{P}\left(\pi^i_{[3,4]}(s_0,t)\right) > 0.38$*. We list the shortest paths with their probabilities:*

$\mathcal{G}_1$			M	$\mathcal{G}_2$		
$\pi^1_{=3}(s_0,s_2)$	$s_0{\cdot}s_1{\cdot}s_2{\cdot}s_2$	0.08	s_2	$\pi^1_{\leqslant 2}(s_2,t)$	$s_2{\cdot}t_1{\cdot}t$	0.5
$\pi^2_{=3}(s_0,s_2)$	$s_0{\cdot}s_2{\cdot}s_2{\cdot}s_2$	0.012		$\pi^2_{\leqslant 2}(s_2,t)$	$s_2{\cdot}t_2{\cdot}t$	0.3
$\pi^1_{=3}(s_0,t_1)$	$s_0{\cdot}s_1{\cdot}s_2{\cdot}t_1$	0.2	t_1	$\pi^1_{\leqslant 2}(t_1,t)$	$t_1{\cdot}t$	1
$\pi^2_{=3}(s_0,t_1)$	$s_0{\cdot}s_2{\cdot}s_2{\cdot}t_1$	0.03				
$\pi^1_{=3}(s_0,t_2)$	$s_0{\cdot}s_1{\cdot}s_2{\cdot}t_2$	0.12	t_2	$\pi^1_{\leqslant 2}(t_2,t)$	$t_2{\cdot}t$	1
$\pi^2_{=3}(s_0,t_2)$	$s_0{\cdot}s_2{\cdot}s_2{\cdot}t_2$	0.018				

- $k=1$*:*

$$Q^1_{[3,4]}(s_0,t) = \Big\{ \underbrace{\pi^1_{=3}(s_0,s_2)\circ\pi^1_{\leqslant 2}(s_2,t)}_{0.04},\ \underbrace{\pi^1_{=3}(s_0,t_1)\circ\pi^1_{\leqslant 2}(t_1,t)}_{\underline{0.2}},\ \underbrace{\pi^1_{=3}(s_0,t_2)\circ\pi^1_{\leqslant 2}(t_2,t)}_{0.12} \Big\}$$

$\pi^1_{[3,4]}(s_0,t) = \pi^1_{=3}(s_0,t_1)\circ\pi^1_{\leqslant 2}(t_1,t) = s_0{\cdot}s_1{\cdot}s_2{\cdot}t_1{\cdot}t$ $\qquad \sum = 0.2 < 0.38$

- $k=2$*:*

$$Q^2_{[3,4]}(s_0,t) = \Big\{ 0.04,\ \underbrace{\pi^2_{=3}(s_0,t_1)\circ\pi^1_{\leqslant 2}(t_1,t)}_{0.03},\ \underbrace{\pi^1_{=3}(s_0,t_1)\circ\pi^2_{\leqslant 2}(t_1,t)}_{0},\ \underline{0.12} \Big\}$$

$\pi^2_{[3,4]}(s_0,t) = \pi^1_{=3}(s_0,t_2)\circ\pi^1_{\leqslant 2}(t_2,t) = s_0{\cdot}s_1{\cdot}s_2{\cdot}t_2{\cdot}t$ $\qquad \sum = 0.32 < 0.38$

- $k=3$*:*

$$Q^3_{[3,4]}(s_0,t) = \Big\{ \underline{0.04},\ 0.03,\ \underbrace{\pi^2_{=3}(s_0,t_2)\circ\pi^1_{\leqslant 2}(t_2,t)}_{0.018},\ \underbrace{\pi^1_{=3}(s_0,t_2)\circ\pi^2_{\leqslant 2}(t_2,t)}_{0} \Big\}$$

$\pi^3_{[3,4]}(s_0,t) = \pi^1_{=3}(s_0,s_2)\circ\pi^1_{\leqslant 2}(s_2,t) = s_0{\cdot}s_1{\cdot}s_2{\cdot}s_2{\cdot}t_1{\cdot}t$ $\qquad \sum = 0.36 < 0.38$

- $k=4$*:*

$$Q^4_{[3,4]}(s_0,t) = \Big\{ \underbrace{\pi^2_{=3}(s_0,s_2)\circ\pi^1_{\leqslant 2}(s_2,t)}_{0.006},\ \underbrace{\pi^1_{=3}(s_0,s_2)\circ\pi^2_{\leqslant 2}(s_2,t)}_{0.0024},\ \underline{0.03},\ 0.018 \Big\}$$

$\pi^4_{[3,4]}(s_0,t) = \pi^1_{=3}(s_0,s_2)\circ\pi^1_{\leqslant 2}(s_2,t) = s_0{\cdot}s_2{\cdot}s_2{\cdot}t_1{\cdot}t$ $\qquad \sum = 0.39 > 0.38$

The removal of 0.2 *in* $k=1$ *has two replacement for* $k=2$*:* 0.03 *and* 0*. Likewise, the removal of* 0.12 *in* $k=2$ *is replaced by* 0.018 *and* 0*. The procedure stops at* $k=4$ *and the smallest counterexample is* $\{\pi^i_{[3,4]}(s_0,t) \mid 1 \leqslant i \leqslant 4\}$*.* ♦

Algorithm and Time Complexity. To compute $\pi^k_{[h_l,h_u]}(s,t)$, the following steps will be taken:

1. Computing the intermediate vertex set M in $\mathcal{G}_1$ costs $\mathcal{O}(h_l mn)$, cf. Section 3.3.4.
2. Computing $Q^1_{[h_l,hu]}(s,t)$ (resp. $Q^1_{\geqslant h_l}(s,t)$) costs $\mathcal{O}(h_u mn)$ (resp. $\mathcal{O}(h_l mn + n^2 \log n)$), cf. Section 3.3.4.
3. Compute the successive $k-1$ paths. We distinguish the two cases $\Phi \, \mathsf{U}^{[h_l,h_u]} \, \Psi$ and $\Phi \, \mathsf{U}^{\geqslant h_l} \, \Psi$. For each k, two new paths $\sigma_1 = \pi^{k_1+1}_{=h_l}(s,v)$ and $\sigma_2 = \pi^{k_2+1}_{\leqslant h+1}(v,t)$ (resp. $\sigma_2' = \pi^{k_2+1}(v,t)$) should be computed, combined and added into $Q^k_{[h_l,h_u]}(s,t)$. The following argument is based on the proof of Theorem 3.41.

   - Case $\Phi \, \mathsf{U}^{[h_l,h_u]} \, \Psi$: In this case, σ_1 can be computed by aREA$^{=h}$ and σ_2 by aREA$^{\leqslant h}$. In each round, the function *Nextpath* is invoked h_l and $h+1$ times for σ_1 and σ_2, respectively. For round i, there are $|M|+i-1$ elements in Q^i. This is illustrated in Table 3.2. Recall that $\mathcal{O}(|M|) = n$. The total time complexity to compute the successive $k-1$ paths is

$$\begin{aligned} & \mathcal{O}\Big((h_l + h)\big(\log(n+1) + \cdots + \log(n+k-1)\big)\Big) \\ = \; & \mathcal{O}\Big(h_u \log\big((n+1)\cdots(n+k-1)\big)\Big) \; = \; \mathcal{O}\Big(h_u \log \frac{(n+k)!}{n!}\Big). \end{aligned}$$

i-th round	#calls of *NextPath* for			take min from Q^k $\mathcal{O}(\log \lvert Q^k \rvert)$
	σ_1 (aREA$^{=h}$)	σ_2 (aREA$^{\leqslant h}$)	σ_2' (REA)	
1	h_l	$h+1$	$n+1$	$\log n$
2	h_l	$h+1$	$n+1$	$\log(n+1)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
k	h_l	$h+1$	$n+1$	$\log(n+k-1)$

Table 3.2: Time complexity for computing the next $k-1$ paths

   - Case $\Phi \, \mathsf{U}^{\geqslant h_l} \, \Psi$: In this case, σ_1 coincides with the previous case and σ_2' can be computed by the REA algorithm [JM99]. The only difference here is the number of invocation of *NextPath* in each round, which is $n+1$ instead of $h+1$. The extra 1 is due to the new state t. Likewise, the time complexity of computing the successive $k-1$ paths is $\mathcal{O}\Big((h_l+n)\log\frac{(n+k)!}{n!}\Big)$.

The total complexity is $\mathcal{O}\Big(h_u mn + h_u \log\frac{(n+k)!}{n!}\Big)$ for case $\Phi \, \mathsf{U}^{[h_l,h_u]} \, \Psi$ and $\mathcal{O}\Big(h_l mn + n^2 \log n + (h_l+n)\log\frac{(n+k)!}{n!}\Big)$ for $\Phi \, \mathsf{U}^{\geqslant h_l} \, \Psi$.

3.5 Summary

In this chapter we have provided the theoretical and algorithmic foundations for counterexample generation in probabilistic model checking, in particular for discrete-time Markov chains. The main contributions are:

- formally defining (strongest) evidences and (smallest) counterexamples for model checking DTMC against PCTL of the form $\mathcal{P}_{\leqslant p}(\Phi \, \mathsf{U}^I \, \Psi)$;
- casting of the concepts of strongest evidence and smallest counterexample as (variants of) shortest path (SP) problems;
- proposing new algorithm aREA and its variants for computing differently bounded KSP problems.

Table 3.3 gives an overview of which algorithms can be applied when checking properties with (non-strict) probability upper bounds. All cases can be treated by standard SP algorithms or their amendments. Note that n and m are the number of states and transitions in the Markov chain, h, h_l, h_u are the hop bound in the formula, and k is the number of shortest paths. These algorithms are central to counterexample generation for PCTL, both for upper and lower probability bounds and for other models and logics. We will extend those results in Chapter 5.

Until	Model Transformation	Strongest Evidence		
		Path Problem	Algorithm	Time Complexity
$\Phi\ \mathsf{U}\ \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$	SP	Dijkstra	$\mathcal{O}(m+n\log n)$
$\Phi\ \mathsf{U}^{\leqslant h}\ \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$	HSP	BF$^{\leqslant h}$/Viterbi	$\mathcal{O}(hm)$
$\Phi\ \mathsf{U}^{=h}\ \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi]}$	FSP	BF$^{=h}$/Viterbi	$\mathcal{O}(hmn)$
$\Phi\ \mathsf{U}^{[h_l,h_u]}\ \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi]}$ $\mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$	FSP + HSP	BF$^{=h}$ + BF$^{\leqslant h}$/ Viterbi	$\mathcal{O}(h_u mn)$
$\Phi\ \mathsf{U}^{\geqslant h_l}\ \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi]}$ $\mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$	FSP + SP	BF$^{=h}$ + Dijkstra	$\mathcal{O}(h_l mn+n^2\log n)$

Until	Model Transformation	Smallest Counterexample		
		Path Problem	Algorithm	Time Complexity
$\Phi\ \mathsf{U}\ \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$	KSP	Eppstein REA	$\mathcal{O}(m+n\log n+k)$ $\mathcal{O}(m+kn\log\frac{m}{n})$
$\Phi\ \mathsf{U}^{\leqslant h}\ \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$	HKSP	aREA$^{\leqslant h}$	$\mathcal{O}(hm+hk\log(\frac{m}{n}))$
$\Phi\ \mathsf{U}^{=h}\ \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi]}$	FKSP	aREA$^{=h}$	$\mathcal{O}(hmn+hk\log n)$
$\Phi\ \mathsf{U}^{[h_l,h_u]}\ \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi]}$ $\mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$	KFSP + HKSP	aREA$^{=h}$ + aREA$^{\leqslant h}$	$\mathcal{O}\Big(h_u mn+h_u\log\frac{(n+k)!}{n!}\Big)$
$\Phi\ \mathsf{U}^{\geqslant h_l}\ \Psi$	$\mathcal{G}_{\mathcal{D}[\neg\Phi]}$ $\mathcal{G}_{\mathcal{D}[\neg\Phi\wedge\neg\Psi]\langle t_\Psi\rangle}$	FKSP + KSP	aREA$^{=h}$ + REA	$\mathcal{O}\Big(h_l mn+n^2\log n+(h_l{+}n)\log\frac{(n+k)!}{n!}\Big)$

Table 3.3: Overview of the counterexample problems

Chapter 4

Compact Counterexample Representations

In this chapter, we propose an alternative way to represent a counterexample. Unlike the path enumeration approach in Chapter 3, the regular expression based approach is investigated. A need for using compact counterexample representation is demonstrated by the size of the counterexample for the synchronous leader election protocol [IR90]. We use regular expressions for the succinct representation because they are commonly known, are easy to understand, and may be very compact. The idea is to represent a DTMC by a deterministic finite-state automaton (DFA, for short) and obtain regular expressions by applying successive state elimination where the order of state elimination is determined heuristically (e.g., [DM04][HW07]). The computation of the probability of a regular expression is performed using the approach advocated by Daws [Daw04] for parametric model checking of DTMCs. This boils down to a recursive evaluation which is guaranteed to be exact (i.e., no rounding errors), provided the transition probabilities are rational. We provide the details of this approach and show its result when applied to the leader election protocol. We briefly argue that model reduction such as bisimulation [LS91] and SCC elimination [lGM02] can be used to obtain even more compact counterexamples. Finally, we show the generation of counterexamples on the Crowds protocol [RR98], a protocol for anonymous web browsing that has been adopted, among others, to Bluetooth [VSV05] and wireless Internet [AFHL04].

4.1 Motivation

Smallest counterexamples may contain an excessive number of evidences, which is illustrated by the violation of $s \models \mathcal{P}_{\leqslant 0.9999}(\Diamond\, a)$ in the DTMC in Fig. 4.1. The smallest counterexample consists of the evidences $s{\cdot}(u{\cdot}s)^0{\cdot}u{\cdot}t, \ldots, s{\cdot}(u{\cdot}s)^{k-1}{\cdot}u{\cdot}t$, where $(u{\cdot}s)^i$ is a short form of traversing the loop $s{\cdot}u{\cdot}s$ for i times and k is the smallest integer such

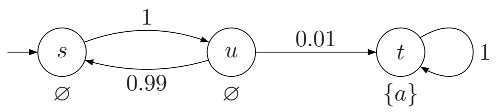


Figure 4.1: A DTMC with excessive number of evidences

that $1 - 0.99^{k-1} > 0.9999$ holds. As a result, the smallest counterexample has $k = 689$ evidences. In fact, the large number of evidences degrades the significance of each evidence. It also provides too much information to easily locate the reason of violation.

To illustrate that such phenomena also occur in real-life cases, we made a prototypical implementation (in Python [Dam08]) to enable generating counterexamples for more practical case studies. Our implementation uses the same input format as the probabilistic model checker MRMC [KKZ05]. Using the export facilities of PRISM [KNP04], counterexamples can be generated for various case studies.

Case Study — Synchronous Leader Election Protocol. Let us report on one case study: *synchronous leader election protocol* [IR90]. In this protocol, N processes are arranged in a unidirectional ring to elect a leader. For this purpose, they randomly select an identity (id, for short) according to a uniform distribution on $\{1, \ldots, K\}$. We call each such selection by all processes a *configuration*. By means of synchronous message passing, processes send their ids around the ring till every process sees all the ids of others, and can thus determine whether a leader (the one with the highest unique id) can be elected. If yes, the protocol terminates; if no, a new round will be started.

Example 4.1 *If $N = 4$ and $K = 3$, then a possible configuration is $(id_1 = 1, id_2 = 3, id_3 = 3, id_4 = 1)$. Under this configuration, there are no unique ids, thus a new round should be started. Another possible configuration is $(id_1 = 2, id_2 = 3, id_3 = 3, id_4 = 1)$. It has two unique ids ($id_1 = 2$ and $id_4 = 1$), among which the highest id, i.e. process* 1 *will be elected as the leader.* ♦

We intend to find a counterexample for formula $\mathcal{P}_{\leqslant p}(\Diamond\, leader\_elected)$, where *leader_elected* characterizes the global state of the protocol in which a leader has been selected. It is clear that a leader will be elected eventually. What interests us, is the number of evidences needed to converge to probability 1. Especially, we are interested in the relationship between the number of evidences and the bound p and R, where R is the round number. Starting a new round means that each process re-selects an id and repeats the procedure.

4.1.1 Experimental Results

To find the number of evidences contained in a counterexample, we used the PRISM-model of the protocol [Pri] and ran the counterexample generation using our imple-

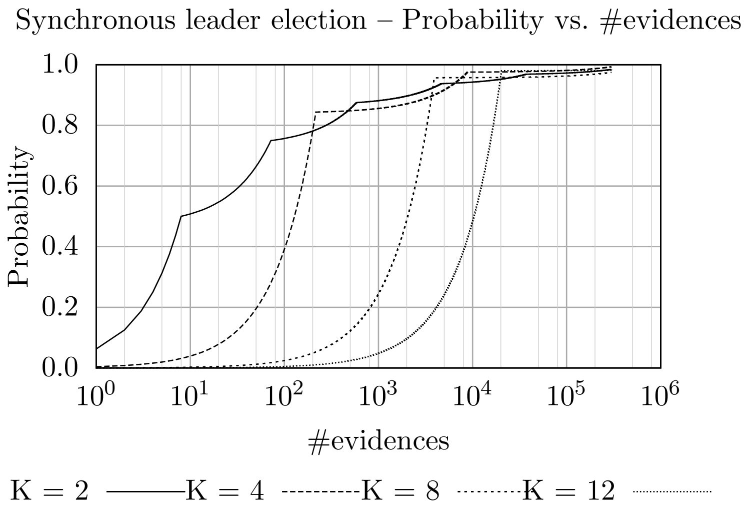


Figure 4.2: Probability vs. number of evidences for leader election ($N = 4$)

mented algorithm. The results for a fixed N ($N = 4$) and varying K are depicted in Fig. 4.2, where the y-axis is the accumulated probability and the x-axis (log-scale) is the number of evidences that are contained in a counterexample. The abrupt changes in the curves correspond to the start of a new round, i.e., a new election, in the protocol. As the probability of all evidences in one round is the same, the curves in Fig. 4.2 are actually piecewise linear if the x-axis were not log-scale. The curves shift more to the right when K increases since there are more possible configurations and thus more evidences. The larger K is, the more quickly the probability of the counterexample approaches 1. This is due to the fact that it is less probable that no process selects a unique id. All curves approach 1, which indicates that almost surely eventually a leader will be elected. The number of evidences in a counterexample, however, grows drastically to millions; whereas the probability of having elected a leader decreases drastically in each round, thus the probability per-evidence decreases tremendously.

4.1.2 Mathematical Analysis

To obtain more insight into this rapid growth of the size of a counterexample, we carry out a brief combinatorial analysis. Let us first consider the number of possibilities (denoted $W(N, K)$) of putting N *labeled* balls into K *labeled* boxes such that each box contains at least two balls. Actually, $W(N, K)$ characterizes the number of possibilities of assigning K ids to N processes such that each id is assigned to more than one process, in which case a leader is not selected. $W(N, K)$ can be solved by using the "associated

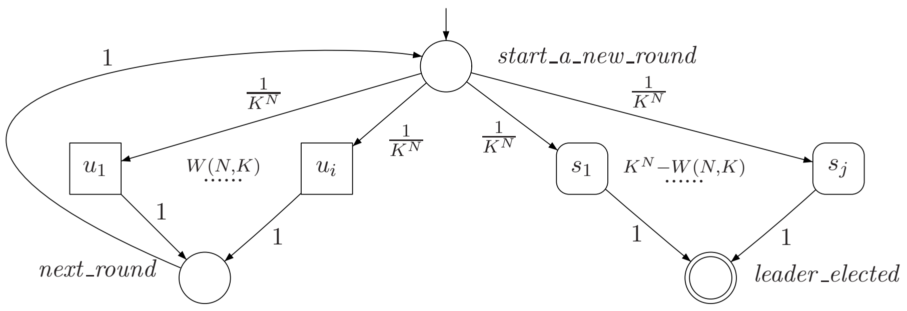


Figure 4.3: The leader election model

Stirling number of the second kind (S_2)" [Com74]:

$$W(N,K) = \sum_{j=1}^{\min(\lfloor N/2 \rfloor, K)} S_2(N,j) \frac{K!}{(K-j)!}, \tag{4.1}$$

where $S_2(N,K) = K{\cdot}S_2(N-1,K) + (N-1){\cdot}S_2(N-2,K-1)$ indicates the number of ways to put N labeled balls into K *unlabeled* boxes. Obviously, it makes no sense to have more than $\lfloor N/2 \rfloor$ boxes, or else it would be impossible to allocate all the balls in the right way. The factor $\frac{K!}{(K-j)!}$ expresses that there are $K!$ ways to permute the boxes (including the empty ones); for these empty boxes the order does not matter, so we divide by $(K-j)!$.

The non-recursive equation for $S_2(N,K)$ is:

$$S_2(N,K) = \sum_{i=0}^{K} (-1)^i \binom{N}{i} \left(\sum_{j=0}^{K-i} (-1)^j \frac{(K-i-j)^{N-i}}{j!(K-i-j)!} \right). \tag{4.2}$$

For each round in the leader election protocol, the number of possibilities for a process to choose an id is K^N. Thus, the probability that N processes with K ids elect a leader in round R, denoted by $P(N,K,R)$, is:

$$P(N,K,R) = \left(\frac{W(N,K)}{K^N} \right)^{R-1} \frac{K^N - W(N,K)}{K^N}, \tag{4.3}$$

where $\left(\frac{W(N,K)}{K^N} \right)^{R-1}$ is the probability that a leader is not elected in the first $(R-1)$ rounds and $\frac{K^N - W(N,K)}{K^N}$ indicates the probability that a leader is elected in the R-th round.

We now calculate the probabilities of each evidence per round using (4.3). The model of the synchronous leader election protocol is depicted in Fig. 4.3. When we *start_a_new_round*, there are K^N possible configurations, among which in $W(N,K)$ (square states, *u*nsuccessful) configurations no unique id will be selected. For these states, we start the *next_round*, while in $K^N - W(N,K)$ (round-angle states, *s*uccessful) configurations a unique id will be selected with a *leader_elected*. Thus:

Proposition 4.2 *The number of evidences that can reach the state leader_elected in the R-th round is:*

$$\#Evi(N, K, R) = W(N, K)^{R-1} \cdot \left(K^N - W(N, K)\right).$$

Proposition 4.2 shows that the number of evidences is exponential in R. Note that $W(N, K)$ is exponential in N and K, which makes $\#Evi(N, K, R)$ double exponential.

The number of evidences thus grows extremely fast. This results in two problems. First, it leads to storage problems as counterexamples may simply get too large to be kept in memory. Secondly, and more importantly, counterexamples will be incomprehensible to the user. We therefore need to find ways to reduce the number of evidences in a counterexample, and to obtain a compact and user-friendly representation. To that purpose we suggest to use *regular expressions*.

4.2 Regular Expression Counterexamples for Unbounded Reachability

This approach is inspired by classical automata theory and is based on representing sets of paths by regular expressions. A major difference with usual regular expressions is that we need to keep track of the transition probabilities. To tackle this, we adopt the approach proposed by Daws [Daw04]. He uses regular expressions to represent sets of paths and calculates the exact rational value of the probability measure in DTMC model checking (provided all transition probabilities are rational). We adapt this approach to obtain compact representations of counterexamples. We restrict to the reachability properties, i.e., no nested formulae. This restriction is due to technical reasons that will become clear later (page 66). The main idea is to consider a counterexample as a set of probable branches (sub-expressions) that go from the initial state to the goal state and to provide a function to evaluate the probability measure of those expressions. We first consider the unbounded reachability properties and assume that the DTMC at our disposal has been subject to the transformation in Step 1, cf. Section 3.2.1. For other bounded reachability forms, we will discuss in Section 4.3.

4.2.1 Turning A DTMC into An Automaton

Definition 4.3 (DFA of a DTMC) *For* DTMC $\mathcal{D} = (S, \mathbf{P}, L)$ *with initial state* $\hat{s} \in S$ *and goal state* t*, let the* deterministic finite automaton (DFA) $\mathcal{A}_\mathcal{D} = (S', \Sigma, \tilde{s}, \delta, \{t\})$*, where:*

- $S' = S \cup \{\tilde{s}\}$ *is the state space with start state* $\tilde{s} \notin S$*;*
- $\Sigma = \left\{\left(\mathbf{P}(s, s'), s'\right) \mid \mathbf{P}(s, s') > 0\right\} \cup \left\{(1, \tilde{s})\right\} \subseteq (0, 1] \times S$ *is the (finite) alphabet;*

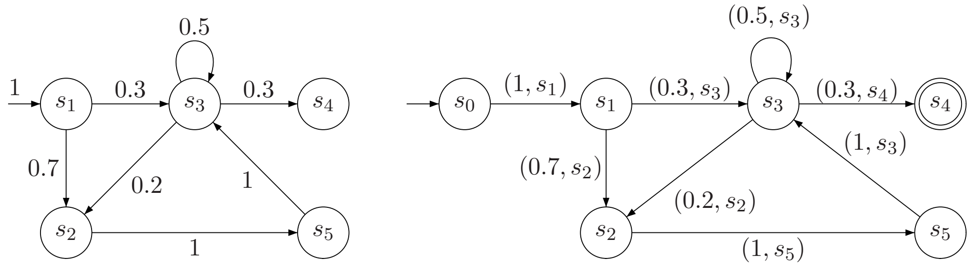


Figure 4.4: An example DTMC $\mathcal{D}$ and its automaton $\mathcal{A}_{\mathcal{D}}$

- *$\delta \subseteq S' \times \Sigma \times S'$ is the transition relation such that $\delta\,(s, (p, s')) = s'$ iff $\mathbf{P}(s, s') = p$, and $\delta(\tilde{s}, (1, \hat{s})) = \hat{s}$;*
- *$t \in S$ is the accepting state.*

The automaton is equipped with a start state $\tilde{s}$ with a transition of probability one to the initial state of $\mathcal{D}$. Symbols in the alphabet are pairs (p, s) with p a probability and s a state. Transition $s \xrightarrow{p} s'$ in $\mathcal{D}$ is turned into a transition from s to s' labeled with (p, s'). (Obviously, this yields a deterministic automaton.) This is a slight, though important deviation from [Daw04], where labels are just probabilities. The probabilities are needed to determine the path probabilities (see Def. 4.5), while the target states are used for recovering the evidences. For simplicity, probability labels are omitted if they are clear from the context.

Example 4.4 *Fig. 4.4 (left) depicts an abstract example of a* DTMC $\mathcal{D}$ *with initial state $\hat{s} = s_1$ and goal state $t = s_4$, and its* DFA $\mathcal{A}_{\mathcal{D}}$ *(right). The new start state is $\tilde{s} = s_0$, which has a transition equipped with symbol* $(1, s_1)$ *to* s_1*.* ♦

4.2.2 Evaluation of Regular Expressions

Regular expressions will be used to represent a counterexample C. Let $\mathcal{R}(\Sigma)$ be the set of regular expressions over the finite alphabet Σ. It contains the elements of Σ, the empty word ε, and is closed under union $(|)$, concatenation $(.)$ and Kleene star $(*)$. Let $\mathcal{L}(r) \in \Sigma^*$ denote the regular language (a set of words) described by the regular expression $r \in \mathcal{R}(\Sigma)$ and $\mathcal{L}(\Sigma)$ denote the regular language that can be generated by any regular expression over Σ. The length $|z|$ and $|r|$ denote the number of symbols in the word z and regular expression r, respectively. For instance, for the alphabet $\Sigma = \{a, b, c\}$, the regular expression $r = a^*.(b|c) \in \mathcal{R}(\Sigma)$ can generate the word $z = aaac \in \mathcal{L}(r)$, where $|r| = 3$ and $|z| = 4$. We sometimes omit . and write $r.r'$ as rr' for short.

To determine the probability of a counterexample C (denoted $\mathbb{P}(C)$) from its regular expression r, we define the evaluation function as follows:

Definition 4.5 ([Daw04], Evaluating regular expressions) *Let* $val : \mathcal{R}(\Sigma) \mapsto \mathbb{R}$ *be defined as:*

$$val(\varepsilon) = 1 \qquad val(r|r') = val(r) + val(r')$$
$$val((p,s)) = p \qquad val(r.r') = val(r) \times val(r')$$

$$val(r^*) = \begin{cases} 1, & \text{if } val(r) = 1 \\ \frac{1}{1-val(r)}, & \text{otherwise} \end{cases} \tag{4.4}$$

If we limit the transition probabilities to be rational values, then exact values are obtained. It can be seen by the following theorem:

Theorem 4.6 *Let* r *be the regular expression for* DFA $\mathcal{A}_{\mathcal{D}} = (S', \Sigma, \tilde{s}, \delta, \{t\})$ *where* $\mathcal{D} = (S, \mathbf{P}, L)$ *with initial state* $\hat{s}$*. Then:*

$$val(r) = \mathbb{P}\big(Paths^{\star}_{\min}(\hat{s}, \Diamond\, at_t)\big).$$

Proof: The theorem is stated in [Daw04] without a proof. We prove it here by structural induction over r. Since it is a folklore [Kle56] that the regular expression r generates the same language as the automaton with $\tilde{s}$ the start state and t the accepting state, it consequently holds that $\mathcal{L}(r) = Paths^{\star}_{\min}(\hat{s}, \Diamond\, at_t)$. Let $\mathbb{P}(r)$ denote the probability of the set $Paths^{\star}_{\min}(\hat{s}, \Diamond\, at_t)$. It remains to show that $val(r) = \mathbb{P}(r)$.

The regular expression r between $\tilde{s}$ and t can be read as the labeling of the transition from $\tilde{s}$ to t, i.e., $\tilde{s} \xrightarrow{r} t$. We distinguish the following cases:

- If $r = \varepsilon$, then $\tilde{s} = t$ and the reachability probability is 1, i.e., $\mathbb{P}(r) = 1 \overset{\text{def.}}{=} val(\varepsilon)$.
- If $r = (p, t)$, there is only one step from $\tilde{s}$ to t and the probability is p, i.e., $\mathbb{P}((p,t)) = p \overset{\text{def.}}{=} val((p,t))$.
- If $r = r_1|r_2$, it can be seen as that there are two possible paths to reach from $\tilde{s}$ to t, labeled with r_1 and r_2, respectively. The probability is thus the sum of the probability of the two paths, see Fig. 4.5(a). By the induction hypothesis, $\mathbb{P}(r_1) = val(r_1)$ and $\mathbb{P}(r_2) = val(r_2)$, thus
$$\mathbb{P}(r_1|r_2) = \mathbb{P}(r_1) + \mathbb{P}(r_2) \overset{\text{I.H.}}{=} val(r_1) + val(r_2) \overset{\text{def.}}{=} val(r).$$
- If $r = r_1.r_2$, it can be seen as that there are two successive (or concatenated) paths to reach from $\tilde{s}$ to t, labeled with r_1 and r_2, respectively. The probability is thus the product of the probability of the two subpaths, see Fig. 4.5(b). Similarly,
$$\mathbb{P}(r_1.r_2) = \mathbb{P}(r_1) \times \mathbb{P}(r_2) \overset{\text{I.H.}}{=} val(r_1) \times val(r_2) \overset{\text{def.}}{=} val(r).$$

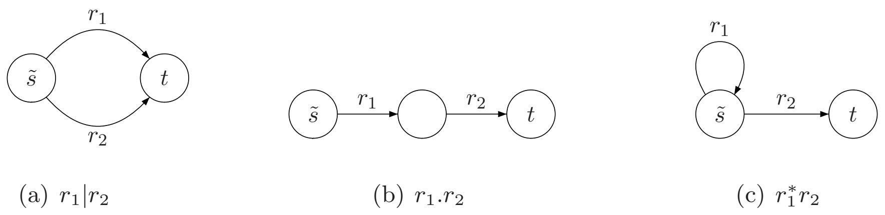


Figure 4.5: The intuition of function $val(r)$

- If $r = r_1^*$ and $val(r_1) = 1$ (the first case in (4.4)), then $\tilde{s} = t$ and $\mathbb{P}(r) = 1 = val(r)$. If $val(r_1) < 1$ (the second case in (4.4)), then there exists $r_2 \neq \varepsilon$ such that $r = r_1^* r_2$, see Fig. 4.5(c). Due to the geometrical series,

$$\begin{aligned}\mathbb{P}(r) &= \sum_{i=0}^{\infty} \mathbb{P}(r_1)^i \cdot \mathbb{P}(r_2) = \frac{1}{1-\mathbb{P}(r_1)} \cdot \mathbb{P}(r_2) \\ &\overset{\text{I.H.}}{=} \frac{1}{1-val(r_1)} \cdot val(r_2) \overset{\text{def.}}{=} val(r_1^*) \cdot val(r_2) \\ &\overset{\text{def.}}{=} val(r_1^* r_2) = val(r).\end{aligned}$$

■

Note that r_1^* cannot stand alone unless $val(r_1) = 1$, otherwise there must exist an outgoing transition with probability at most $1 - val(r_1) > 0$. This explains why there exists $r_2 \neq \varepsilon$. For $val(r_1) < 1$, $val(r_1^*)$ is not interpreted as *a* probability (since it is greater than 1) but as the *sum* of an infinite set of probabilities $(\sum_{i=0}^{\infty} val(r_1^i))$.

In the following, we define a maximal union subexpression which plays the role of a *compact evidence*, i.e., the evidences in a *compact counterexample* (the regular expression counterexample):

Definition 4.7 *r_1 is a* maximal union subexpression (MUS) *of a regular expression r if $r = r_1 | r_2$ modulo* $(\mathbf{R_1})$-$(\mathbf{R_3})$*, for some $r_2 \in \mathcal{R}(\Sigma)$, where:*

$$\begin{array}{lrcl} (\mathbf{R_1}) & r' & \equiv & r' \mid \varepsilon \\ (\mathbf{R_2}) & r_1' \mid r_2' & \equiv & r_2' \mid r_1' \\ (\mathbf{R_3}) & r_1' \mid (r_2' \mid r_3') & \equiv & (r_1' \mid r_2') \mid r_3 \end{array}$$

$(\mathbf{R_1})$-$(\mathbf{R_3})$ are equations and $\equiv$ denotes the language equivalence between regular expressions. r_1 is maximal because it is at the topmost level of a union operator. If the topmost level operator is not union, then $r_1 = r$ (cf. $\mathbf{R_1}$). A regular expression represents a set of paths and each MUS can be regarded as a main branch from the start state to the accepting state.

Example 4.8 *A regular expression for the automaton $\mathcal{A}_{\mathfrak{D}}$ in Fig. 4.4 (right) is:*

$$r_0 = \underbrace{s_1 s_3 s_3^* s_4}_{r_1} \mid \underbrace{s_1(s_2|s_3 s_3^* s_2)(s_5 s_3 s_3^* s_2)^* s_5 s_3 s_3^* s_4}_{r_2}.$$

r_1 and r_2 are the MUS*s of r_0 with $val(r_1) = 1 \times 0.3 \times \frac{1}{1-0.5} \times 0.3 = 0.18$ and $val(r_2) = 0.82$. We can distribute $\mid$ over $.$ in r_2 and obtain two more* MUS*s instead: $r_3 = s_1 s_2 (s_5 s_3 s_3^* s_2)^* s_5 s_3 s_3^* s_4$ and $r_4 = s_1 s_3 s_3^* s_2 (s_5 s_3 s_3^* s_2)^* s_5 s_3 s_3^* s_4$. The regular expressions r_1, r_3 and r_4 characterize all paths from s_1 to s_4, which fall into the above three branches. Note that r_1* cannot *be written as $s_1 s_3^+ s_4$, since from the full form of $r_1 = (1, s_1)(0.3, s_3)(0.5, s_3)^*(0.3, s_4)$, the probability of the first s_3 is different from that of s_3^*.* ♦

4.2.3 Regular expressions as counterexamples

The equivalence of DFAs and regular expressions, as well as converting DFAs to regular expressions has been widely studied. Several techniques are known, e.g., the transitive closure method [Kle56], Brzozowski's algebraic method [Brz64][BS86], and the state elimination method [DK01][Lin01]. The *state elimination* method is based on removing states one by one, while labeling transitions by regular expressions. It terminates once only the start and accepting state remain; the transition connecting these states is labeled with the resulting regular expression. This technique is suitable for manual inspection but is less straightforward to implement. *The transitive closure method* gives a clear and simple implementation but tends to create rather long regular expressions. *The algebraic method* is elegant and generates reasonably compact regular expressions. For a more detailed comparison, see [Neu05]. In order to obtain a minimal counterexample in an *on-the-fly* manner, we take the state elimination method. This allows us to stop once the value of the obtained regular expression exceeds the probability threshold. The algebraic method does not support this.

By using regular expressions for representing counterexamples, we will, instead of obtaining evidences one by one, derive a larger number of evidences at a time, which hopefully yields a quick convergence to the required probability threshold and a clear explanation of the violation. As a result, we will not insist on obtaining the smallest counterexample but instead prefer to find the branches (MUSs) with large probabilities and short length. Thus, a (good) regular expression should be:

(i) shorter (wrt. its length), to improve comprehensibility;

(ii) more probable, such that it is more informative and the algorithm will terminate with less MUSs;

(iii) minimal, where a compact counterexample is *minimal* if the omission of any of its MUSs would no longer result in a counterexample.

However, it has been recently proven that (i) the size of a shortest regular expression of a given DFA cannot be efficiently approximated [GS07]. Therefore, it is not easy to, e.g., by state elimination, compute an optimal removal sequence for state elimination in polynomial time [HW07]. We could adapt the heuristics proposed in e.g. [HW07][DM04] to get a better order to eliminate states. For (ii), we could take the advantage of the SP algorithms as well as the model-checking results. The states on the more probable evidences should be eliminated first. The minimality (iii) will be guaranteed by the following algorithm.

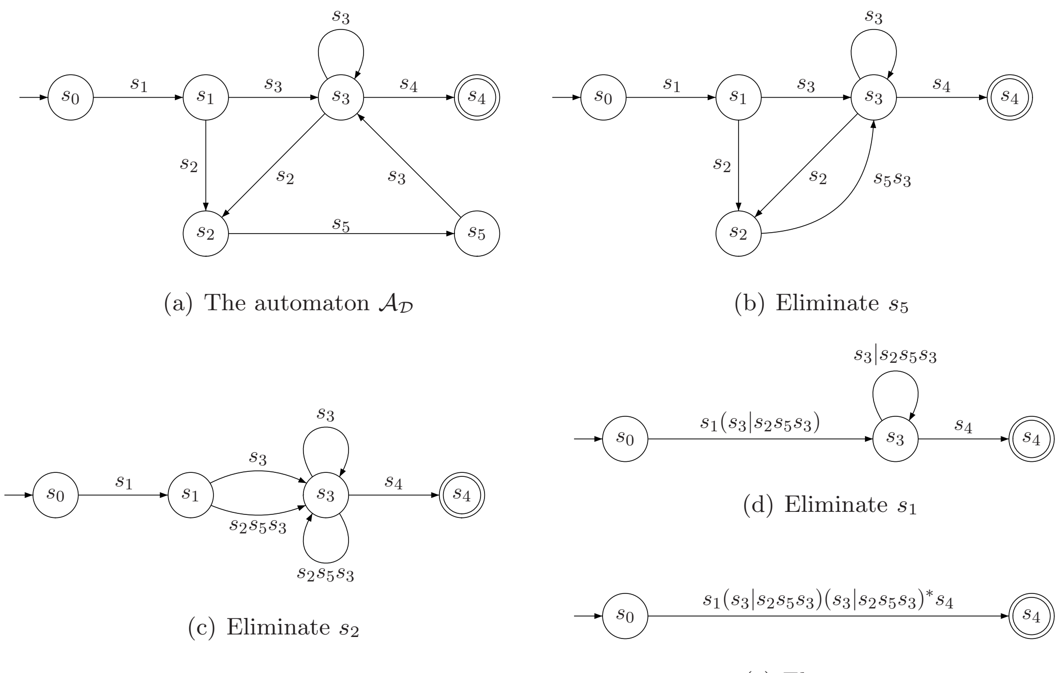
We take the following iterative strategy: In each iteration, we take the strongest evidence $\sigma = \tilde{s}{\cdot}\hat{s}{\cdot}s_1 \cdots s_j{\cdot}t$ in the remaining automaton — recall that this amounts to an SP problem — and eliminate all the intermediate states on σ (i.e., $\hat{s}, s_1, \ldots, s_j$) one by one according to a recently proposed heuristics [HW07]. After eliminating a state, possibly a new MUS r_k, say, is created where k MUSs have been created so far, and $val(r_k)$ can be determined. If $\sum_{i=1}^{k} val(r_i) > p$, then the algorithm terminates. Otherwise, the transition labeled with r_k is removed from the automaton and either a next state is selected for elimination or a new evidence is to be found, cf. Alg. 3.

Algorithm 3 Regular expression counterexamples

```
Require: DFA $\mathcal{A}_\mathcal{D} = (S, \Sigma, \tilde{s}, \delta, \{t\})$, and $p \in [0,1]$
Ensure: regular expression $r \in \mathcal{R}(\Sigma)$ with $val(r) > p$
 1: $\mathcal{A} := \mathcal{A}_\mathcal{D}$;   $pr := 0$;   Priority queue $q := \varnothing$;   $k := 1$;
 2: while $pr \leqslant p$ do
 3:     $\sigma :=$ the strongest evidence in $\mathcal{A}$ by Dijkstra's algorithm;
 4:     forall $s' \in \sigma \setminus \{\tilde{s}, t\}$ do $q$.enqueue($s'$); endforall;
 5:     while $q \neq \varnothing$ do
 6:         $\mathcal{A} :=$ eliminate($q$.dequeue());
 7:         $r_k :=$ the created MUS;
 8:         $pr := pr + val(r_k)$;
 9:         $\mathcal{A} :=$ eliminate($r_k$);
10:         if $(pr > p)$ then break; else $k := k + 1$;
11:     endwhile;
12: endwhile;
13: return $r_1 \mid \ldots \mid r_k$;
```

Priority queue q keeps the states to be eliminated in the current iteration. Those states come from the current strongest evidences. The order in which states are dequeued from q is given by the heuristics in [HW07], which will be briefly introduced later. The function "eliminate($\cdot$)" can both eliminate states and regular expressions, where the latter simply means the deletion of the transitions labeled with the regular expression.

Example 4.9 *Let us apply the algorithm to $\mathcal{A}_\mathcal{D}$ of Fig. 4.4 and consider $\mathcal{P}_{\leqslant 0.7}(\Diamond\, s_4)$.*



(a) The automaton $\mathcal{A}_{\mathcal{D}}$

(b) Eliminate s_5

(c) Eliminate s_2

(d) Eliminate s_1

(e) Eliminate s_3

Figure 4.6: An example of state elimination

In the first iteration, $s_0{\cdot}s_1{\cdot}s_2{\cdot}s_5{\cdot}s_3{\cdot}s_4$ is found as the strongest evidence. Assuming the order to eliminate the states by [HW07] *is s_5, s_2, s_1, s_3 (this will be explained in Example 4.13), we obtain the regular expression $r_0 = s_1(s_3|s_2s_5s_3)(s_3|s_2s_5s_3)^*s_4$ with $val(r_5) = 1$. The process of the elimination is illustrated in Fig. 4.6. Since all states are eliminated and the threshold 0.7 is exceeded, the algorithm terminates. This expression gives a clear reason that traversing the cycle s_3 or $s_2s_5s_3$ infinitely many times causes the probability exceeding 0.7.*

Let us change the elimination order *to s_5, s_1, s_3, s_2. Then the regular expression is $r_1 = s_1s_3s_3^*s_4 \mid s_1s_2(s_5s_3s_3^*s_2)^*s_5s_3s_3^*s_4$. After eliminating s_3, the first* MUS *$r_2 = s_1s_3s_3^*s_4$ is generated and the probability is $0.18 < 0.7$. The algorithm continues (i.e., eliminates s_2) to find more* MUS*s, till r_1 is found. Note that r_1 is longer than r_0, and thus less intuitive to comprehend. The cycles s_3 and $s_3s_2s_5$ are however indicated.*

Let us pick a less probable evidence *$s_0{\cdot}s_1{\cdot}s_3{\cdot}s_4$ to be eliminated in the first iteration. After eliminating s_1 and s_3, the resulting expression is $r_2{=}s_1s_3s_3^*s_4$. Then r_2 is removed from the automaton and the strongest evidence in the remaining automaton is $s_0{\cdot}s_2{\cdot}s_5{\cdot}s_4$. After eliminating s_5, s_2, we obtain the regular expression: $s_1s_2(s_5s_3s_3^*s_2)^*s_5s_3s_3^*s_4$. The final regular expression is again r_1 and the analysis in the last case applies.* ♦

Proposition 4.10 *The regular expression counterexample generated by Alg. 3 is minimal.*

This property immediately follows from the fact that Alg. 3 terminates immediately once the cumulative probability exceeds the threshold. We would like to emphasize that the regular expression representation is not applicable to formulae with nested probabilistic operators, e.g., $\mathcal{P}_{\leqslant p_1}\big(\Diamond\mathcal{P}_{\leqslant p_2}(\Diamond\, a)\big)$. However, this is not a real constraint in practice, since those formulae are rarely used [Gru08]. In addition, it is important to mention that the algorithm in this section not only applies to non-strict probability bounds, but also to strict bounds as, e.g., $\mathcal{P}_{<p}(\Diamond\, a)$. For instance, consider the DTMC in Fig. 3.1 (page 26). The regular expression counterexample for the violation of $\mathcal{P}_{<\frac{1}{2}}(\Diamond\, a)$ in state s is s^*t with probability $\frac{1}{2}$. Recall that the KSP characterization in Chapter 3 only works for the non-strict probability bounds.

Example 4.11 (Leader election protocol revisited) *We conclude this section by reconsidering the leader election protocol. For the original* DTMC*, the regular expression, denoted $r(N,K)$, is:*

$$start.\big((u_1|\cdots|u_i)\,.next.start\big)^*.\,(s_1|\cdots|s_j)\,.leader,$$

where start, next and leader are the obvious short forms. The regular expression lists all the unsuccessful configurations, as well as the successful ones. As a result, $|r(N,K)| = K^N{+}4$. Compared to the number of evidences computed directly, i.e., $\sum_{i=1}^{R}\#Evi(N,K,i)$, $|r(N,K)|$ is much shorter, but it still has an exponential length. On the other hand, however, the structure of $r(N,K)$ clearly indicates the reason of violation, i.e., the repeated unsuccessful configurations followed by a successful one. ♦

4.2.4 Heuristics to Obtain Shorter Regular Expressions

For the completeness of the algorithm, we now explain how the heuristics work to determine the order of the states to be eliminated. A detailed description of the state elimination algorithm can be found in [Woo87]. As summarized in [GH08], two groups of algorithms have been studied on choosing the elimination order: the first group has a tail-recursive specification and are most easily implemented by an iterative program [DM04][EKSW05][MMR05]; the other is based on the divide-and-conquer paradigm [HW07], suggesting a recursive implementation. Here we pick one representative from each group: [DM04] and [HW07] and show the main ideas.

4.2.4.1 Least Priced State First

The approach by Delgado and Morais [DM04] calculates for each state a *price*[1], which indicates how "expensive" it is to remove this state. The least priced state will be removed first.

[1]In the original text, it is called "weight". We use "price" to differentiate the weight in a weighted digraph in Chapter 3.

Let $\mathcal{A}_{\mathcal{D}} = (S', \Sigma, \tilde{s}, \delta^*, \{t\})$ be a *nondeterministic finite automaton* (NFA). Note that we allow the extended transition relation, which is the transitive closure of δ such that if r is a regular expression, then $\delta^* \subseteq S \times \mathcal{R}(\Sigma) \times S$. Graphically, the regular expression r is labeled on the edge between s and s'. For a state we define In and Out to be the number of edges going into and out from it. Let $|r_{in}^{(k)}|$ (resp. $|r_{out}^{(k)}|$) denote the size (i.e., the number of symbols) of the regular expression on the k-th incoming (resp. outgoing) edge, not including the self-loop. Similarly, $|r_{loop}|$ denotes the size of the regular expression on the self-loop (if any) on the state.

Example 4.12 *For state s_3 in Fig. 4.6(b), $In = Out = 2$, and*

$$|\underbrace{r_{in}^{(1)}}_{s_3}| = 1 \qquad |\underbrace{r_{in}^{(2)}}_{s_5 s_3}| = 2 \qquad |\underbrace{r_{out}^{(1)}}_{s_2}| = |\underbrace{r_{out}^{(2)}}_{s_4}| = 1 \qquad |\underbrace{r_{loop}}_{s_3}| = 1. \qquad \blacklozenge$$

Formally, using the convention $\sum_{j=1}^{0} a^{(j)} = 0$, the price of a state s can be computed as:

$$price = \underbrace{\sum_{j=1}^{In} \left(|r_{in}^{(k)}| \times (Out - 1)\right)}_{\text{sum of in-edge price}} + \underbrace{\sum_{j=1}^{Out} \left(|r_{out}^{(k)}| \times (In - 1)\right)}_{\text{sum of out-edge price}} + \underbrace{|r_{loop}| \times (In \times Out - 1)}_{\text{sum of loop price}}.$$

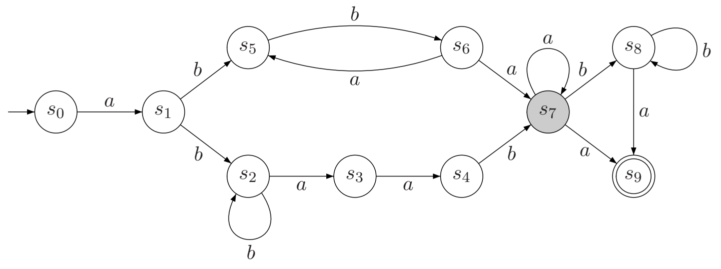
Intuitively, the *price of the graph* is the sum of the sizes of all regular expressions on the edges of the graph. The price of a state given above characterizes the price that will be added to the price of the graph by removing that state. The time to compute the price of states is negligible. The price of s_3 in the above example is 8.

As is shown by experiments in [DM04], "less price" does not necessarily mean "better choice". Some more strategies are taken to improve the heuristics, but these are more time-consuming. We don't elaborate on these extensions here.

4.2.4.2 Structure-Oriented Chopping

Han and Wood viewed the problem from the perspective of the structure of the automaton [HW07]. Essentially, the states on the important positions, denoted as *bridge states*, should be eliminated as late as possible. Bridge states s_b (i) are neither the start state $\tilde{s}$ nor the accepting state t; (ii) appear at least once on each $\tilde{s}$-t path; and (iii) once a path visited s_b, it will never visit the states prior to s_b again. For instance, the automaton in Fig. 4.7(a) has bridge states s_1 and s_7.

Vertical Chopping. Using the bridge states we can divide the automaton $\mathcal{A}$ into several sub-automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$. The chopping is illustrated as in Fig. 4.7(b). Note that the language of the automaton does not change under the vertical chopping: $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1).\mathcal{L}(\mathcal{A}_2) \ldots \mathcal{L}(\mathcal{A}_k)$. Consequently, $r = r_1.r_2 \ldots r_k$ where r and r_i are the regular expressions in $\mathcal{A}$ and $\mathcal{A}_i$, respectively. This takes $\mathcal{O}(|S| + |\delta|)$ times.


(a) Bridge states s_1 and s_7

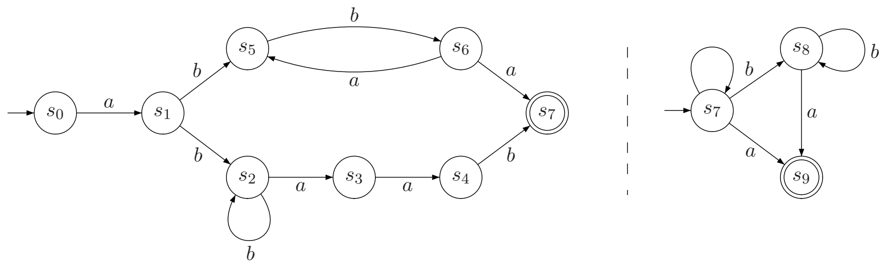

(b) Vertical chopping at state s_7

Figure 4.7: Vertical chopping

Horizontal Chopping. For each sub-automaton $\mathcal{A}_i$ that has been vertically chopped (without bridge states), we can apply the horizontal chopping illustrated in Fig. 4.8 to divide the automaton into $\mathcal{A}_{i1}, \ldots, \mathcal{A}_{i\ell}$. The language of the automaton does not change under the vertical chopping: $\mathcal{L}(\mathcal{A}_i) = \mathcal{L}(\mathcal{A}_{i1})|\mathcal{L}(\mathcal{A}_{i2})|\ldots|\mathcal{L}(\mathcal{A}_{i\ell})$. Consequently, $r_i = r_{i1}|r_{i2}|\ldots|r_{i\ell}$ where r_i and r_{ij} are the regular expressions in $\mathcal{A}_i$ and $\mathcal{A}_{ij}$, respectively. This takes $\mathcal{O}(|S| + |\delta|)$ times.

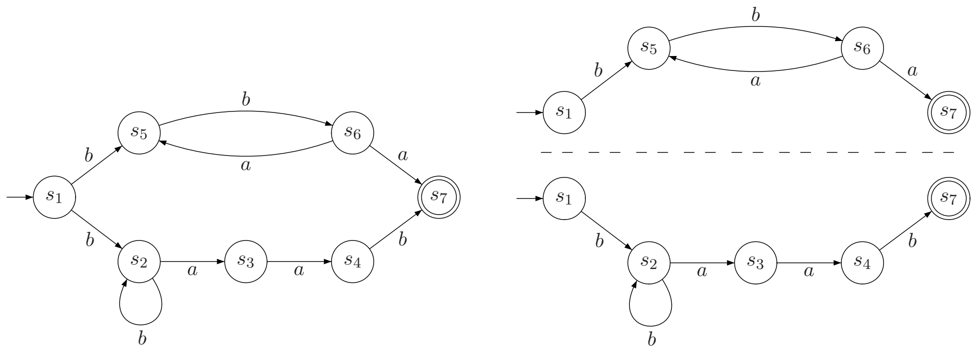


Figure 4.8: Horizontal chopping

Once the horizontal chopping is performed, some states become the bridge states of the sub-automaton and the whole procedure of "finding bridge state - vertical chopping - horizontal chopping" can be applied again, until no further chopping is possible. The removal of a sequence is easily done. In the worst case, it might be not able to perform any vertical or horizontal chopping. Brute force can be instead applied.

Example 4.13 *For the automaton in Fig. 4.6(a) (page 65), the bridge states are s_1 and s_3. By vertical chopping, we obtain three sub-automata in Fig. 4.9.*

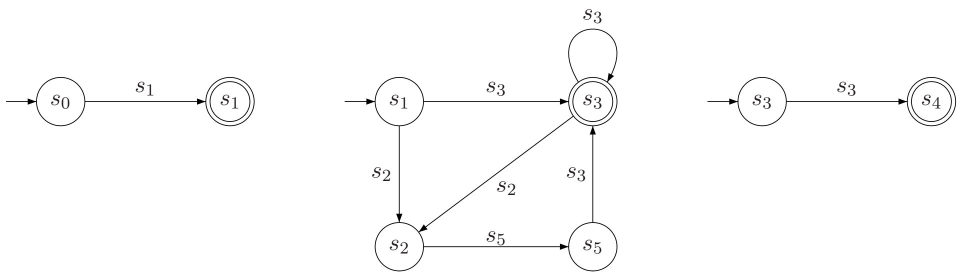


Figure 4.9: Vertical chopping of the automaton in Fig. 4.6(a)

For the automaton in the middle in Fig. 4.9, horizontal chopping can be further applied and it yields the two sub-automata in Fig 4.10. Note that the right sub-automaton in Fig 4.10 has the bridge state s_2. The above procedure means that s_2 should be eliminated after s_5, while s_1 and s_3 should remain as they are start and accepting state. ♦

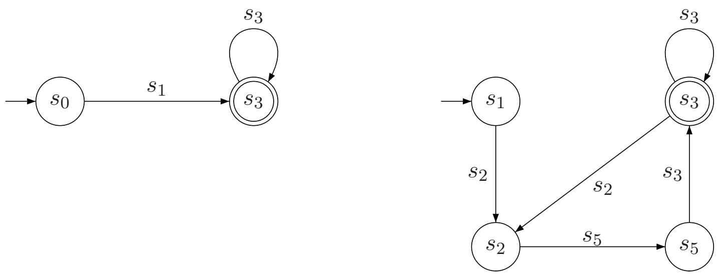


Figure 4.10: Horizontal chopping of the middle automaton in Fig. 4.9

4.3 Regular Expression Counterexamples for Bounded Reachability

For bounded reachability formula $\Diamond^{\leqslant h} at_t$, a regular expression, e.g. $r = r_1 \,|\, r_2^*$, may not be valid because it is possible that the length of the words generated by r_1 or the expansion of r_2 exceeds h. Thus, $val(r)$ might be larger than the actual probability. In order to obtain a precise valuation, we consider *constrained regular expressions*.

Definition 4.14 (Constrained regular expressions) *For $r \in \mathcal{R}(\Sigma)$ and $h \in \mathbb{N}_{\geqslant 0}$, $r\langle h\rangle$ and $r[h]$ are* constrained regular expressions *where $\mathcal{L}(r\langle h\rangle) = \{z \in \mathcal{L}(r) \mid |z| \leqslant h\}$ and $\mathcal{L}(r[h]) = \{z \in \mathcal{L}(r) \mid |z| = h\}$.*

Intuitively $r\langle i\rangle$ (resp. $r[i]$) is the regular expression that can generate the set of words with at most (resp. exactly) i symbols. In fact, $\mathcal{L}(r[h]) \subseteq \mathcal{L}(r\langle h\rangle) \subseteq \mathcal{L}(r)$ and $\mathcal{L}(r\langle h\rangle)$ can be expressed equivalently by a union of possible enumerations, namely,

$$\begin{aligned}\mathcal{L}(r\langle h\rangle) &= \mathcal{L}\big(\ r[0] \mid r[1] \mid \cdots \mid r[h]\ \big) \\ &= \mathcal{L}\big(r[0]\big) \cup \mathcal{L}\big(r[1]\big) \cup \cdots \cup \mathcal{L}\big(r[h]\big).\end{aligned}$$

Example 4.15 *Let $r = abbc \,|\, (a \,|\, bc)^*$ and $h = 3$. $\mathcal{L}(r\langle 3\rangle) = \bigcup_{i=0}^{3} \mathcal{L}(r[i])$ and*

$$\begin{array}{ll}\mathcal{L}(r[0]) = \{\varepsilon\} & \mathcal{L}(r[2]) = \{aa, bc\} \\ \mathcal{L}(r[1]) = \{a\} & \mathcal{L}(r[3]) = \{aaa, abc, bca\}\end{array}$$

Note that $abbc \in \mathcal{L}(r)$ but $abbc \notin \mathcal{L}(r\langle 3\rangle)$. ♦

Constrained regular expressions can be obtained in the same way as presented just before, only their valuation is different:

Definition 4.16 *For $r \in \mathcal{R}(\Sigma)$ and $h \in \mathbb{N}_{\geqslant 0}$, the function val for $r\langle h\rangle$ and $r[h]$ is defined by:*

$$\begin{aligned}
val(r\langle h\rangle) &= \sum_{i=0}^{h} val(r[h]) \\
val(\varepsilon[h]) &= \begin{cases} 1, & \text{if } h = 0 \\ 0, & \text{otherwise}\end{cases} \\
val((p,s)[h]) &= \begin{cases} p, & \text{if } h = 1 \\ 0, & \text{otherwise}\end{cases} \\
val((r_1|r_2)[h]) &= val(r_1[h]) + val(r_2[h]) \\
val((r_1.r_2)[h]) &= \sum_{i=0}^{h} val(r_1[i]){\cdot}val(r_2[h-i]) \\
val(r^*[h]) &= val(\varepsilon\langle h\rangle) + \sum_{i=1}^{h} val(r[i]) \cdot val(r^*[h-i])
\end{aligned}$$

Note that the complexity of the above evaluation function is, however, very high. It remains to establish that constrained regular expressions are counterexamples for bounded until-formulae.

Theorem 4.17 *Let r be the regular expression for* DFA *$\mathcal{A}_{\mathcal{D}} = (S', \Sigma, \tilde{s}, \delta, \{t\})$ where $\mathcal{D} = (S, \mathbf{P}, L)$ with initial state $\hat{s}$ and $h \in \mathbb{N}_{\geqslant 0}$. Then:*

$$\begin{aligned}val(r\langle h\rangle) &= \mathbb{P}\big(Paths^{\star}_{\min}(\hat{s}, \Diamond^{\leqslant h} at_t)\big) \\ val(r[h]) &= \mathbb{P}\big(Paths^{\star}_{\min}(\hat{s}, \Diamond^{=h} at_t)\big).\end{aligned}$$

Proof: The proof is similar to the proof for Theorem 4.6. Let $\mathbb{P}(r\langle h\rangle)$ and $\mathbb{P}(r[h])$ denote the probability of the set of paths $Paths^{\star}_{\min}(\hat{s}, \Diamond^{\leqslant h} at_t)$ and $Paths^{\star}_{\min}(\hat{s}, \Diamond^{=h} at_t)$, respectively. Those two sets only contain paths with at most and exactly h hops, respectively. We don't elaborate to do the case distinction here. A general note is that $val(r[h])$ only considers the probabilities of all paths with h hops. Furthermore, $val(r\langle h\rangle) = \sum_{i=0}^{h} val(r[h])$ guarantees that the probability of paths with more than h hops will not be considered. ■

For the more general bounded reachability properties $\Diamond^{[h_l,h_u]} at_t$, due to the two-phase nature specified in Section 3.3.4, it can be divided into generating $r_1[h_l]$ in $\mathcal{D}_1 = \mathcal{D}[\neg\Phi]$ and $r_2\langle h_u - h_l + 1\rangle$ (r_2, if $h_u = \infty$) in $\mathcal{D}_2 = \mathcal{D}[\neg\Phi \wedge \neg\Psi]\langle t_\Phi\rangle$. Thus, the regular expression will be $r_1[h_l].{}_v r_2\langle h_u - h_l + 1\rangle$, where v is the intermediate state on the h_l-th position and $._v$ denotes the concatenation on state v. Recall that M is the set of all intermediate states.

Corollary 4.18 *Let r_1 be the regular expression for* DFA $\mathcal{A}_{\mathcal{D}_1} = (S', \Sigma, \tilde{s}, \delta, \{v\})$ *where $\mathcal{D}_1$ has initial state s and let r_2 be the regular expression for* DFA $\mathcal{A}_{\mathcal{D}_2} = (S'', \Sigma, \tilde{v}, \delta', \{t\})$ *where $\mathcal{D}_2$ has initial state v and $h_l, h_u \in \mathbb{N}_{\geqslant 0}$. Then:*

$$\sum_{v\in M} val\big(r_1[h_l].{}_v r_2\langle h_u - h_l + 1\rangle\big) = \mathbb{P}\Big(Paths^{\star}_{\min}(\hat{s}, \Diamond^{[h_l,h_u]} at_t)\Big).$$

4.4 Model Reduction

Regular expression counterexamples are feasible when the excessive number of evidences are caused by traversing loops. Clearly, the number of states also affects the size of the regular expression. Thus, any model reduction prior to counterexample generation would be helpful. Two strategies may be utilized to slim down the model size, viz. *bisimulation minimization* and SCC *minimization*. Bisimulation minimization [KKZJ07][LS91] lumps bisimilar states and yields a typically much smaller quotient DTMC. Strongly-connected-component (SCC) minimization [lGM02][ADvR08], instead, only lumps SCCs and also yields a quotient DTMC. It worth mentioning the distributed implementation of both techniques: [BHKvdP08] for bisimulation and [BCvdP09] for SCC minimization. An evidence $[s_0]_R{\cdot}[s_1]_R{\cdots}[s_n]_R$ in the quotient DTMC represents a set of evidences in the original DTMC, viz., $\{s'_0{\cdot}s'_1{\cdots}s'_n \mid s'_i \in [s_i]_R \text{ and } 0 \leqslant i \leqslant n\}$, where $[s_i]_R$ is the equivalence class under equivalence R with s_i as its representative. Note that the relation R can be bisimulation $\sim$ or SCC. Bisimulation minimization preserves both unbounded and bounded probabilistic reachability properties, while SCC minimization only preserves the former one.

In the following, we continue the leader election example to show the effect of the model reduction on the size of the regular expression counterexamples.

- The regular expression counterexample in the bisimulation quotient DTMC in Fig. 4.11 is:

$$r^{\sim}(N,K) = \mathit{start}.\,(u.\mathit{next}.\mathit{start})^{*}.s.\mathit{leader},$$

  where $u_1, \ldots, u_i$ are wrapped as u; $s_1, \ldots, s_j$ as s in Fig. 4.3. Note that $|r^{\sim}(N,K)| = 6$ is independent of N and K.

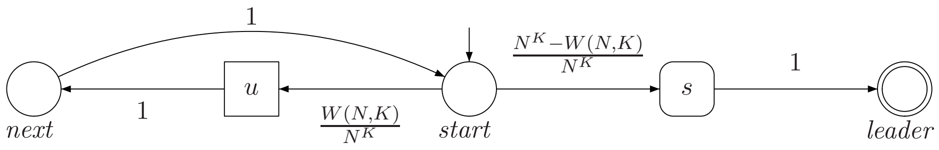


Figure 4.11: The bisimulation minimized model for leader election protocol

- The SCC-quotient DTMC is obtained by replacing the left half of the model (an SCC) by the (macro) initial state $\underline{\mathit{start}}$, as shown in Fig. 4.12. The regular expression counterexample is:

$$r_{\text{SCC}}(N,K) = \underline{\mathit{start}}.(s_1|\cdots|s_j).\mathit{leader},$$

  where the intuition of staying in $\underline{\mathit{start}}$ is "still unsuccessful".

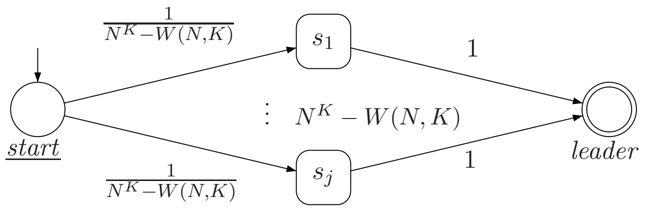


Figure 4.12: The SCC-minimized model for leader election protocol

- Applying both bisimulation- and SCC-minimization techniques yields the model in Fig. 4.13 and the regular expression counterexample is:

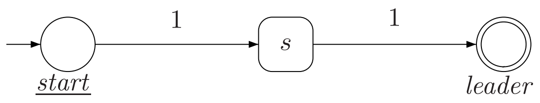


Figure 4.13: The bisimulation- and SCC-minimized model for leader election protocol

Note that in this case, too much information is abstracted away such that it only indicates that eventually with probability 1 a leader will be elected.

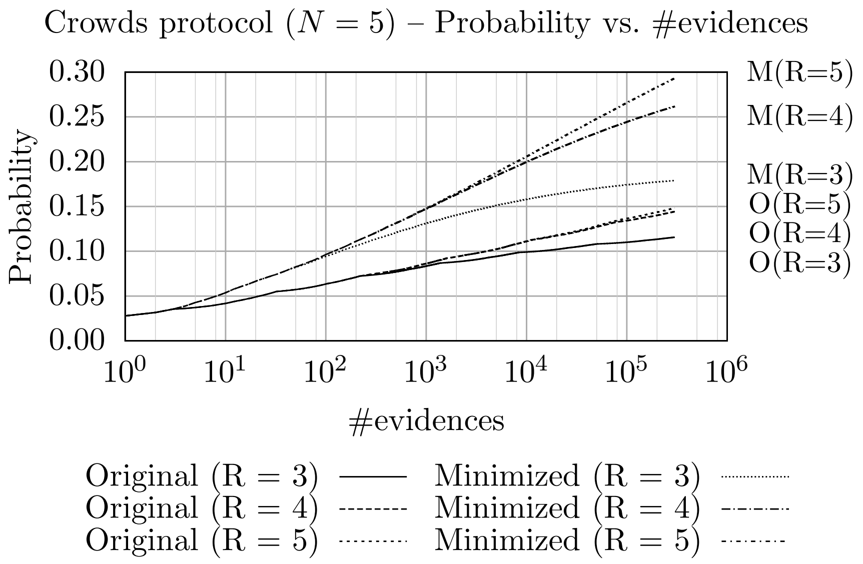


Figure 4.14: Probability vs. number of evidences for Crowds protocol ($N = 5$)

4.5 Case Study

We now illustrate our techniques on a more serious example. The *Crowds* protocol [RR98] is aimed to provide users with a mechanism for anonymous Web browsing. The main idea behind Crowds is to hide each user's communication by routing randomly within a group of similar users. Even if a local eavesdropper or a bad group member observes a message being sent by a particular user, it can never be sure whether the user is the actual sender, or is simply routing another user's message.

The protocol works in the following way: 1) The sender selects a crowd member at random (possibly itself), and forwards the message to it, encrypted by the corresponding pairwise key. 2) The selected router flips a biased coin. With probability $1 - PF$, where PF (forwarding probability) is a parameter of the system, it delivers the message directly to the destination. With probability PF, it selects a crowd member at random (possibly itself) as the next router in the path, and forwards the message to it, re-encrypted with the appropriate pairwise key. The next router repeats this step.

In our experiments, we assume that 1) if a sender has been observed by the bad member twice, then it has been *positively identified* (*Pos* for short), thus the anonymity is not preserved; 2) the bad member will deliver the message with probability 1, as in [Shm04]. This protocol is executed every time a crowd member wants to establish an anonymous connection to a Web server. We call one run of the protocol a *session* and denote the number of sessions by R. Other parameters are the number N of good members and the number M of bad members.

We take the Crowds protocol modeled using PRISM [Pri] and the property is

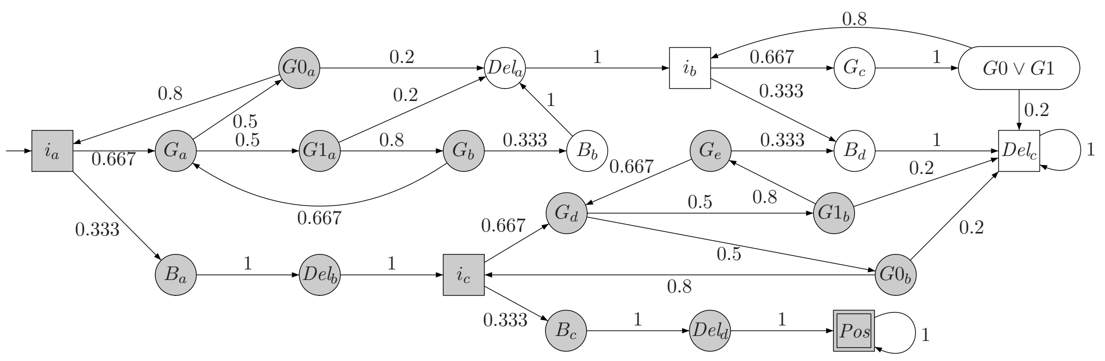


Figure 4.15: Quotient DTMC for Crowds protocol ($N = 2, M = 1, R = 2, PF = 0.8$)

$\mathcal{P}_{\leqslant p}(\Diamond\, Pos)$ which characterizes the probability threshold that the original sender's id 0 is positively identified by any of the bad members. The relation between the number of evidences and the probability threshold for different number of sessions R is shown in Fig. 4.14 ($N = 5$, $M = 1$, $PF = 0.8$), both for the original DTMC and its bisimulation quotient. As one evidence in the minimized model may represent a set of evidences in the original model, given the same number of evidences, the cumulative probability in the minimized model is higher than that in the original model. This is illustrated in Fig. 4.14 by the fact that the curves for the minimized model are above the ones for the original model. It also holds that the larger the round number we allow, the larger the cumulative probability is, as the smaller round case is subsumed in the larger round case. For instance, in the case of $R = 2$, only consecutively sending to two bad member causes a Pos (shortly as BB); whereas in the case of $R = 3$, besides BB, there are two more situations that causes Pos, viz. GBB and BGB (G represents sending to a good member). This also explains why the curves overlap in the beginning of the x-axis.

We choose a configuration with a small state space ($N = 2$, $M = 1$, $R = 2$, and $PF = 0.8$) as this suffices to illustrate the algorithm. Bisimulation minimization reduces the state space from 77 to 34; cf. the quotient DTMC in Fig. 4.15. To make the figure comprehensible, sequences of states with probability 1 are depicted by a square state. States i, G, B, Del, Pos represent $i$nitiating a new session, sending a message to a Good member, to a Bad member, a message being Delivered, a Positive result obtained, respectively. $G0$ and $G1$ are the two good members, where $G0$ is assumed always to be the original sender when a new session starts. $G0 \vee G1$ is a lumped state where either $G0$ or $G1$ is reached. The subscripts $a, b, ...$ are to distinguish the states in similar situations. Since the goal state Pos can be reached by only the gray states, the regular expression (thus the automaton) only depends on those states. Note that Del_a and Del_b denote the end of the first session, while Del_c and Del_b denote the end of the second. Only the case that two messages are both delivered by the bad member indicates a positive identification of the sender.

An intermediate automaton (see Fig. 4.16) can be derived after eliminating some states. The start state $\tilde{s}$ of the automaton is also omitted. This shows the basic structure of the model: i_a and i_c are the starting points of two sessions. The horizontal transitions indicate the observation of $G0$ by the bad member, which lead to Pos. In each session, a message can be forwarded to $G0$ or $G1$ many times (captured by the self-loops). Once a message is delivered, a new session is assumed to be started (the transitions back to i_a and i_c). Thus, a regular expression that can be generated from the automaton is $r = r_0 r_1^* r_2 r_3^* r_4$, where:

$$\begin{aligned}
r_0 &= (1, i_a),\\
r_1 &= (0.667, G_a)(0.267, G1_a.G_b.G_a)^*(0.4, G0_a.i_a),\\
r_2 &= (0.333, B_a.Del_b.i_c),\\
r_3 &= (0.667, G_d)(0.267, G1_b.G_e.G_d)^*(0.4, G0_b.i_c),\\
r_4 &= (0.333, B_c.Del_d.Pos).
\end{aligned}$$

If we omit the probabilities and the subscripts and merge the stuttering steps G, then we obtain:

$$r' = i\,\underbrace{(G.(G1.G)^*G0.i)^*}_{good}.\underbrace{(B.Del.i)}_{bad}.\underbrace{(G.(G1.G)^*G0.i)^*}_{good}.\underbrace{B}_{bad},$$

which is highly compact and informative in the sense that it indicates the observation of the bad members twice with arbitrary number of observing the good members. r' can be further compacted if the SCCs are identified and replaced by self-loops. In this case, $r'' = i.i^*.(B.Del.i).i^*.B$.

The probability of r is $val(r) = 0.274$, which coincides with the model checking result. These probabilities depend, among others, on the parameters of the protocol (N, M, R, PF, etc.). For instance, the probability of the strongest evidence is $(\frac{M}{N+M})^R = (\frac{1}{3})^2 = \frac{1}{9}$, which loops 0 times at r_1 and r_3. The probability of r_2 and r_4 is $\frac{a}{1-a} = \frac{4}{11}$, where a is the probability of the inner loop: $\frac{1}{N+M} \cdot PF \cdot (1 - \frac{M}{N+M}) = 0.267$, as is shown in the intermediate automaton. Note that this closed-form expression can now be used for arbitrary parameter values.

4.6 Summary

Motivated by some experimental results on the generation of counterexamples for model-checking DTMCs, as an alternative and more compact way of enumerating paths as counterexamples (KSP), we proposed to use regular expressions to represent

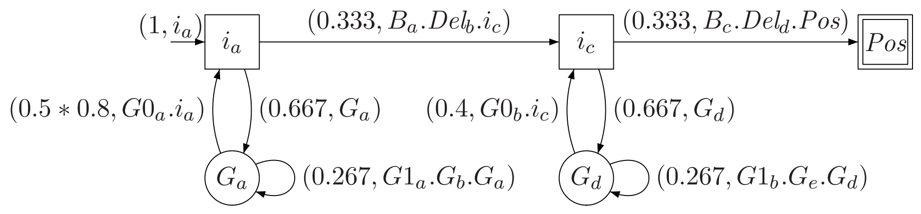


Figure 4.16: A more compact automaton

counterexamples, which is shown to be correct, and yields promising results. Bisimulation and SCC minimization may be explored to slim down the counterexample size. Constrained regular expressions for bounded-until formulae is a topic for further study as evaluating those regular expressions is expensive.

Chapter 5

Extensions

So far we have considered the generation of counterexamples for PCTL formulae of the form $\mathcal{P}_{\leqslant p}(\phi)$ (Chapter 3 and 4) and $\mathcal{P}_{<p}(\phi)$ (Chapter 4 only) for $0 < p < 1$. In this chapter, we will show how these results can be extended to

1. formulae with *lower bound on probabilities*, i.e., $\mathcal{P}_{\geqslant p}(\phi)$ and $\mathcal{P}_{>p}(\phi)$;
2. *qualitative fragment* of PCTL, i.e., $\mathcal{P}_{=1}(\phi)$ and $\mathcal{P}_{>0}(\phi)$;
3. *other discrete-time models*: MRMs and MDPs;
4. *other logics*: LTL and PCTL*; and
5. the *continuous-time model*: CTMCs.

5.1 Lower Bounds

In order to generate counterexamples for formulae of the form $\mathcal{P}_{\trianglerighteq p}(\phi)$ ($\trianglerighteq \in \{\geqslant, >\}$), we propose a reduction to the case with upper probability bounds. This is done by a transformation of the given DTMC and PCTL formula, based on which the algorithms presented before can be applied. For simplicity, we assume $\trianglerighteq = \geqslant$ and all the reasonings remain the same for $\trianglerighteq = >$ except that infinite counterexamples can only be captured by regular expressions rather than by path enumeration.

Recall that $\equiv$ is the semantic equivalence (Def. 2.17, page 17) over PCTL or PCTL* formulae. For clarity, we use $\overset{\text{PCTL}^*}{\equiv}$ instead of $\equiv$ when the right hand side is a PCTL* but not a PCTL formula (i.e., a PCTL$^*\backslash$PCTL formula). For any interval $I \subseteq \mathbb{N}_{\geqslant 0}$,

$$\begin{aligned}
\mathcal{P}_{\geqslant p}\big(\Phi \,\mathsf{U}^I\, \Psi\big) \quad &\equiv \quad \mathcal{P}_{\leqslant 1-p}\Big(\underbrace{(\Phi \wedge \neg\Psi)}_{\overset{\text{def}}{=}\ \Phi^*}\mathsf{W}^I\underbrace{(\neg\Phi \wedge \neg\Psi)}_{\overset{\text{def}}{=}\ \Psi^*}\Big) \\
&\overset{\text{PCTL}^*}{\equiv} \quad \mathcal{P}_{\leqslant 1-p}\big((\Phi^*\,\mathsf{U}^I\,\Psi^*) \vee (\Box^I \Phi^*)\big) \qquad (5.1)
\end{aligned}$$

Eq. (5.1) is due to the semantics of W^I. In the following, we show how $\mathcal{P}_{\leqslant 1-p}\big((\Phi^* \,\mathsf{U}^I\, \Psi^*) \vee (\Box^I \Phi^*)\big)$ can be further transformed to the form $\mathcal{P}_{\leqslant 1-p}(\Phi' \,\mathsf{U}^{I'}\, \Psi')$ or the like, such that our earlier algorithms can be applied. As before, we distinguish different bounds I.

5.1.1 Unbounded Until — U

Let at_b be a new atomic proposition such that $s \models at_{B_{\Phi^*}}$ iff $s \in B$ where B is a BSCC and $B \subseteq Sat(\Phi^*)$. Recall that a BSCC B is a maximal strongly connected subgraph that has no transitions leaving B. For *finite* DTMCs, we have the following theorem:

Theorem 5.1 *For any* PCTL *formulae* Φ^*, Ψ^**, it holds that:*

$$\begin{aligned}\mathcal{P}_{\leqslant 1-p}\left((\Phi^* \,\mathsf{U}\, \Psi^*) \vee (\Box\, \Phi^*)\right) &\overset{\text{PCTL}^*}{\equiv} \mathcal{P}_{\leqslant 1-p}\big((\Phi^* \,\mathsf{U}\, \Psi^*) \vee (\Phi^* \,\mathsf{U}\, at_{B_{\Phi^*}})\big) &(5.2)\\ &\equiv \mathcal{P}_{\leqslant 1-p}\big(\Phi^* \,\mathsf{U}\, (\Psi^* \vee at_{B_{\Phi^*}})\big) &(5.3)\end{aligned}$$

Proof: To show (5.2), it is sufficient to prove that $\Box\,\Phi \equiv \Phi \,\mathsf{U}\, at_{B_\Phi}$. To this end, for any $\rho \in Paths^\omega$, it holds that $\rho \models \Box\,\Phi$ iff $\forall i \geqslant 0.\ \rho[i] \models \Phi$. Since the DTMC is finite, there exists a BSCC B_Φ and $s' \in B_\Phi$, such that $\exists\, i \geqslant 0.\ \rho[i] = s'$, (i.e., $\rho[i] \models at_{B_\Phi}$) and $\forall j < i.\ \rho[j] \models \Phi$. Due to the semantics, $\rho \models \Phi \,\mathsf{U}\, at_{B_\Phi}$.

To show (5.3), it is sufficient to prove that $\Phi \,\mathsf{U}\, (\Psi \vee \Psi') \overset{\text{PCTL}^*}{\equiv} (\Phi \,\mathsf{U}\, \Psi) \vee (\Phi \,\mathsf{U}\, \Psi')$. For any $\rho \in Paths^\omega$, it holds that

$$\begin{aligned} & \rho \models (\Phi \,\mathsf{U}\, \Psi) \vee (\Phi \,\mathsf{U}\, \Psi') \\ \text{iff}\quad & \rho \models \Phi \,\mathsf{U}\, \Psi \quad \text{or} \quad \rho \models \Phi \,\mathsf{U}\, \Psi' \\ \text{iff}\quad & \exists\, i \geqslant 0.\ \rho[i] \models \Psi \text{ and } \forall j < i.\ \rho[j] \models \Phi \quad \text{or} \\ & \exists\, i \geqslant 0.\ \rho[i] \models \Psi' \text{ and } \forall j < i.\ \rho[j] \models \Phi \\ \text{iff}\quad & \exists\, i \geqslant 0.\ \rho[i] \models \Psi \vee \Psi' \text{ and } \forall j < i.\ \rho[j] \models \Phi \\ \text{iff}\quad & \rho \models \Phi \,\mathsf{U}\, (\Psi \vee \Psi') \end{aligned}$$

■

Prior to explaining the algorithms, we define a model transformation $\mathcal{D}[\![\Phi]\!]$ of $\mathcal{D}$, where we make $\mathbf{P}(s,s) = 1$ iff $s \models \Phi$ (s_Φ is made sinking). Note that in essence, this transformation is the same as $\mathcal{D}[\Phi]$ (page 26), where $\mathbf{P}(s,s) = 0$ iff $s \models \Phi$ (s_Φ is made absorbing). We distinguish the two transformations only for technical reasons.

Algorithmically, we first transform DTMC $\mathcal{D}$ to $\mathcal{D}[\![\neg\Phi^* \wedge \neg\Psi^*]\!][\![\Psi^*]\!] = \mathcal{D}[\![\neg\Phi^* \vee \Psi^*]\!]$. Those states are absorbing because the $(\neg\Phi^* \wedge \neg\Psi^*)$-states will falsify the path formula and the Ψ^*-states are the goal states. All BSCCs in $\mathcal{D}[\![\neg\Phi^* \vee \Psi^*]\!]$ are then obtained and the states in B_{Φ^*} are labeled with $at_{B_{\Phi^*}}$. The obtained DTMC now acts as the starting point for applying all the model transformations and algorithms in Chapter 3 and 4 to generate a counterexample for $\mathcal{P}_{\leqslant 1-p}\big(\Phi^* \,\mathsf{U}\, (\Psi^* \vee at_{B_{\Phi^*}})\big)$.

5.1.2 Bounded Until — $\mathsf{U}^{\leqslant h}$

For finite h, identifying all states in BSCCs B_{Φ^*} is not sufficient, as a path satisfying $\Box^{=h}\,\Phi^*$ may never reach such a BSCC. Instead, we transform the DTMC $\mathcal{D}$ to $\mathcal{D}[\![\neg\Phi^*\wedge\neg\Psi^*]\!][\![\Psi^*]\!] = \mathcal{D}[\![\neg\Phi^*\vee\Psi^*]\!]$ as in the previous case. Let $at_{\Phi^*_h}$ be an atomic proposition such that $s' \models at_{\Phi^*_h}$ iff there exists $\sigma \in \mathit{Paths}^\star(s)$ such that $\sigma[h] = s'$ and $\sigma \models \Box^{=h}\,\Phi^*$. It directly follows that $\Box^{\leqslant h}\Phi^* \equiv \Phi\ \mathsf{U}^{=h}\ at_{\Phi^*_h}$ (†). Besides, since the Ψ^*-states are absorbing in $\mathcal{D}[\![\Psi^*]\!]$, it follows that $\Phi^*\ \mathsf{U}^{\leqslant h}\ \Psi^* \equiv (\Phi^*\vee\Psi^*)\ \mathsf{U}^{=h}\ \Psi^*$ (‡). We have the following theorem:

Theorem 5.2 *For any* PCTL *formulae* Φ^*,Ψ^**, it holds in* DTMC $\mathcal{D}[\![\Psi^*]\!]$ *that:*

$$
\begin{aligned}
& \mathcal{P}_{\leqslant 1-p}\Big((\Phi^*\ \mathsf{U}^{\leqslant h}\ \Psi^*)\vee(\Box^{\leqslant h}\Phi^*)\Big) \\
\overset{\text{PCTL}^*}{\equiv}\;& \mathcal{P}_{\leqslant 1-p}\Big(\big((\Phi^*\vee\Psi^*)\ \mathsf{U}^{=h}\ \Psi^*\big)\vee(\Phi^*\ \mathsf{U}^{=h}\ at_{\Phi^*_h})\Big) && (5.4)\\
\overset{\text{PCTL}^*}{\equiv}\;& \mathcal{P}_{\leqslant 1-p}\Big(\big((\Phi^*\vee\Psi^*)\ \mathsf{U}^{=h}\ \Psi^*\big)\vee\big((\Phi^*\vee\Psi^*)\ \mathsf{U}^{=h}\ at_{\Phi^*_h}\big)\Big) && (5.5)\\
=\;& \mathcal{P}_{\leqslant 1-p}\big((\Phi^*\vee\Psi^*)\ \mathsf{U}^{=h}\ (\Psi^*\vee at_{\Phi^*_h})\big) && (5.6)
\end{aligned}
$$

Proof: (5.4) directly follows from (†) and (‡). The proof for (5.6) is very similar to that for (5.3), except for adding the hop constraint. It remains to prove (5.5). For any $\rho \in \mathit{Paths}^\omega$ in $\mathcal{D}[\![\Psi^*]\!]$, it holds that

$$
\begin{aligned}
& \rho \models \big((\Phi^*\vee\Psi^*)\ \mathsf{U}^{=h}\ \Psi^*\big)\vee(\Phi^*\ \mathsf{U}^{=h}\ at_{\Phi^*_h}) \\
\text{iff}\quad & \rho \models (\Phi^*\vee\Psi^*)\ \mathsf{U}^{=h}\ \Psi^* \text{ or } \rho \models \Phi^*\ \mathsf{U}^{=h}\ at_{\Phi^*_h} \\
\text{iff}\quad & \rho[h] \models \Psi^* \text{ and } \forall\, j<h.\ \rho[j]\models \Phi^*\vee\Psi^* \qquad \text{or} \\
& \rho[h] \models at_{\Phi^*_h} \text{ and } \forall\, j<h.\ \rho[j]\models \Phi^* \\
\text{iff}^\star\quad & \rho[h] \models \Psi^* \text{ and } \forall\, j<h.\ \rho[j]\models \Phi^*\vee\Psi^* \qquad \text{or} \\
& \rho[h] \models at_{\Phi^*_h} \text{ and } \forall\, j<h.\ \rho[j]\models \Phi^*\vee\Psi^* \\
\text{iff}\quad & \rho \models (\Phi^*\vee\Psi^*)\ \mathsf{U}^{=h}\ \Psi^* \text{ or } \rho \models (\Phi^*\vee\Psi^*)\ \mathsf{U}^{=h}\ at_{\Phi^*_h} \\
\text{iff}\quad & \rho \models \big((\Phi^*\vee\Psi^*)\ \mathsf{U}^{=h}\ \Psi^*\big)\vee\big((\Phi^*\vee\Psi^*)\ \mathsf{U}^{=h}\ at_{\Phi^*_h}\big)
\end{aligned}
$$

Now we justify why iff* holds. Let $\mathit{Paths}_1 = \{\rho \mid \rho \models \Phi^*\ \mathsf{U}^{=h}\ at_{\Phi^*_h}\}$ and $\mathit{Paths}_2 = \{\rho \mid \rho \models (\Phi^*\vee\Psi^*)\ \mathsf{U}^{=h}\ at_{\Phi^*_h}\}$. Since all Ψ^*-states are made absorbing in $\mathcal{D}[\![\Psi^*]\!]$, for any path $\rho \in \mathit{Paths}_2 \setminus \mathit{Paths}_1$, $\rho[h] \neq at_{\Phi^*}$. In other words, such ρ does not exist and $\mathit{Paths}_2 \setminus \mathit{Paths}_1 = \varnothing$. Thus iff* holds. The remaining iff's are easy to see.

■

Note that in this theorem, we use $=$ instead of $\equiv$ as the equivalence only holds in DTMC $\mathcal{D}[\![\Psi^*]\!]$ but not in all DTMCs. $=$ means the values of the probabilities are equal.

The problem of finding counterexamples for formula $\mathcal{P}_{\geqslant p}(\Phi \, \mathsf{U}^{\leqslant h} \, \Psi)$ in $\mathcal{D}$ now reduces to that of $\mathcal{P}_{\leqslant 1-p}\big((\Phi^* \vee \Psi^*) \, \mathsf{U}^{=h} \, (\Psi^* \vee at_{\Phi^*_h})\big)$ in $\mathcal{D}[\![\neg\Phi^* \vee \Psi^*]\!]$.

5.1.3 Point Interval Until — $\mathsf{U}^{=h}$, Lower-Bounded Until — $\mathsf{U}^{\geqslant h}$

For these two cases, a counterexample can be derived by checking the following two equivalent until formulae. Here we omit the proofs since they are similar to the previous two cases.

Theorem 5.3 *For any* PCTL *formulae* Φ^*, Ψ^*, *it holds that:*

$$\begin{aligned}
& \mathcal{P}_{\leqslant 1-p}\Big((\Phi^* \, \mathsf{U}^{=h} \, \Psi^*) \vee (\Box^{=h}\Phi^*)\Big) \\
\overset{\text{PCTL}^*}{\equiv} \;& \mathcal{P}_{\leqslant 1-p}\Big(\Big(\Phi^* \, \mathsf{U}^{=h} \, \Psi^*\Big) \vee (\Phi^* \, \mathsf{U}^{=h} \, at_{\Phi^*_h})\Big) \\
\equiv \;& \mathcal{P}_{\leqslant 1-p}\Big(\Phi^* \, \mathsf{U}^{=h} \, (\Psi^* \vee at_{\Phi^*_h})\Big)
\end{aligned}$$

Theorem 5.4 *For any* PCTL *formulae* Φ^*, Ψ^*, *it holds that:*

$$\begin{aligned}
& \mathcal{P}_{\leqslant 1-p}\Big((\Phi^* \, \mathsf{U}^{\geqslant h} \, \Psi^*) \vee (\Box^{\geqslant h}\Phi^*)\Big) \\
\overset{\text{PCTL}^*}{\equiv} \;& \mathcal{P}_{\leqslant 1-p}\Big(\Big(\Phi^* \, \mathsf{U}^{\geqslant h} \, \Psi^*\Big) \vee (\Phi^* \, \mathsf{U}^{\geqslant h} \, at_{B_{\Phi^*}})\Big) \\
\equiv \;& \mathcal{P}_{\leqslant 1-p}\Big(\Phi^* \, \mathsf{U}^{\geqslant h} \, (\Psi^* \vee at_{B_{\Phi^*}})\Big)
\end{aligned}$$

Note that in two cases, $\mathcal{D}$ is transformed to $\mathcal{D}[\![\neg\Phi^* \wedge \neg\Psi^*]\!]$ (or $\mathcal{D}[\neg\Phi^* \wedge \neg\Psi^*]$). In other words, Ψ^*-states should not be made absorbing, since they may be visited several times before hitting h hops.

5.1.4 Interval Until — $\mathsf{U}^{[h_l,h_u]}$

The DTMC $\mathcal{D}$ is transformed into $\mathcal{D}[\![\neg\Phi^* \wedge \neg\Psi^*]\!]$. We note that

$$\begin{aligned}
& \mathcal{P}_{\leqslant 1-p}(\Phi^* \, \mathsf{W}^{[h_l,h_u]} \, \Psi^*) \\
\equiv \;& \mathcal{P}_{\leqslant 1-p}\Big((\Phi^* \, \mathsf{U}^{[h_l,h_u]} \, \Psi^*) \vee (\Box^{[h_l,h_u]}\Phi^*)\Big) \\
\equiv \;& \mathcal{P}_{\leqslant 1-p}\Big((\Phi^* \, \mathsf{U}^{[h_l,h_u]} \, \Psi^*) \vee (\Box^{=h_u}\Phi^*)\Big) \\
\equiv \;& \mathcal{P}_{\leqslant 1-p}\Big(\underbrace{(\Phi^* \, \mathsf{U}^{[h_l,h_u]} \, \Psi^*)}_{\phi_1} \vee \underbrace{(\Phi^* \, \mathsf{U}^{=h_u} \, \underline{at}_{\Phi^*_{h_u}})}_{\phi_2}\Big)
\end{aligned} \tag{5.7}$$

Note that $s \models \underline{at}_{\Phi^*_{h_u}}$ iff $s \models at_{\Phi^*_{h_u}}$ and $s \not\models \Psi^*$. However, (5.7) cannot be transformed into an equivalent single-until formula. We therefore partition the set of paths $Paths(s, \Phi^* \, \mathsf{W}^{[h_l,h_u]} \, \Psi^*)$ into two sets $Paths(s, \phi_1)$ and $Paths(s, \phi_2)$. Note that by the

Original formula	Model trans.	Reduced formula
$\mathcal{P}_{\geqslant p}(\Phi \mathsf{U} \Psi)$	$\mathcal{D}[\![\neg\Phi^* \vee \Psi^*]\!]$	$\mathcal{P}_{\leqslant 1-p}\big(\Phi^* \mathsf{U} (\Psi^* \vee at_{B_{\Phi^*}})\big)$
$\mathcal{P}_{\geqslant p}(\Phi \mathsf{U}^{\leqslant h} \Psi)$	$\mathcal{D}[\![\neg\Phi^* \vee \Psi^*]\!]$	$\mathcal{P}_{\leqslant 1-p}\big((\Phi^* \vee \Psi^*) \mathsf{U}^{=h} (\Psi^* \vee at_{\Phi^*_h})\big)$
$\mathcal{P}_{\geqslant p}(\Phi \mathsf{U}^{=h} \Psi)$	$\mathcal{D}[\![\neg\Phi^* \wedge \neg\Psi^*]\!]$	$\mathcal{P}_{\leqslant 1-p}\Big(\Phi^* \mathsf{U}^{=h} (\Psi^* \vee at_{\Phi^*_h})\Big)$
$\mathcal{P}_{\geqslant p}(\Phi \mathsf{U}^{\geqslant h} \Psi)$	$\mathcal{D}[\![\neg\Phi^* \wedge \neg\Psi^*]\!]$	$\mathcal{P}_{\leqslant 1-p}\big(\Phi^* \mathsf{U}^{\geqslant h} (\Psi^* \vee at_{B_{\Phi^*}})\big)$
$\mathcal{P}_{\geqslant p}(\Phi \mathsf{U}^{[h_l,h_u]} \Psi)$	$\mathcal{D}[\![\neg\Phi^* \wedge \neg\Psi^*]\!]$	$\mathcal{P}_{\leqslant 1-p}\big((\Phi^* \mathsf{U}^{[h_l,h_u]} \Psi^*) \vee (\Phi^* \mathsf{U}^{=h_u} \underline{at}_{\Phi^*_{h_u}})\big)$

Table 5.1: Problem reduction for generating counterexamples for $\mathcal{P}_{\geqslant p}(\phi)$

slightly different labeling $\underline{at}_{\Phi^*_{h_u}}$ (rather than $at_{\Phi^*_{h_u}}$) the two sets are disjoint, as for any path $\rho \in \mathit{Paths}(s, \Phi^* \mathsf{W}^{[h_l,h_u]} \Psi^*)$, if $\rho[h_u] \models \Psi^*$ then $\rho \in \mathit{Paths}(s, \phi_1)$; if $\rho[h_u] \models \underline{at}_{\Phi^*_{h_u}}$ then $\rho \in \mathit{Paths}(s, \phi_2)$.

A counterexample can be computed by first computing a counterexample for $\mathcal{P}_{\leqslant 1-p}(\phi_1)$. If such a counterexample C_1 exists, then C_1 is also a counterexample for $\mathcal{P}_{\leqslant 1-p}(\Phi^* \mathsf{W}^{[h_l,h_u]} \Psi^*)$; otherwise let $p' = \mathit{Prob}(s, \phi_1)$ and then we compute a counterexample C_2 for $\mathcal{P}_{\leqslant 1-p-p'}(\phi_2)$. $C_1 \cup C_2$ is a counterexample for $\mathcal{P}_{\leqslant 1-p}(\Phi^* \mathsf{W}^{[h_l,h_u]} \Psi^*)$.

5.1.5 Summary

In the above explained way, counterexamples for different until-formulae with a lower bound on their probabilities are obtained by considering formulae on slightly adapted DTMCs with upper bounds on probabilities (cf. Table 5.1). Intuitively, the fact that s refutes $\mathcal{P}_{\geqslant p}(\phi)$ is witnessed by showing that violating paths of s are "too probable", i.e., carry more probability mass than p.

5.2 Qualitative Fragment of PCTL

Quantitative questions relate to the numerical value of the probability with which the property holds in the system; *qualitative* questions ask whether the property holds with probability 0 or 1. Typically, a qualitative property can be checked using graph analysis, i.e., by just considering the underlying digraph of the DTMC and ignoring the transition probabilities. With the qualitative fragment of PCTL we can specify properties that hold *almost surely* (i.e., with probability 1) or, dually, *almost never* (i.e., with probability 0). The qualitative fragment of PCTL only allows 0 and 1 as probability bounds and only covers unbounded (weak) until [BK08]. Formally,

Definition 5.5 ([BK08]) *State formulae in the* qualitative fragment *of* PCTL *(over*

AP) *are formed according to the following grammar:*

$$\Phi ::= \mathrm{tt} \mid a \mid \Phi \wedge \Phi \mid \neg\Phi \mid \mathcal{P}_{>0}(\phi) \mid \mathcal{P}_{=1}(\phi)$$

where $a \in \mathrm{AP}$, $\phi := \Phi_1 \,\mathsf{U}\, \Phi_2$ *is a path formula with state formulae* Φ_1, Φ_2.

Due to the fact that $\mathcal{P}_{=0}(\phi) \equiv \neg\mathcal{P}_{>0}(\phi)$ and $\mathcal{P}_{<1}(\phi) \equiv \neg\mathcal{P}_{=1}(\phi)$, it suffices to only consider formulae of the form $\mathcal{P}_{>0}(.)$ and $\mathcal{P}_{=1}(.)$. Qualitative PCTL is closely related to CTL:

Lemma 5.6 ([BK08]) *For state* s *of* DTMC $\mathcal{D}$, *it holds that:*

$$\begin{array}{lcl} s \models \mathcal{P}_{>0}(a \,\mathsf{U}\, b) & \text{iff} & s \models \exists(a \,\mathsf{U}\, b) \\ s \models \mathcal{P}_{=1}(a \,\mathsf{U}\, b) & \text{iff} & s \models \forall\big(\ \exists(a \,\mathsf{U}\, b) \ \ \mathsf{W} \ \ b\ \big) \end{array}$$

As a result, a counterexample for a qualitative PCTL property is a counterexample for the corresponding CTL formula. For the violation of CTL formula $\forall\big(\exists(a \,\mathsf{U}\, b) \,\mathsf{W}\, b\big)$ in state s, it suffices to find one path $\sigma \in \mathit{Paths}^\star(s)$ such that $\sigma \models \big(\exists(a \,\mathsf{U}\, b) \wedge \neg b\big) \,\mathsf{U}\, \big(\neg\exists(a \,\mathsf{U}\, b) \wedge \neg b\big)$. Counterexamples for formulae of the form $\exists(a \,\mathsf{U}\, b)$ can be obtained using the techniques in [SG03].

5.3 Markov Reward Models

Both DTMCs and PCTL can be extended with costs, or dually, rewards, which can specify standard and complex measures in a precise, unambiguous and clear manner. A reward function w is added to the DTMC, which associates a real reward (or cost) to any transition. Formally, $w_i : S \times S \to \mathbb{R}_{\geqslant 0}$ for $1 \leqslant i \leqslant c$, where c is the number of resources in the model. $w_i(s, s')$ denotes the reward for resource i earned when taking transition $s \to s'$. The *cumulative reward* along a finite path σ is the sum of the reward on each transition along the path. Formally, $w_i(\sigma) = \sum_{l \geqslant 0}^{|\sigma|-1} w_i(\sigma[l], \sigma[l+1])$.

Let $J_i \subseteq \mathbb{R}_{\geqslant 0}$ $(1 \leqslant i \leqslant c)$ be an interval on the real line and $p \in [0, 1]$. We use $\vec{J}$ to denote the vector of intervals, i.e., $\vec{J} = \{J_1, \ldots, J_c\}$. The formula $\mathcal{P}_{\leqslant p}(\Phi \,\mathsf{U}_{\vec{J}}\, \Psi)$ asserts that with probability at most p, Ψ will be satisfied such that all preceding states satisfy Φ, and that the cumulative reward w_i until reaching the Ψ-state lies in the interval J_i, for $1 \leqslant i \leqslant c$. The formal semantics can be found in [AHK03]. Note that the hop constraint $\leqslant h$ can be considered as a reward constraint over a simple auxiliary reward structure, which assigns cost 1 to each edge.

It holds that $s \not\models \mathcal{P}_{\leqslant p}(\Phi \,\mathsf{U}_{\vec{J}}\, \Psi)$ iff $\mathit{Prob}(s, \Phi \,\mathsf{U}_{\vec{J}}\, \Psi) > p$. As before, we cast the SE problem into an SP problem. Obviously, the weight (probability) of a path is of primary concern, which is required to be optimal. The rewards are of secondary concern; they are not required to be optimal but need to fulfill some constraints. This is exactly an instance of the (resource) constrained shortest path (CSP) problem which is NP-complete [HSP83]. Approximation or heuristic methods are surveyed in [KKKvM04]. There are some special cases of CSP problems. For the case $c = 1$ (a single resource) and if this resource increases in a constant unit for each edge (e.g., hop counts), the CSP problem, as is mentioned before, can be solved in PTIME. For the case $c = 1$ and not with a uniformly allocated resource and the case for $c = 2$, the CSP problem is *not* strong NP-complete since there are pseudo-polynomial algorithms to solve it exactly, in which the computational complexity depends on the values of edge weights in addition to the graph size [Jaf84]. The other cases are strong NP-complete problem.

For finding smallest counterexamples, we need to obtain k shortest paths subject to multiple constraints, denoted k-CSP or KMCSP [LR01], which is NP-complete. The KMCSP problem only received scant attention; an exact solution is given in [LR01].

5.4 MDP, LTL and PCTL*

In this section we will deal with counterexample generation for probabilistic model checking regarding *Markov decision processes* (MDPs), *linear temporal logic* (LTL), and PCTL*. Prior to this, we first show the relationship among various combinations of different models and logics in Figure 6.2.

Let us explain the figure in more details. Two horizontal dashed lines classify all these model checking problems according to the logic, i.e., PCTL, LTL and PCTL*, from bottom-up, respectively. The left vertical dashed line classifies them according to the model, i.e., DTMC and MDP. In MDP, we further distinguish those with or without fairness conditions by the right vertical dashed line. The solid arrow from A to B means that the model checking problem of A can be reduced to that of B. The reductions using the same technique are labeled with the same number, where:

- Reduction I transforms a PCTL* formula $\mathcal{P}_{\bowtie p}(\varphi)$ to an LTL formula φ'' by replacing all maximal state subformulae with fresh atomic propositions. Note that we don't deal with the next operator in PCTL formulae, as it is straightforward to provide a set of successor states to which the sum of the one-step probability refutes the probability bound.

- Reduction II transforms an LTL formula to a PCTL path formula (a reachability property, to be exact) by constructing the product of the model (DTMC or MDP) with the deterministic Rabin automaton that corresponds to the LTL formula. The product remains to be a DTMC or MDP and it remains to calculate the probability to reach certain "accepting states" in the product model.

- Reduction III is the link from MDPs to DTMCs, where a simple scheduler suffices to induce a DTMC from an MDP as long as only hop-unbounded until operator

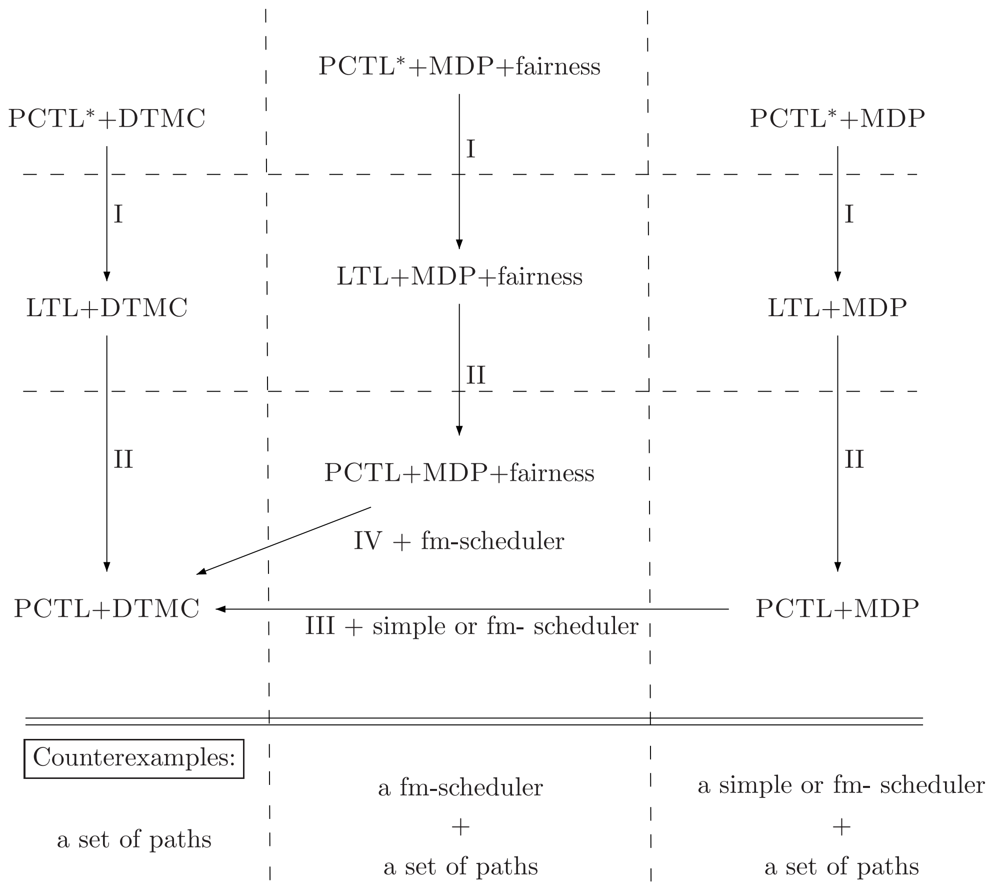


Figure 5.1: Probabilistic model checking lattice

is used in the path formulae. For other hop-bounded until operators, a finite memory scheduler is required.

- Reduction IV reduces the MDP with fairness conditions to DTMCs, where a fair scheduler should be investigated. Generally in this case, a finite-memory instead of simple scheduler suffices [BK98].

We note that all those model checking problems form a lattice, where the least element in the lattice is DTMC+PCTL. Recall that a counterexample for this is a set of paths. The same can be applied on LTL+DTMC and PCTL*+DTMC since they both can be reduced to PCTL+DTMC. For MDPs without fairness, a simple or fm-scheduler is additionally needed in order to get a DTMC. A counterexample is then composed of two parts: a simple or fm-scheduler and a set of paths in the obtained DTMC. For MDPs with fairness conditions, a finite-memory scheduler and a set of paths form a counterexample. This is shown in the lower part of Fig. 6.2. Since all cases can be finally reduced to model checking DTMC wrt. PCTL formulas, we omit the details here.

5.5 CTMC Counterexamples

This section considers the generation of evidences and counterexamples for model checking CSL [ASSB00][BHHK03] on CTMCs. For time-unbounded properties expressed in CSL e.g. $\mathcal{P}_{\leqslant 0.5}(\Phi \, \mathsf{U} \, \Psi)$, the algorithms in Chapter 3 and 4 can be exploited in the embedded DTMC (cf. page 11). Properties that involve time, e.g. $\mathcal{P}_{\leqslant 0.5}(\Phi \, \mathsf{U}^{\leqslant t} \Psi)$ $(t \in \mathbb{R})$, however, require other strategies. The continuous-time setting is unfortunately different and more complicated than the discrete one. In particular, an evidence cannot be a single timed path (an alternating sequence of states and time instants) as such paths have probability zero. Instead, we consider *symbolic evidences* for $\Phi \, \mathsf{U}^{\leqslant t} \, \Psi$, i.e., time-abstract paths—finite state sequences—that satisfy $\Phi \, \mathsf{U} \, \Psi$. A symbolic evidence induces a set of concrete evidences, viz. the set of timed paths on the same state sequence whose duration does not exceed t. A counterexample is a set of symbolic evidences such that their probability sum exceeds p.

The main contribution of this section is an algorithm for computing informative (symbolic) evidences and counterexamples, i.e., evidences with large probability and small-sized but very probable counterexamples. We first indicate how the likelihood of symbolic evidences can be computed, both numerically and analytically. The latter approach exploits the fact that symbolic evidences are in fact acyclic CTMCs for which closed-form solutions exist [MRT87] to compute the transient probability, and thus the probability of the symbolic evidence (Lemma 5.10). We then consider the problem of how to find symbolic evidences such that small counterexamples result. The strategy from Chapter 3, i.e., using KSP algorithms on a discretized CTMC (obtained by uniformization [Jen53]) is applied. Although the algorithm might not result in an optimal sequence of evidences (i.e., generating smallest counterexample cannot be guaranteed), it is simple, rather efficient, and the minimality of a counterexample can be guaranteed.

5.5.1 The Likelihood of a Symbolic Evidence

We define a *symbolic evidence* for violating $\mathcal{P}_{\leqslant p}(\Phi \, \mathsf{U}^{\leqslant t} \, \Psi)$ in state s to be a finite time-abstract path $\xi \in \mathit{Paths}_{\min}(s, \Phi \, \mathsf{U} \, \Psi)$. Actually, a symbolic evidence for $\phi = \Phi \, \mathsf{U}^{\leqslant t} \, \Psi$ is a finite path that minimally satisfies $\Phi \, \mathsf{U} \, \Psi$ and it represents a set of (infinite) timed paths in the CTMC, denoted $\mathit{Paths}_{\leqslant t}(\xi)$, such that (1) those paths have a common initial state sequence, viz. ξ; and (2) the total duration of the prefix ending at the last state of ξ is at most t. We define the probability of a symbolic evidence ξ to be $\mathbb{P}_{\leqslant t}(\xi)$, and

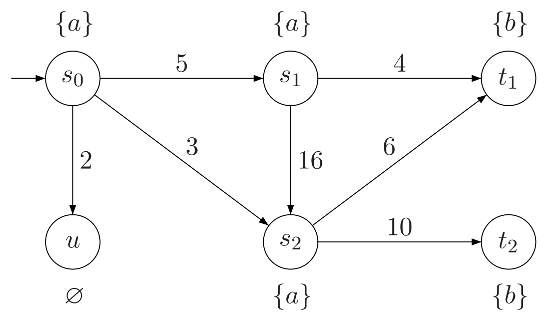


Figure 5.2: An example CTMC $\mathcal{C}$

it can be easily shown that for a set C of (prefix containment-free) symbolic evidences, the probability is $\mathbb{P}(C) = \sum_{\xi \in C} \mathbb{P}_{\leqslant t}(\xi)$. A (*symbolic*) *counterexample* for $\mathcal{P}_{\leqslant p}(\phi)$ where $\phi = \Phi \, \mathsf{U}^{\leqslant t} \, \Psi$ is a set C of symbolic evidences for ϕ such that $\mathbb{P}(C) > p$.

Example 5.7 *For the* CTMC $\mathcal{C}$ *in Fig. 5.2 and* CSL *formula* $\mathcal{P}_{\leqslant 0.45}(a \, \mathsf{U}^{\leqslant 1} \, b)$, *the symbolic evidences are* $\xi_1 = s_0{\cdot}s_2{\cdot}t_2$, $\xi_2 = s_0{\cdot}s_1{\cdot}s_2{\cdot}t_2$, $\xi_3 = s_0{\cdot}s_1{\cdot}t_1$, *and so on. These paths all minimally satisfy* $a \, \mathsf{U} \, b$ *in the embedded* DTMC*. The symbolic evidence* ξ_2 *contains e.g. the following* CTMC *path:* $s_0 \xrightarrow{0.5} s_1 \xrightarrow{0.25} s_2 \xrightarrow{0.05} t_2 \in Paths_{\leqslant 1}(\xi_2)$, *as the duration of the path is* $0.5 + 0.25 + 0.05 < 1$. *Without specifying the details (will become clear later), the probabilities of the symbolic evidences are:* $\mathbb{P}_{\leqslant 1}(\xi_1) = 0.24998$, $\mathbb{P}_{\leqslant 1}(\xi_2) = 0.24994$ *and* $\mathbb{P}_{\leqslant 1}(\xi_3) = 0.16667$. $C = \{\xi_1, \xi_2\}$ *is a counterexample since* $\mathbb{P}(C) > 0.45$, *but* $C' = \{\xi_1, \xi_3\}$ *is not.* ♦

Assume we have symbolic evidence $\xi = s_0{\cdot}s_1 \cdots s_l$ at our disposal. The probability $\mathbb{P}_{\leqslant t}(\xi)$ of this evidence—in fact, the probability of all concrete evidences of ξ up to time t—is given by:

$$\int_0^t \left(\underbrace{den(s_0, s_1, t_0)}_{\text{one step}} \cdot \underbrace{\left(\cdots \left(\int_0^{t - \sum_{i=0}^{l-2} t_i} den(s_{l-1}, s_l, t_{l-1}) \cdot dt_{l-1} \right) \cdots \right)}_{\text{the rest } s_1 \cdots s_l} \right) dt_0 \tag{5.8}$$

where $den(s_0, s_1, t_0)$ is the probability density function (cf. page 11) of $s_0 \rightarrow s_1$ winning the race at time instant t_0 in the interval $[0, t]$. The corresponding probability is thus derived by the outermost integral. Suppose the transition $s_0 \rightarrow s_1$ takes place at time instant t_0. Then the possible time instant for the second transition $s_1 \rightarrow s_2$ to take place is in $[0, t - t_0]$. This determines the range of the second outermost integral. The rest is likewise. The innermost integral determines the residence time in state s_{l-1}, the one-but-last state in ξ.

To avoid computing this (somewhat involved) integral directly by numerical techniques we resort to a simpler technique. The main idea is to isolate the time-abstract

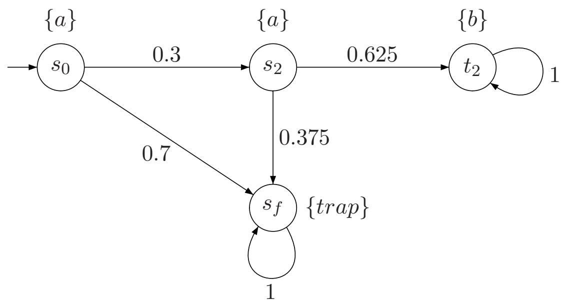


Figure 5.3: CTMC $\mathcal{C}_{\xi_1}$ induced by symbolic evidence ξ_1

path ξ from the entire CTMC. This yields a simple acyclic CTMC, i.e., an acyclic phase-type distribution [Neu81] which can be solved either analytically or numerically.

Transformation into an Acyclic CTMC. Consider CTMC $\mathcal{C}$ and CSL path-formula $\phi = \Phi \, \mathsf{U}^{\leqslant t} \, \Psi$. Note that the CTMCs are assumed to have been subject to the transformation $\mathcal{C}[\![\neg\Phi \wedge \neg\Psi]\!][\![\Psi]\!] = \mathcal{C}[\![\neg\Phi \vee \Psi]\!]$ (cf. [BHHK03], or the discrete-time case, page 78). For each evidence ξ, we may compute its probability by first constructing a CTMC $\mathcal{C}_\xi$ and then computing some transient probability in $\mathcal{C}_\xi$.

Definition 5.8 (CTMC induced by symbolic evidence) *Let* $\mathcal{C} = (S, \mathbf{P}, E, L)$ *be a* CTMC *and* $\xi = s_0{\cdot}s_1 \cdots s_l$ *be a symbolic evidence in which all states are pairwise distinct*[1]*. The* CTMC $\mathcal{C}_\xi$ *induced by* ξ *on* $\mathcal{C}$ *is defined by* $\mathcal{C}_\xi = (S_\xi, \mathbf{P}_\xi, E_\xi, L_\xi)$ *where:*

- $S_\xi = \{s_0, \ldots, s_l, s_f\}$ *with* $s_f \notin S$*;*
- $\mathbf{P}_\xi(s_i, s_{i+1}) = \mathbf{P}(s_i, s_{i+1})$, $\mathbf{P}_\xi(s_i, s_f) = 1 - \mathbf{P}_\xi(s_i, s_{i+1})$ *for* $0 \leqslant i < l$ *and* $\mathbf{P}_\xi(s, s) = 1$ *for* $s = s_l$ *or* $s = s_f$*;*
- $E_\xi(s_i) = E(s_i)$ *and* $E_\xi(s_f) = 0$*;*
- $L_\xi(s_i) = L(s_i)$ *and* $L_\xi(s_f) = \{trap\}$*.*

Stated in words, $\mathcal{C}_\xi$ is the CTMC obtained from $\mathcal{C}$ by incorporating all states in ξ, and deleting all outgoing transitions from these states except $s_i \rightarrow s_{i+1}$. The total probability mass of these omitted transitions becomes the probability to move to the trap state s_f. It follows directly that $\mathcal{C}_\xi$ is acyclic when ignoring the self-loops of the absorbing states.

Example 5.9 *Consider* CTMC $\mathcal{C}$ *in Fig. 5.2 and symbolic evidence* $\xi_1 = s_0{\cdot}s_2{\cdot}t_2$*. The induced* CTMC $\mathcal{C}_{\xi_1}$ *is shown in Fig. 5.3.* ♦

[1]This is not a restriction since it is always possible to rename a state along ξ while keeping e.g. its exit rate and labeling the same.

The following result states that computing the probability of symbolic evidence ξ boils down to a (standard) transient analysis of the induced CTMC by ξ.

Lemma 5.10 *For* CTMC $\mathcal{C}$ *and symbolic evidence* ξ *for* $\phi = \Phi \, \mathsf{U}^{\leqslant t} \, \Psi$, $\mathbb{P}^{\mathcal{C}}_{\leqslant t}(\xi) = \wp^{\mathcal{C}_\xi}_{s_l}(t)$, *where* $\wp^{\mathcal{C}_\xi}_{s_l}(t)$ *is the transient probability of being in state* s_l*, the last state of* ξ*, at time* t *given the initial state* s_0 *in the* CTMC $\mathcal{C}_\xi$.

Proof: Since ξ is a symbolic evidence, it follows that

$$\mathbb{P}^{\mathcal{C}}_{\leqslant t}(\xi) = \Pr\left\{\rho \in \mathit{Paths}^{\omega}_{\mathcal{C}}(s_0) \mid \xi = \mathit{abs}(\rho)[..l] \ \wedge \ \exists x \leqslant t.\, \rho@x = s_l\right\}.$$

Note that the function *abs* abstracts a timed path to be a time-abstract path and $\rho'[..l]$ is the prefix until s_l in path ρ'. Since the Ψ-state s_l is made absorbing in $\mathcal{C}$, it follows that once ρ reaches s_l at time instant x, it will stay there at all later time instants. This reduces the above equation to

$$\mathbb{P}^{\mathcal{C}}_{\leqslant t}(\xi) = \Pr\left\{\rho \in \mathit{Paths}^{\omega}_{\mathcal{C}}(s_0) \mid \xi = \mathit{abs}(\rho)[..l] \ \wedge \ \rho@t = s_l\right\}.$$

Due to the construction of $\mathcal{C}_\xi$, only the paths that go along ξ can reach s_l. Moreover, s_l is made absorbing and the exit rate of each state on ξ as well as their rate to its successor on ξ remain unchanged. Thus,

$$\mathbb{P}^{\mathcal{C}}_{\leqslant t}(\xi) = \Pr\left\{\rho \in \mathit{Paths}^{\omega}_{\mathcal{C}_\xi}(s_0) \mid \rho@t = s_l\right\} = \wp^{\mathcal{C}_\xi}_{s_l}(t). \qquad \blacksquare$$

This result enables us to exploit the well-known algorithms for the transient analysis of CTMCs to determine the likelihood of a symbolic evidence. By uniformization techniques, it yields an *approximate* solution up to an a priori user-defined accuracy and is part of the standard machinery in model checkers such as PRISM [KNP04] and MRMC [KKZ05]. Alternatively, we can exploit the fact that $\mathcal{C}_\xi$ is acyclic (when ignoring the self-loops at the absorbing states) and use the closed-form expression for transient distributions in acyclic CTMCs as proposed by Marie et al. [MRT87]. This yields an *exact* solution.

5.5.2 Finding Probable Symbolic Evidences

The remainder of the section is concerned with determining (symbolic) counterexamples and symbolic evidences. As in conventional model checking, the intention is to obtain *comprehensible* counterexamples. We interpret this as counterexamples of minimal cardinality (i.e., it would not be a counterexample when lacking any symbolic evidence it contains). The framework for generating such counterexamples iteratively is presented in Alg. 4.

Algorithm 4 Minimal counterexample generation for CTMCs

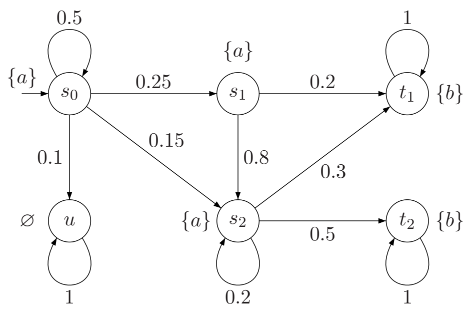
```
Require: CTMC $\mathcal{C}$, CSL formula $\mathcal{P}_{\leqslant p}(\Phi \, \mathsf{U}^{\leqslant t} \, \Psi)$
Ensure: $C = \{\xi^1, \ldots, \xi^k\}$ with $\mathbb{P}(C) > p$
1: $k := 1$;    $pr := 0$;
2: while $pr \leqslant p$ do
3:     determine symbolic evidence $\xi^k$;
4:     compute $\mathbb{P}_{\leqslant t}(\xi^k)$;
5:     $pr := pr + \mathbb{P}_{\leqslant t}(\xi^k)$;
6:     $k := k + 1$;
7: end while;
8: return   $\xi^1, \ldots, \xi^k$;
```

The termination of this algorithm is guaranteed as the violation of the property has been already established prior to invoking it. Evidently, the smaller the index k is, the more succinct the counterexamples are. We have already shown how to determine $\mathbb{P}_{\leqslant t}(\xi)$, i.e., the probability of a symbolic evidence (cf. line 4). It remains to clarify how symbolic evidences can be obtained in such a way that small counterexamples may result (line 3). As symbolic evidences are just state sequences, the first idea is to reduce the CTMC to a DTMC and adapt the algorithms (KSP and its variants) in Chapter 3.

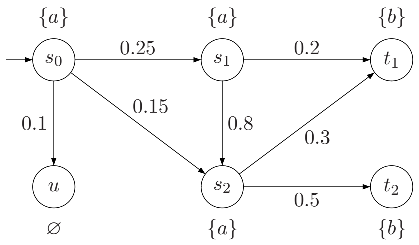
Model Discretization. There are two ways to discretize a CTMC: the embedded DTMC and the uniformized DTMC. First consider the uniformized DTMC $\mathcal{U}$ and its variant $\mathcal{U}^{\otimes}$ where all the self-loops are removed. Note that $\mathcal{U}^{\otimes}$ is an FPS instead of a DTMC. If $\mathcal{U}^{\otimes}$ is normalized, we would obtain the embedded DTMC of $\mathcal{C}$. The embedded DTMC does not involve any timing aspects as the probabilities therein only characterize the races of transitions after the delay, while the probabilities in $\mathcal{U}^{\otimes}$ also take delays into consideration. In particular, with less probable outgoing transitions from state s, it is more probable (and thus takes longer) to stay in s.

While a set of paths (traversing the same loop(s) for different times) in $\mathcal{U}$ corresponds to the same time-abstract path in $\mathcal{C}$ (many-to-one), each path in $\mathcal{U}^{\otimes}$ corresponds to a time-abstract path in $\mathcal{C}$ (one-to-one). This explains why we consider $\mathcal{U}^{\otimes}$ instead of $\mathcal{U}$. Besides, the probability of the removed self-loops (viz., delays) can be recovered easily by taking one minus the total probability of a state.

To obtain a sequence of symbolic evidences (line 3 in Alg. 4), the KSP algorithm can be applied to the weighted digraph for $\mathcal{U}^{\otimes}$. We omit the details here. Since every path the KSP algorithm generates is a symbolic evidence, it greatly decreases the time of computation, thus the whole procedure may terminate rather quickly. However, since the time bound in the property is not considered, it cannot always guarantee that the more probable symbolic evidence is generated earlier, i.e., $\mathbb{P}_{\leqslant t}(\xi^i) \geqslant \mathbb{P}_{\leqslant t}(\xi^j)$ for $i < j$



(a) Uniformized DTMC $\mathcal{U}$



(b) $\mathcal{U}^{\otimes}$

Figure 5.4: Model transformation

might be possible. The following example illustrates this.

Example 5.11 *Consider the* CTMC $\mathcal{C}$ *in Fig. 5.2 and formula* $\Phi \, \mathsf{U}^{\leqslant t} \, \Psi$*. The uniformized* DTMC $\mathcal{U}$ *and* $\mathcal{U}^{\otimes}$ *with the uniformization rate* $q = 20$ *are illustrated in Fig. 5.4. For symbolic evidences* $\xi_2 = s_0{\cdot}s_1{\cdot}s_2{\cdot}t_2$ *and* $\xi_3 = s_0{\cdot}s_2{\cdot}t_1$*, the probabilities are shown in the following table:*

		$\xi_2 = s_0{\cdot}s_1{\cdot}s_2{\cdot}t_2$		$\xi_3 = s_0{\cdot}s_2{\cdot}t_1$
$\mathcal{U}^{\otimes}$		0.100	$<$	0.045
$\mathcal{C}$	$t = 0.1$	0.04478	$<$	0.06838
$\mathcal{C}$	$t = 1$	0.24994	$>$	0.16362

For different time bounds (i.e., $t = 0.1$ *and* 1*), the order of the probabilities of two symbolic evidences in* $\mathcal{C}$ *might be reversed, however, this cannot be reflected by their probabilities in* $\mathcal{U}^{\otimes}$*, which has only one fixed order. Note that the probabilities of the two paths in* $\mathcal{U}^{\otimes}$ *are computed as in DTMCs.* ♦

This implies that for symbolic evidences ξ and ξ' and arbitrary time bound t, $\mathbb{P}^{\mathcal{U}^{\otimes}}(\xi) > \mathbb{P}^{\mathcal{U}^{\otimes}}(\xi')$ cannot guarantee that $\mathbb{P}^{\mathcal{C}}_{\leqslant t}(\xi) > \mathbb{P}^{\mathcal{C}}_{\leqslant t}(\xi')$. A direct consequence is that the sequence of generated symbolic evidences might not be optimal. The counterexamples may thus be less comprehensive, because evidences with large probability might not be included, instead more low-probability evidences are contained. In [HK07b] we proposed an algorithm that attempts to overcome this problem by taking the time bound into account and some heuristics to improve its performance. This algorithm may derive a better sequence of evidences than the one stated above, however, it usually takes much more time. We therefore omit the details here and refer the reader to [HK07b].

5.6 Summary

In this chapter we extend the work in Chapter 3 and 4 to more probabilistic models and logics. We first discuss the path enumeration approach (Chapter 3). For most of the extensions, e.g., formulae with lower-bounds on probability, MDPs or even CTMCs, they can be reduced to the base case, i.e., DTMC with $\mathcal{P}_{\leqslant p}(\phi)$. For counterexamples for the qualitative fragment of PCTL due to its equivalence with CTL formulae, the reduction to CTL counterexample generation becomes natural. Counterexamples for Markov reward models follow the similar scheme as the base case, but require more advanced shortest paths algorithms (NP-complete). When considering the regular expression counterexample approach (Chapter 4), it is applicable when a DTMC and a reachability property is at our disposal, which is true for most cases above except for the reward models. The problem now becomes that in a *weighted* automaton, given two states and a reward bound, how to generate a regular expression that corresponds to the set of paths between the two states whose accumulative reward is less than the bound. This might be an interesting future work.

5.6.1 Related Work

Counterexample generation for probabilistic model checking and the possible application of counterexamples are recently a hot research topic. We summarize some of the most relevant works here:

Directed Explicit State Space Search. Aljazzar and Leue investigated to apply directed explicit state space search to counterexample generation in DTMC and CTMC model checking of PCTL and CSL properties in [AHL05][AL06]. The presented search algorithms can explore the state space on the fly and can be further guided by heuristics to improve the performance of the algorithm. The algorithms are implemented based on PRISM and a number of case studies has been reported to illustrate its applicability, efficiency and scalability.

Counterexamples for cpCTL. Andrés and van Rossum [AvR08] introduced the logic cpCTL, which extends PCTL with conditional probability, allowing one to express that the probability that "ϕ is true given that ψ is true" is at least p. The algorithms for model checking DTMCs as well as MDPs against cpCTL have been studied. Furthermore, the notion of counterexamples for cpCTL model checking was presented together with an algorithm to generate such counterexamples. A counterexamples for DTMCs is defined along the similar fashion as in this dissertation but with a pair of measurable sets of paths. This pair of sets of paths consists of the set of the "condition" paths and the "event" paths. (Recall the conditional probability characterizes the probability of certain event given certain condition.) The counterexamples for MDPs contain additionally a scheduler.

Counterexamples for LTL. Schmalz, Varacca and Völzer proposed in [SVV09] how to present and compute a counterexample in probabilistic LTL model checking for DTMCs. In qualitative probabilistic model checking, a counterexample is presented as a pair (α, γ), where α and γ are finite words such that all paths that extend α and have infinitely many occurrences of γ violate the specification. In quantitative probabilistic model checking, a counterexample is presented as a pair (W, R) where W is a set of such finite words α and R is a set of such finite words γ. Moreover, it is also presented how the counterexample can help the user to identify the underlying error in the system through an interactive game with the model checker.

Counterexamples for MDPs.

- Andrés, D'Argenio and van Rossum [ADvR08] also observed the potential problem of the excessive number of evidences and proposed the notion of "witnesses" that groups together the evidences belonging to the same (macro) evidence in the SCC-quotient DTMC or MDP. A SCC minimization algorithm was presented and it turns out that the problem of generating counterexamples for general properties over MDPs can be reduced, in several steps, to the easy problem of generating counterexamples for reachability properties over acyclic DTMCs.

- The algorithm of SCC minimization in a Markov chain has been proposed by le Guen and Marie [lGM02].

- Aljazzar and Leue also considered generating counterexamples for MDPs against a fragment of PCTL viz. path formulae with hop-unbounded until operator in [AL09]. Three methods are presented and compared through experiments. The first two methods are the one proposed in this thesis and its improved version by using K* algorithm [AL08b] instead of Eppstein's algorithm or REA. The third method applies K* directly on MDPs to collect the most probable MDP execution traces contributing to the violation. AND/OR trees are then adopted to adequately extract from the collected execution sequences the most informative counterexample and to compute its probability. The experiments demonstrate the conditions under which each method is appropriate to be used.

Probabilistic CEGAR for MDPs.

- Hermanns, Wachter and Zhang discussed how counterexample-guided abstraction refinement can be developed in a probabilistic setting [HWZ08]. By using predicate abstraction for a guarded command language, an MDP is derived. The probabilistic reachability property was studied. Counterexamples are applied to guide the abstraction-refinement in a novel way using the strongest evidence idea of this dissertation: a counterexample is a finite set *afp* of abstract finite paths carrying enough abstract probability mass. The problem of computing the realizable probability mass of *afp* is formulated in terms of a weighted MAX-SMT problem. The set *afp* is built incrementally in an on-the-fly manner, until either enough probability is realizable, or *afp* cannot be enriched with sufficient probability mass to make the probability threshold realizable, in which case the counterexample is spurious.

- Chadha and Viswanathan presented a CEGAR framework for MDPs, where an MDP $\mathcal{M}$ is abstracted by another MDP $\mathcal{A}$ defined using an equivalence on the states of $\mathcal{M}$ [CV08]. A counterexample for an MDP $\mathcal{M}$ and property Ψ will be a pair $(\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ is an MDP violating the property Ψ that is simulated by $\mathcal{M}$ via the relation $\mathcal{R}$. This simulation relation is implicitly assumed to be part of the counterexample. Besides the definition for the notion of a counterexample, the main contributions were the algorithms to compute counterexamples, check their validity and perform automatic refinement based on an invalid counterexample.

Bounded Model Checking for Counterexample Generation. Wimmer, Braitling and Becker [WBB09] proposed to apply bounded model checking to generate counterexamples for DTMCs. Novel optimization techniques like loop-detection are applied not only to speed up the counterexample computation, but also to reduce the size of the counterexamples significantly. Some experiments were reported to demonstrate the practical applicability of the method.

Counterexample Visualization. Aljazzar and Leue presented an approach to support the debugging of stochastic system models using interactive visualization [AL08a]. It is to facilitate the identification of causal factors in the potentially very large sets of execution traces that form counterexamples in stochastic model checking. The visualization is interactive and allows the user to focus on the most meaningful aspects of a counterexample. The visualization method has been implemented in the prototype tool DIPRO and some case studies were reported.

Proof Refutations for Probabilistic Programs. McIver, Morgan and Gonzalia studied the convincing presentation of counterexamples to a proposed specification-to-implementation claim $\mathit{spec} \sqsubseteq \mathit{imp}$ (refinement) in the context of probabilistic systems [MMG08]. A geometric interpretation of the probabilistic/demonic semantic domain allows both refinement success and refinement failure to be encoded as linear satisfaction problems, which can then be analyzed automatically by an SMT solver. This allows the generation of counterexamples in an independently and efficiently checkable form. In many cases the counterexamples can subsequently be converted into “source level” hints for the verifier.

Chapter 6

Parameter Synthesis

This chapter studies a parametric version of CTMCs, a novel variant of CTMCs in which rate expressions over variables (or parameters) with bounded range indicate the average speed of state changes. The rate expressions are expressed using polynomial rings over reals, allowing e.g., $3\vartheta_1 \cdot \vartheta_2$ or $\vartheta_1^3 - \vartheta_1 \cdot \vartheta_2$. Given a time-bounded reachability property with a probability threshold, the main task is to find the set of parameter values (forming the synthesis region) that can guarantee the validity of the property on the derived CTMCs.

To this end, we propose a symbolic and a non-symbolic approach to approximate the synthesis region. The symbolic approach amounts to solving a polynomial function over the rate parameters, while the non-symbolic approach is to first instantiate the parameters and then reduce the problem to the non-parametric CTMC case. Both approaches are based on a grid discretization on the parameter ranges together with a refinement technique. The feasibility of each proposed approach is shown by synthesizing the parameter ranges for a storage system that incorporates error checking, i.e., after each operation (read/write), there is a possibility to check whether an error occurred [CY95a]. Finally we compare the two approaches and analyze the respective error bound and time complexity.

6.1 Parametric CTMCs

Parameters and Constraints. For a set $\mathcal{X}$ of m variables (or parameters) $x_1, \ldots, x_m$, expressions in the polynomial ring $\mathbb{R}[\mathcal{X}]$ over the reals $\mathbb{R}$ are formed by the grammar:

$$\vartheta ::= c \mid x \mid \vartheta + \vartheta \mid \vartheta \cdot \vartheta,$$

where $\vartheta \in \mathbb{R}[\mathcal{X}]$, $c \in \mathbb{R}$ and $x \in \mathcal{X}$. The operations $+$ and $\cdot$ are addition and multiplication, respectively. A *valuation* is a function $v : \mathcal{X} \to \mathbb{R}$ assigning a real value to a variable. We assume that all variables have a closed interval with finite bounds as the range of possible values, i.e., $v(x_i) \in [l_i, u_i]$ and $l_i, u_i \in \mathbb{R}$, for $1 \leqslant i \leqslant m$. The set of all valuation functions is $\mathbb{R}^{\mathcal{X}}$. $\vartheta[v]$ denotes the valuation of polynomial $\vartheta \in \mathbb{R}[\mathcal{X}]$ by instantiating $x_i \in \mathcal{X}$ by $v(x_i)$. Note that we do not restrict to linear expressions as this will not simplify matters in our setting (as explained later).

An *atomic constraint* is of the form $\vartheta \bowtie c$, where $\vartheta \in \mathbb{R}[\mathcal{X}]$, $\bowtie \in \{<, \leqslant, >, \geqslant\}$ and $c \in \mathbb{R}$. A *constraint* is a conjunction of atomic constraints. A valuation v satisfies the constraint $\vartheta \bowtie c$ if $\vartheta[v] \bowtie c$. A *region* $\zeta \subseteq \mathbb{R}^{\mathcal{X}}$ is a set of valuations satisfying a constraint.

Definition 6.1 (pCTMCs) *A* parametric continuous-time Markov chain *over the set $\mathcal{X}$ of parameters is a triple $\mathcal{C}^{(\mathcal{X})} = (S, \mathbf{R}^{(\mathcal{X})}, L)$, where S is a finite state space, L is the labeling function and $\mathbf{R}^{(\mathcal{X})} : S \times S \to \mathbb{R}[\mathcal{X}]$ is the parametric rate matrix.*

As in CTMCs, we assume s_0 to be the unique initial state for simplicity. The parametric infinitesimal matrix $\mathbf{Q}^{(\mathcal{X})}$ and the exit rate vector $\vec{E}^{(\mathcal{X})}$ are defined in a similar way as $\mathbf{R}^{(\mathcal{X})}$.

Definition 6.2 (Instance CTMCs) *For pCTMC $\mathcal{C}^{(\mathcal{X})}$ and valuation function v, $\mathcal{C}^{(\mathcal{X})}[v]$ (or simply $\mathcal{C}[v]$ when $\mathcal{X}$ is clear from the context) is the* instance CTMC *of $\mathcal{C}^{(\mathcal{X})}$ obtained by instantiating $x_i \in \mathcal{X}$ by $v(x_i)$.*

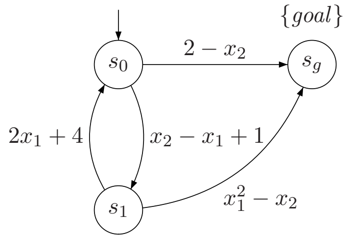
For a pCTMC $\mathcal{C}^{(\mathcal{X})} = (S, \mathbf{R}^{(\mathcal{X})}, L)$ over the set of parameters $\mathcal{X} = \{x_1, \ldots, x_m\}$, the *initial region* ζ_0 satisfies the following constraints:

$$\underbrace{\bigwedge_{i=1}^{m} l_i \leqslant x_i \leqslant u_i}_{\text{range constraints}} \wedge \underbrace{\bigwedge_{s,s' \in S} \mathbf{R}^{(\mathcal{X})}(s, s') \geqslant 0}_{\text{rate constraints}}, \tag{6.1}$$

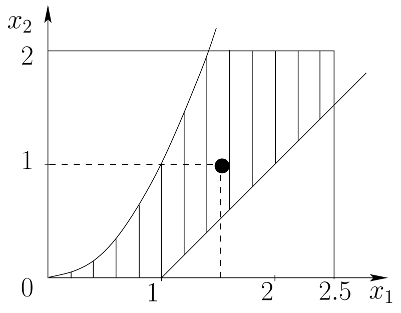
where $\bigwedge_{i=1}^{m} l_i \leqslant x_i \leqslant u_i$ is the set of *range constraints* of the parameters and $\bigwedge_{s,s' \in S} \mathbf{R}^{(\mathcal{X})}(s, s') \geqslant 0$ is the set of *rate constraints* ruling out the negative rates. Note that we assume each parameter x_i is bounded by l_i and u_i. This is a reasonable assumption in practice since those parameters usually have some physical meanings (e.g., speed, height, temperature, etc.) that mostly have an upper- and lower-bound. The regions satisfying the range and rate constraints are called *range region* (ζ_{range}) and *rate region* (ζ_{rate}), respectively. Note that $\zeta_0 = \zeta_{range} \cap \zeta_{rate}$.

Example 6.3 *In Fig. 6.1(a), we illustrate a pCTMC over $\{x_1, x_2\}$ with the range constraints: $0 \leqslant x_1 \leqslant 2.5$ and $0 \leqslant x_2 \leqslant 2$. The rate constraints are as follows:*

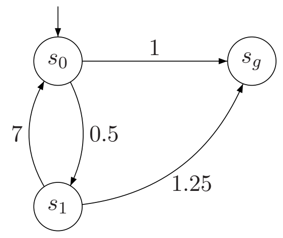
$$2x_1 + 4 \geqslant 0 \ \wedge \ 2 - x_2 \geqslant 0 \ \wedge \ x_2 - x_1 + 1 \geqslant 0 \ \wedge \ x_1^2 - x_2 \geqslant 0.$$



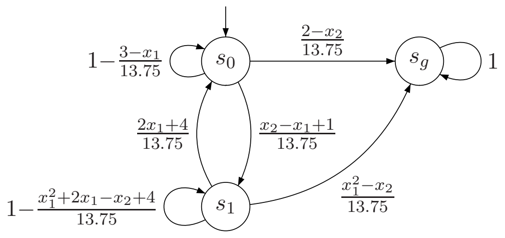
(a) *p*CTMC



(b) initial region ζ_0



(c) instance CTMC



$1-\frac{3-x_1}{13.75}$
$\frac{2-x_2}{13.75}$
s_0
s_g
1
$\frac{2x_1+4}{13.75}$
$\frac{x_2-x_1+1}{13.75}$
$1-\frac{x_1^2+2x_1-x_2+4}{13.75}$
s_1
$\frac{x_1^2-x_2}{13.75}$

(d) uniformized *p*CTMC

Figure 6.1: *p*CTMCs and related concepts

ζ_{range} *is the rectangular area in Fig. 6.1(b), while* ζ_{rate} *is the area between the two curves. The initial region* ζ_0 *is the shaded area. Any point in* ζ_0 *will induce an instance* CTMC*, e.g., for the valuation* $v(x_1) = 1.5$*,* $v(x_2) = 1$*, the instance* CTMC *is shown in Fig. 6.1(c).* ♦

6.2 Probabilistic Time-Bounded Reachability

In this chapter, we consider a very important fragment of CSL, i.e., the *probabilistic time-bounded reachability* properties, i.e., given a set of goal states, labeling with *goal*, does the probability of reaching those goal states within time t lie in the interval $[p_l, p_u]$? This is formalized in CSL as $\mathcal{P}_{[p_l,p_u]}(\Diamond^{\leqslant t} goal)$, where $0 \leqslant p_l \leqslant p_u \leqslant 1$. Note that the interval $[p_l, p_u]$ may also be open, or half-open. The following lemma states that the reachability probability can be reduced to computing the transient probability:

Lemma 6.4 (**[BHHK03]**) *Given a* CTMC $\mathcal{C} = (S, \mathbf{R}, L)$*, and* $s_g \in S$ *an absorbing state:*

$$Prob(s_0, \Diamond^{\leqslant t} goal) = \sum_{s_g \models goal} \wp_{s_g}(t), \tag{6.2}$$

where $Prob(s_0, \Diamond^{\leqslant t} goal)$ is the reachability probability of s_g from s_0 within t time units and $\wp_{s_g}(t)$ is the transient probability of state s_g at time t.

This lemma also applies to pCTMCs in the sense that every valuation $v \in \zeta_0$ will induce an instance CTMC, to which this lemma applies. Without loss of generality, we assume s_g is the *unique* goal state and write $\Diamond^{\leqslant t} s_g$ instead. In the remainder of this section, we focus on deriving an expression for $\wp_{s_g}(t)$ with parameters in $\mathcal{X}$. The expression for $\wp_{s_g}(t)$ (the reachability probability) is the basis of solving the synthesis problem symbolically.

6.2.1 Computing Uniformization Rate q

Given pCTMC $\mathcal{C}^{(\mathcal{X})} = (S, \mathbf{R}^{(\mathcal{X})}, L)$ with the set $\mathcal{X}$ of parameters and initial region ζ_0, the uniformization rate q is at least the largest number that an exit rate can take in ζ_0. Formally,

$$q \geqslant \max_{1 \leqslant i \leqslant n} \left\{ \max_{v \in \zeta_0} \left\{ E^{(\mathcal{X})}(s_i)[v] \right\} \right\} \tag{6.3}$$

Note that q is a *constant*. It suffices to first maximize each expression $E^{(\mathcal{X})}(s_i)$ (the objective function) within the closed region ζ_0 (the inner max), and then take the maximum out of n candidates (the outer max), where $n = |S|$. Typically, we take the minimal value of q that fulfills (6.3), i.e., $\geqslant$ is replaced by $=$. Prior to discussing how to obtain a solution to (6.3), we first give an example:

Example 6.5 *For the pCTMC in Fig. 6.1(a) and the initial region ζ_0 in Fig. 6.1(b), the exit rate of s_0 and s_1 is given by the expressions $g_0(x_1) := 3 - x_1$ and $g_1(x_1, x_2) := x_1^2 + 2x_1 - x_2 + 4$, respectively. The maximal value of g_0 in ζ_0 is 3 and 13.75 for g_1. Therefore, we take the uniformization rate $q = 13.75$. The uniformized pCTMC for this rate is shown in Fig. 6.1(d).* ♦

In general, determining the inner max in (6.3) boils down to solving a *nonlinear programming* (NLP) problem [Avr03], where the constraints are provided by the initial region. Note that both the objective function $E^{(\mathcal{X})}(s_i)$ and the constraints (6.1) are polynomial expressions. For some special cases, e.g., one parameter or linear expression rates, the NLP problem can be simplified. Fig. 6.2 summarizes the techniques that can be applied. The highest degree of the objective function and constraints is indicated along the y-axis, whereas the number of variables x_i of the objective function and constraints is indicated along the x-axis. Each dot represents a combination. We partition the x-y plane into 5 areas (indicated (I) through (V)) by dashed lines, where the dots in the same area can be solved by the techniques specified in the graph. In details:

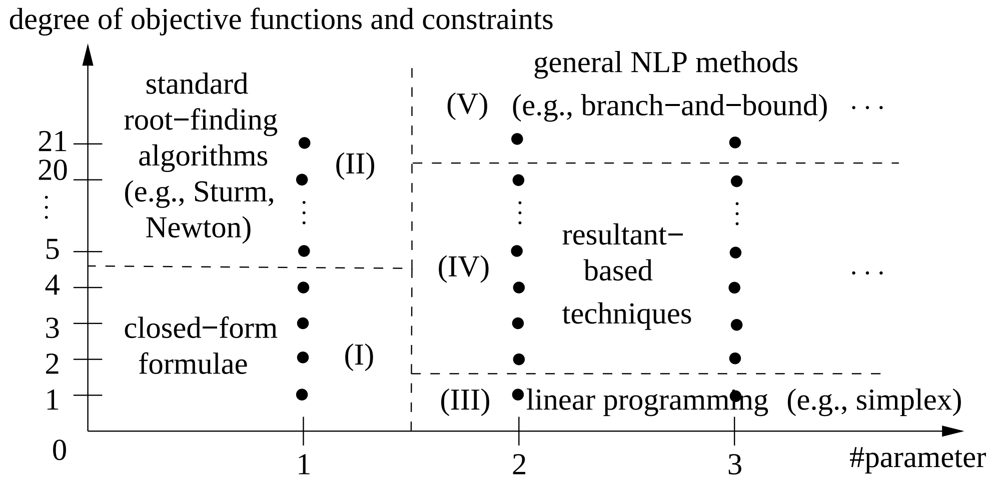

degree of objective functions and constraints
standard root−finding algorithms (e.g., Sturm, Newton)
(II)
general NLP methods
(V) (e.g., branch−and−bound)
(IV)
resultant−based techniques
closed−form formulae
(I)
(III) linear programming (e.g., simplex)
21
20
5
4
3
2
1
0
1
2
3
#parameter

Figure 6.2: Methods for maximizing polynomials under constraints

- For the case of one parameter and degree at most 4 (area (I)), the maximal value can be obtained by first deriving a closed-form expression and then solving the expression. For polynomials of higher degree, this is *impossible* as proven by Galois [Ste89]. In particular, the Galois' theory states that it is not possible to give an algebraic formula for the zeros of a polynomial of degree 5 and higher.

- For the case of one parameter x and the degree at least 5 (area (II)), standard *root-finding algorithms* can be applied. To be more precise, the roots of the first derivative of the polynomial are to be found as the extreme values, among which together with the boundary values of x we can obtain the maximal values of the polynomial. The prevailing techniques are *Newton's method* [Ham73], *Sturm's method* [HK88], *Laguerre's method* [Lag83], to name a few.

- For the case of more than one parameter and degree one (area (III)), it boils down to solving a *linear programming problem* [Sch98], where, amongst others, the *simplex algorithm* [CLRS01b] and the *interior point method* [NW99] are well-known solution techniques and are quite efficient.

- For the case of more than one parameter and degree from 2 to 20 (area (IV)), the *resultant-based techniques* or *Gröbner bases methods* [Fro98] perform better than branch-and-bound techniques (see below). Note that 20 is an estimation due to performance considerations.

- The remaining cases (area (V)) are general NLP problems and can be solved numerically by *branch-and-bound techniques* [Han92].

By using the above algorithms in different cases, the inner max can be determined. The outer max directly follows, so does the uniformization rate q. Given q, it is then possible to obtain a uniformized pCTMC, based on which the transient probability (i.e., the reachability probability) can be computed. This is specified in the next subsection.

6.2.2 Computing the Symbolic Expressions of $\tilde{\vec{\wp}}(t)$

Recall that $\tilde{\vec{\wp}}(t)$ is the ε-approximation of the transient probability vector $\vec{\wp}(t)$. The reachability probability to s_g within time t equals $\wp_{s_g}(t)$, and thus can be ε-approximated by $\tilde{\wp}_{s_g}(t)$. Similarly, we truncate the infinite sum of $\vec{\wp}(t)$ to obtain $\tilde{\vec{\wp}}(t)$. Recall that k_ε is the truncation point in the infinite sum series of $\vec{\wp}(t)$. k_ε is computed in such a way ((2.4), page 14) that it is *independent of the rates* in the CTMC. This implies that k_ε coincides with the non-parametric case.

The transient probability vector

$$\tilde{\vec{\wp}}(t) = \tilde{\vec{\wp}}(0) \cdot \sum_{i=0}^{k_\varepsilon} e^{-qt} \frac{(qt)^i}{i!} \left(\mathbf{P}^{(\mathcal{X})}\right)^i \tag{6.4}$$

can be computed by vector-matrix multiplication. For given q and t, $e^{-qt}\frac{(qt)^i}{i!}$ is a constant, while $\left(\mathbf{P}^{(\mathcal{X})}\right)^i$ contains parameters. Let $\deg_{x_i}(\mathbf{P}^{(\mathcal{X})})$ denote the maximal degree of parameter x_i in all expressions in $\mathbf{P}^{(\mathcal{X})}$. For instance, $\deg_{x_1}(\mathbf{P}^{(\mathcal{X})}) = 2$ and $\deg_{x_2}(\mathbf{P}^{(\mathcal{X})}) = 1$ for the pCTMC in Fig. 6.1(a). Note that $\deg_{x_i}(\mathbf{P}^{(\mathcal{X})}) = \deg_{x_i}(\mathbf{R}^{(\mathcal{X})})$. The degree of a polynomial is the sum of the degrees of all its variables. Thus, $\left(\mathbf{P}^{(\mathcal{X})}\right)^{k_\varepsilon}$ has degree at most $\hat{k_\varepsilon}$, where

$$\hat{k_\varepsilon} = k_\varepsilon \cdot \sum_{i=1}^{m} \deg_{x_i}(\mathbf{P}^{(\mathcal{X})}). \tag{6.5}$$

This is because at most k_ε times a vector-matrix multiplication will be taken and the degree of one multiplier (the matrix) is $\sum_{i=1}^{m} \deg_{x_i}(\mathbf{P}^{(\mathcal{X})})$. The order of $\deg_{x_i}(\mathbf{P}^{(\mathcal{X})})$ is $\mathcal{O}(\mathcal{K})$, where $\mathcal{K} := \max_{x_i \in \mathcal{X}} \deg_{x_i}(\mathbf{P}^{(\mathcal{X})})$ is the maximal degree of the polynomial expressions appearing in the pCTMC. Recall that k_ε tends to be of the order $\mathcal{O}(qt)$, if qt is large. Thus, $\hat{k_\varepsilon}$ is of order $\mathcal{O}(qtm\mathcal{K})$.

Given q and t, by (6.4) we can derive the transient probability $\tilde{\wp}_{s_g}(t)$ which is a polynomial function over the parameters $x_1, \ldots, x_m$, i.e.,

$$f(x_1, \ldots, x_m) = \sum_{j=(i_1,\ldots,i_m)} a_j \cdot x_1^{i_1} \cdots x_m^{i_m}, \tag{6.6}$$

where $i_\ell \leqslant k_\varepsilon \cdot \deg_{x_\ell}(\mathbf{P}^{(\mathcal{X})})$ $(1 \leqslant \ell \leqslant m)$, and $a_j \in \mathbb{R}$. The degree of f is at most $\hat{k_\varepsilon}$.

In order to better explain (6.6), we will consider an example with an unrealistically small k_ε. Realistic k_ε's are usually too large to derive a polynomial that can be illustrated as an example and a small k_ε can already show the main idea of (6.6).

Example 6.6 *Consider the uniformized* pCTMC *in Fig. 6.1(d) and assume* $q = 13.75$, $t = 1$ *and* $k_\varepsilon = 2$. *Then,*

$$\begin{aligned}\vec{\wp}(t) &= \vec{\wp}(0)\cdot\sum_{i=0}^{k_\varepsilon} e^{-qt}\frac{(qt)^i}{i!}\left(\mathbf{P}^{(\mathcal{X})}\right)^i \quad = \quad \vec{\wp}(0)\cdot e^{-qt}\cdot\sum_{i=0}^{k_\varepsilon}\frac{(qt)^i}{i!}\left(\mathbf{P}^{(\mathcal{X})}\right)^i \\ &= \vec{\wp}(0)\cdot e^{-qt}\cdot\left(\frac{(qt)^0}{0!}\mathbf{P}^{(\mathcal{X})0} + \frac{(qt)^1}{1!}\mathbf{P}^{(\mathcal{X})1} + \frac{(qt)^2}{2!}\mathbf{P}^{(\mathcal{X})2}\right)\end{aligned}$$

where $\vec{\wp}(0) = (1, 0, 0)$ *and*

$$\mathbf{P}^{(\mathcal{X})} = \begin{pmatrix} 1 - \frac{3-x_1}{13.75} & \frac{x_2-x_1+1}{13.75} & \frac{2-x_2}{13.75} \\ \frac{2x_1+4}{13.75} & 1 - \frac{x_1^2+2x_1-x_2+4}{13.75} & \frac{x_1^2-x_2}{13.75} \\ 0 & 0 & 1 \end{pmatrix}.$$

The resulting polynomial is

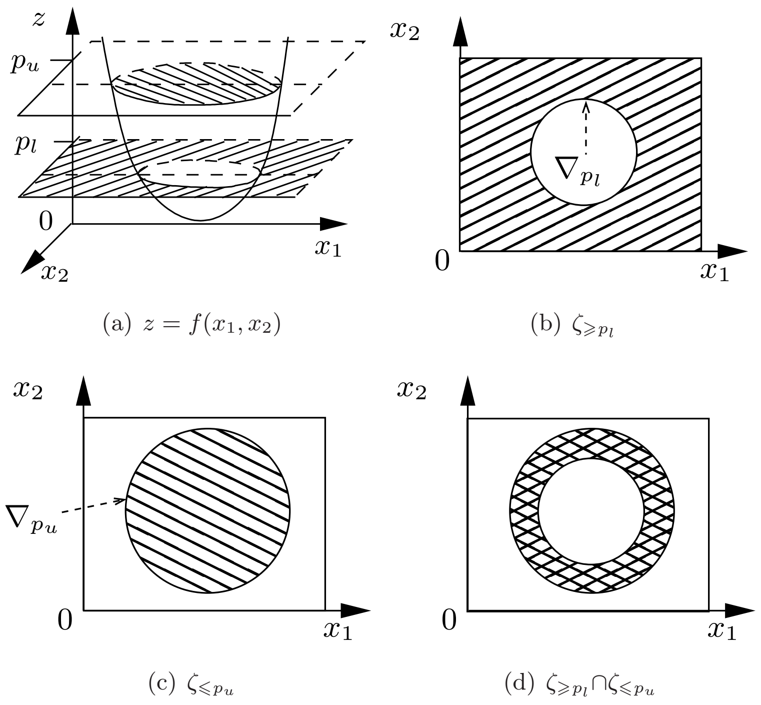
$$f(x_1, x_2) = \vec{\wp}_{s_g}(t) = \frac{1}{2}e^{-qt}\cdot(-x_1^3 + x_1^2x_2 - x_2^2 + x_1x_2 - 8.75x_1 - 16.75x_2 + 53),$$

where $\frac{1}{2}e^{-qt} = 5.3385 \times 10^{-7}$. *Note that* $\deg_{x_1}(\mathbf{P}^{(\mathcal{X})}) = 2$, $\deg_{x_2}(\mathbf{P}^{(\mathcal{X})}) = 1$ *and* $\mathcal{K} = \max\{2, 1\} = 2$. *Furthermore, each power of* x_1 *(resp.* x_2*) in* $f(x_1, x_2)$ *is less than* $k_\varepsilon * \deg_{x_1}(\mathbf{P}^{(\mathcal{X})}) = 2 * 2 = 4$ *(resp.* $k_\varepsilon * \deg_{x_2}(\mathbf{P}^{(\mathcal{X})}) = 2 * 1 = 2$*) and the degree of* $f(x_1, x_2)$ *is at most* $k_\varepsilon \sum_{i=1}^{m} \deg_{x_i}(\mathbf{P}^{(\mathcal{X})}) = 2 * (2 + 1) = 6$. ♦

Note that k_ε is usually much larger than $\deg_{x_i}(\mathbf{R}^{(\mathcal{X})})$ and thus the degree of the polynomial expression in (6.6) is not much affected if we would restrict rate expressions in pCTMCs to be linear. This symbolic expression $\tilde{\wp}_{s_g}(t)$ (i.e., $f(x_1, \ldots, x_m)$) is the basis of the symbolic synthesis approach in Section 6.4.

6.3 Parameter Synthesis Framework

The parameter synthesis problem we will consider is to determine all the values that parameters can take such that the satisfaction of the property $\mathcal{P}_{[p_l,p_u]}(\Diamond^{\leqslant t} s_g)$ is ensured in any derived instance model. We define the *synthesis region* $\zeta_{syn} \subseteq \zeta_0$ to be the set of valuations, such that each valuation (or point) $v = (x'_1, \ldots, x'_m)$ therein induces an instance CTMC $\mathcal{C}[v]$, for which $f(x'_1, \ldots, x'_m) \in [p_l, p_u]$. The main task is to find the (approximate) synthesis region ζ_{syn}. For the sake of easy visualization, we restrict to pCTMCs with at most two parameters. Our techniques can be applied to more-parameter cases, however, the computational complexity will grow drastically.



(a) $z = f(x_1, x_2)$ (b) $\zeta_{\geqslant p_l}$ (c) $\zeta_{\leqslant p_u}$ (d) $\zeta_{\geqslant p_l} \cap \zeta_{\leqslant p_u}$

Figure 6.3: An example synthesis region

6.3.1 Synthesis Regions

Given the pCTMC $\mathcal{C}^{(x_1,x_2)}$ and property $\mathcal{P}_{[p_l,p_u]}(\Diamond^{\leqslant t} s_g)$, the transient probability $z = f(x_1, x_2)$ defines a surface, see Fig. 6.3(a) for an (artificial) example. For $z \bowtie p$ ($\bowtie \in \{<, \leqslant, >, \geqslant\}$ and constant $p \in [0, 1]$), the projection of the surface on the x_1x_2-plane $z = p$ (in particular in region ζ_{range}) is a region $\zeta_{\bowtie p}$ with boundary curve ∇_p. $\zeta_{\bowtie p}$ is the set of points (x_1, x_2) such that $f(x_1, x_2) \bowtie p$. The boundary curve ∇_p is given by $f(x_1, x_2) - p = 0$. The region $\zeta_{[p_l,p_u]}$ is the intersection of $\zeta_{\geqslant p_l}$ and $\zeta_{\leqslant p_u}$, where p_l and p_u are the probability bounds on the reachability property $\Diamond^{\leqslant t} s_g$.

Example 6.7 *Consider the surface in Fig. 6.3(a). The shaded areas in Fig. 6.3(b) and 6.3(c) depict the region $\zeta_{\geqslant p_l}$ and $\zeta_{\leqslant p_u}$ derived from the projection of this surface on $z = p_l$ and $z = p_u$. The intersection $\zeta_{\geqslant p_l} \cap \zeta_{\leqslant p_u}$ (Fig. 6.3(d)) is the synthesis region $\zeta_{[p_l,p_u]}$, given ζ_{range} the rectangular area.* ♦

Note that in general it is impossible to get the exact shape of the boundary curve $f(x_1, x_2) - p = 0$ (as $f(x_1, x_2)$ is a high-degree polynomial) as well as the exact synthesis region. As a result, we use *a set of linear line segments* (or equivalently, *a set of linear inequations*) to approximate the boundary curves, thus the approximate synthesis region is *a set of polygons*. It is also called *polygon(al) approximation* in the literature [Mon70][ABW90][JTW07]. We will provide two different approaches for polygon approximation in Section 6.4 and 6.5. Prior to this, we first introduce discretization — the basis of the following approximation algorithms and then give the general framework to synthesize polygon regions.

6.3.2 Discretization

Theoretically, every point in the initial region has to be checked for the satisfaction in the induced CTMC. However, the initial region is dense, i.e., it contains uncountably many points. A common approach to cope with this is to discretize the "continuous" region and obtain only finitely many "sample" points, which might represent, or to be more exact, approximate the original region.

Given the parameter set $\mathcal{X} = \{x_1, x_2\}$ and ζ_{range}, we specify a discretization step $\Delta_i \in \mathbb{R}_{>0}$ for each parameter x_i $(i = 1, 2)$, such that $u_i - l_i = N_i\Delta_i$. Thus, the range $[l_i, u_i]$ of values that variable x_i can take is partitioned into N_i subintervals:

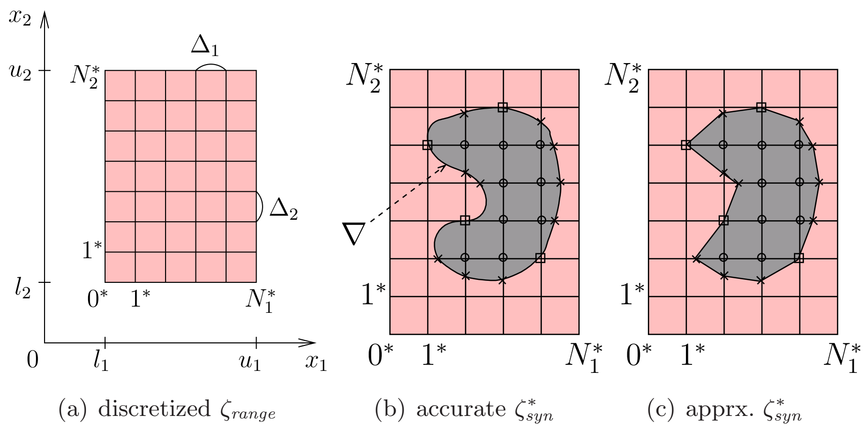
$$\Big[l_i,\, l_i{+}\Delta_i\Big],\ \Big(l_i{+}\Delta_i,\, l_i{+}2\Delta_i\Big],\ \ldots,\ \Big(l_i{+}(N_i{-}1)\Delta_i,\, l_i{+}N_i\Delta_i\Big].$$

The values $l_i + j\Delta_i$ $(0 \leqslant j \leqslant N_i)$ are assigned the indices $0, 1, \ldots, N_i$. We thus obtain a 2-dimensional grid, where the *grid points* are of the form $gp = (j_1, j_2)$ for $0 \leqslant j_i \leqslant N_i$ with the valuation $(l_1 + j_1\Delta_1, l_2 + j_2\Delta_2)$ and a *grid cell* is a smallest rectangle with grid points as its four vertices. Two grid points $gp = (i, j)$ and $gp' = (i', j')$ are *adjacent* if $|i - i'| + |j - j'| = 1$. The region ζ_{range} consists of at most $(N_1 + 1)(N_2 + 1)$ grid points and each point gp induces an instance CTMC $\mathcal{C}[gp]$ by the valuation of gp. The transient probability $\tilde{\wp}^{\mathcal{C}[gp]}_{s_g}(t)$ is computed by standard methods. A grid point gp is *positive* (denoted $gp = \top$) if $\tilde{\wp}^{\mathcal{C}[gp]}_{s_g}(t) \in (p_l, p_u)$; gp is *neutral* (denoted $gp =$⌶) if $\tilde{\wp}^{\mathcal{C}[gp]}_{s_g}(t) \in \{p_l, p_u\}$ and gp is *negative* (denoted $gp = \perp$) otherwise.

It is important to realize that this yields a discretization in the sense that instead of checking each point in the dense region ζ_{range}, we only check the discrete grid points as "samples".

Example 6.8 *Given the light gray range region ζ_{range} formed by $[l_1, u_1]$ and $[l_2, u_2]$, the grid discretization is as in Fig. 6.4(a), where $\Delta_1 = \Delta_2$. The grid indices are marked with $*$ to distinguish them from the indices on the axes. Given the accurate synthesis region as the dark gray area in Fig. 6.4(b), the positive and neutral grid points are marked with circle and square, respectively. The negative grid points are marked with nothing.* ♦

It is usually convenient to choose a global Δ (i.e., $\Delta = \Delta_1 = \Delta_2$) such that the approximation error is the same in each dimension. From now on we will consider only global Δ. Besides, since the grid points near the boundary curve are much more



x_2
u_2
l_2
Δ_1
Δ_2
N_2^*
1^*
0^*
N_1^*
0
l_1
u_1
x_1
∇
(a) discretized ζ_{range}
(b) accurate ζ^*_{syn}
(c) apprx. ζ^*_{syn}

Figure 6.4: Grid discretization

important, it is possible to use a *non-uniform* grid, e.g., smaller grid steps near the curve and larger grid steps away from the curve. Actually, in the second approach of the polygon approximation (cf. Section 6.5), only the grid cells that are close to the boundary curve are refined. This will become clear later.

6.3.3 General Framework

In this subsection, we present the general framework to obtain an approximate synthesis region (cf. Alg. 5) which consists of two main steps:

- in Step 1, a first approximation ζ^*_{syn} of the synthesis region is obtained while ignoring the rate constraint ζ_{rate}. To achieve this, the grid discretization is first applied (Step 1.1, Section 6.3.2), based on which the set of points $\Im$ needed for constructing the polygon(s) is computed (Step 1.2). We consider two different approaches to implement this step in Section 6.4 and 6.5, where several rounds of refinements may be performed. Consequently, the points in $\Im$ are computed differently. Having $\Im$ at our disposal, it then amounts to connect these points (Step 1.3, Section 6.4.3) to form the polygon(s) ζ^*_{syn}.
- in Step 2, $\zeta_{syn} = \zeta^*_{syn} \cap \zeta_{rate}$ is computed as the final synthesis region such that all the points that will induce a negative rate (thus not a CTMC at all) are removed. We first discretize the region ζ_{rate} (Step 2.1) by the same grid as in the end of Step 1.2. This grid is not necessarily the same as the initial one in Step 1.1, as it might have been refined. The intersection is then done by intersecting ζ^*_{syn} and ζ_{rate} (Step 2.2, Section 6.4.4).

Algorithm 5 Framework for obtaining approximate synthesis regions.

```
Require: pCTMC $\mathcal{C}^{(x_1,x_2)}$, formula $\mathcal{P}_{[p_l,p_u]}(\Diamond^{\leqslant t} s_g)$,
         initial discretization step $\Delta$, termination discretization step $\Delta_{\min}$
Ensure: approximate synthesis region $\zeta_{syn}$ (a set of polygons)
 1: compute $\zeta^*_{syn} := \zeta_{\leqslant p_u} \cap \zeta_{\geqslant p_l} \cap \zeta_{range}$;
    1.1: discretize $\zeta_{range}$ on $x_1x_2$-plane by grid step $\Delta$;
    1.2: compute the set of points $\Im$ that are needed to form polygon(s);
         1.2.1 if necessary, refine certain grids with $\Delta'$ ($\Delta_{\min} \leqslant \Delta' < \Delta$);   goto Step 1.2;
    1.3: connect those obtained points to form approximate polygons (i.e., $\zeta^*_{syn}$);
 2: compute $\zeta_{syn} := \zeta^*_{syn} \cap \zeta_{rate}$;
    2.1: discretize $\zeta_{rate}$ on $x_1x_2$-plane with the same grid (possibly being refined);
    2.2: intersect $\zeta^*_{syn}$ and $\zeta_{rate}$;
```

6.4 The Symbolic Approach

The symbolic approach is based on the expression $\tilde{\wp}_{s_g}(t)$ derived in Section 6.2.2. The main idea is that by solving some polynomials we find all intersection points (sometimes short as int. pts.) between the boundary curves $\nabla_{p_l}, \nabla_{p_u}$ and the grid lines. These intersection points (forming the set $\Im$) are then connected to approximate ∇_{p_l} and ∇_{p_u}. The resulting area bounded by the approximate curves is a set of polygons (polyhedra, in case of more than 2 parameters). This idea is illustrated by the following example:

Example 6.9 (Continuing Example 6.8) *For the boundary curve and the accurate synthesis region ζ^*_{syn} (the dark gray area) in Fig. 6.4(b). Besides the grid points, the intersection points are marked with* X*. One possible approximate synthesis region is the dark gray polygon in Fig. 6.4(c).* ♦

Computing Intersection Points. To determine all intersection points between the boundary curves ∇_{p_l} and ∇_{p_u} and the grid lines, for each grid line $x_i = j\Delta$ ($i = 1, 2$ and $0 \leqslant j \leqslant N_i$), we solve the following system of equations:

$$\begin{cases} f(x_1, x_2) - p_l = 0 \\ x_i = j\Delta \end{cases} \qquad \begin{cases} f(x_1, x_2) - p_u = 0 \\ x_i = j\Delta \end{cases}$$

which boils down to finding the roots of a *single variable* polynomial function, which in general can be solved by standard root-finding algorithms [Ham73][HK88][Pan00]. Since the polynomial function is usually of (very) high degree, the method in [SBFT03] is more applicable. In total, we need to solve $2(N_1 + N_2 + 2)$ polynomials, as we have 2 curves and $N_i + 1$ grid lines in dimension i. Note that by increasing the number of parameters in a pCTMC the total time needed to find all intersection points increases exponentially. At the end of this step, we have obtained the set $\Im$ of intersection points. Note that the size of $\Im$ might increase in the refinement process.

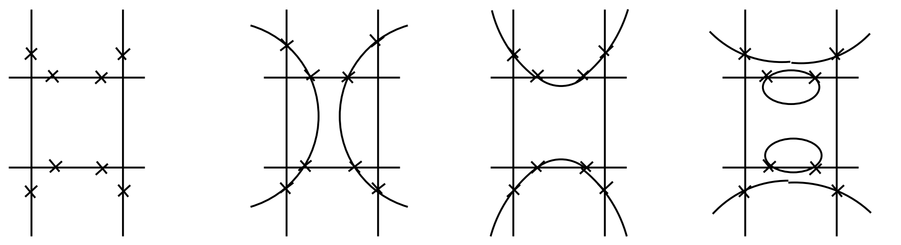

(a) Set of Int. Pts. $\Im$ (b) Region 1 (c) Region 2 (d) Region 3

Figure 6.5: Ambiguity in connecting intersection points

6.4.1 Labeling Grid and Intersection Points

Given the set of intersection points $\Im$, a natural question is how to connect them. Since the boundary of the accurate region can be of arbitrary shape, the set $\Im$ might have more than one way to be connected. We give the following example:

Example 6.10 *For the set $\Im$ of intersection points indicated by* X *in Fig. 6.5(a), we show in Fig. 6.5(b)-6.5(d) several possible ways to connect them.* ♦

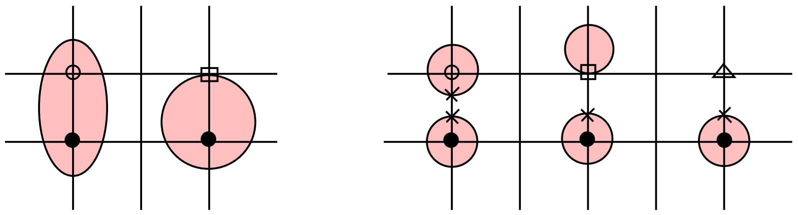
The following algorithm attempts to reduce or even resolve the potential ambiguity.

Data Structures. A polygon ζ is represented by a *tree*, such that one (positive) grid point gp inside ζ is picked as the *root* and all other (positive) grid points in ζ are *internal nodes*, while the intersection points (possibly neutral grid points) are *leaf nodes*. This terminology will be used interchangeably in the remainder of this chapter. Since the synthesis region may consist of more than one polygon (tree), we need a root tag to indicate to which tree a leaf or an internal node belongs. A leaf or an internal node without a root tag is called an *orphan*. The approximate synthesis region ζ^*_{syn} is represented by a set of polygons, i.e., sets of line segments.

Labeling Intersection Points and Grid Points. Labeling intersection and grid points aims to determine to which polygon an intersection point belongs. This information is the basis for the algorithm of connecting the intersection points (Section 6.4.3). Let $\mathsf{a}, \mathsf{b}, \ldots$ denote grid points ($\circ/\bullet$ for positive and $\triangle$ for negative), $\mathsf{1}, \mathsf{2}, \ldots$ for intersection points (X), and $\mathsf{A}, \mathsf{B}, \ldots$ for grid cells. We also use $\#\mathrm{IntPts}(gp_1, gp_2)$ to denote the number of intersection points between grid points gp_1 and gp_2 (gp_1, gp_2 inclusive). We first notice the following facts:

Fact 6.11 *Given two adjacent grid points gp_1, gp_2 with $gp_1 = \top$,*

1. *if ($gp_2 = \top$ and $\#\mathrm{IntPts}(gp_1, gp_2) = 0$) or ($gp_2 =$⌶ and $\#\mathrm{IntPts}(gp_1, gp_2) = 1$ where the intersection point is gp_2) then gp_1, gp_2 belong to the same polygon;*



(a) $\#\mathrm{IntPts}(curt, adj) = 0$ (b) $\#\mathrm{IntPts}(curt, adj) > 0$

Figure 6.6: Labeling criteria

2. *if* $gp_2 = \perp$, *then* $\#\mathrm{IntPts}(gp_1, gp_2) > 0$.

Note that if $gp_2 = \top\!\!\!\perp$, then gp_2 is an intersection point in $\Im$. The labeling algorithm will start with any positive grid point *curt* and label it as a root node of, say tree *tree*, after which we search its four adjacent grid points *adj*. According to the above facts, we have the following labeling criteria:

(i) if $adj = \top$ or $\top\!\!\!\perp$ and $\#\mathrm{IntPts}(curt, adj) = 0$, then set $adj \in tree$;

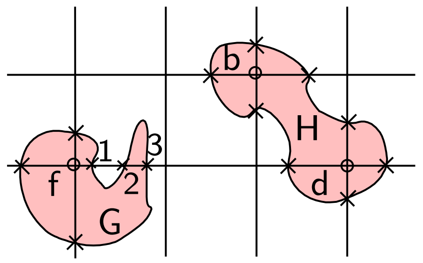
(ii) if $\#\mathrm{IntPts}(curt, adj) > 0$, then set $ip \in tree$, where ip is the intersection point that is closest to *curt*.

Basically, (i) is based on Fact 1 and (ii) is based on Fact 2. $\#\mathrm{IntPts}(curt, adj) > 0$ ensures that there exists at least one intersection point between *curt* and *adj* and the closest one to *curt* must be on the same polygon as *curt*. The two criteria are illustrated in Fig. 6.6, where the black filled grid points are *curt* and let *adj* be *curt*'s upper neighbor. Fig. 6.6(a) shows the two cases of (i) where $\#\mathrm{IntPts}(curt, adj) = 0$ and $adj = \top, \top\!\!\!\perp$, whereas Fig. 6.6(b) shows the three cases of (ii) where $\#\mathrm{IntPts}(curt, adj) > 0$ and $adj = \top, \top\!\!\!\perp, \perp$, respectively.

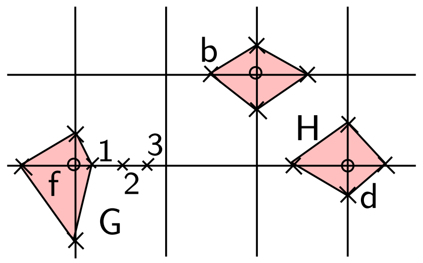
Let $GP_\top$ and $GP_\top(tree)$ be the set of all positive grid points and positive grid points assigned to the tree *tree*. We will continue labeling until $GP_\top(tree)$ becomes stable (i.e., no more change on the labeling of any grid or intersection points). This means that the labeling of this particular polygon (*tree*) is finished (thus $GP_\top(tree)$ is obtained). If $GP_\top \backslash GP_\top(tree) \neq \varnothing$ (i.e., there exist other polygons), then another root node for a new tree will be chosen and the labeling process will continue; otherwise the labeling of all the points has been finished.

6.4.2 Refinement

As is typical for polygon approximation algorithms, the above labeling algorithm cannot guarantee to find all regions correctly. We give some examples:

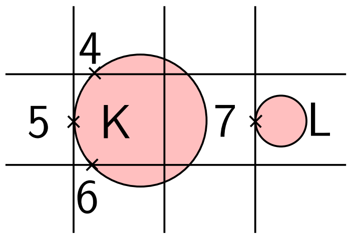


(a) Accurate synthesis regions

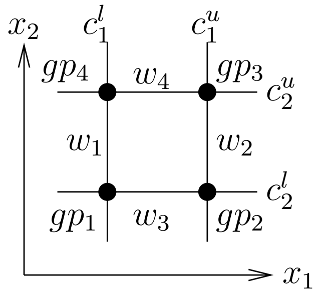


(b) Approximate synthesis regions

Figure 6.7: Motivation for refinement



(a) #leaves in a grid cell with a tangent point



(b) For proving Theorem 6.14

Figure 6.8: Each grid cell has even number of intersection points

Example 6.12 *According to the labeling criteria, only point* 1 *in Fig.* 6.7(a) *will be labeled to be in tree* f*, and* 2, 3 *will remain to be orphans (cf. Fig.* 6.7(b) *for the approximate regions). As another example, the grid points* b, d *in grid cell* H *are in the same region, but since they are not adjacent, they will be identified as the roots of two different trees (and thus give rise to two polygons).* ♦

The main cause is that the grid is too coarse to have enough "sample points" that can capture some of the fine details on the curves. This can be repaired by a grid refinement, as explained below.

First we consider grid cells that do not have any grid point as an intersection point. In other words, all the grid points are either positive or negative, but not neutral. Let #leaves(gc) denote the number of intersection points on the four sides of grid cell gc. For instance, grid cell H in Fig. 6.7(a) has four leaves. If a leaf point is the tangent point between a boundary curve and one of the grid sides, then it will be counted twice (see the intersection points 5 and 7 in Fig. 6.8(a) with #leaves(K) $= 4$ and #leaves(L) $= 2$). We prove in the following that #leaves(gc) is always even.

Lemma 6.13 *Let gp_1, gp_2 be two adjacent grid points with labeling in $\{\top, \bot\}$. Then*

$$\begin{array}{lll} gp_1 \odot gp_2 = \mathrm{tt} & \text{iff} & \#\mathrm{IntPts}(gp_1, gp_2) \textit{ is even, and} \\ gp_1 \odot gp_2 = \mathrm{ff} & \text{iff} & \#\mathrm{IntPts}(gp_1, gp_2) \textit{ is odd.} \end{array}$$

where $gp \odot gp' = \mathrm{tt}$ if gp and gp' have the same labeling (both are $\top$ or $\bot$) and $gp \odot gp' = \mathrm{ff}$ otherwise.

Proof: Let $\#\mathrm{IntPts}(gp_1, gp_2) = n$ and assume w.l.o.g. that $gp_1 = (c_0, b)$ and $gp_2 = (c_n, b)$. We can divide the interval $[c_0, c_n]$ between gp_1 and gp_2 into n sub-intervals $[c_i, c_{i+1}]$ $(0 \leqslant i \leqslant n)$ such that each sub-interval contains and only contains one intersection point. Assume $\nabla(x_1, x_2)$ be the boundary curve. Let $\nabla_b(x_1) = \nabla(x_1, b)$ on which the grid line $x_2 = b$ gp_1 and gp_2 are located. Since $\nabla(x_1, x_2)$ is continuous, it holds that for any sub-interval $[c_i, c_{i+1}]$ $(0 \leqslant i \leqslant n)$, $\nabla_b(c_i) \cdot \nabla_b(c_{i+1}) < 0$. It holds that

$$n \text{ is even} \quad \text{iff} \quad \prod_{0 \leqslant i \leqslant n} \big(\nabla_b(c_i) \cdot \nabla_b(c_{i+1})\big) > 0 \quad \text{iff} \quad \nabla_b(c_0) \cdot \nabla_b(c_n) > 0 \quad \text{iff} \quad gp_1 \odot gp_2 = \mathrm{tt},$$

$$n \text{ is odd} \quad \text{iff} \quad \prod_{0 \leqslant i \leqslant n} \big(\nabla_b(c_i) \cdot \nabla_b(c_{i+1})\big) < 0 \quad \text{iff} \quad \nabla_b(c_0) \cdot \nabla_b(c_n) < 0 \quad \text{iff} \quad gp_1 \odot gp_2 = \mathrm{ff}.$$

■

Theorem 6.14 *Given a boundary curve $\nabla(x_1, x_2) = 0$ and a grid cell gc formed by $x_1 = c_1^l$, $x_1 = c_1^u$ and $x_2 = c_2^l$, $x_2 = c_2^u$. gc's four grid points gp_i $(i \in \{1, 2, 3, 4\})$ have labeling in $\{\top, \bot\}$, then*

$$|\nabla(c_1^l, x_2) = 0| + |\nabla(c_1^u, x_2) = 0| + |\nabla(x_1, c_2^l) = 0| + |\nabla(x_1, c_2^u) = 0| \text{ is even},$$

where $|\nabla(c, x_2) = 0|$ (resp. $|\nabla(x_1, c) = 0|$) defines the number of x_2 in $[c_2^l, c_2^u]$ (resp. x_1 in $[c_1^l, c_1^u]$) with which the respective equation holds.

Proof: We prove by contraposition. Let $|\nabla(c_1^l, x_2) = 0| = w_1$, $|\nabla(c_1^u, x_2) = 0| = w_2$, $|\nabla(x_1, c_2^l) = 0| = w_3$, $|\nabla(x_1, c_2^u) = 0| = w_4$ and suppose $w_1 + w_2 + w_3 + w_4$ is odd. Without loss of generality, we distinguish two cases: (1) w_1 is odd and w_2, w_3, w_4 are even; (2) w_1 is even and w_2, w_3, w_4 are odd. Let the four grid points of gc be $gp_1 = (c_1^l, c_2^l)$, $gp_2 = (c_1^u, c_2^l)$, $gp_3 = (c_1^u, c_2^u)$ and $gp_4 = (c_1^l, c_2^u)$ (cf. Fig. 6.8(b)). For case (1), due to Lemma 6.13, $gp_1 \odot gp_4 = \mathrm{ff}$ and $gp_1 \odot gp_2 = gp_2 \odot gp_3 = gp_3 \odot gp_4 = \mathrm{tt}$. This means that $W := (gp_1 \odot gp_4) \odot (gp_1 \odot gp_2) \odot (gp_2 \odot gp_3) \odot (gp_3 \odot gp_4) = \mathrm{ff}$[1], which contradicts the fact that $W = (gp_1 \odot gp_1) \odot (gp_2 \odot gp_2) \odot (gp_3 \odot gp_3) \odot (gp_4 \odot gp_4) = \mathrm{tt}$.

[1] $\odot$ is actually the xnor operator, where tt$\odot$tt=tt, ff$\odot$ff=tt, tt$\odot$ff=ff, ff$\odot$tt=ff.

For case (2), it is similar. ■

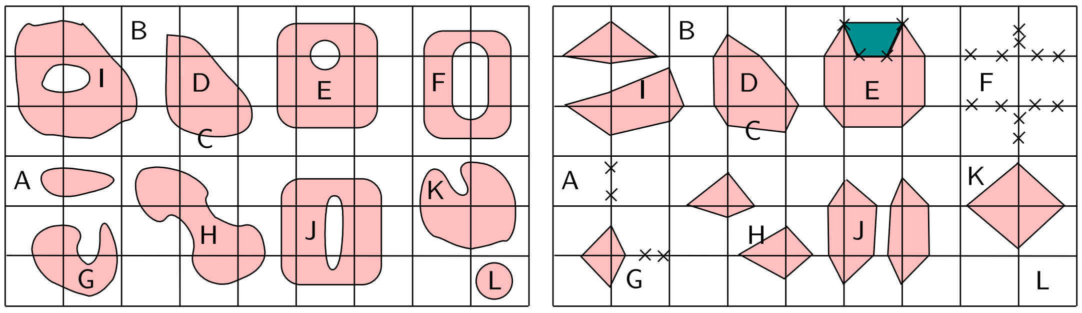
For those grids that have neutral grid points, we can parallel move the grid with a very small distance (e.g., $10^{-3}\Delta$) such that none of the four grid points are not intersection points any more and then we can compute the intersection points of the grid. This reduces the problem to the one that has been solved above, i.e., #leaves(gc) is even.

We also define the *number of positive grid points* of a grid cell gc, denoted by $\#\mathrm{GP}_\top(gc)$. For instance, $\#\mathrm{GP}_\top(\mathsf{H}) = 2$. Note that $\#\mathrm{GP}_\top(gc)$ can be at most 4.

Where to Refine? Let us now explain when refinement is necessary. We list some combinations of #leaves(gc) and $\#\mathrm{GP}_\top(gc)$ in Table 6.1, each with an example grid cell in Fig. 6.9. For different combinations, they might require a refinement ("to refine?"=*yes*) or not ("to refine?"=*no*).

$\#\mathrm{GP}_\top$	#leaves=2 example cell	to refine?	#leaves=4 example cell	to refine?
0	A	*yes*	F	*yes*
1	B	*no*	G	*yes*
2	C	*no*	H	*yes*
3	D	*no*	I	*yes*
4	E	*yes*	J	*yes*

Table 6.1: Refinement criteria



(a) Accurate synthesis regions (b) Approximate synthesis regions

Figure 6.9: Refinement criteria: #leaves(gc) and $\#\mathrm{GP}_\top(gc)$

- #leaves(gc) $= 2 \wedge \#\mathrm{GP}_\top(gc) \in \{1,2,3\}$

  We choose not to refine the grid gc in this case, as there is no ambiguity in connecting the intersection points.

- $\#\text{leaves}(gc) = 2 \wedge \#\text{GP}_\top(gc) \in \{0, 4\}$

  A refinement is required in this case as some area is smaller than a grid cell, and unknown or ambiguity arises (see grid cells A and E). Note A has only orphan leaves and more than that the shape of their region is unknown. For E, all the four vertices (intersection points) of the dark gray trapezoid belong to the same polygon according to the algorithm, however, it is unknown how these four points are connected as sides of a (larger) polygon.

- $\#\text{leaves}(gc) \geqslant 4$

  Typically, the more intersection points gc has, the more possible that some locally abrupt behavior (or protuberances) of the boundary curve occurs in gc. This happens when the area of interest is smaller than a grid cell. It means that the discretization is too coarse and needs a refinement.

  This can be seen by all the $\#\text{leaves}(gc) = 4$ cases shown in the table, where grid cells H, I, J split one connected region into two separated polygons, while grid cells F and G have orphan leaves. None of those grid cells yield a close approximation.

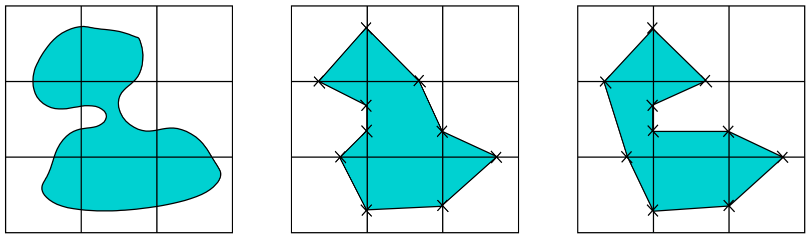
  For $\#\text{leaves}(gc) > 4$, the grid cell gc will be refined due to the similar reasons.

- $\#\text{leaves}(gc) = 0 \wedge \#\text{GP}_\top(gc) \in \{0, 4\}$

  The grid cell gc is assumed to be either completely outside the polygon ($\#\text{GP}_\top(gc) = 0$) or completely inside ($\#\text{GP}_\top(gc) = 4$). Thus, there is no need for refinement in this case.

How to Refine? The table can be used as a criterion to check whether or not a grid cell needs to be refined. Once we have identified the suspicious grid cell gc, the following question is how to refine it? There can be different strategies for refinement [Ste73], e.g., global vs. local; with uniform or non-uniform steps; how to reduce the discretization steps, etc. The strategies highly depend on the application, i.e., the structure of the polygons.

For the sake of simplicity, we consider one strategy, namely, the *local* and *bisectional* refinement. To be more exact, we will refine locally the area of 9 grid cells with gc in the center. Note that it is also possible to refine more or fewer (than 9) grid cells as the "local area". A new discretization with step size $\frac{1}{2}\Delta$ will be performed on those grid cells. The whole procedure (intersection points computation-labeling) is repeated for the newly appeared grid points and intersection points. Note that in the neighborhood of the new points, some old points might be re-labeled, e.g., the tree tag being changed, or a tag being added to orphan points, etc. The labeling-refinement procedure will continue until either the discretization step is less than the user-defined Δ_{min} or there are no grid cells to be refined due to our criterion.



(a) Actual boundary (b) Approx. boundary 1 (c) Approx. boundary 2

Figure 6.10: Connecting intersection points not uniquely

6.4.3 Constructing the Polygons

In case that the algorithm terminates before $\Delta_{\min}$ is reached, the refinement criterion guarantees that there does not exist any grid cell with more than 2 intersection points. Hence, obtaining polygon regions amounts to connecting the intersection points which share the same root within the same grid cell, see Fig. 6.4(c). Otherwise (if $\Delta_{\min}$ is reached), when there are only two intersection points in one grid, they can be connected according to the same rule as above; when there are more than two, then we may connect them arbitrarily. This might give rise to ambiguity (cf. Example 6.15) or omissions (e.g., the rightmost circle in grid cell L in Fig. 6.9(a)). However, these regions are only bounded in very limited areas, given a small discretization step. Thus, the undetected areas can be safely ignored within the predefined error bound. Note that to obtain a more precise approximation, we can take other discretization techniques, say, adding diagonal lines as well. In this case, a cell is a triangle, where our algorithm can be adapted easily.

Example 6.15 *Given the accurate synthesis region and grid as in Fig. 6.10(a), there are two ways to connect the intersection points as in Fig. 6.10(b) and 6.10(c). This kind of ambiguity can be avoided if each grid cell has at most 2 intersection points.* ♦

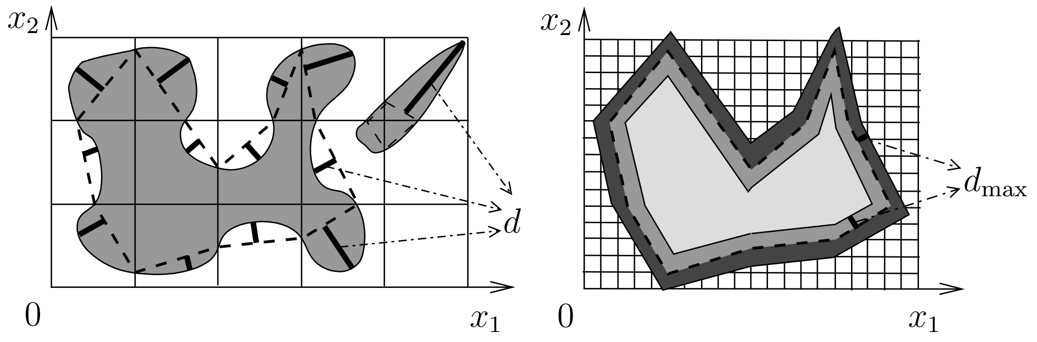
6.4.4 Region Intersection

By the end of the last section, we have derived the approximate synthesis region ζ^*_{syn} in terms of a set of polygons (represented by sets of linear inequations). Similarly, the rate region ζ_{rate} can also be polygonally approximated in the above way. To intersect the two regions, it suffices to take the intersection of the two sets of linear inequations.

Main Algorithm. So far we have explained each step of the framework (cf. Alg. 5). The algorithm realizing the symbolic approach is shown in Alg. 6.

Algorithm 6 Obtaining approx. synthesis regions symbolically

```
Require: pCTMC $\mathcal{C}^{(x_1,x_2)}$, formula $\mathcal{P}_{[p_l,p_u]}(\Diamond^{\leqslant t} s_g)$,
         initial discretization step $\Delta$, termination discretization step $\Delta_\perp$
Ensure: approximate synthesis region $\zeta_{syn}$ (a set of polygons)
1: compute $\zeta^*_{syn} := \zeta_{\leqslant p_u} \cap \zeta_{\geqslant p_l} \cap \zeta_{range}$;
   1.1: discretize $\zeta_{range}$ on $x_1x_2$-plane by grid step $\Delta$;
   1.2 /* compute the set of points $\Im$ that are needed to form polygon(s) */
   1.2.1 find all int. pts. between $\nabla_{p_l}, \nabla_{p_u}$ with the grid lines;
         /* label intersection and grid points with root tags */
   1.2.2 while (there exists a positive orphan grid point gp) do
   1.2.3    make gp the root of a new tree;
   1.2.4    gp is set as an unexplored tree node;
   1.2.5    while (tree gp has unexplored node curt) do
   1.2.6       curt is set to be explored;
   1.2.7       for each (curt's adjacent node adj) do
   1.2.8          if ($adj = \top \wedge adj$ is orphan $\wedge$ #IntPts(curt,adj)=0) then
   1.2.9             let adj have the same root as curt;
   1.2.10            adj is set as an unexplored tree node;
   1.2.11         elseif (#IntPts(curt,adj) > 0)
   1.2.12            find the leaf node lp closest to curt;
   1.2.13            make lp's root the same as curt's root;
   1.2.14         end if;
   1.2.15   end while;
   1.2.16 end while;
         /* refine the discretization steps, if necessary */
   1.2.17 if $\min\{\Delta_1, \Delta_2\} \geqslant \Delta_{\min}$ then
   1.2.18    for each (gc that should refined according to the criteria)
   1.2.19         refine the nine grids with gc in the middle;
   1.2.20         for the new grid points and intersection points, repeat from 1.2.1;
   1.3: connect the obtained points in $\Im$ to form approximate polygons (i.e., $\zeta^*_{syn}$);
2: compute $\zeta_{syn} := \zeta^*_{syn} \cap \zeta_{rate}$;
   2.1: discretize $\zeta_{rate}$ on $x_1x_2$-plane with the same grid (possibly being refined);
   2.2: intersect $\zeta^*_{syn}$ and $\zeta_{rate}$ grid by grid;
```



(a) find largest distance $d_{\max}$ (b) ζ_{syn}, $\zeta_{syn}^{\max}$, $\zeta_{syn}^{\min}$

Figure 6.11: Error bound analysis

6.4.5 Efficiency and Accuracy

Time Complexity. In the worst case, the discretization step is $\Delta_{\min}$ and there are $N_i = \frac{u_i - l_i}{\Delta_{\min}}$ $(i = 1, 2)$ subintervals. Obtaining the closed-formed expression of $f(x_1, x_2)$ (cf. (6.6)), takes $\mathcal{O}(n^2qt)$ time by computing the transient probability vector. For the initialization, $2(N_1{+}N_2{+}2)$ polynomial equations have to be solved with precision $2^{-\beta}$ where β is the bit precision. Using the algorithm in [Pan00], this is of the time complexity $\mathcal{O}\left(\hat{k}_\varepsilon^2 \log(\hat{k}_\varepsilon) \log(\beta\hat{k}_\varepsilon)\right)$, where $\hat{k}_\varepsilon$ is the degree of the polynomial $f(x_1, x_2)$ (cf. (6.5)). For the points labeling part, the time complexity is of the order of the number of grid points, i.e., $\mathcal{O}(N_1N_2)$. Evaluating $f(x_1, x_2)$ at each grid point takes $\mathcal{O}(\hat{k}_\varepsilon)$ time. To sum them up, we have:

Theorem 6.16 *The worst case time complexity of the symbolic synthesis region algorithm is:*

$$\mathcal{O}\left(n^2qt + \hat{k}_\varepsilon N_1N_2 + \hat{k}_\varepsilon^2 \log(\hat{k}_\varepsilon) \log(\beta\hat{k}_\varepsilon)(N_1{+}N_2)\right),$$

where $\hat{k}_\varepsilon$ is of the order $\mathcal{O}(qtm\mathcal{K})$ with $\mathcal{K}$ the maximal degree of the polynomial expressions in the pCTMC.

Error Bound. An important question is how accurate the derived synthesis region is. Let us explain this by using Fig. 6.11(a), where the accurate region is the gray area and its approximation is the dashed polygon. Let d_i be the distance between the line segment approximating the curve and the tangent (with the same slope) to the curve in the grid cell i. Let $d_{\max} = \max_i\{d_i\}$ be the largest of such distance. It is, however, very costly to compute every d_i and thus $d_{\max}$. In practice, $d_{\max}$ is taken to be $\sqrt{\Delta_1^2 + \Delta_2^2}$ which is the maximal value it can take. The top-rightmost distance in Fig. 6.11(a) is very close to this upper bound.

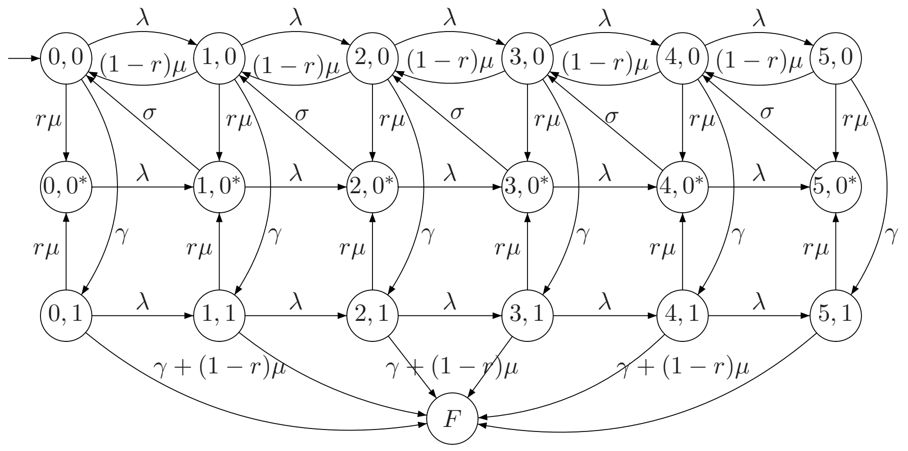


Figure 6.12: A storage system with probabilistic error checking ($qc = 5$)

Given the approximate polygon ζ_{syn} (dashed polygon in Fig. 6.11(b)) and $d_{\max}$, we can construct polygons $\zeta_{syn}^{\max}$ (the largest polygon in Fig. 6.11(b)) and $\zeta_{syn}^{\min}$ (the smallest polygon), where distance $d_{\max}$ is added and subtracted from the boundary of ζ_{syn}, respectively. The points in $\zeta_{syn}^{\max} \backslash \zeta_{syn}^{\min}$ *may* induce a valid CTMC, while the points in $\zeta_{syn}^{\min}$ *always* induce a valid CTMC. $\zeta_{syn}^{\min}$ can be regarded as the "safe" synthesis region.

6.4.6 Case Study

To see how the symbolic approach works in real-life cases, we apply it to a concrete case study from the literature. A *storage system with error checking* incorporates redundant data for discovery and recovery from errors caused by hardware or software faults [CY95a]. The use of redundancy enhances the reliability of the storage system but unavoidably degrades the system's performance because of the extra processing time required for error checking. Typically, on every request it is checked whether an error has occurred or not. *Probabilistic error checking* can be applied to reduce the error checking overhead. In particular, each access operation will be followed by an error checking with probability $r \in [0, 1]$, instead of almost surely (i.e., $r = 1$). Such a storage system can be modeled by a pCTMC as indicated in Fig. 6.12.

The storage system is *1-correctable*, i.e., the system can recover from a single error, but fails (state F) if two or more errors occur. We suppose that all access operations (reads and writes) as well as the error checking are atomic and all delays involved (such as arrivals, checks, etc.) are exponentially distributed. Access operation requests *arrive* with rate λ and are *served* with rate μ; the hardware/software will *fail* with rate γ, while the *error checking* takes place with rate σ. The arrived but not yet served requests are queued. We assume a queue capacity $qc = 5$. The states of the pCTMC are of the form (i, j), where i is the number of queued access operation requests and j the number of errors (0 or 1); an asterisk indicates that an error check is being performed. The property of interest is $\mathcal{P}_{\leqslant p}(\Diamond^{\leqslant t} s_F)$.

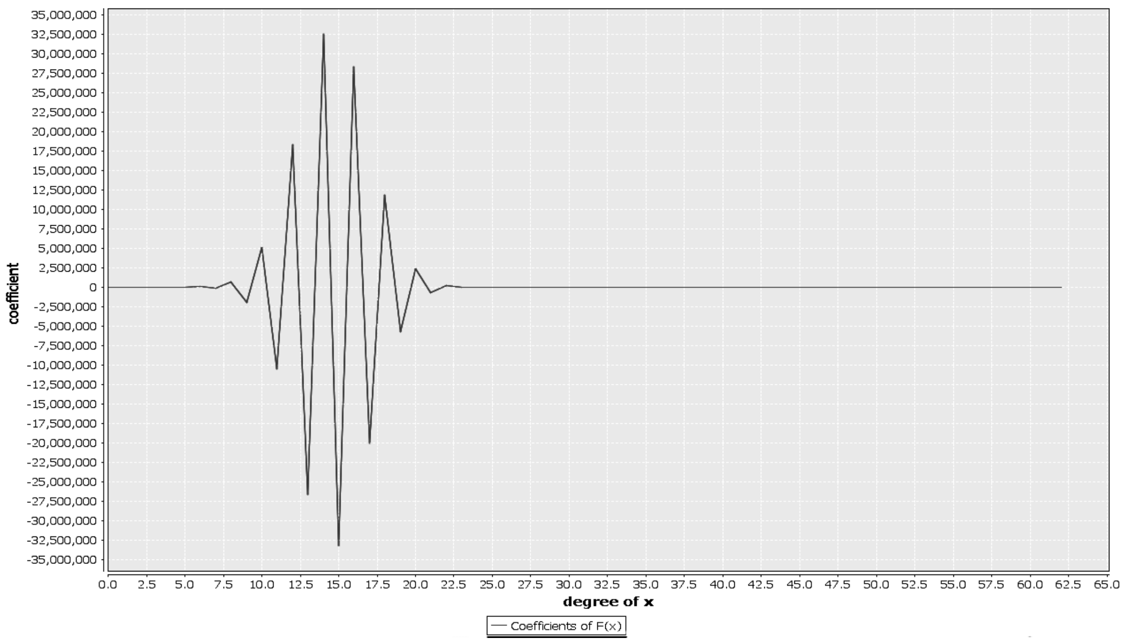
Typically, the probability r can be *logically adjusted* to guarantee some given specification. In the following, we show the experimental results for one parameter r, i.e., we determine for which error checking probability r it is guaranteed that the probability to reach the fail state (within the deadline) is low.

The first step of the symbolic approach is to establish the polynomial f — the transient probability (cf. (6.6), page 100). Since the coefficients of the polynomial range over a potentially very large domain, we applied the Java-supported data structure, the BigDecimal class, which results in very accurate calculation. We report the results for the following configurations:

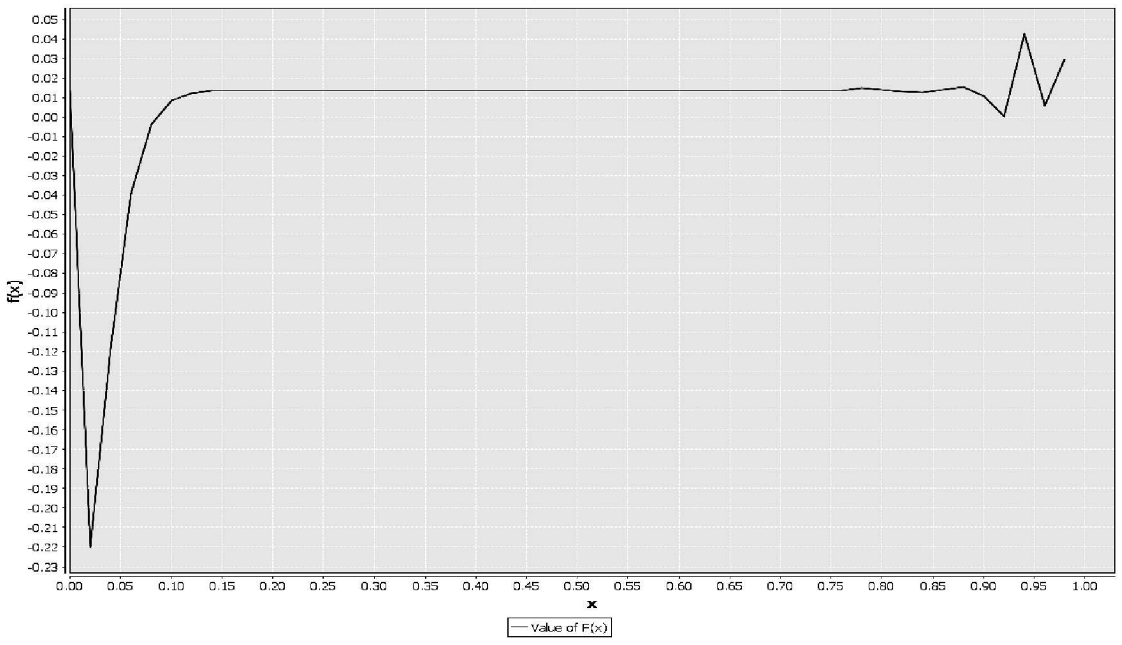
system parameters	program parameters	time	queue capacity	
			$qc = 6$	$qc = 10$
$\sigma = 0.5$				
$\mu = 0.5$	$\varepsilon = 10^{-3}$	$t = 100$	✓	✓
$\lambda = 0.3$	$\Delta = 0.5$	$t = 150$		✓
$\gamma = 5 \times 10^{-5}$		$t = 200$	✓	✓

Table 6.2: Experiment configurations

The coefficients and values of the resulting function $f(x)$ are shown in Fig. 6.13 and 6.14 for different time t and queue capacity qc. Here, the variable x is the error checking probability r. For the case $(t = 150, qc = 10)$, Fig. 6.13(a) shows the relationship between the degree of x and the coefficients. Note that the coefficients range from -10^7 to 10^7, however, the abrupt change of coefficients only appears for degrees from between 10 and 20. For the other degrees, the coefficients are very small, almost close to 0. This polynomial in Fig. 6.13(a) has a high degree (up to 60, see the x-axis) and presents a pathological shape, which might suffer from numerical errors. This also explains why we apply the BigDecimal data structure — basically to avoid the numerical errors that are introduced by the less accurate coefficients. The efficiency of the algorithm, however, is relative low, also due to the BigDecimal data structure. Having the function $f(x)$ at our disposal, the values of $f(x)$ can be easily computed, given $x \in [0, 1]$. The results are shown in Fig. 6.13(b). Note that for $x \in [0, 0.08]$, the value of $f(x)$ — the transient probability — is less than 0, which indicates that there exist errors in the computation, as for every $x \in [0, 1]$, the reachability probability to the fail state should be positive. Note that $x = 0$ means that there is no error checking at all. Although the *Simple Newton's Root Finder* and the *Lindsey-Fox Grid Search*

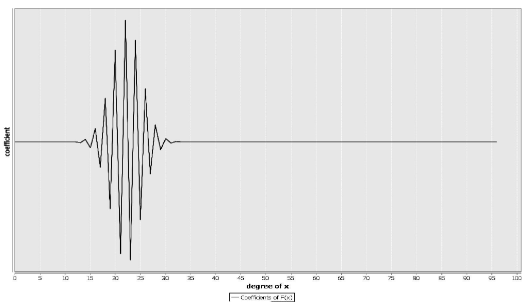


(a) coefficients of $f(x)$ $(t = 150, qc = 10)$

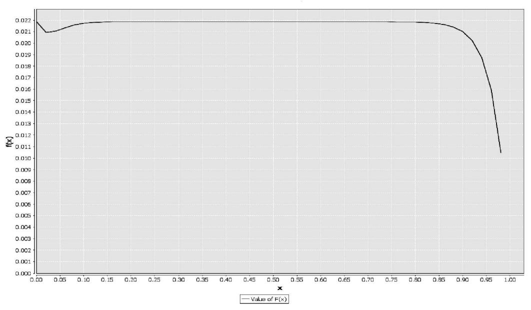


(b) value of $f(x)$ $(t = 150, qc = 10)$

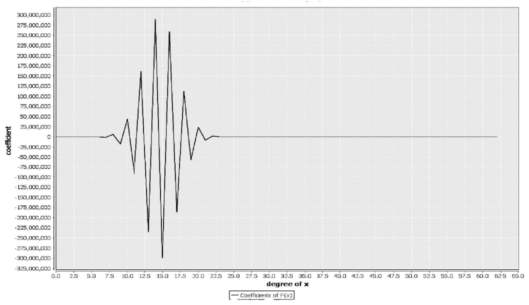
Figure 6.13: Coefficients and values of $f(x)$ for $t = 150$ and $qc = 10$



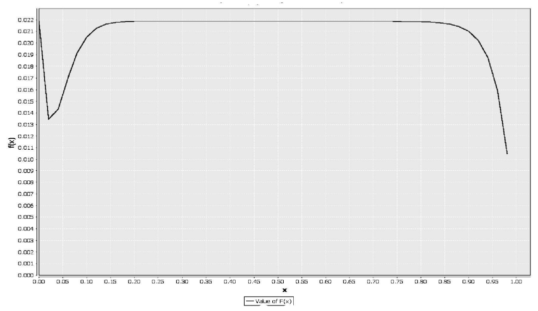
(a) coefficients of $f(x)$ $(t = 100, qc = 6)$



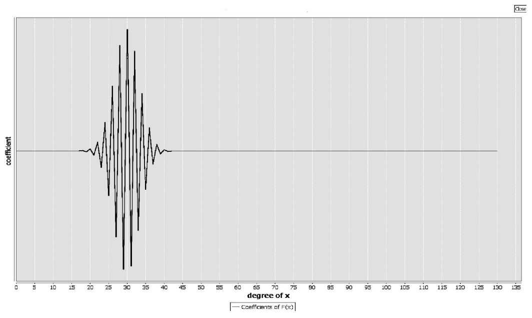
(b) value of $f(x)$ $(t = 100, qc = 6)$



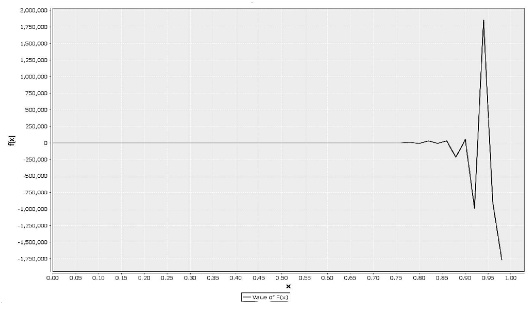
(c) coefficients of $f(x)$ $(t = 100, qc = 10)$



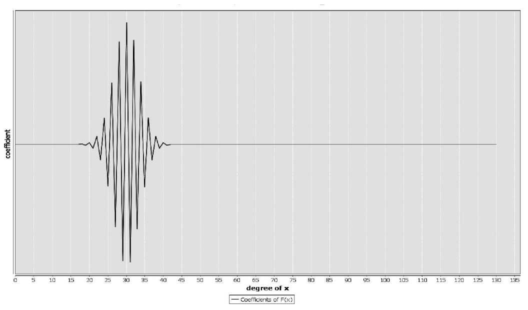
(d) value of $f(x)$ $(t = 100, qc = 10)$



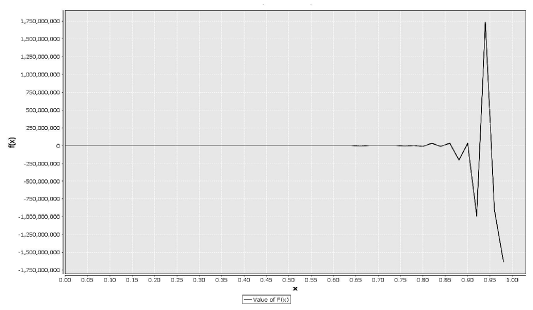
(e) coefficients of $f(x)$ $(t = 200, qc = 6)$



(f) value of $f(x)$ $(t = 200, qc = 6)$



(g) coefficients of $f(x)$ $(t = 200, qc = 10)$



(h) value of $f(x)$ $(t = 200, qc = 10)$

Figure 6.14: Coefficients and values of $f(x)$ for different t and qc

algorithm described in [SBFT03] have also been implemented, the feasibility of the polynomial itself is suspicious. Thus we don't report the results for the solvers here. For the other cases, as illustrated in Fig. 6.14, the curves share a similar pattern and are subject to similar problems.

Due to the above experimental results and reasoning, given this specific kind of polynomials, neither the correctness nor the performance of the proposed algorithm is satisfactory. We will, therefore, propose an alternative approach in the next section. Note that it is still worth investigating the current approach further. The idea of looking into the intersection points is natural and straightforward. The algorithm for labeling the grid and intersection points (and determining the "ownership" of a point) is novel and can be adapted to similar problems. We believe that given other "normal" polynomials, where the numerical instability is no longer a problem, this approach could work.

6.5 The Non-Symbolic Approach

Due to the high numerical instability of the obtained polynomials, we propose this non-symbolic approach which does not involve any polynomials. Instead, we collect and explore more grid points (i.e., "samples"), for each of which the pCTMC is first instantiated to be a CTMC, followed by a standard transient analysis (e.g., by uniformization) on this obtained non-parametric CTMC. Given the resulting transient probability, it is easy to decide whether a grid point is inside the synthesis region (if $\tilde{\wp}_{s_g}(t)$ lies in $[p_l, p_u]$) or not (otherwise). The main difficulty in this approach is how to closely approximate the boundary curve by exploring as few grid points as possible. This is done by choosing a good refinement strategy that only refines the grid cells where boundary curves go through. This will become clear later.

6.5.1 Marking and Refinement

Marking. In the marking step, we first determine for all grid points whether they are positive, neutral, or negative (Section 6.3.2, page 103). We then mark all grid cells gc according to its four grid points as follows:

$$gc = \begin{cases} \top & \text{if } gc\text{'s grid points are } 4\top \text{ or } 3\top 1⌶ \\ \bot & \text{if } gc\text{'s grid points are } 4\bot \text{ or } 3\bot 1⌶ \\ ? & \text{otherwise} \end{cases}$$

where e.g. $3\top 1⌶$ means 3 out of 4 four grid points are positive and 1 is neutral. A grid cell gc is *positive* $\top$ (resp. *negative* $\bot$) if it is assumed to be totally in (resp. outside) the synthesis region. gc is *unknown* ? if the boundary curves might go through the grid cell. Note that if two adjacent grid points have opposite values (i.e., one positive and one negative), then there must exist at least one intersection point on the line between the two grid points. In other words, the boundary curve goes through this line (thus also this grid).

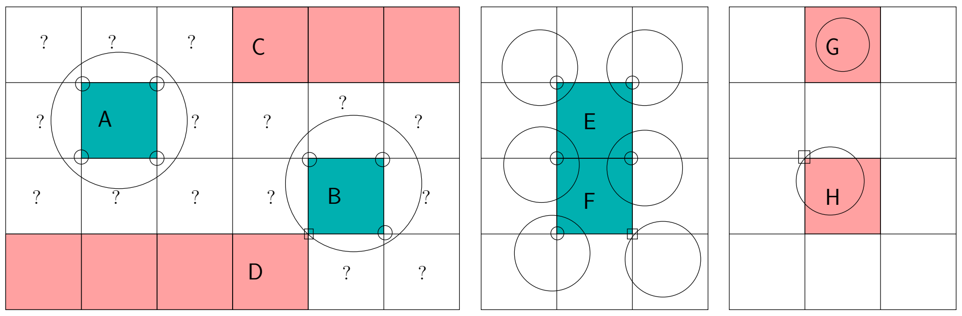
The above grid-cell marking gives an approximate criterion to determine whether a given grid cell is inside or outside the synthesis region. We illustrate the different cases in the following example.

Example 6.17 *Given the grid and accurate synthesis regions (big circles) in Fig. 6.15(a), the positive (resp. neutral) grid points are marked with small circles (resp. squares). All the unmarked grid points are negative. The positive grid cells* A *($4\top$) and* B *($3\top 1\mathbb{I}$) are marked with $\top$ (dark gray), while the negative* C *($4\bot$) and* D *($3\bot 1\mathbb{I}$) are marked with $\bot$ (light gray). The neutral grid cells are marked with ?.* ♦

Note that it is an approximate criterion since it is possible to have both "false positive" and "false negative" grid cells, where the former falsely includes the invalid region into the approximate synthesis region and the latter falsely excludes the valid region from the approximate synthesis region.

Example 6.18 *For grid cells* E *($4\top$) and* F *($3\top 1\mathbb{I}$) in Fig. 6.15(b), they will be marked with $\top$ according to the criterion. However, not the entire regions in* E *and* F *belong to the accurate synthesis region. Likewise,* G *and* H *in Fig. 6.15(c) will be marked with $\bot$ even though there exist positive regions in them.* ♦

We remark that the false region problem can be alleviated when small grid steps are applied. Since then the "false" regions are small and even if they are neglected, it is still acceptable. So far we have marked all the grid cells; in the following we will show how to refine the ?-marked grid cells.

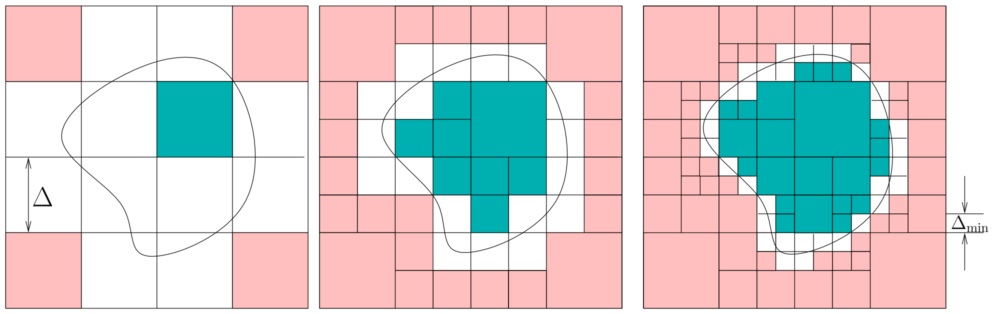


(a) Marking criterion (b) False positive (c) False negative

Figure 6.15: Grid cell marking

Refinement. For a grid cell gc that needs a refinement, we half the grid step size in each dimension and obtain four small grid cells out of gc. For each of the small grid cells, the same marking-and-refinement procedure is applied until the pre-defined grid step size $\Delta_{\min}$ is reached.

Example 6.19 *Consider the accurate synthesis region and the initial grid as in Fig. 6.16(a), we refine all the white grid cells (the ?-grid cells) and obtain the refined grid as in Fig. 6.16(b). We can further mark and refine such that Fig. 6.16(c) will be derived. The refinement will terminate once the minimal grid step $\Delta_{\min}$ has been reached.* ♦



(a) Initial grid (b) 1st refinement (c) 2nd refinement

Figure 6.16: Refinement iterations

6.5.2 Obtaining the Approximate Synthesis Region

Determine and Connecting the Points. When the marking-refinement process terminates, we can determine the set $\Im$ of points that are needed to connect the approximated polygon(s). First of all, all the neutrally marked grid points are for sure on the boundary and they are in $\Im$. Secondly, for each unknown grid cell gc, if its two adjacent grid points have opposite values, then there exists one intersection point (i.e., a point on the boundary curve) on the side between the two grid points. We then simply take the middle point of this side as the intersection point, i.e., being in $\Im$. This is an approximation and the error bound here, i.e., the maximal distance from the accurate and approximate intersection points is $\frac{1}{2}\Delta_{\min}$. The same maximal and minimal synthesis regions $\zeta_{syn}^{\max}, \zeta_{syn}^{\min}$ can be derived as in Section 6.4.5 (page 114), where $d_{\max} = \frac{\sqrt{2}}{4}\Delta_{\min}$. This is illustrated in Fig. 6.17, where the outermost and innermost polygons are $\zeta_{syn}^{\max}$ and $\zeta_{syn}^{\min}$, respectively. The middle polygon is the approximation we choose. Note the distance $d_{\max}$ between the two adjacent polygons, they can be no larger than $\frac{\sqrt{2}}{4}\Delta_{\min}$.

Given the set $\Im$ of points, we apply the same procedure as in Section 6.4.3 to connect them. We omit the details here.

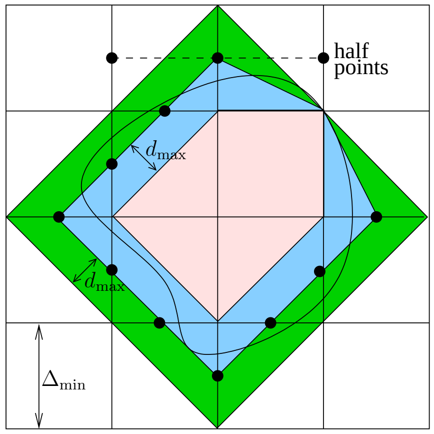


Figure 6.17: Error bound analysis

Main Algorithm. We present the main algorithm in Alg. 7. In the algorithm, we keep a list of unprocessed grid points and cells GP_{new} and GC_{new}, respectively. The mark-refinement process is from step 1.2.3 to 1.2.16, where step 1.2.4 - 1.2.7 is the grid points marking and step 1.2.8 - 1.2.10 is the grid cells marking. The refinement process is from step 1.2.11 to 1.2.15, where the split of every grid cell yields four sub-grid cells and five new grid points. We determine the set of points $\Im$ in step 1.2.17 - 1.2.19.

Time Complexity. Computing the transient probability (i.e., the reachability probability) takes $\mathcal{O}(n^2qt)$ time. For the first round (i.e., the round without any refinements, or to be more exact, the discretization step is the initial one: Δ), there are $(N_1+1)(N_2+1)$ CTMCs to instantiate (i.e., for each grid point), where each instantiation takes $\mathcal{O}(\hat{m})$ time, where $\hat{m}$ is the number of non-constant elements in $\mathbf{R}^{(\mathcal{X})}$. The marking procedure for N_1N_2 grid cells takes $\mathcal{O}(N_1N_2)$ time. Thus, for the first round, it takes $\mathcal{O}(\hat{m}(N_1+1)(N_2+1)+N_1N_2) = \mathcal{O}(\hat{m}N_1N_2)$.

We continue to discuss the second round, i.e., after the first refinement. Let us assume that c $(0 < c \leqslant 1)$ percent of the "fresh" grid cells is refined. "Fresh" grid cells are the newly generated grid cells in the last refinement round. In the first round, there are N_1N_2 grid cells and all of them are fresh. Recall that for each grid cell that is refined, 4 new small grid cells as well as 5 new grid points will be generated. Therefore, there are cN_1N_2 grid cells in the first round that have to be refined and in the second round $4cN_1N_2$ grid cells and $5cN_1N_2$ grid points will be newly generated[1]. In the second round, the time complexity is thus $\mathcal{O}(5c\hat{m}N_1N_2+4cN_1N_2) = \mathcal{O}(c\hat{m}N_1N_2)$. For the $(j+1)$-th round (after j-th refinement), the time complexity is $\mathcal{O}(c^j\hat{m}N_1N_2)$. Let

[1]Note that since some new grid points are shared by two neighboring grid cells, the actual number of new grid points is less than $5cN_1N_2$. However, for simplicity, we assume it is $5cN_1N_2$.

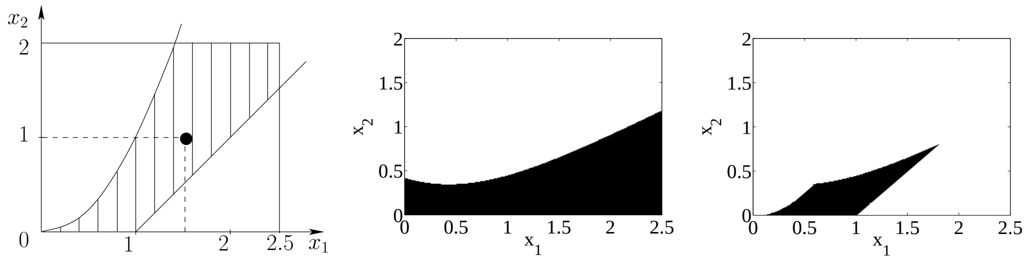
Algorithm 7 Obtaining approximate synthesis regions non-symbolically.

```
Require: pCTMC $\mathcal{C}^{(x_1,x_2)}$, formula $\mathcal{P}_{[p_l,p_u]}(\Diamond^{\leqslant t} s_g)$,
         initial discretization step $\Delta$, termination discretization step $\Delta_\perp$
Ensure: approximate synthesis region $\zeta_{syn}$ (a set of polygons)
1: compute $\zeta^*_{syn} := \zeta_{\leqslant p_u} \cap \zeta_{\geqslant p_l} \cap \zeta_{range}$;
   1.1: discretize $\zeta_{range}$ on $x_1x_2$-plane by grid step $\Delta$;
   1.2 /* compute the set of points $\Im$ that are needed to form polygon(s) */
   1.2.1: insert all grid cells to $GC_{new}$;
   1.2.2: insert all grid points to $GP_{new}$;
   1.2.3: while $\Delta > \Delta_{\min}$ do                      /* successive refinement*/
   1.2.4:     for each grid point $gp \in GP_{new}$ do
   1.2.5:         $GP_{new} := GP_{new} - \{gp\}$;
   1.2.6:         mark $gp$ with $\top$, $\perp$ or $\mathbb{I}$;
   1.2.7:         if $gp = \mathbb{I}$ then $\Im := \Im \cup \{gp\}$;      /* $gp$ on the boundary */
   1.2.8:     for each grid cell $gc \in GC_{new}$ do
   1.2.9:         $GC_{new} := GC_{new} - \{gc\}$;
   1.2.10:        mark $gc$ with $\top$, $\perp$ or ? and put it in $GC_\top$, $GC_\perp$ or $GC_?$;
   1.2.11:    for each $gc \in GC_?$ do
   1.2.12:        $GC_? := GC_? - \{gc\}$;
   1.2.13:        $\Delta := \Delta/2$ and split $gc$ into four sub-grid cells $gc_1, gc_2, gc_3, gc_4$;
   1.2.14:        $GC_{new} := GC_{new} \cup \{gc_1, gc_2, gc_3, gc_4\}$;
   1.2.15:        $GP_{new} := GP_{new} \cup \bigcup_{1\leqslant i\leqslant 5}\{gp_i\}$;
   1.2.16: end while;
     /* end of refinement, obtaining more approximate boundary points*/
   1.2.17: for each grid cell $gc \in GC_?$ do
   1.2.18:     if $\exists$ grid points $gp_1, gp_2$ of $gc$ such that
                 ($gp_1$ and $gp_2$ are adjacent) and ($gp_1 = \top \wedge gp_2 = \perp$) then
   1.2.19:        $\Im := \Im \cup \{$the middle point of $gp_1$ and $gp_2\}$;
   1.3: connect the obtained points in $\Im$ to form approximate polygons (i.e., $\zeta^*_{syn}$);
2: compute $\zeta_{syn} := \zeta^*_{syn} \cap \zeta_{rate}$;
   2.1: discretize $\zeta_{rate}$ on $x_1x_2$-plane with the same grid (possibly being refined);
   2.2: intersect $\zeta^*_{syn}$ and $\zeta_{rate}$ grid by grid;
```

j be the smallest integer satisfying $\frac{\Delta}{2^j} \leqslant \Delta_{\min}$, i.e., $j \geqslant \log_2 \frac{\Delta}{\Delta_{\min}}$. (Note that j is the number of refinements.) The total time complexity is the sum of that of each round and equals $\mathcal{O}(\frac{c(1-c^j)}{1-c}\hat{m}N_1N_2)$. Therefore we have the following:

Theorem 6.20 *The worst case time complexity of the non-symbolic synthesis region algorithm is:*

$$\mathcal{O}(\frac{c(1-c^{\frac{\Delta}{\Delta_{\min}}})}{1-c}\hat{m}N_1N_2).$$



(a) ζ_{rate} (b) ζ^*_{syn} (c) $\zeta_{syn} = \zeta_{rate} \cap \zeta^*_{syn}$

Figure 6.18: Example synthesis region

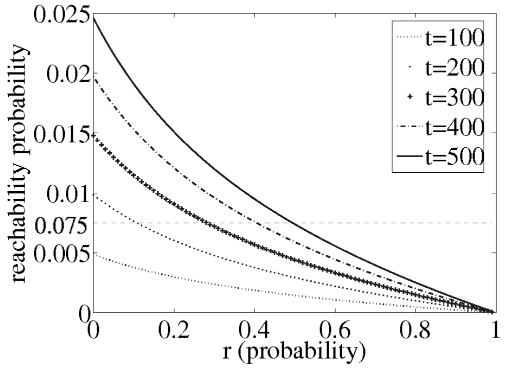
6.5.3 Case Study

In the following we will apply the non-symbolic approach to find synthesis regions in our running example:

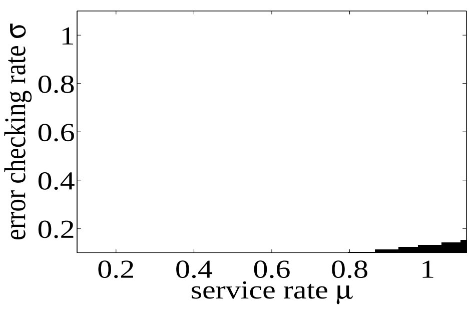
The Running Example. Consider the pCTMC in Fig. 6.1, and let the discretization steps $\Delta_1 = \Delta_2 = 0.01$, uniformization error bound $\varepsilon = 10^{-6}$ and property $\mathcal{P}_{\geqslant 0.5}(\Diamond^{\leqslant 0.5} s_g)$. Given the rate region ζ_{rate} as in Fig. 6.18(a), ζ^*_{syn} and ζ_{syn} are as shown in Fig. 6.18(b) and Fig. 6.18(c), respectively. We omit the grid lines so as to make the figure readable. Due to the fact that the refinement step is small and the boundary curve is smooth and has no abrupt changes, we do not apply any refinement in this case.

The Storage System. We consider the same storage system case study as in Section 6.4.6 (page 115). The property of interest is $\mathcal{P}_{\leqslant p}(\Diamond^{\leqslant t} s_F)$. Typically, the probability r can be *logically adjusted* to guarantee some given specification. On the other hand, μ, σ, and γ can be *physically adjusted* by changing the hardware/software. In the following, we show the experimental results for 1) one parameter r, i.e., for which error checking probability r can we guarantee that the probability to reach the fail state (within the deadline) is low, e.g., less than 0.0075? 2) two parameters μ and σ, i.e., how fast should read/write requests be handled and errors be checked in order to obtain a low failure probability? In all computations the error bound for uniformization is $\varepsilon = 10^{-6}$.

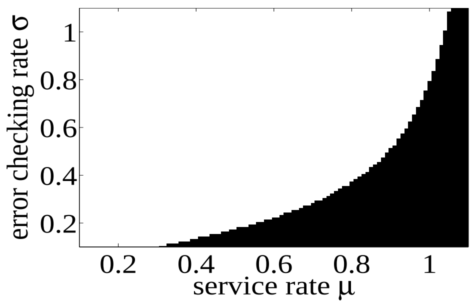
One Parameter: r. Let $\lambda = 0.3$ (0.3 access operation requests per second), $\mu = 0.5$, $\sigma = 0.5$ and $\gamma = 5 \times 10^{-5}$ (an average time of two consecutive errors is approximately 5 days). The parameter r has initial range $[0, 1]$ and the discretization step $\Delta = 0.01$. For the specification $\Phi_1 = \mathcal{P}_{\leqslant 0.0075}(\Diamond^{\leqslant t} s_F)$, where $t \in \{100, \ldots, 500\}$, the synthesis region is an interval as shown in Fig. 6.19(a), where the probability threshold $p = 0.0075$ is marked by a dashed line. For $t = 100$, the entire range $r \in [0, 1]$ is safe; intuitively, it is



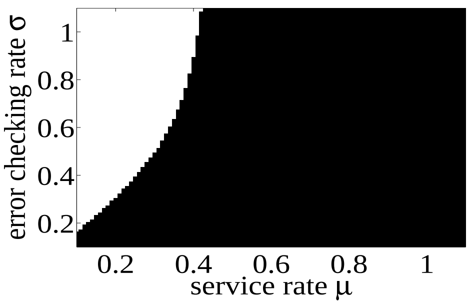
(a) 1 parameter r



(b) 2 parameters μ, σ $(r = 0.3)$



(c) 2 parameters μ, σ $(r = 0.5)$



(d) 2 parameters μ, σ $(r = 0.7)$

Figure 6.19: Synthesis regions for the storage system

less probable to fail given a small period of time. For $t = 200, \ldots, 500$, r approximately lies in the intervals $[0.1, 1], [0.28, 1], [0.41, 1], [0.5, 1]$, respectively. This shows that the larger the time bound is, the higher the error checking probability r should be in order to satisfy Φ_1.

Two Parameters: μ **and** σ**.** Let $\lambda = 0.3$, $\gamma = 5 \times 10^{-5}$ and $r = 0.3$, 0.5 or 0.7. The parameter ranges are $\mu \in [0.1, 1.1]$ and $\sigma \in [0.1, 1.1]$, with $\Delta_\mu = \Delta_\sigma = 0.01$. The initial region ζ_0 is the same rectangular area as ζ_{range}. For the specification $\Phi_2 = \mathcal{P}_{\leqslant 0.002}(\diamondsuit^{\leqslant 200} s_F)$, the synthesis regions are the black regions as shown in Fig. 6.19(b) through 6.19(d) for different values of r. Notice that the shape of the boundary curves is simple (i.e. without local protuberances), refinement is thus not performed. However, the stairs-like shape is due to the grid discretization. A refinement will make the curve smoother.

The higher the error checking probability r is, the larger the region for which Φ_2 holds. If error checking takes longer (i.e., with a low error checking rate σ), then it is less probable to fail. This is due to the assumption that during the error checking, no error will occur. This assumption is true because the states from the middle layer in

Fig.6.12 (page 115) do not have a direct transition to the failure state. On the other hand, if a request is served faster (i.e., with high service rate μ), then it is less probable to fail. This is because given a fixed failure rate γ, a larger service rate μ can make the CTMC stay more in the middle layer, where it is less probable to fail. In practice, an error checking is preferred to be carried out fast for the sake of efficiency. We, therefore, can adjust the combination of μ and σ to meet the specification and enhance the efficiency.

6.6 Summary

The central question that we considered is: for a CTMC with parametric random delays, can we find sets of parameter values for which a given specification is satisfied? We presented two algorithms that can yield an approximation of these values for CTMCs and time-bounded reachability specifications. To the best of our knowledge, this is the first result considering parameter synthesis approach in this setting. We make some comparison on the two approaches as follows.

6.6.1 Comparison

As we have shown in the case study, the polynomials we obtain in the *symbolic approach* are usually of very high degree and the coefficients in the polynomials range over a very large domain. These facts explain that solving the polynomials would in most cases suffer from numerical instability, and consequently derive inaccurate (or even incorrect) transient probabilities (which fall outside $[0, 1]$). That is to say, although getting intersection points seems to be the most direct way to derive the points needed for the polygons, in practice it does not work well.

In contrast, the *non-symbolic approach* usually takes more rounds of refinement to locate the points; however, the computation is much easier and with less numerical instability. The reason is that it is not done by exploring any polynomials, but by instantiating the CTMCs per point and applying the uniformization technique. It is worth mentioning that the Runge-Kutta-like methods [RT88][RST89] can also be applied to compute the transient probabilities, instead of using uniformization here. The Runge-Kutta-like methods might perform better under some special conditions, e.g., stiff Markov chains [RT88][MT91].

Both approaches might fail to detect some very small regions (e.g., within one grid) or connect the points in a different way as they should be (cf. e.g., Fig. 6.10). However, these can all be fixed if a smaller grid step is taken.

The symbolic approach might not be fully applicable in practice in our setting. However, given a different objective function, i.e., a function that can be solved stably, it would then be possible to apply it. The non-symbolic approach is more practical in this case. Our experiments are mainly based on it.

6.6.2 Related Work

Parameter synthesis has been studied for the following models:

Probabilistic Systems.

- Daws presented a language-theoretic approach to symbolic model checking of PCTL over DTMCs [Daw04]. The probability with which a path formula is satisfied is represented by a regular expression. A recursive evaluation of the regular expression yields an exact rational value when transition probabilities are rational, and rational functions when some probabilities are left unspecified as parameters of the system. This allows for parametric model checking by evaluating the regular expression for different parameter values.

- Lanotte, Maggiolo-Schettini and Troina studied a model of parametric probabilistic transition systems (PPTSs), where probabilities on transitions may be parameters [LMST07]. It was shown how instances of the parameters can be found to satisfy a given property, or the instances that either maximize or minimize the reachability probability.

- Hahn and Zhang considered the parametric probabilistic reachability problem [HHZ09]: Given a parametric Markov model with rewards and with non-determinism, compute the closed form expression representing the probability of reaching a given set of states. In the setting of reward models, the computation of parametric expected rewards was considered, while for MDPs they establish the maximal parametric reachability probability.

- Chamseddine et al. considered a variant of probabilistic timed automata called parametric determinate probabilistic timed automata [CDF$^+$08]. Such automata are fully probabilistic and it is possible to stay at a node only for a given amount of time. The residence time within a node may be given in terms of a parameter, and hence its concrete value is assumed to be unknown. A method was given for computing the expected time for a parametric determinate probabilistic timed automaton to reach an absorbing node. The method consists in constructing a variant of a Markov chain with costs (where the costs correspond to durations), and is parametric in the sense that the expected absorption time is computed as a function of the model's parameters.

Timed Automata.

- Alur, Henzinger and Vardi addressed the problem of deriving symbolic constraints on the timing properties required of real-time systems (e.g., message delivery within the time it takes to execute two assignment statements) by introducing parametric timed automata — finite-state machines whose transitions are constrained with parametric timing requirements [AHV93]. They showed that the emptiness question is in general undecidable. However, an algorithm was provided for checking the emptiness of the same restricted classes of parametric timed automata.

- It is known that the reachability problem for timed automata is decidable when the coefficients in the guards are rational numbers. Puri showed that the reachability problem is undecidable when the coefficients are chosen from the set $\{1, \sqrt{2}\}$ [Pur99]. A consequence of this is that the parameter synthesis problem for timed automata with even a single parameter is undecidable.

- Hune et al. presented an extension of the model checker UPPAAL, capable of synthesizing linear parameter constraints for the correctness of parametric timed automata [HRSV02]. It also showed that the emptiness problem for a subclass of parametric timed automata — L/U automata — is decidable.

- Bruyère, Dall'Olio and Raskin considered the problem of model-checking a parametric extension of the logic TCTL over timed automata and establish its decidability [BDR03]. Given a timed automaton, it was shown that the set of durations of runs starting from a region and ending in another region is definable in Presburger arithmetic (when the time domain is discrete) or in the theory of the reals (when the time domain is dense). With this logical definition, the parametric model-checking problem for the logic TCTL can easily be solved. More generally, it is possible to effectively characterize the values of the parameters that satisfy the parametric TCTL formula.

- Zhang and Cleaveland presented a local algorithm for solving the universal parametric real-time model-checking problem: given a real-time system and a temporal formula, both of which may contain parameters, and a constraint over the parameters, does every allowed parameter assignment ensure that the real-time system satisfies the formula [ZC05]? The approach relies on translating these model-checking problems into predicate equation systems, and then using an efficient proof-search algorithm to solve these systems.

- Bruyère and Raskin studied the model-checking and parameter synthesis problems of the logic TCTL over discrete-timed automata where parameters are allowed both in the model (timed automaton) and in the property (temporal formula) [BR07]. It was shown that the model-checking problem of TCTL extended with parameters is undecidable over discrete-timed automata with only one parametric clock. The undecidability result needs equality in the logic. It would become decidable if equality in the logic is not allowed.

Hybrid Systems.

- HyTech [AHH96][HHWT97] is an automatic tool for the analysis of hybrid automata. HyTech computes the condition under which a linear hybrid system satisfies a temporal requirement. Hybrid systems are specified as collections of automata with discrete and continuous components, and temporal requirements are verified by symbolic model checking. If the verification fails, then HyTech generates a diagnostic error trace.

- Frehse presented the tool PHAVer for the exact verification of safety properties of hybrid systems with piecewise constant bounds on the derivatives, so-called linear hybrid automata [Fre08]. Affine dynamics are handled by on-the-fly overapproximation and partitioning of the state space based on user-provided constraints and the dynamics of the system. PHAVer features exact arithmetic in a robust implementation that supports arbitrarily large numbers. To force termination and manage the complexity of the polyhedral computations, the methods to conservatively limit the number of bits and constraints of polyhedra were also proposed.

Chapter 7

Model Checking CTMCs Against Timed Automata Specifications

This chapter considers the problem of verifying CTMCs versus *linear* real-time specifications, which are given as timed automata. Concretely speaking, we explore the following problem: given a CTMC $\mathcal{C}$ and a linear real-time property provided as a *deterministic timed automaton* [AD94] (DTA) $\mathcal{A}$, what is the probability of the set of paths of $\mathcal{C}$ which are accepted by $\mathcal{A}$ ($\mathcal{C} \models \mathcal{A}$)? We consider two kinds of acceptance conditions: the reachability condition (in DTA$^\Diamond$) and the Muller acceptance condition (in DTA$^\omega$). The former accepts (finite) paths which reach some final state and the latter accepts (infinite) paths that infinitely often visit some set of final states.

We set off to show that this problem is well-defined in the sense that the path set is *measurable*. For DTA$^\Diamond$, we have shown that computing this probability is reducible to computing the reachability probability in a piecewise deterministic Markov process (PDP) [Dav93], a model that is used in, e.g., stochastic control theory and financial mathematics. This result relies on a product construction of CTMC $\mathcal{C}$ and DTA $\mathcal{A}$, denoted $\mathcal{C} \otimes \mathcal{A}$, yielding *deterministic Markov timed automata* (DMTA), a variant of DTA in which, besides the usual ingredients of timed automata, like guards and clock resets, the location residence time is exponentially distributed. We show that for DTA$^\Diamond$ $\mathcal{A}$ the probability of $\mathcal{C} \models \mathcal{A}$ coincides with the reachability probability of accepting paths in $\mathcal{C} \otimes \mathcal{A}$. The underlying PDP of a DMTA is obtained by a slight adaptation of the standard region construction. The desired reachability probability is characterized as the least solution of a system of *integral equations* that is obtained from the PDP. Finally, this probability is shown to be approximated by solving a system of *partial differential equations* (PDEs). For single-clock DTA$^\Diamond$, we show that the system of integral equations can be transformed into a system of *linear equations*, where the coefficients are solutions of some *ordinary differential equations* (ODEs), which can have either an analytical solution (for small state space) or an arbitrarily

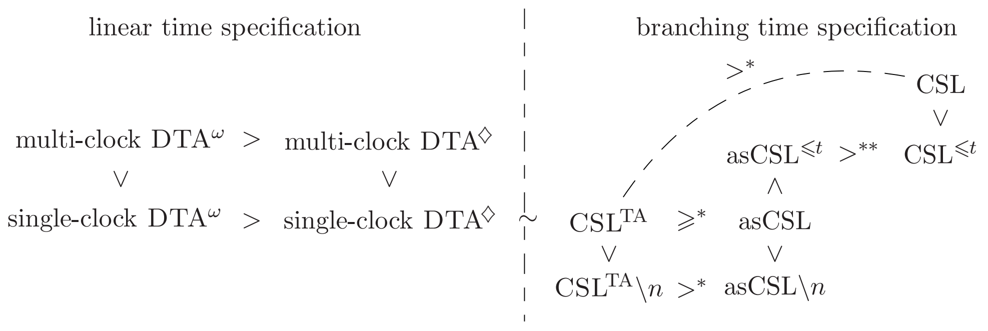

linear time specification
branching time specification
multi-clock DTA$^\omega$ > multi-clock DTA$^\Diamond$
single-clock DTA$^\omega$ > single-clock DTA$^\Diamond$
$\sim$
$>^*$
CSL
asCSL$^{\leqslant t}$ $>^{**}$ CSL$^{\leqslant t}$
CSL$^{\mathrm{TA}}$ $\geqslant^*$ asCSL
CSL$^{\mathrm{TA}}\backslash n$ $>^*$ asCSL$\backslash n$

Figure 7.1: A comparison of the expressive power of different specifications

closely approximated solution that can be computed efficiently. For DTA$^\omega$, by finding the accepting BSCCs in the region graph, the ω-regular acceptance condition is proven to be reducible to the finite paths case, i.e., the reachability problem.

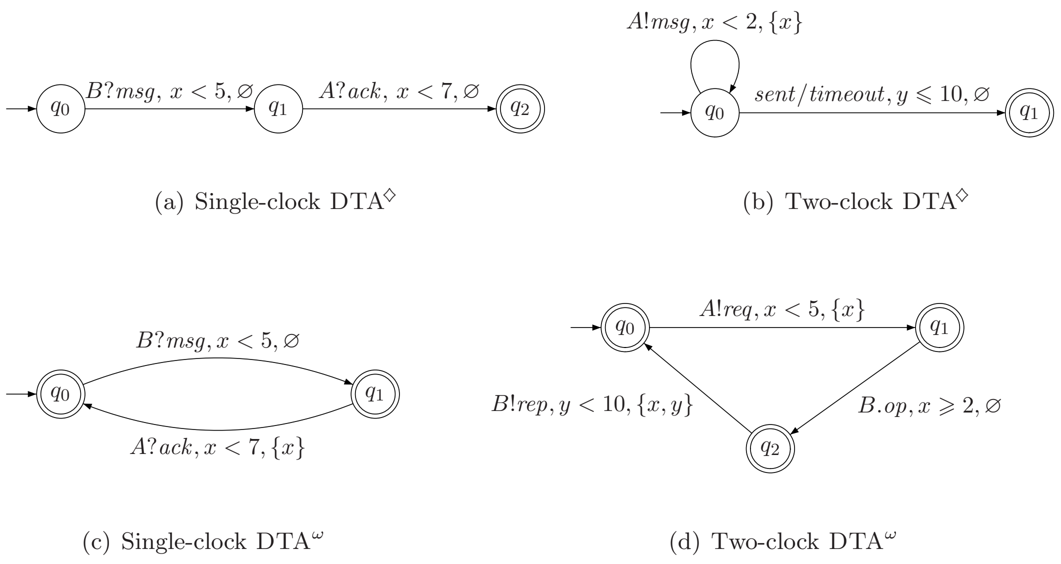
All the proofs of the theorems, propositions and lemmata can be found in the appendix.

Motivation of Using Timed Automata Specifications. To justify the use of timed automata as specifications, we will now exemplify the possible properties that can be specified by a timed automaton. Prior to that, we first make a comparison among the expressive power of CSL [BHHK03], asCSL [BCH$^+$07], CSL$^{\mathrm{TA}}$ [DHS09], DTA$^\Diamond$ [AD94] and DTA$^\omega$ [AD94] with arbitrary number of clocks.

Let $\mathbf{A}, \mathbf{B}$ be logic specifications. We write $\mathbf{A} \geqslant \mathbf{B}$ if $\mathbf{A}$ is *at least as expressive* as $\mathbf{B}$; $\mathbf{A} > \mathbf{B}$, if $\mathbf{A}$ is *strictly more expressive* than $\mathbf{B}$ and $\mathbf{A} = \mathbf{B}$ if they are *equally expressive* in the CTMC model. $\mathbf{A} \geqslant \mathbf{B}$ if for any $\Phi_B \in \mathbf{B}$, there exists $\Phi_A \in \mathbf{A}$ such that for any CTMC $\mathcal{C}$ and state s, $s \models^{\mathcal{C}} \Phi_A$ iff $s \models^{\mathcal{C}} \Phi_B$. $\mathbf{A} = \mathbf{B}$ iff $\mathbf{A} \geqslant \mathbf{B}$ and $\mathbf{B} \geqslant \mathbf{A}$. $\mathbf{A} > \mathbf{B}$ if $\mathbf{A} \geqslant \mathbf{B}$ and there exists $\Phi'_A \in \mathbf{A}$ such that there does not exist an equivalent formula $\Phi'_B \in \mathbf{B}$. When $\mathbf{A}$ and $\mathbf{B}$ are automata, the relations are defined in a similar way.

We illustrate the comparison of the expressive power of these specifications in Fig. 7.1, where CSL$^{\mathrm{TA}}\backslash n$ and asCSL$\backslash n$ represent the formulae without nesting of probability operators in respective logics; while asCSL$^{\leqslant t}$ and CSL$^{\leqslant t}$ represent the formulae with time intervals starting from 0 in respective logics. $\sim$ means that the path formulae of a CSL$^{\mathrm{TA}}$ is represented by a single-clock DTA$^\Diamond$. Those marked with $*$ are proven in [DHS09] and the one marked with $**$ is proven in [BCH$^+$07]. The remaining relations are straightforward to establish. Note that in the spectrum in Fig. 7.1, the one-clock DTA$^\Diamond$ as well as the CSL$^{\mathrm{TA}}$ build the link between the linear-time and branching-time specifications.

In the remainder of this chapter, we will focus on verifying the properties specified by the four variants of linear-time timed automata specifications in the left half in


$B?msg,\, x < 5, \varnothing$
$A?ack,\, x < 7, \varnothing$
q_0
q_1
q_2
$A!msg, x < 2, \{x\}$
$sent/timeout, y \leqslant 10, \varnothing$
$B?msg, x < 5, \varnothing$
$A?ack, x < 7, \{x\}$
$A!req, x < 5, \{x\}$
$B!rep, y < 10, \{x, y\}$
$B.op, x \geqslant 2, \varnothing$

(a) Single-clock DTA$^\Diamond$

(b) Two-clock DTA$^\Diamond$

(c) Single-clock DTA$^\omega$

(d) Two-clock DTA$^\omega$

Figure 7.2: Example properties specified by DTA$^\Diamond$ and DTA$^\omega$

Fig. 7.1. Their usage as specifications is shown, each by an example, in Fig. 7.2.

- Single-clock DTA$^\Diamond$

  It is pointed out in [DHS09] that "*A typical example is the responsiveness property 'with probability at least* 0.75*, a message sent at time* 0 *by a system* A *will be received before time* 5 *by system* B *and the acknowledgment will be back at* A *before time* 7*', a property that cannot be expressed in either* CSL *or* asCSL". The corresponding single-clock DTA$^\Diamond$ is as in Fig. 7.2(a), where the formal syntax and semantics will become clear later in Section 7.1.1.

- Multi-clock DTA$^\Diamond$

  In some situations, however, a single clock might not be sufficient, e.g., *with which probability that from time* 0 *system* A *will keep trying to send a message, where the interval between every two successive trials is less than* 2 *time units, and either the message has been successfully sent or the timeout is at time* 10*?* This is illustrated in Fig. 7.2(b). Actually, it is a time-bounded property, with a global clock y that is never reset.

- Single-clock DTA$^\omega$

  On the other hand, if one is interested in the infinite behavior of a system, then timed ω-automata can be used. In this chapter we focus on DTA with Muller acceptance conditions (DTA$^\omega$). For the single-clock case, an example property is: *with which probability does it globally hold that "a message sent at time* 0 *by system A will be received before time* 5 *by system B and the acknowledgment will be received by A before time* 7*"?* (cf. Fig. 7.2(c)) Note that the single-clock DTA$^\diamond$ in Fig. 7.2(a) only requires the responsiveness to hold once, however, the single clock DTA$^\omega$ requires that the responsiveness holds infinitely often.

- Multi-clock DTA$^\omega$

  The following property needs more than one clock for infinite behaviors: *with which probability does it globally hold that "starting from time* 0*, system A will send out a request before time* 5*; it takes system B at least* 2 *time units to operate on this request, and A will receive a reply from B before time* 10*"?* (cf. Fig. 7.2(d))

The above properties are widely used in practice, therefore, verifying timed automata specifications becomes very crucial.

7.1 Problem Statement

In the remainder of this chapter, we will first focus on the specifications for finite behaviors (Section 7.2 -7.3), and then we deal with the infinite behaviors (Section 7.4).

7.1.1 Deterministic Timed Automata

(Clock) Variables and Valuations. Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a set of variables in $\mathbb{R}$. An $\mathcal{X}$-valuation is a function $\eta : \mathcal{X} \to \mathbb{R}$ assigning to each variable x a value $\eta(x)$. Let $\mathcal{V}(\mathcal{X})$ denote the set of all valuations over $\mathcal{X}$. A *constraint* over $\mathcal{X}$, denoted by g, is a subset of $\mathbb{R}^n$. Let $\mathcal{B}(\mathcal{X})$ denote the set of constraints over $\mathcal{X}$. An $\mathcal{X}$-valuation η *satisfies* constraint g, denoted as $\eta \models g$ if $(\eta(x_1), \ldots, \eta(x_n)) \in g$.

Occasionally we use a special case of *nonnegative* variables, called *clocks*. We write $\vec{0}$ for the valuation that assigns 0 to all clocks. For a subset $X \subseteq \mathcal{X}$, the *reset* of X, denoted $\eta[X := 0]$, is the valuation η' such that $\forall x \in X.\ \eta'(x) := 0$ and $\forall x \notin X.\ \eta'(x) := \eta(x)$. For $\delta \in \mathbb{R}_{\geqslant 0}$, $\eta + \delta$ is the valuation η'' such that $\forall x \in \mathcal{X}.\ \eta''(x) := \eta(x) + \delta$, which implies that all clocks proceed at the same speed, or equivalently, $\forall x_i \in \mathcal{X}.\ \dot{x}_i = 1$. A *clock constraint* on $\mathcal{X}$ is an expression of the form $x \bowtie c$, or $x - y \bowtie c$, or the conjunction of any clock constraints, where $x, y \in \mathcal{X}$, $\bowtie \in \{<, \leqslant, >, \geqslant\}$ and $c \in \mathbb{N}$.

Definition 7.1 (DTA) *A* deterministic timed automaton *is a tuple* $\mathcal{A} = (\Sigma, \mathcal{X}, Q, q_0, Q_{\mathbf{F}}, \to)$ *where*

- Σ *is a finite* alphabet*;*
- $\mathcal{X}$ *is a finite set of* clocks*;*
- Q *is a nonempty finite set of* locations*;*
- $q_0 \in Q$ *is the* initial location*;*

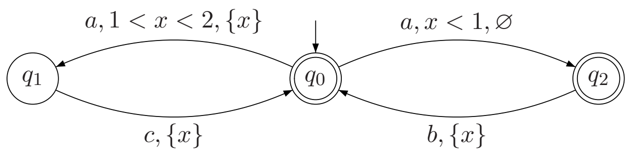


Figure 7.3: DTA with Muller acceptance conditions (DTA$^\omega$)

- $\rightarrow \in Q \times \Sigma \times \mathcal{B}(\mathcal{X}) \times 2^{\mathcal{X}} \times Q$ *is an* edge relation *satisfying:* $q \xrightarrow{a,g,X} q'$ *and* $q \xrightarrow{a,g',X'} q''$ *with* $g \neq g'$ *implies* $g \cap g' = \varnothing$*; and*
- $Q_{\mathbf{F}}$ *is the* Y acceptance condition*, where*
  - ▶ *if* Y=reachability*, then* $Q_{\mathbf{F}} := Q_F \subseteq Q$ *is a set of* accepting locations*;*
  - ▶ *if* Y=Muller*, then* $Q_{\mathbf{F}} := Q_{\mathcal{F}} \subseteq 2^Q$ *is the* acceptance family*.*

We refer to $q \xrightarrow{a,g,X} q'$ as an *edge*, where $a \in \Sigma$ is the input symbol, the *guard* g is a clock constraint on the clocks of $\mathcal{A}$, $X \subseteq \mathcal{X}$ is a set of clocks to be reset and q' is the successor location. The intuition is that the DTA $\mathcal{A}$ can move from location q to location q' when the input symbol is a and the guard g holds, while the clocks in X should be reset when entering q'. Note that we don't consider diagonal constraints like $x - y \bowtie c$ in DTA. However, it is known that this does not harm the expressiveness of a TA [BPDG98].

We will denote DTA$^\diamond$ and DTA$^\omega$ for the DTA with reachability and Muller acceptance conditions, respectively; while with DTA we denote the general case covering both DTA$^\diamond$ and DTA$^\omega$. As a convention, we assume each location $q \in Q_F$ in DTA$^\diamond$ is a sink.

An (infinite) *timed path* in $\mathcal{A}$ is of the form $\theta = q_0 \xrightarrow{a_0,t_0} q_1 \xrightarrow{a_1,t_1} \cdots$, satisfying that $\eta_0 = \vec{0}$, and for all $j \geqslant 0$, it holds that $t_j > 0$, $\eta_j + t_j \models g_j$ and $\eta_{j+1} = (\eta_j + t_j)[X_j := 0]$, where η_j is the clock evaluation on *entering* q_j. Let *inf*(θ) denote the set of states $q \in Q$ such that $q = q_i$ for infinitely many $i \geqslant 0$. Furthermore, all the definitions on paths in CTMCs can be adapted.

Definition 7.2 (DTA accepting paths) *An infinite path θ is* accepted *by a* DTA$^\diamond$ *if there exists some $i \geqslant 0$ such that $\theta[i] \in Q_F$; θ is accepted by a* DTA$^\omega$ *if inf$(\theta) \in Q_{\mathcal{F}}$.*

Example 7.3 (DTA$^\diamond$ and DTA$^\omega$) *An example* DTA$^\diamond$ *is shown in Fig. 7.4(c) (page 140) over the alphabet $\{a, b\}$. The reachability acceptance condition is characterized by the accepting location set $Q_F = \{q_1\}$. The unique initial location is q_0 and the guards $x < 1$ and $1 < x < 2$ are disjoint on the edges labeled with a and emanating from q_0. This guarantees the determinism.*

We then consider the DTA$^\omega$ *in Fig. 7.3 over* $\Sigma = \{a, b, c\}$. *The unique initial location is* q_0 *and the Muller acceptance family is* $Q_{\mathcal{F}} = \{\{q_0, q_2\}\}$. *Since* $Q_{\mathcal{F}}$ *is a singleton, we can indicate it in the figure by the double-lined states. Any accepting path should cycle between the states* q_0 *and* q_1 *for* finitely *many times, and between states* q_0 *and* q_2 *for* infinitely *many times. The determinism is guaranteed of the similar reason.*

♦

Remark 7.4 (Muller not Büchi) *According to* [AD94], *the expressive power of* (deterministic) timed Muller automata (D)TMA[1] *and* (deterministic) timed Büchi automata (D)TBA *has the following relation:*

$$\text{TMA} = \text{TBA} > \text{DTMA} > \text{DTBA}.$$

Also notice that DTMA *are closed under all Boolean operators (union, intersection and complement), while* DTBA *are* not *closed under complement. These two points justify our choice of* DTMA *(or* DTA$^\omega$*) instead of* DTBA.

Remark 7.5 (Successor location) *Due to the determinism, we can replace the transition relation* $\rightarrow \in Q \times \Sigma \times \mathcal{B}(\mathcal{X}) \times 2^{\mathcal{X}} \times Q$ *by a function* $succ : Q \times \Sigma \times \mathcal{B}(\mathcal{X}) \mapsto 2^{\mathcal{X}} \times Q$. *Namely, given a location* q, *an action* a *and a guard* g, *there is at most one successor location* q'. *Note that the set of reset clocks is irrelevant to the successor location. Therefore, if only the successor location is of interest, then we can thus simplify the function* $succ$ *to* $\widetilde{succ} : Q \times \Sigma \times \mathcal{B}(\mathcal{X}) \mapsto Q$, *i.e.,* $q' = \widetilde{succ}(q, a, g)$.

7.1.2 The Model Checking Problem

To simplify the notations, we assume w.l.o.g., that a CTMC has only one initial state s_0, i.e., $\alpha(s_0) = 1$, and $\alpha(s) = 0$ for $s \neq s_0$.

Definition 7.6 (CTMC paths accepted by a DTA) *Given a* CTMC $\mathcal{C} = (S, \text{AP}, L, s_0, \mathbf{P}, E)$ *and a* DTA $\mathcal{A} = (2^{\text{AP}}, \mathcal{X}, Q, q_0, Q_{\mathbf{F}}, \rightarrow)$, *a* CTMC *path* $\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \cdots$ *is* accepted by $\mathcal{A}$ *if the* DTA *path*

$$q_0 \xrightarrow{L(s_0),t_0} \underbrace{\widetilde{succ}\big(q_0, L(s_0), g_0\big)}_{q_1} \xrightarrow{L(s_1),t_1} \underbrace{\widetilde{succ}\big(q_1, L(s_1), g_1\big)}_{q_2} \cdots$$

is accepted by $\mathcal{A}$, *where* $\eta_0 = \vec{0}$, g_i *is the unique guard (if it exists) such that* $\eta_i + t_i \models g_i$ *and* $\eta_{i+1} = (\eta_i + t_i)[X_i := 0]$.

The model checking problem on CTMC $\mathcal{C}$ against DTA $\mathcal{A}$ is to compute the probability of the set of paths in $\mathcal{C}$ that can be accepted by $\mathcal{A}$. Formally, let

$$Paths^{\mathcal{C}}(\mathcal{A}) := \{\, \rho \in Paths_{\mathcal{C}} \mid \rho \text{ is accepted by DTA } \mathcal{A} \,\}.$$

[1]In this thesis, DTMA is referred to as DTA$^\omega$.

Prior to computing the probability of this set, we first prove its measurability:

Theorem 7.7 *For any* CTMC $\mathcal{C}$ *and* DTA $\mathcal{A}$, $\mathit{Paths}^{\mathcal{C}}(\mathcal{A})$ *is measurable.*

Proof: The proof can be found in Appendix A.1, page 163. ■

The following sections deal with computing the probability of $\mathit{Paths}^{\mathcal{C}}(\mathcal{A})$, denoted $\mathit{Prob}^{\mathcal{C}}(\mathcal{A}) = \Pr\big(\mathit{Paths}^{\mathcal{C}}(\mathcal{A})\big)$, under both reachability and Muller acceptance conditions.

7.2 Product of CTMC and DTA

As the traditional way of verifying the automata specifications, a product between the model and the automaton is built first and the (adapted) property can then be checked on the product model. Our approach is carried out in the same fashion. In this section, we focus on building the product (and some more transformations on the product) and in Section 7.3 and 7.4, the probability measure $\mathit{Prob}^{\mathcal{C}}(\mathcal{A})$ will be computed.

7.2.1 Deterministic Markovian Timed Automata

We will first exploit the product of a CTMC and a DTA, which is what we call a *deterministic Markovian timed automaton*:

Definition 7.8 (DMTA) *A* deterministic Markovian timed automaton *is a tuple* $\mathcal{M} = (\mathit{Loc}, \mathcal{X}, \ell_0, \mathit{Loc}_{\mathbf{F}}, E, \rightsquigarrow)$, *where*

- Loc *is a finite set of* locations*;*
- $\mathcal{X}$ *is a finite set of* clocks*;*
- $\ell_0 \in \mathit{Loc}$ *is the* initial location*;*
- $\mathit{Loc}_{\mathbf{F}}$ *is the* acceptance condition *with* $\mathit{Loc}_{\mathbf{F}} := \mathit{Loc}_F \subseteq \mathit{Loc}$ *the reachability condition and* $\mathit{Loc}_{\mathbf{F}} := \mathit{Loc}_{\mathcal{F}} \subseteq 2^{\mathit{Loc}}$ *the Muller condition;*
- $E : \mathit{Loc} \to \mathbb{R}_{\geqslant 0}$ *is the* exit rate function*; and*
- $\rightsquigarrow \subseteq \mathit{Loc} \times \mathcal{B}(\mathcal{X}) \times 2^{\mathcal{X}} \times \mathit{Distr}(\mathit{Loc})$ *is an* edge relation *satisfying* (ℓ, g, X, ζ), $(\ell, g', X', \zeta') \in \rightsquigarrow$ *with* $g \neq g'$ *implies* $g \cap g' = \varnothing$.

The set of clocks $\mathcal{X}$ and the related concepts, e.g., clock valuation, clock constraints are defined as for DTA. We refer to $\ell \overset{g,X}{\rightsquigarrow} \zeta$ for distribution $\zeta \in \mathit{Distr}(\mathit{Loc})$ as an *edge* and refer to $\ell \xmapsto[\zeta(\ell')]{g,X} \ell'$ as a *transition* of this edge. The intuition is that when entering location ℓ, the DMTA chooses a residence time which is governed by the exponential distribution, i.e., the probability to leave ℓ within t time units is $1 - e^{-E(\ell)t}$. When it decides to jump, at most one edge, say $\ell \overset{g,X}{\rightsquigarrow} \zeta$, due to the determinism, is enabled and the probability to jump to ℓ' is given by $\zeta(\ell')$. The DMTA is *deterministic* as it has a unique initial location and disjoint guards for all edges emanating from any location. Similar as in DTAs, DMTA$^\Diamond$ and DMTA$^\omega$ are defined in an obvious way and DMTA refers to both cases.

Definition 7.9 (Paths in DMTAs) *Given a* DMTA $\mathcal{M}$, *an* (*infinite*) symbolic path *is of the form:*

$$\ell_0 \xmapsto[p_0]{g_0,X_0} \ell_1 \xmapsto[p_1]{g_1,X_1} \ell_2 \cdots$$

where $p_i = \zeta_i(\ell_{i+1})$ *is the transition probability of* $\ell_i \xmapsto[\zeta_i(\ell_{i+1})]{g_i,X_i} \ell_{i+1}$.

An infinite path *in* $\mathcal{M}$ (*induced from the symbolic path*) *is of the form* $\tau = \ell_0 \xrightarrow{t_0} \ell_1 \xrightarrow{t_1} \ell_2 \cdots$ *and has the property that* $\eta_0 = \vec{0}$, $(\eta_i + t_i) \models g_i$, *and* $\eta_{i+1} = (\eta_i + t_i)[X_i := 0]$ *where* $i \geqslant 0$ *and* η_i *is the clock valuation of* $\mathcal{X}$ *in* $\mathcal{M}$ *on* entering *location* ℓ_i.

The path τ *is* accepted *by a* DMTA$^\Diamond$ *if there exists* $n \geqslant 0$, *such that* $\tau[n] \in Loc_F$. *It is* accepted *by* DMTA$^\omega$ *iff* $\mathit{inf}(\tau) \in Loc_{\mathcal{F}}$, *i.e.,* $\exists L_F \in Loc_{\mathcal{F}}$ *such that* $\mathit{inf}(\tau) = L_F$.

All definitions on paths in CTMC*s can be carried over to* DMTA *paths.*

DMTA Semantics. First we characterize the *one-jump* probability $\ell \xmapsto[\mathbf{P}(\ell,\ell')]{g,X} \ell'$ within time interval I starting at clock valuation η, denoted $p_\eta(\ell, \ell', I)$, as follows:

$$p_\eta(\ell,\ell',I) = \int_I \underbrace{E(\ell)\cdot e^{-E(\ell)\tau}}_{\text{(i) density to leave } \ell \text{ at } \tau} \cdot \underbrace{\mathbf{1}_g(\eta+\tau)}_{\text{(ii) } \eta+\tau\models g?} \cdot \underbrace{\mathbf{P}(\ell,\ell')}_{\text{(iii) probabilistic jump}} d\tau \quad (7.1)$$

Actually, (i) characterizes the delay τ at location ℓ which is exponentially distributed with rate $E(\ell)$; (ii) is the *characteristic function*, where $\mathbf{1}_g(\eta + \tau) = 1$, if $\eta + \tau \models g$; 0, otherwise. It compares the current valuation $\eta + \tau$ with g and rules out the paths that cannot lead to ℓ'; and (iii) indicates the probabilistic jump. Note that (i) and (iii) are features from CTMCs while (ii) is from DTA. The characteristic function is Riemann integrable as it is bounded and its support is an interval, therefore $p_\eta(\ell, \ell', I)$ is well-defined.

Based on the one-jump probability, we can now consider the probability of a set of paths. Given DMTA $\mathcal{M}$, $C(\ell_0, I_0, \ldots, I_{n-1}, \ell_n)$ is the cylinder set where $(\ell_0, \ldots, \ell_n) \in Loc^{n+1}$ and $I_i \subseteq \mathbb{R}_{\geqslant 0}$. It denotes a set of paths τ in $\mathcal{M}$ such that $\tau[i] = \ell_i$ and $\tau\langle i\rangle \in I_i$. Let $\Pr^{\mathcal{M}}_{\eta_0}(C(\ell_0, I_0, \ldots, I_{n-1}, \ell_n))$ denote the probability of $C(\ell_0, I_0, \ldots, I_{n-1}, \ell_n)$ such that the initial clock valuation in location ℓ_0 is η_0. We define $\Pr^{\mathcal{M}}_{\eta_0}(C(\ell_0, I_0, \ldots, I_{n-1}, \ell_n)) := \mathbb{P}^{\mathcal{M}}_0(\eta_0)$, where $\mathbb{P}^{\mathcal{M}}_i(\eta)$ for $0 \leqslant i \leqslant n$ is defined as:

$\mathbb{P}_n^{\mathcal{M}}(\eta) = 1$ and for $0 \leqslant i < n$, we note that there exists a transition from ℓ_i to ℓ_{i+1} with $\ell_i \xmapsto[p_i]{g_i, X_i} \ell_{i+1}$ $(0 \leqslant i < n)$ and thus we define

$$\mathbb{P}_i^{\mathcal{M}}(\eta) = \int_{I_i} \underbrace{E(\ell_i){\cdot}e^{-E(\ell_i)\tau}{\cdot}\mathbf{1}_{g_i}(\eta+\tau){\cdot}p_i}_{(\star)} \cdot \underbrace{\mathbb{P}_{i+1}^{\mathcal{M}}(\eta')}_{(\star\star)}\ d\tau,$$

where $\eta' := (\eta+\tau)[X_i := 0]$. Intuitively, $\mathbb{P}_i^{\mathcal{M}}(\eta_i)$ is the probability of the suffix cylinder set starting from ℓ_i and η_i to ℓ_n. It is recursively computed by the product of the probability of taking a transition from ℓ_i to ℓ_{i+1} within time interval I_i (cf. $(\star)$ and (7.1)) and the probability of the suffix cylinder set from ℓ_{i+1} and η_{i+1} on (cf. $(\star\star)$). For the same reason as $p_\eta(\ell, \ell', I)$ was well-defined, $\mathbb{P}_i^{\mathcal{M}}(\eta)$ is well-defined.

Example 7.10 (DMTA$^\Diamond$ and DMTA$^\omega$) *The* DMTA$^\Diamond$ *in Fig. 7.4(a) has initial location ℓ_0 with two edges, with guards $x < 1$ and $1 < x < 2$. We use the small black dots to indicate distributions. Assume t time units elapsed. If $t < 1$, then the upper edge is enabled and the probability to go to ℓ_1 within time t is $p_{\vec{0}}(\ell_0, \ell_1, t) = (1-e^{-r_0 t}){\cdot}1$, where $E(\ell_0) = r_0$; no clock is reset. It is similar for $1 < t < 2$, except that x will be reset. $Loc_F = \{q_3\}$. It is obvious to see the determinism in this automaton. The* DMTA$^\omega$ *in Fig. 7.5(c) has Muller acceptance family $Loc_{\mathcal{F}} = \big\{\{\ell_1, \ell_2, \ell_3\}, \{\ell_4, \ell_5, \ell_6\}\big\}$.* ♦

7.2.2 Product DMTAs

Given a CTMC $\mathcal{C}$ and a DTA $\mathcal{A}$, the product $\mathcal{C} \otimes \mathcal{A}$ is a DMTA defined by:

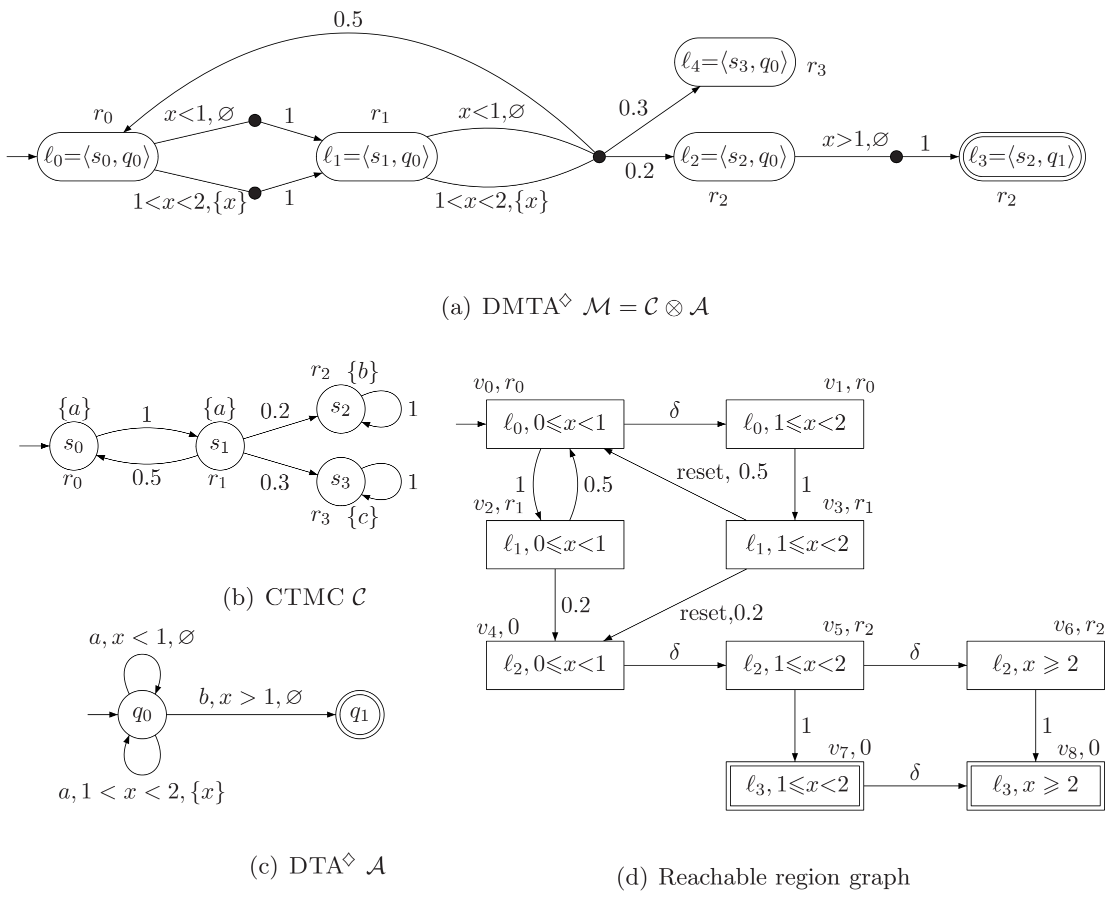
Definition 7.11 (Product of CTMC and DTA) *Let $\mathcal{C} = (S, \mathrm{AP}, L, s_0, \mathbf{P}, E)$ be a* CTMC *and $\mathcal{A} = (2^{\mathrm{AP}}, \mathcal{X}, Q, q_0, Q_{\mathbf{F}}, \rightarrow)$ be a* DTA. *We define $\mathcal{C} \otimes \mathcal{A} = (Loc, \mathcal{X}, \ell_0, Loc_{\mathbf{F}}, E, \rightsquigarrow)$ as the product* DMTA, *where*

- $Loc := S \times Q$; $\quad \ell_0 := \langle s_0, q_0\rangle$; $\quad E(\langle s, q\rangle) := E(s)$;
- $Loc_{\mathbf{F}} = Loc_F := S \times Q_F$, *if* $Q_{\mathbf{F}} = Q_F$; *(reachability condition)*

  $Loc_{\mathbf{F}} = Loc_{\mathcal{F}} := \bigcup_{F \in Q_{\mathcal{F}}} S \times F$, *if* $Q_{\mathbf{F}} = Q_{\mathcal{F}}$; *(Muller condition)*
- $\rightsquigarrow$ *is defined as the smallest relation defined by the rule:*

$$\frac{\mathbf{P}(s, s') > 0 \ \wedge\ q \xrightarrow{L(s), g, X} q'}{\langle s, q\rangle \overset{g,X}{\rightsquigarrow} \zeta},\ \text{such that } \zeta(\langle s', q'\rangle) = \mathbf{P}(s, s').$$

Example 7.12 (Product DMTA$^\Diamond$) *Let* CTMC $\mathcal{C}$ *and* DTA$^\Diamond$ $\mathcal{A}$ *be in Fig. 7.4(b) and 7.4(c), the product* DMTA$^\Diamond$ $\mathcal{C} \otimes \mathcal{A}$ *is as in Fig. 7.4(a). Since $Q_F = \{q_1\}$ in $\mathcal{A}$, the set of accepting locations in* DMTA$^\Diamond$ *is $Loc_F = \{\langle s_2, q_1\rangle\} = \{\ell_3\}$.* ♦



(a) DMTA$^\diamondsuit$ $\mathcal{M} = \mathcal{C} \otimes \mathcal{A}$

(b) CTMC $\mathcal{C}$

(c) DTA$^\diamondsuit$ $\mathcal{A}$

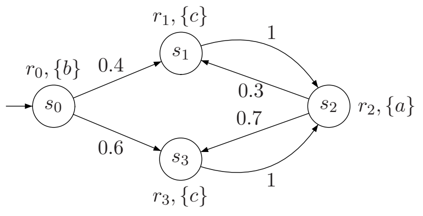
(d) Reachable region graph

Figure 7.4: Example product DMTA$^\diamondsuit$ of CTMC $\mathcal{C}$ and DTA$^\diamondsuit$ $\mathcal{A}$

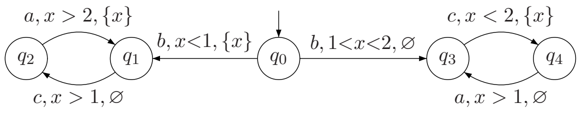
Example 7.13 (Product DMTA$^\omega$) *For the* CTMC $\mathcal{C}$ *in Fig. 7.5(a) and the* DTA$^\omega$ $\mathcal{A}^\omega$ *in Fig. 7.5(b) with acceptance family* $Q_\mathcal{F} = \big\{\{q_1, q_2\}, \{q_3, q_4\}\big\}$, *the product* DMTA$^\omega$ $\mathcal{C}{\otimes}\mathcal{A}^\omega$ *is shown in Fig. 7.5(c).* $Loc_\mathcal{F} = \big\{\{\langle s_i, q_1\rangle, \langle s_j, q_2\rangle\}, \{\langle s'_i, q_3\rangle, \langle s'_j, q_4\rangle\}\big\}$, *for any* $s_i, s'_i, s_j, s'_j \in S$, *in particular,* $Loc_\mathcal{F} = \big\{\{\ell_1, \ell_2, \ell_3\}, \{\ell_4, \ell_5, \ell_6\}\big\}$. ♦

Remark 7.14 *It is easy to see from the construction that* $\mathcal{C} \otimes \mathcal{A}$ *is indeed a* DMTA. *The determinism of the* DTA $\mathcal{A}$ *guarantees that the induced product is also deterministic. In* $\mathcal{C} \otimes \mathcal{A}$, *from each location there is at most one "action" possible, viz.* $L(s)$. *We can thus omit actions from the product* DMTA.

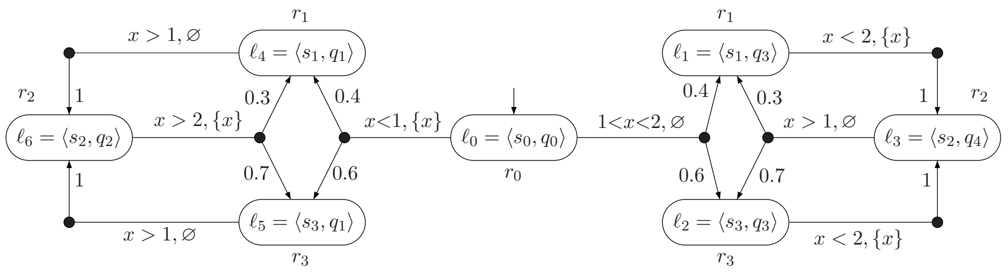
For DTA$^\diamondsuit$ $\mathcal{A}$ with the set of accepting locations Loc_F, we denote $Paths^{\mathcal{C}\otimes\mathcal{A}}(\diamondsuit\, Loc_F) := \{\, \tau \in Paths^{\mathcal{C}\otimes\mathcal{A}} \mid \tau \text{ is accepted by } \mathcal{C}{\otimes}\mathcal{A} \,\}$ as the set of accepted paths in $\mathcal{C}{\otimes}\mathcal{A}$. Recall that $Paths^{\mathcal{C}}(\mathcal{A})$ is the set of paths in CTMC $\mathcal{C}$ that are accepted



(a) CTMC $\mathcal{C}$



(b) DTA$^\omega$ $\mathcal{A}^\omega$


$x > 1, \varnothing$
r_1
$\ell_4 = \langle s_1, q_1 \rangle$
r_2
1
0.3
0.4
$x > 2, \{x\}$
$x<1, \{x\}$
$\ell_6 = \langle s_2, q_2 \rangle$
$\ell_0 = \langle s_0, q_0 \rangle$
r_0
0.7
0.6
1
$\ell_5 = \langle s_3, q_1 \rangle$
$x > 1, \varnothing$
r_3
r_1
$\ell_1 = \langle s_1, q_3 \rangle$
$x < 2, \{x\}$
0.4
0.3
1
r_2
$1<x<2, \varnothing$
$x > 1, \varnothing$
$\ell_3 = \langle s_2, q_4 \rangle$
0.6
0.7
1
$\ell_2 = \langle s_3, q_3 \rangle$
$x < 2, \{x\}$
r_3

(c) DMTA$^\omega$ $\mathcal{C} \otimes \mathcal{A}^\omega$

Figure 7.5: Example product DMTA$^\omega$ of CTMC $\mathcal{C}$ and DTA$^\omega$ $\mathcal{A}^\omega$

by DTA $\mathcal{A}$. For any n-ary tuple J, let $J{\downharpoonleft_i}$ denote the i-th entry in J, for $1 \leqslant i \leqslant n$. For a $(\mathcal{C}\otimes\mathcal{A})$-path $\tau = \langle s_0, q_0 \rangle \xrightarrow{t_0} \langle s_1, q_1 \rangle \xrightarrow{t_1} \cdots$, let $\tau{\downharpoonleft_1} := s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \cdots$, and for any set Π of $(\mathcal{C}\otimes\mathcal{A})$-paths, let $\Pi{\downharpoonleft_1} = \bigcup_{\tau\in\Pi} \tau{\downharpoonleft_1}$.

Lemma 7.15 *For any* CTMC $\mathcal{C}$ *and* DTA$^\diamondsuit$ $\mathcal{A}$, $\mathit{Paths}^{\mathcal{C}}(\mathcal{A}) = \mathit{Paths}^{\mathcal{C}\otimes\mathcal{A}}(\diamondsuit\, \mathit{Loc}_F){\downharpoonleft_1}$.

Proof: The proof can be found in Appendix A.2, page 165. ■

The following theorem establishes the link between CTMC $\mathcal{C}$ and DMTA$^\diamondsuit$ $\mathcal{C} \otimes \mathcal{A}$.

Theorem 7.16 *For any* CTMC $\mathcal{C}$ *and* DTA$^\diamondsuit$ $\mathcal{A}$,

$$\Pr^{\mathcal{C}}\left(\mathit{Paths}^{\mathcal{C}}(\mathcal{A})\right) = \Pr^{\mathcal{C}\otimes\mathcal{A}}_{\vec{0}}\left(\mathit{Paths}^{\mathcal{C}\otimes\mathcal{A}}(\diamondsuit \mathit{Loc}_F)\right).$$

Proof: The proof can be found in Appendix A.3, page 165. ■

7.2.3 Region Construction for DMTA

In the remainder of this section, we focus on how to compute the probability measure $\Pr^{\mathcal{C}\otimes\mathcal{A}}_{\vec{0}}\left(\mathit{Paths}^{\mathcal{C}\otimes\mathcal{A}}(\diamondsuit\, \mathit{Loc}_F)\right)$ in an effective way. Since the state space $\big\{(\ell, \eta) \mid \ell \in$

$Loc, \eta \in \mathcal{V}(\mathcal{X})\}$ of $\mathcal{C} \otimes \mathcal{A}$ is uncountable, we start with adopting the standard *region construction* [AD94] to DMTA$^\diamondsuit$ to discretize the state space into a finite one. As we will see in Section 7.2.5, this allows us to obtain a piecewise-deterministic Markov process from a DMTA$^\diamondsuit$ in a natural way.

As usual, a region is a constraint. For regions $\Theta, \Theta' \in \mathcal{B}(\mathcal{X})$, Θ' is the *successor region* of Θ if for all $\eta \models \Theta$ there exists $\delta \in \mathbb{R}_{>0}$ such that $\eta + \delta \models \Theta'$ and for all $\delta' < \delta$, $\eta + \delta' \models \Theta \vee \Theta'$. A region Θ *satisfies* a guard g (denoted $\Theta \models g$) iff $\forall \eta \models \Theta.\ \eta \models g$. A *reset operation* on region Θ is defined as $\Theta[X := 0] := \{\eta[X := 0] \mid \eta \models \Theta\}$.

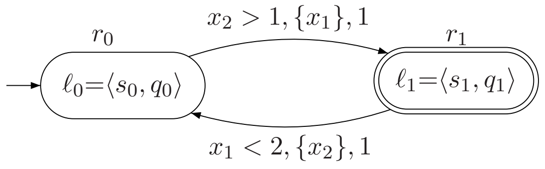
Definition 7.17 (Region graph of DMTA$^\diamondsuit$) *Given DMTA$^\diamondsuit$ $\mathcal{M} = (Loc, \mathcal{X}, \ell_0, Loc_F, E, \rightsquigarrow)$, the region graph is $\mathcal{G}(\mathcal{M}) = (V, v_0, V_F, \Lambda, \hookrightarrow)$, where*

- *$V := Loc \times \mathcal{B}(\mathcal{X})$ is a finite set of* vertices*, consisting of a location ℓ in $\mathcal{M}$ and a region Θ;*
- *$v_0 \in V$ is the* initial vertex *if $(\ell_0, \vec{0}) \in v_0$;*
- *$V_F := \{v \mid v{\downharpoonright_1} \in Loc_F\}$ is the set of* accepting vertices*;*
- *$\hookrightarrow \subseteq V \times \big(([0,1] \times 2^{\mathcal{X}}) \cup \{\delta\}\big) \times V$ is the* transition (edge) relation*, such that:*
  - ▶ *$v \overset{\delta}{\hookrightarrow} v'$ is a* delay transition *if $v{\downharpoonright_1} = v'{\downharpoonright_1}$ and $v'{\downharpoonright_2}$ is a successor region of $v{\downharpoonright_2}$;*
  - ▶ *$v \overset{p,X}{\hookrightarrow} v'$ is a* Markovian transition *if there exists some transition $v{\downharpoonright_1} \xmapsto[p]{g,X} v'{\downharpoonright_1}$ in $\mathcal{M}$ such that $v{\downharpoonright_2} \models g$ and $v{\downharpoonright_2}[X := 0] \models v'{\downharpoonright_2}$; and*
- *$\Lambda : V \to \mathbb{R}_{\geqslant 0}$ is the* exit rate function *where $\Lambda(v) := E(v{\downharpoonright_1})$ if there exists a Markovian transition from v, $\Lambda(v) := 0$ otherwise.*

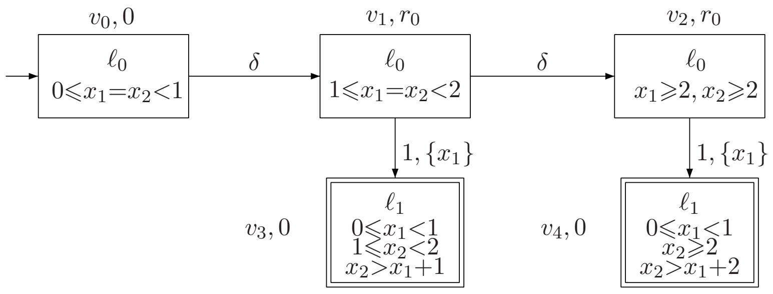
Note that in the obtained region graph, Markovian transitions emanating from any boundary region do *not* contribute to the reachability probability as the time to hit the boundary is always zero (i.e., $\flat(v, \eta) = 0$ in (7.5), page 147). Therefore, we can remove all the Markovian transitions emanating from boundary regions and then collapse each of them with its unique *non-boundary* (direct) successor. In the sequel, by slightly abusing the notation we still denote this *collapsed* region graph as $\mathcal{G}(\mathcal{M})$.

Remark 7.18 (Exit rates) *The exit rate $\Lambda(v)$ is set to 0 if there is only delay transition from v. The probability to take the delay transition within time t is $e^{-\Lambda(v)t} = 1$ and the probability to take Markovian transitions is 0.*

Example 7.19 *For the DMTA$^\diamondsuit$ $\mathcal{C} \otimes \mathcal{A}$ in Fig. 7.6(a), the reachable part (forward reachable from the initial vertex and backward reachable from the accepting vertices) of the collapsed region graph $\mathcal{G}(\mathcal{C} \otimes \mathcal{A})$ is shown in Fig. 7.6(b). The accepting vertices are sinks.* ♦


r_0
r_1
$\ell_0{=}\langle s_0, q_0\rangle$
$\ell_1{=}\langle s_1, q_1\rangle$
$x_2 > 1, \{x_1\}, 1$
$x_1 < 2, \{x_2\}, 1$

(a) DMTA$^\Diamond$ $\mathcal{M} = \mathcal{C} \otimes \mathcal{A}$


$v_0, 0$
ℓ_0 $0{\leqslant}x_1{=}x_2{<}1$
δ
v_1, r_0
ℓ_0 $1{\leqslant}x_1{=}x_2{<}2$
δ
v_2, r_0
ℓ_0 $x_1{\geqslant}2, x_2{\geqslant}2$
$1, \{x_1\}$
$1, \{x_1\}$
$v_3, 0$
ℓ_1 $0{\leqslant}x_1{<}1$ $1{\leqslant}x_2{<}2$ $x_2{>}x_1{+}1$
$v_4, 0$
ℓ_1 $0{\leqslant}x_1{<}1$ $x_2{\geqslant}2$ $x_2{>}x_1{+}2$

(b) Reachable region graph $\mathcal{G}(\mathcal{C} \otimes \mathcal{A})$

Figure 7.6: Example of a region graph

Notice that DMTA$^\Diamond$ and DMTA$^\omega$ have the same locations and edge relations. The only difference is their acceptance condition. This guarantees that their obtained region graphs are the same except for the definition and interpretation of the final set V_F. We will present how V_F is derived in the region graph for DMTA$^\omega$ in Section 7.4.

7.2.4 Piecewise-Deterministic Markov Processes

The model PDP was introduced by Davis in 1984 [Dav84]. We abbreviate it as PDP instead of literally PDMP, following the convention by Davis [Dav93]. A PDP constitutes a general framework that can model virtually any stochastic system without diffusions [Dav93] and for which powerful analysis and control techniques exist [LL85][LY91][CD88]. A PDP is a stochastic process of hybrid type, i.e., the stochastic process concerns both a discrete location and a continuous variable. The class of PDPs was recognized as a very wide class holding many types of stochastic hybrid system. This makes PDP a useful model for an enormous variety of applied problems in engineering, operations research, management science and economics; examples include queueing systems, stochastic scheduling, fault detection in systems engineering, etc.

Given a set H, let $\Pr : \mathcal{F}(H) \to [0, 1]$ be a probability measure on the measurable space $(H, \mathcal{F}(H))$, where $\mathcal{F}(H)$ is a σ-algebra over H. Let *Distr*(H) denote the set of probability measures on this measurable space.

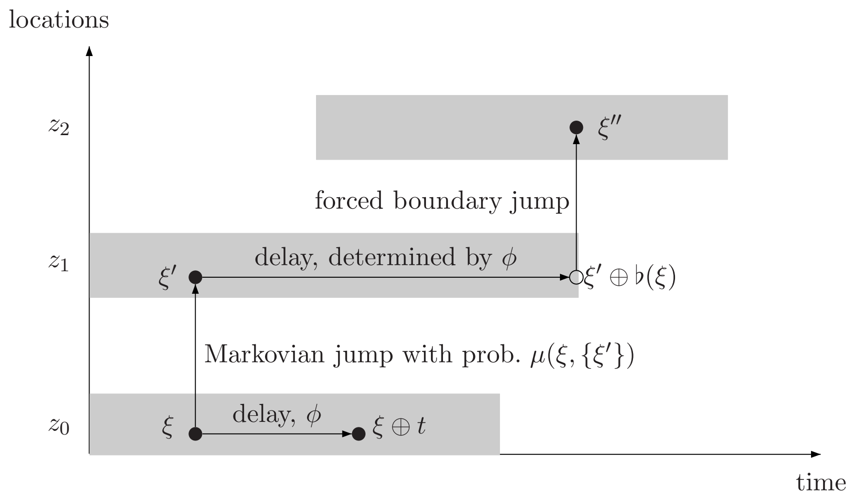


Figure 7.7: The behavior of a PDP

Definition 7.20 (PDP [Dav93]) *A* piecewise-deterministic (Markov) process *is a tuple $\mathcal{Z} = (Z, \mathcal{X}, Inv, \phi, \Lambda, \mu)$ with:*

- *Z is a finite set of* locations*;*
- *$\mathcal{X}$ is a finite set of* variables*;*
- *$Inv : Z \to \mathcal{B}(\mathcal{X})$ is an* invariant function*;*
- *$\phi : Z \times \mathcal{V}(\mathcal{X}) \times \mathbb{R} \to \mathcal{V}(\mathcal{X})$ is a* flow function[1]*;*
- *$\Lambda : \mathbb{S} \to \mathbb{R}_{\geqslant 0}$ is an* exit rate function*;*
- *$\mu : \mathring{\mathbb{S}} \cup \partial\mathbb{S} \to Distr(\mathbb{S})$ is the* transition probability function, *where:*

$\mathbb{S} := \{\xi := (z, \eta) \mid z \in Z, \eta \models Inv(z)\}$ is the state space of the PDP *$\mathcal{Z}$, $\mathring{\mathbb{S}}$ is the interior of $\mathbb{S}$ and $\partial\mathbb{S} = \bigcup_{z \in Z}\{z\} \times \partial Inv(z)$ is the boundary of $\mathbb{S}$ with $\partial Inv(z) = \overline{Inv(z)} \setminus \mathring{Inv}(z)$ as the boundary of $Inv(z)$, $\mathring{Inv}(z)$ the interior of $Inv(z)$ and $\overline{Inv(z)}$ the closure of $Inv(z)$. Functions Λ and μ satisfy the following conditions:*

- *$\forall \xi \in \mathbb{S}.\ \exists \epsilon(\xi) > 0.$ function $t \mapsto \Lambda(\xi \oplus t)$ is integrable on $[0, \epsilon(\xi)[$, where $\xi \oplus t = \big(z, \phi(z, \eta, t)\big)$, for $\xi = (z, \eta)$;*
- *Function $\xi \mapsto \mu(\xi, A)$[2] is measurable for any $A \in \mathcal{F}(\mathbb{S})$, where $\mathcal{F}(\mathbb{S})$ is a σ-algebra generated by the countable union $\bigcup_{z \in Z}\{z\} \times A_z$ with A_z being a subset of $\mathcal{F}(Inv(z))$ and $\mu(\xi, \{\xi\}) = 0$.*

We will explain the behavior of a PDP by the aid of Fig. 7.7. A PDP consists of a finite set of *locations* each with a *location invariant* over a set of *variables*. A *state* consists of a location and a valuation of the variables. A PDP is only allowed to stay

[1]The flow function is the solution of a system of ODEs with a Lipschitz continuous vector field.
[2]$\mu(\xi, A)$ is a shorthand for $(\mu(\xi))(A)$.

in location z when the constraint $Inv(z)$ is satisfied. If e.g., $Inv(z)$ is $x_1^2 - 2x_2 \leqslant 1.5 \wedge x_3 > 2$, then its interior $\overset{\circ}{Inv}(z)$ is $x_1^2 - 2x_2 < 1.5 \wedge x_3 > 2$ and its closure $\overline{Inv(z)}$ is $x_1^2 - 2x_2 \leqslant 1.5 \wedge x_3 \geqslant 2$, and the boundary $\partial Inv(z)$ is $x_1^2 - 2x_2 = 1.5 \wedge x_3 = 2$. In Fig. 7.7, there are three locations z_0, z_1, z_2 and the gray zones are the valid valuations for respective locations. A state is a black dot. A boundary state is a white dot. When a new state $\xi = (z, \eta)$ is entered and $Inv(z)$ is valid, i.e., $\xi \in \mathbb{S}$, the PDP can (i) either *delay* to state $\xi' = (z, \eta') \in \mathbb{S} \cup \partial\mathbb{S}$ according to both the flow function ϕ and the time delay t (in this case $\xi' = \xi \oplus t$); (ii) or take a *Markovian jump* to state $\xi'' = (z'', \eta'') \in \mathbb{S}$ with probability $\mu(\xi, \{\xi''\})$. Note that the residence time of a location is exponentially distributed. When the variable valuation satisfies the boundary (i.e., $\xi \in \partial\mathbb{S}$), the PDP is *forced to take a boundary jump* and leave the current location z with probability $\mu(\xi, \{\xi''\})$ to state ξ''.

The flow function ϕ defines the time-dependent behavior in a single location, in particular, how the variable valuations change when time elapses. State $\xi \oplus t$ is the timed successor of state ξ (on the same location) given that t time units have passed. The PDP is piecewise-deterministic because in each location (one piece) the behavior is deterministically determined by ϕ. The process is *Markovian* as the current state contains all the information to predict the future progress of the process.

The embedded *discrete-time Markov process* (DTMP) $emb(\mathcal{Z})$ of the PDP $\mathcal{Z}$ has the same state space $\mathbb{S}$ as $\mathcal{Z}$. The (one-jump) *transition probability* from a state ξ to a set $A \subseteq \mathbb{S}$ of states (on different locations as ξ), denoted $\hat{\mu}(\xi, A)$, is given by [Dav93]:

$$\hat{\mu}(\xi, A) = \int_0^{\flat(\xi)} (\mathcal{Q}\mathbf{1}_A)(\xi \oplus t) \cdot \Lambda(\xi \oplus t)\, e^{-\int_0^t \Lambda(\xi \oplus \tau) d\tau}\, dt \tag{7.2}$$

$$+ \; (\mathcal{Q}\mathbf{1}_A)(\xi \oplus \flat(\xi)) \cdot e^{-\int_0^{\flat(\xi)} \Lambda(\xi \oplus \tau) d\tau}, \tag{7.3}$$

where $\flat(\xi) = \inf\{t > 0 \mid \xi \oplus t \in \partial\mathbb{S}\}$ is the minimal time to hit the boundary if such time exists; $\flat(\xi) = \infty$ otherwise. $(\mathcal{Q}\mathbf{1}_A)(\xi) = \int_{\mathbb{S}} \mathbf{1}_A(\xi')\mu(\xi, d\xi')$ is the accumulative (one-jump) transition probability from ξ to A and $\mathbf{1}_A(\xi)$ is the characteristic function such that $\mathbf{1}_A(\xi) = 1$ when $\xi \in A$ and $\mathbf{1}_A(\xi) = 0$ otherwise. Term (7.2) specifies the probability to delay to state $\xi \oplus t$ (on the same location) and take a Markovian jump from $\xi \oplus t$ to A. Note the delay t can take a value from $[0, \flat(\xi))$. Term (7.3) is the probability to stay in the same location for $\flat(\xi)$ time units and then it is forced to take a boundary jump from $\xi \oplus \flat(\xi)$ to A since $Inv(z)$ is invalid.

Example 7.21 *Fig. 7.8 depicts a* 3*-location* PDP $\mathcal{Z}$ *with one variable* x*, where* $Inv(z_0)$ *is* $x < 2$ *and* $Inv(z_1)$*,* $Inv(z_2)$ *are both* $x \in [0, \infty)$*. Solving* $\dot{x} = 1$ *gives the flow function* $\phi(z_i, \eta(x), t) = \eta(x) + t$ *for* $i = 0, 1, 2$*. The state space of* $\mathcal{Z}$ *is* $\{(z_0, \eta) \mid 0 < \eta(x) < 2\} \cup \{(z_1, \mathbb{R})\} \cup \{(z_2, \mathbb{R})\}$*. Let exit rate* $\Lambda(\xi) = 5$ *for any* $\xi \in \mathbb{S}$*. For* $\eta \models Inv(z_0)$*, let* $\mu\big((z_0, \eta), \{(z_1, \eta)\}\big) := \frac{1}{3}$*,* $\mu\big((z_0, \eta), \{(z_2, \eta)\}\big) := \frac{2}{3}$ *and the boundary*

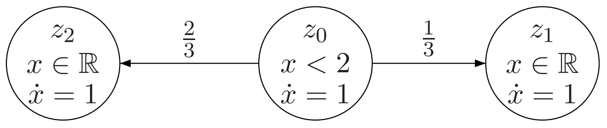


Figure 7.8: An example PDP $\mathcal{Z}$

measure $\mu\big((z_0, 2), \{(z_1, 2)\}\big) := 1$. *Given state* $\xi_0 = (z_0, 0)$ *and the set of states* $A = (z_1, \mathbb{R})$, *the time for* ξ_0 *to hit the boundary is* $\flat(\xi_0) = 2$. *Then* $(\mathcal{Q}\mathbf{1}_A)(\xi_0 \oplus t) = \frac{1}{3}$ *if* $t < 2$, *and* $(\mathcal{Q}\mathbf{1}_A)(\xi_0 \oplus t) = 1$ *if* $t = 2$. *In* emb$(\mathcal{Z})$, *the transition probability from state* ξ_0 *to* A *is:*

$$\hat{\mu}(\xi_0, A) = \int_0^2 \frac{1}{3} \cdot 5 \cdot e^{-\int_0^t 5 \ d\tau} \ dt + 1 \cdot e^{-\int_0^2 5 \ d\tau} = \frac{1}{3} + \frac{2}{3} e^{-10}. \qquad \blacklozenge$$

7.2.5 From Region Graph to PDP

We can now define the underlying PDP of a DMTA$^\diamondsuit$ by using the region graph $\mathcal{G}(\mathcal{M})$. Actually, a region graph is a PDP.

Definition 7.22 (PDP for DMTA$^\diamondsuit$) *For* DMTA$^\diamondsuit$ $\mathcal{M} = (Loc, \mathcal{X}, \ell_0, Loc_F, E, \rightsquigarrow)$ *and region graph* $\mathcal{G}(\mathcal{M}) = (V, v_0, V_F, \Lambda, \hookrightarrow)$, *let* PDP $\mathcal{Z}(\mathcal{M}) = (V, \mathcal{X}, Inv, \phi, \Lambda, \mu)$ *where for any* $v \in V$,

- $Inv(v) := v\lfloor_2$ *and the state space* $\mathbb{S} := \big\{(v, \eta) \mid v \in V, \eta \in Inv(v)\big\}$;
- $\phi(v, \eta, t) := \eta + t$ *for* $\eta \models Inv(v)$;
- $\Lambda(v, \eta) := \Lambda(v)$ *is the exit rate of state* (v, η);
- [boundary jump] *for each delay transition* $v \overset{\delta}{\hookrightarrow} v'$ *in* $\mathcal{G}(\mathcal{M})$ *we have* $\mu(\xi, \{\xi'\}) := 1$, *where* $\xi = (v, \eta)$, $\xi' = (v', \eta)$ *and* $\eta \models \partial Inv(v)$;
- [Markovian jump] *for each Markovian transition* $v \overset{p,X}{\hookrightarrow} v'$ *in* $\mathcal{G}(\mathcal{M})$ *we have* $\mu(\xi, \{\xi'\}) := p$, *where* $\xi = (v, \eta)$, $\eta \models Inv(v)$ *and* $\xi' = (v', \eta[X := 0])$.

From now on we write $\Lambda(v)$ instead of $\Lambda(v, \eta)$ as they coincide.

7.3 Model Checking DTA$^\diamondsuit$ Specifications

With the model and problem transformation presented in the last section, we are now ready to model check CTMC against DTA$^\diamondsuit$ specifications. We first consider the general case, i.e., DTA$^\diamondsuit$ with arbitrary number of clocks and then the special case of single clock DTA$^\diamondsuit$ specifications is investigated.

7.3.1 General DTA$^\Diamond$ Specifications

Recall that the aim of model checking is to compute the probability of the set of paths in CTMC $\mathcal{C}$ accepted by a DTA$^\Diamond$ $\mathcal{A}$. For the general case, we have proven that this is reducible to computing the reachability probability in the product $\mathcal{C} \otimes \mathcal{A}$ (Theorem 7.16, page 141), which can be further reduced to computing the reachability probability in a corresponding PDP (Theorem 7.23 below), which will be established in Section 7.3.1.1. The characterization by a system of integral equations is usually difficult to solve. Therefore we propose an approach to approximate the reachability probabilities in Section 7.3.1.2.

7.3.1.1 Characterizing Reachability Probabilities

Computing $\Pr_{\vec{0}}^{\mathcal{C}\otimes\mathcal{A}}\big(Paths^{\mathcal{C}\otimes\mathcal{A}}(\Diamond\, Loc_F)\big)$ is now reduced to computing the (time-unbounded) reachability probability in the PDP $\mathcal{Z}(\mathcal{C} \otimes \mathcal{A})$ — basically the region graph of $\mathcal{C} \otimes \mathcal{A}$ — given the initial state $(v_0, \vec{0})$ and the set of goal states $\{(v,\eta) \mid v \in V_F, \eta \in Inv(v)\}$ $\big((V_F,\cdot)$ for short$\big)$. Reachability probabilities of untimed events in a PDP $\mathcal{Z}$ can be computed in the embedded DTMP $emb(\mathcal{Z})$. Note that the set of locations of $\mathcal{Z}$ and $emb(\mathcal{Z})$ are equal. In the sequel, let $\mathcal{D}$ denote $emb(\mathcal{Z})$.

For each vertex $v \in V$, we define recursively $Prob^{\mathcal{D}}\big((v,\eta),(V_F,\cdot)\big)$ $\big($or shortly $Prob_v^{\mathcal{D}}(\eta)\big)$ as the probability to reach the goal states $(V_F,\cdot)$ in $\mathcal{D}$ from state (v,η).

- for the delay transition $v \overset{\delta}{\hookrightarrow} v'$,

$$Prob_{v,\delta}^{\mathcal{D}}(\eta) = e^{-\Lambda(v)\flat(v,\eta)} \cdot Prob_{v'}^{\mathcal{D}}\big(\eta + \flat(v,\eta)\big). \tag{7.4}$$

  Recall that $\flat(v,\eta)$ is the minimal time for (v,η) to hit the boundary $\partial Inv(v)$.

- for the Markovian transition $v \overset{p,X}{\hookrightarrow} v'$,

$$Prob_{v,v'}^{\mathcal{D}}(\eta) = \int_0^{\flat(v,\eta)} p\cdot\Lambda(v)\cdot e^{-\Lambda(v)\tau}\cdot Prob_{v'}^{\mathcal{D}}\big((\eta+\tau)[X:=0]\big)\, d\tau. \tag{7.5}$$

Overall, for each vertex $v \in V$, we obtain:

$$Prob_v^{\mathcal{D}}(\eta) = \begin{cases} Prob_{v,\delta}^{\mathcal{D}}(\eta) + \sum_{v \overset{p,X}{\hookrightarrow} v'} Prob_{v,v'}^{\mathcal{D}}(\eta), & \text{if } v \notin V_F \\ 1, & \text{otherwise} \end{cases}. \tag{7.6}$$

Note that here the notation η is slightly abused. It represents a vector of clock variables (see Example 7.25). Eq. (7.4) and (7.5) are derived based on (7.3) and (7.2), respectively. In particular, the multi-step reachability probability is computed using a sequence of one-step transition probabilities.

Hence we obtain a system of *integral equations* (7.6). One can read (7.6) either in the form $f(\xi) = \int_{Dom(\xi)} K(\xi, \xi') f(d\xi')$, where K is the kernel and $Dom(\xi)$ is the domain of integration depending on the continuous state space $\mathbb{S}$; or in the operator form $f(\xi) = (\mathcal{J} f)(\xi)$, where $\mathcal{J}$ is the integration operator. Generally, (7.6) does *not* necessarily have a unique solution. It turns out that the reachability probability $Prob^{\mathcal{D}}_{v_0}(\vec{0})$ coincides with the least fixpoint of the operator $\mathcal{J}$ (denoted by lfp$\mathcal{J}$) i.e., $Prob^{\mathcal{D}}_{v_0}(\vec{0}) = (\mathrm{lfp}\mathcal{J})(v_0, \vec{0})$. Formally, we have:

Theorem 7.23 *For any* CTMC $\mathcal{C}$ *and* DTA$^{\diamond}$ $\mathcal{A}$, $\Pr^{\mathcal{C}\otimes\mathcal{A}}_{\vec{0}}\big(Paths^{\mathcal{C}\otimes\mathcal{A}}(\diamond Loc_F)\big)$ *is the least solution of* $Prob^{\mathcal{D}}_{v_0}(\cdot)$*, where* $\mathcal{D}$ *is the embedded* DTMP *of* $\mathcal{C} \otimes \mathcal{A}$.

Proof: The proof can be found in Appendix A.4, page 167. ■

Remark 7.24 *Clock valuations* η *and* η' *in region* Θ *may induce different reachability probabilities. The reason is that* η *and* η' *may have different periods of time to hit the boundary, thus the probability for* η *and* η' *to either delay or take a Markovian transition may differ. This is in contrast with the traditional timed automata theory as well as probabilistic timed automata* [KNSS02]*, where* η *and* η' *are not distinguished.*

Example 7.25 *For the region graph in Fig. 7.6(b), the system of integral equations for* v_1 *in location* ℓ_0 *is as follows for* $1 \leqslant x_1 = x_2 < 2$*:*

$$Prob^{\mathcal{D}}_{v_1}(x_1, x_2) = Prob^{\mathcal{D}}_{v_1,\delta}(x_1, x_2) + Prob^{\mathcal{D}}_{v_1,v_3}(x_1, x_2),$$

where

$$Prob^{\mathcal{D}}_{v_1,\delta}(x_1, x_2) = e^{-(2-x_1)r_0} \cdot Prob^{\mathcal{D}}_{v_2}(2, 2)$$

and

$$Prob^{\mathcal{D}}_{v_1,v_3}(x_1, x_2) = \int_0^{2-x_1} r_0 \cdot e^{-r_0 \tau} \cdot Prob^{\mathcal{D}}_{v_3}(0, x_2 + \tau)\, d\tau$$

where $Prob^{\mathcal{D}}_{v_3}(0, x_2 + \tau) = 1$*. The integral equations for* v_2 *can be derived similarly.* ♦

7.3.1.2 Approximating Reachability Probabilities

Finally, we discuss how to obtain a solution of (7.6). The integral equations (7.6) are *Volterra equations of the second type* [AW95]. For a general reference on solutions to Volterra equations, cf., e.g. [Cor91]. As an alternative option to solve (7.6), we proceed to give a general formulation of $\Pr^{\mathcal{C}}\big(Paths^{\mathcal{C}}(\mathcal{A})\big)$ using a system of *partial differential equations* (PDEs). Let the *augmented* DTA$^{\diamond}$ $\mathcal{A}[t_f]$ be obtained from $\mathcal{A}$ by adding a new clock variable y which is never reset and a clock constraint $y < t_f$ on all edges entering the accepting locations in Loc_F, where t_f is a finite (and usually very large) integer. The purpose of this augmentation is to ensure that the value of all clocks reaching Loc_F is bounded. It is clear that $Paths^{\mathcal{C}}(\mathcal{A}[t_f]) \subseteq Paths^{\mathcal{C}}(\mathcal{A})$. More precisely, $Paths^{\mathcal{C}}(\mathcal{A}[t_f])$ coincides with those paths which can reach the accepting states of $\mathcal{A}$ within the time bound t_f. Note that $\lim_{t_f \to \infty} \Pr^{\mathcal{C}}(Paths^{\mathcal{C}}(\mathcal{A}[t_f])) = \Pr^{\mathcal{C}}(Paths^{\mathcal{C}}(\mathcal{A}))$. We can approximate $\Pr^{\mathcal{C}}(Paths^{\mathcal{C}}(\mathcal{A}))$ by solving the PDEs with a large t_f as follows:

Proposition 7.26 *Given a* CTMC $\mathcal{C}$, *an augmented* DTA$^\diamondsuit$ $\mathcal{A}[t_f]$ *and the underlying* PDP $\mathcal{Z}(\mathcal{C} \otimes \mathcal{A}[t_f]) = (V, \mathcal{X}, Inv, \phi, \Lambda, \mu)$, $\Pr^{\mathcal{C}}\left(Paths^{\mathcal{C}}(\mathcal{A}[t_f])\right) = \hbar_{v_0}(0, \vec{0})$ $\big($*which is the probability to reach the final states in* $\mathcal{Z}$ *starting from initial state* $(v_0, \vec{0}_{\mathcal{X}\cup\{y\}}$[1]$)\big)$ *is the unique solution of the following system of* PDE*s:*

$$\frac{\partial \hbar_v(y,\eta)}{\partial y} + \sum_{i=1}^{|\mathcal{X}|} \frac{\partial \hbar_v(y,\eta)}{\partial \eta^{(i)}} + \Lambda(v)\cdot \sum_{v \overset{p,X}{\hookrightarrow} v'} p\cdot(\hbar_{v'}(y, \eta[X := 0]) - \hbar_v(y,\eta)) = 0,$$

where $v \in V \setminus V_F$, $\eta \models Inv(v)$, $\eta^{(i)}$ *is the i'th clock variable and* $y \in [0, t_f)$. *For every* $\eta \models \partial Inv(v)$ *and transition* $v \overset{\delta}{\hookrightarrow} v'$, *the boundary conditions take the form:* $\hbar_v(y,\eta) = \hbar_{v'}(y,\eta)$. *For every vertex* $v \in V_F$, $\eta \models Inv(v)$ *and* $y \in [0, t_f)$, *we have the following* PDE*:*

$$\frac{\partial \hbar_v(y,\eta)}{\partial y} + \sum_{i=1}^{|\mathcal{X}|} \frac{\partial \hbar_v(y,\eta)}{\partial \eta^{(i)}} + 1 = 0.$$

The final boundary conditions are that for every vertex $v \in V$ *and* $\eta \models Inv(v) \cup \partial Inv(v)$, $\hbar_v(t_f, \eta) = 0$.

Proof: The proof can be found in Appendix A.5, page 169. ■

7.3.2 Single-Clock DTA$^\diamondsuit$ Specifications

For single-clock DTA$^\diamondsuit$ specifications, we can simplify the system of integral equations obtained in the previous section to a system of *linear* equations where the coefficients are a solution of a system of ODEs that can be calculated efficiently.

Given a DMTA$^\diamondsuit$ $\mathcal{M}$, we denote the set of constants appearing in the clock constraints of $\mathcal{M}$ as $\{c_0, \ldots, c_m\}$ with $c_0 = 0$. We assume the following order: $0 = c_0 < c_1 < \cdots < c_m$. Let $\Delta c_i = c_{i+1} - c_i$ for $0 \leqslant i < m$. Note that for one clock DMTA$^\diamondsuit$, the regions in the region graph $\mathcal{G}(\mathcal{M})$ can be represented by the following intervals: $[c_0, c_1), \ldots, [c_m, \infty)$. We partition the region graph $\mathcal{G}(\mathcal{M}) = (V, v_0, V_F, \Lambda, \hookrightarrow)$, or $\mathcal{G}$ for short, into a set of subgraphs $\mathcal{G}_i = (V_i, V_{Fi}, \Lambda_i, \{M_i, F_i, B_i\})$, where $0 \leqslant i \leqslant m$ and $\Lambda_i(v) = \Lambda(v)$, if $v \in V_i$, 0 otherwise. These subgraphs are obtained by partitioning V, V_F and $\hookrightarrow$ as follows:

[1]denoting the valuation η with $\eta(x) = 0$ for $x \in \mathcal{X} \cup \{y\}$.

- $V = \bigcup_{0 \leqslant i \leqslant m} \{V_i\}$, where $V_i = \{(\ell, \Theta) \in V \mid \Theta \subseteq [c_i, c_{i+1})\}$;
- $V_F = \bigcup_{0 \leqslant i \leqslant m} \{V_{Fi}\}$, where $v \in V_{Fi}$ iff $v \in V_i \cap V_F$;
- $\hookrightarrow = \bigcup_{0 \leqslant i \leqslant m} \{M_i \cup F_i \cup B_i\}$, where
  - M_i is the set of *Markovian transitions* (*without reset*) between vertices inside $\mathcal{G}_i$;
  - F_i is the set of *delay transitions* from the vertices in $\mathcal{G}_i$ to that in $\mathcal{G}_{i+1}$ (Forward);
  - B_i is the set of *Markovian transitions* (*with reset*) from $\mathcal{G}_i$ to $\mathcal{G}_0$ (Backward).

  It is easy to see that M_i, F_i, and B_i are pairwise disjoint.

Since the initial vertex of $\mathcal{G}_0$ is v_0 and the initial vertices of $\mathcal{G}_i$ for $0 < i \leqslant m$ are implicitly given by F_{i-1}, we omit them in the definition.

Example 7.27 *Given the region graph in Fig. 7.9, the vertices are partitioned as indicated by the ovals. The M_i edges are unlabeled while the F_i and B_i edges are labeled with δ and "reset", respectively. The V_F vertices (double circles) may appear in any $\mathcal{G}_i$. Actually, if $v = (\ell, [c_i, c_{i+1})) \in V_F$, then $v' = (\ell, [c_j, c_{j+1})) \in V_F$ for $i < j \leqslant m$. This is true because $V_F = \{(\ell, \mathsf{true}) \mid \ell \in Loc_F\}$. It implies that for each final vertex not in the last region, there is a delay transition from it to the next region, see e.g. the final vertex in $\mathcal{G}_{i+1}$ in Fig. 7.9. The exit rate functions and the probabilities on Markovian edges are omitted in the graph.* ♦

Given a subgraph $\mathcal{G}_i$ ($0 \leqslant i \leqslant m$) of $\mathcal{G}$ with k_i states, let the probability vector $\vec{U}_i(x) = [u_i^1(x), \ldots, u_i^{k_i}(x)]^\top \in \mathbb{R}^{k_i \times 1}$ where $u_i^j(x)$ is the probability to go from vertex $v_i^j \in V_i$ to some vertex in V_F (in $\mathcal{G}$) at time x. Starting from (7.4)-(7.6), we provide a set of integral equations for $\vec{U}_i(x)$ which we later on reduce to a system of linear equations. Distinguish two cases:

Case $0 \leqslant i < m$**:** $\vec{U}_i(x)$ is given by:

$$\vec{U}_i(x) = \int_0^{\Delta c_i - x} \mathbf{M}_i(\tau) \vec{U}_i(x+\tau) d\tau + \int_0^{\Delta c_i - x} \mathbf{B}_i(\tau) d\tau \cdot \vec{U}_0(0) + \mathbf{D}_i(\Delta c_i - x) \cdot \mathbf{F}_i \vec{U}_{i+1}(0), \tag{7.7}$$

where $x \in [0, \Delta c_i]$ and

- $\mathbf{D}_i(x) \in \mathbb{R}^{k_i \times k_i}$ is the delay probability matrix, where for any $0 \leqslant j \leqslant k_i$, $\mathbf{D}_i(x)[j,j] = e^{-E(v_i^j)x}$ (the off-diagonal elements are zero);

- $\mathbf{M}_i(x) = \mathbf{D}_i(x) \cdot \mathbf{E}_i \cdot \mathbf{P}_i \in \mathbb{R}^{k_i \times k_i}$ is the probability density matrix for the Markovian transitions inside $\mathcal{G}_i$, where $\mathbf{P}_i$ and $\mathbf{E}_i$ are the transition probability matrix and exit rate matrix for vertices inside $\mathcal{G}_i$, respectively;

- $\mathbf{B}_i(x) \in \mathbb{R}^{k_i \times k_0}$ is the probability density matrix for the reset edges B_i, where $\mathbf{B}_i(x)[j, j']$ indicates the probability density function to take the Markovian jump with reset from the j-th vertex in $\mathcal{G}_i$ to the j'-th vertex in $\mathcal{G}_0$; and

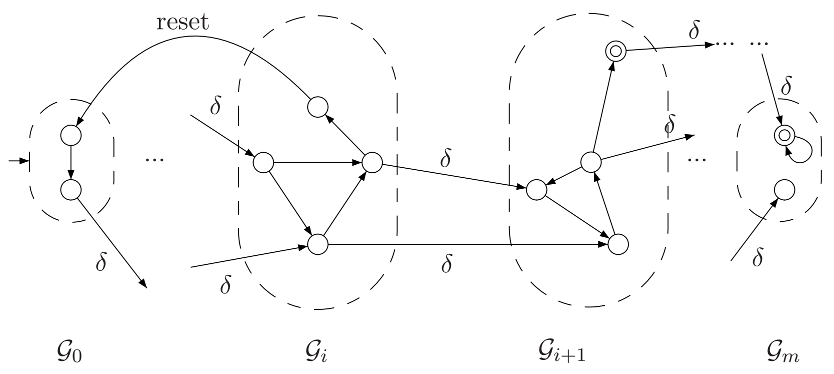


Figure 7.9: Partitioning the region graph

- $\mathbf{F}_i \in \mathbb{R}^{k_i \times k_{i+1}}$ is the incidence matrix for delay edges F_i. More specifically, $\mathbf{F}_i[j, j'] = 1$ indicates that there is a delay transition from the j-th vertex in $\mathcal{G}_i$ to the j'-th vertex in $\mathcal{G}_{i+1}$; 0 otherwise.

Let us explain these equations. The third summand of (7.7) is obtained from (7.4) where $\mathbf{D}_i(\Delta c_i - x)$ indicates the probability to delay until the "end" of region i, and $\mathbf{F}_i \vec{U}_{i+1}(0)$ denotes the probability to continue in $\mathcal{G}_{i+1}$ (at relative time 0). Similarly, the first and second summands are obtained from (7.5); the former reflects the case where clock x is not reset, while the latter considers the reset of x (thus, implying a return to $\mathcal{G}_0$).

Case $i = m$: $\vec{U}_m(x)$ is simplified as follows:

$$\vec{U}_m(x) = \int_0^{\infty} \hat{\mathbf{M}}_m(\tau) \vec{U}_m(x+\tau) d\tau + \vec{1}_F + \int_0^{\infty} \mathbf{B}_m(\tau) d\tau \cdot \vec{U}_0(0) \tag{7.8}$$

where $\hat{\mathbf{M}}_m(\tau)[v, \cdot] = \mathbf{M}_m(\tau)[v, \cdot]$ for $v \notin V_F$, 0 otherwise. $\vec{1}_F$ is a vector such that $\vec{1}_F[v] = 1$ if $v \in V_F$, 0 otherwise. We note that $\vec{1}_F$ stems from the second clause of (7.6), and $\hat{\mathbf{M}}_m$ is obtained by setting the corresponding elements of $\mathbf{M}_m$ to 0. Also note that as the last subgraph $\mathcal{G}_m$ involves infinite regions, it has no delay transitions.

Before solving the system of integral equations (7.7)-(7.8), we first make the following observations:

(i) Due to the fact that inside $\mathcal{G}_i$ there are only Markovian jumps with neither resets nor delay transitions, $\mathcal{G}_i$ with (V_i, Λ_i, M_i) forms a CTMC $\mathcal{C}_i$, say. For each $\mathcal{G}_i$ we define an *augmented* CTMC $\mathcal{C}_i^a$ with state space $V_i \cup V_0$, such that all V_0-vertices are made absorbing in $\mathcal{C}_i^a$. The edges connecting V_i to V_0 are kept and all the edges inside $\mathcal{C}_0$ are removed. The augmented CTMC is used to calculate the probability to start from a vertex in $\mathcal{G}_i$ and take a reset edge within a certain period of time.

(ii) Given any CTMC $\mathcal{C}$ with k states and rate matrix $\mathbf{P} \cdot \mathbf{E}$, the matrix $\mathbf{\Pi}(x)$ is given by:

$$\mathbf{\Pi}(x) = \int_0^x \mathbf{M}(\tau)\mathbf{\Pi}(x-\tau)d\tau + \mathbf{D}(x). \tag{7.9}$$

Intuitively, $\mathbf{\Pi}(t)[j, j']$ indicates the probability to start from vertex j and reach j' at time t.

The following proposition states the close relationship between $\mathbf{\Pi}(x)$ and the transient probability vector:

Proposition 7.28 *Given a* CTMC $\mathcal{C}$ *with initial distribution* α*, rate matrix* $\mathbf{P}\cdot\mathbf{E}$ *and* $\mathbf{\Pi}(t)$*,* $\vec{\wp}(t)$ *satisfies the following two equations:*

$$\begin{aligned} \vec{\wp}(t) &= \alpha \cdot \mathbf{\Pi}(t), & (7.10)\\ \frac{d\vec{\wp}(t)}{dt} &= \vec{\wp}(t) \cdot \mathbf{Q}, & (7.11)\end{aligned}$$

where $\mathbf{Q} = \mathbf{P}\cdot\mathbf{E} - \mathbf{E}$ *is the infinitesimal generator.*

Proof: The proof can be found in Appendix A.6, page 171. ■

$\vec{\wp}(t)$ is the *transient probability vector* with $\wp_s(t)$ indicating the probability to be in state s at time t given the initial probability distribution α. Eq. (7.11) is the celebrated forward Chapman-Kolmogorov equations. According to this proposition, solving the integral equation $\mathbf{\Pi}(t)$ boils down to selecting the appropriate initial distribution vector α and solving the system of ODEs (7.11), which can be done very efficiently using *uniformization*.

Prior to exposing how to solve the system of integral equations by solving a system of *linear* equations, we define $\bar{\mathbf{\Pi}}_i^a \in \mathbb{R}^{k_i \times k_0}$ for an augmented CTMC $\mathcal{C}_i^a$ to be part of $\mathbf{\Pi}_i^a$, where $\bar{\mathbf{\Pi}}_i^a$ only keeps the probabilities starting from V_i and ending in V_0. Actually,

$$\mathbf{\Pi}_i^a(x) = \left(\begin{array}{c|c} \mathbf{\Pi}_i(x) & \bar{\mathbf{\Pi}}_i^a(x) \\ \hline \mathbf{0} & \mathbf{I} \end{array}\right),$$

where $\mathbf{0} \in \mathbb{R}^{k_0 \times k_i}$ is the zero matrix and $\mathbf{I} \in \mathbb{R}^{k_0 \times k_0}$ is the identity matrix.

Theorem 7.29 *For subgraph* $\mathcal{G}_i$ *of* $\mathcal{G}$ *with* k_i *states, it holds for* $0 \leqslant i < m$ *that:*

$$\vec{U}_i(0) = \mathbf{\Pi}_i(\Delta c_i) \cdot \mathbf{F}_i \vec{U}_{i+1}(0) + \bar{\mathbf{\Pi}}_i^a(\Delta c_i) \cdot \vec{U}_0(0), \tag{7.12}$$

where $\mathbf{\Pi}_i(\Delta c_i)$ *and* $\bar{\mathbf{\Pi}}_i^a(\Delta c_i)$ *are for* CTMC $\mathcal{C}_i$ *and the augmented* CTMC $\mathcal{C}_i^a$*, respectively. For case* $i = m$*,*

$$\vec{U}_m(0) = \hat{\mathbf{P}}_i \cdot \vec{U}_m(0) + \vec{1}_F + \hat{\mathbf{B}}_m \cdot \vec{U}_0(0), \tag{7.13}$$

where $\hat{\mathbf{P}}_i(v, v') = \mathbf{P}_i(v, v')$ *if* $v \notin V_F$*;* 0 *otherwise and* $\hat{\mathbf{B}}_m = \int_0^\infty \mathbf{B}_m(\tau)d\tau$*.*

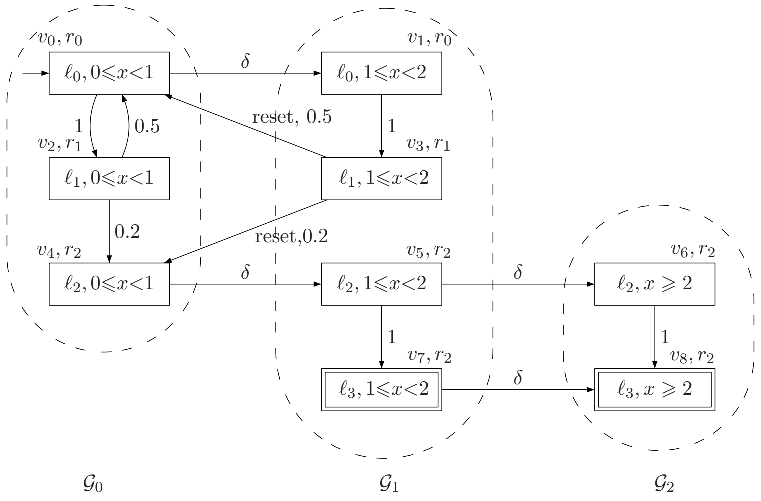


Figure 7.10: Partition the region graph in Fig. 7.4(d)

Proof: The proof can be found in Appendix A.7, page 173. ■

Since the coefficients of the linear equations are all known, solving the system of linear equations yields $\vec{U}_0(0)$, which contains the probability $Prob_{v_0}(0)$ of reaching V_F from initial vertex v_0.

Now we explain how (7.12) is derived from (7.7). The term $\mathbf{\Pi}_i(\Delta c_i) \cdot \mathbf{F}_i \vec{U}_{i+1}(0)$ is for the delay transitions, where $\mathbf{F}_i$ specifies how the delay transitions are connected between $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$. The term $\bar{\mathbf{\Pi}}_i^a(\Delta c_i) \cdot \vec{U}_0(0)$ is for Markovian transitions with reset. $\bar{\mathbf{\Pi}}_i^a(\Delta c_i)$ in the augmented CTMC $\mathcal{C}_i^a$ specifies the probabilities to take first transitions inside $\mathcal{G}_i$ and then a one-step Markovian transition back to $\mathcal{G}_0$. Eq. (7.13) is derived from (7.8). Since it is the last region and time goes to infinity, the time to enter the region is irrelevant (thus set to 0). Thus $\int_0^\infty \hat{\mathbf{M}}_i(\tau)d\tau$ boils down to $\hat{\mathbf{P}}_i$. In fact, the Markovian jump probability inside $\mathcal{G}_m$ can be taken from the embedded DTMC of $\mathcal{C}_m$, which is $\hat{\mathbf{P}}_i$.

Example 7.30 *For the single-clock* DMTA$^\diamondsuit$ *in Fig. 7.4(a) (page 140), we show how to compute the reachability probability* $Prob((v_0, 0), (v_5, \cdot))$ *on the region graph* $\mathcal{G}$ *(cf. Fig. 7.4(d)), which has been partitioned into subgraphs* $\mathcal{G}_0$*,* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *as in Fig. 7.10.*

The matrices for $\mathcal{G}_0$ *are given as*

$$\mathbf{M}_0(x) = \begin{pmatrix} 0 & 1{\cdot}r_0{\cdot}e^{-r_0 x} & 0 \\ 0.5{\cdot}r_1{\cdot}e^{-r_1 x} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \mathbf{F}_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The matrices for $\mathcal{G}_1$ *are given as*

$$\mathbf{M}_1(x) = \begin{pmatrix} 0 & r_0 \cdot e^{-r_0 x} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_2 \cdot e^{-r_2 x} \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{F}_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{B}_1 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{M}_1^a(x) = \begin{pmatrix} 0 & r_0 \cdot e^{-r_0 x} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \cdot r_1 \cdot e^{-r_1 x} & 0 & 0.2 \cdot r_1 \cdot e^{-r_1 x} \\ 0 & 0 & 0 & r_2 \cdot e^{-r_2 x} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The matrices for $\mathcal{G}_2$ are given as

$$\hat{\mathbf{M}}_2(x) = \begin{pmatrix} 0 & r_2 \cdot e^{-r_2 x} \\ 0 & 0 \end{pmatrix} \quad \hat{\mathbf{P}}_2 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

To obtain the system of linear equations, we need:

$$\mathbf{\Pi}_0(1) = \begin{pmatrix} p_{00} & p_{02} & p_{04} \\ p_{20} & p_{22} & p_{24} \\ p_{40} & p_{42} & p_{44} \end{pmatrix} \quad \mathbf{\Pi}_1(1) = \begin{pmatrix} p_{11} & p_{13} & p_{15} & p_{17} \\ p_{31} & p_{33} & p_{35} & p_{37} \\ p_{51} & p_{53} & p_{55} & p_{57} \\ p_{71} & p_{73} & p_{75} & p_{77} \end{pmatrix} \quad \bar{\mathbf{\Pi}}_1^a(1) = \begin{pmatrix} \bar{p}_{10} & \bar{p}_{12} & \bar{p}_{14} \\ \bar{p}_{30} & \bar{p}_{32} & \bar{p}_{32} \\ \bar{p}_{50} & \bar{p}_{52} & \bar{p}_{54} \\ \bar{p}_{70} & \bar{p}_{72} & \bar{p}_{74} \end{pmatrix}$$

All elements in these $\mathbf{\Pi}$*-matrices can be computed by the transient probability in the corresponding* CTMC*s* $\mathcal{C}_0$, $\mathcal{C}_1$ *and* $\mathcal{C}_1^a$ *(cf. Fig. 7.11).*

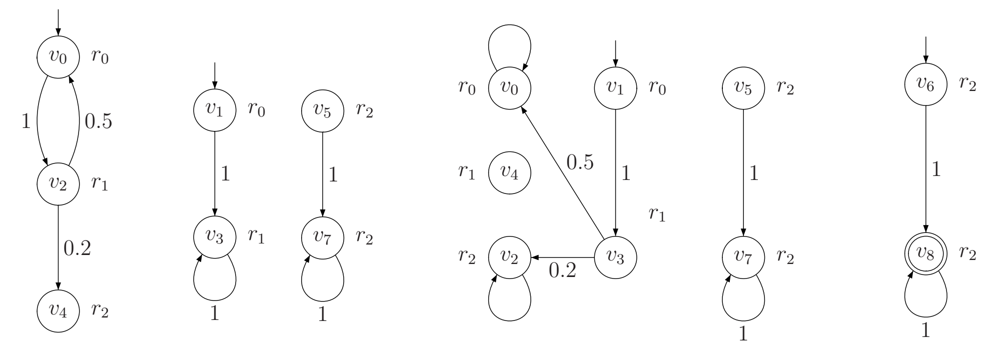
The obtained system of linear equations by applying Theorem 7.29 is:

$$\begin{pmatrix} u_0 \\ u_2 \\ u_4 \end{pmatrix} = \begin{pmatrix} p_{00} & p_{02} & p_{04} \\ p_{20} & p_{22} & p_{24} \\ p_{40} & p_{42} & p_{44} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} u_1 \\ u_3 \\ u_5 \\ u_7 \end{pmatrix}$$

$$\begin{pmatrix} u_1 \\ u_3 \\ u_5 \\ u_7 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{13} & p_{15} & p_{17} \\ p_{31} & p_{33} & p_{35} & p_{37} \\ p_{51} & p_{53} & p_{55} & p_{57} \\ p_{71} & p_{73} & p_{75} & p_{77} \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} u_6 \\ u_8 \end{pmatrix} + \begin{pmatrix} \bar{p}_{10} & \bar{p}_{12} & \bar{p}_{14} \\ \bar{p}_{30} & \bar{p}_{32} & \bar{p}_{32} \\ \bar{p}_{50} & \bar{p}_{52} & \bar{p}_{54} \\ \bar{p}_{70} & \bar{p}_{72} & \bar{p}_{74} \end{pmatrix} \cdot \begin{pmatrix} u_0 \\ u_1 \\ u_3 \end{pmatrix}$$

$$\begin{pmatrix} u_6 \\ u_8 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} u_6 \\ u_8 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

This can be solved easily. ♦



(a) $\mathcal{C}_0$ (b) $\mathcal{C}_1$ (c) $\mathcal{C}_1^a$ (d) $\mathcal{C}_2$

Figure 7.11: Derived CTMCs

Remark 7.31 *We note that for two-clock* DTA$^\Diamond$ *which yield two-clock* DMTA$^\Diamond$*, the approach given in this section fails in general. In the single-clock case, the reset guarantees to jump to $\mathcal{G}_0(0)$ and delay to $\mathcal{G}_{i+1}(0)$ when it is in $\mathcal{G}_i$. However, in the two-clock case, after a delay or reset generally only one clock has a fixed value while the value of the other one is not determined.*

The time-complexity of computing the reachability probability in the single-clock DTA$^\Diamond$ case is $\mathcal{O}(m \cdot |S|^2 \cdot |Loc|^2 \cdot \lambda \cdot \Delta c + m^3 \cdot |S|^3 \cdot |Loc|^3)$, where m is the number of constants appearing in the DTA$^\Diamond$, $|S|$ is the number of states in the CTMC, $|Loc|$ is the number of locations in the DTA$^\Diamond$, λ is the maximal exit rate in the CTMC and $\Delta c = \max_{0 \leq i < m}\{c_{i+1} - c_i\}$. The first term $m \cdot |S|^2 \cdot |Loc|^2 \cdot \lambda \cdot \Delta c$ is due to the uniformization technique for computing transient distribution; and the second term $m^3 \cdot |S|^3 \cdot |Loc|^3$ is the time complexity for solving a system of linear equations with $\mathcal{O}(m \cdot |S| \cdot |Loc|)$ variables.

7.4 Model Checking DTA$^\omega$ Specifications

We now deal with DTA$^\omega$ specifications. Given the product $\mathcal{M}^\omega = (Loc, \mathcal{X}, \ell_0, Loc_\mathcal{F}, E, \rightsquigarrow)$, we first define the region graph $\mathcal{G}^\omega(\mathcal{M}^\omega)$ (or simply $\mathcal{G}^\omega$) as $(V, v_0, V_F^\omega, \Lambda, \hookrightarrow)$ without specifying how the accepting set V_F^ω is defined. This will become clear later. The elements V, v_0, Λ and $\hookrightarrow$ are defined in the same way as in Def. 7.17 (page 142).

The Muller acceptance conditions $Q_\mathcal{F}$ in the DTA$^\omega$ consider the infinite paths that visit the locations in $F \in Q_\mathcal{F}$ infinitely often. For this sake, BSCCs in the *region graph* $\mathcal{G}^\omega$ that consist of set of vertices corresponding to $L_F \in Loc_\mathcal{F}$ are of most importance. Note that it is not sufficient to consider the BSCCs in the DMTA$^\omega$. The reason will

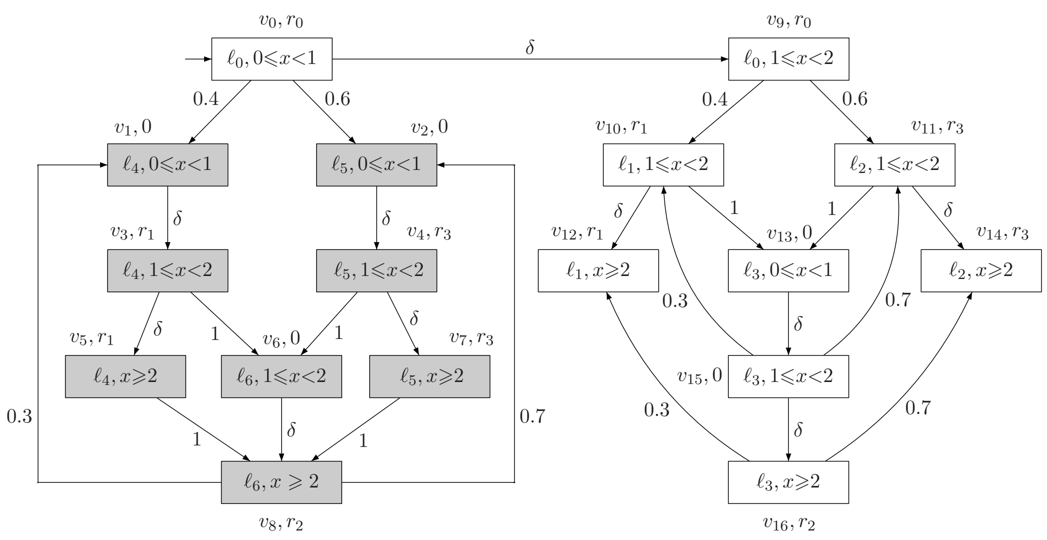


Figure 7.12: Region graph of the product DMTA$^\omega$ in Fig. 7.5(c)

become clear in Remark 7.36. Let $v \in B$ denote that vertex v is in the BSCC B. We define *accepting* BSCC*s* as follows:

Definition 7.32 (*a*BSCC) *Given a product $\mathcal{C} \otimes \mathcal{A}^\omega = (Loc, \mathcal{X}, \ell_0, Loc_{\mathcal{F}}, E, \rightsquigarrow)$ and its region graph $\mathcal{G}^\omega$, a* BSCC *B in $\mathcal{G}^\omega$ is* accepting *if there exists $L_F \in Loc_{\mathcal{F}}$ such that for any $v \in B$, $v{\downarrow_1} \in L_F$. Let $a\mathcal{B}$ denote the set of accepting* BSCC*s in $\mathcal{G}^\omega$.*

Based on $a\mathcal{B}$, we can now define the set of accepting vertices of $\mathcal{G}^\omega$ as $V_F^\omega = \{v \in B \mid B \in a\mathcal{B}\}$. Note that it is *not* an acceptance family but a set of accepting vertices.

Example 7.33 *For the* DMTA$^\omega$ *in Fig. 7.5(c) with $Loc_{\mathcal{F}} = \big\{\{\ell_1, \ell_2, \ell_3\}, \{\ell_4, \ell_5, \ell_6\}\big\}$, the region graph is as in Fig. 7.12. There is one accepting* BSCC*, which has been labeled with gray. This* BSCC *corresponds to the set $\{\ell_4, \ell_5, \ell_6\} \in Loc_{\mathcal{F}}$ in the* DMTA$^\omega$*. There is no* BSCC *corresponding to the set $\{\ell_1, \ell_2, \ell_3\}$ because in the region graph v_{12} and v_{14} are sink vertices connecting to the* SCC*. In other words, the probabilities will leak when $x > 2$ on either ℓ_1 or ℓ_2. This is determined by the guards on the* DTA$^\omega$*.* ◆

We remark on two points: 1) the probability of staying in an *a*BSCC is 1, considering both the delay and Markovian transitions. That is to say, there are no outgoing transitions from which probabilities can "leak"; 2) any two *a*BSCCs are disjoint, such that the probabilities to reach two BSCCs can be added. These two points are later important for the computation of the reachability probability.

Let $Prob^{\mathcal{C}}(\mathcal{A}^\omega)$ be the probability of the set of infinite paths in $\mathcal{C}$ that can be accepted by $\mathcal{A}^\omega$. The following theorem computes $Prob^{\mathcal{C}}(\mathcal{A}^\omega)$ on the region graph:

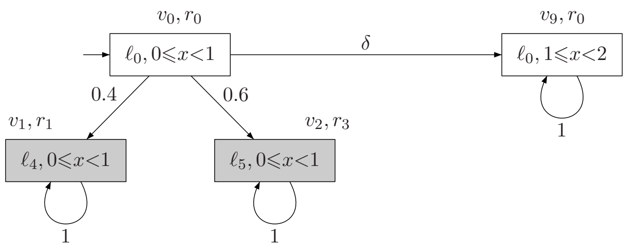


Figure 7.13: The transformed region graph $\mathcal{G}^\omega_{abs}$

Theorem 7.34 *For any* CTMC $\mathcal{C}$, DTA$^\omega$ $\mathcal{A}^\omega$, *and the region graph* $\mathcal{G}^\omega = (V, v_0, V^\omega_F, \Lambda, \hookrightarrow)$ *of the product, it holds that:*

$$Prob^{\mathcal{C}}(\mathcal{A}^\omega) = Prob^{\mathcal{G}^\omega}(v_0, \Diamond V^\omega_F).$$

Proof: The proof can be found in Appendix A.8, page 175. ■

Actually, the region graph $\mathcal{G}^\omega$ can be simplified to $\mathcal{G}^\omega_{abs}$ to compute $Prob^{\mathcal{C}}(\mathcal{A}^\omega)$. $\mathcal{G}^\omega_{abs}$ is obtained by making (i) all vertices in V^ω_F and (ii) all vertices that *cannot* reach V^ω_F absorbing. (i) is justified by the fact that for these $v \in V^\omega_F$, $Prob^{\mathcal{G}}(v, \Diamond V^\omega_F) = 1$; while (ii) is because $Prob^{\mathcal{G}}(v', \Diamond V^\omega_F) = 0$, for v' cannot reach V^ω_F. It is obvious to see that

$$Prob^{\mathcal{G}^\omega}(v_0, \Diamond V^\omega_F) = Prob^{\mathcal{G}^\omega_{abs}}(v_0, \Diamond V^\omega_F).$$

Example 7.35 *The transformed region graph $\mathcal{G}^\omega_{abs}$ of that in Fig. 7.12 is shown in Fig. 7.13. We omit all the vertices that cannot be reached from v_0 in $\mathcal{G}^\omega_{abs}$. In this new model, $V^\omega_F = \{v_1, v_2\}$. We now can perform the approach for computing timed-unbounded reachability probabilities in Section 7.3 such that Eq. (7.4)-(7.6) can be applied. We have: $Prob^{\mathcal{G}^\omega_{abs}}(v_0, \Diamond V^\omega_F) = Prob^{\mathcal{G}^\omega_{abs}}(v_0, \Diamond at_{v_1}) + Prob^{\mathcal{G}^\omega_{abs}}(v_0, \Diamond at_{v_2})$. Note that $Prob^{\mathcal{G}^\omega_{abs}}(v_i, \Diamond V^\omega_F) = 1$ for $i = 1, 2$ and 0 for $i = 9$. For the delay transition $v_0 \overset{\delta}{\hookrightarrow} v_9$,*

$$Prob_{v_0,\delta}(0) = e^{-r_0 \cdot 1} \cdot Prob_{v_9}(1) = e^{-r_0\cdot 1} \cdot 0 = 0.$$

For the Markovian transition $v_0 \overset{0.4,\{x\}}{\hookrightarrow} v_1$,

$$Prob_{v_0,v_1}(0) = \int_0^1 0.4 \cdot r_0 \cdot e^{-r_0\cdot\tau} \cdot Prob_{v_1}(\tau) d\tau = \int_0^1 0.4 \cdot r_0 \cdot e^{-r_0 \cdot \tau} d\tau.$$

A similar reasoning applies to $v_0 \overset{0.6,\{x\}}{\hookrightarrow} v_2$. In the end, we have

$$Prob^{\mathcal{C}}(\mathcal{A}^\omega) = \int_0^1 (0.4 + 0.6) \cdot r_0 \cdot e^{-r_0\cdot\tau} d\tau = \int_0^1 r_0 \cdot e^{-r_0\cdot\tau} d\tau = 1 - e^{-r_0}.$$

♦

Remark 7.36 (Why not BSCCs in the product?) *There are two* BSCC*s in the product* DMTA$^\omega$*: one formed by* $\{\ell_1, \ell_2, \ell_3\}$ *and the other by* $\{\ell_4, \ell_5, \ell_6\}$*. As turned out in the example that only the latter forms a* BSCC *in the region graph while the former does not. This is because the guards on the transitions also play a role on whether a path can be accepted. The impact of guards, however, is not immediately clear in the product* DMTA$^\omega$*, but is implicitly consumed in the region graph. This justifies finding* BSCC*s in the region graph instead of in the product.*

Theorem 7.34 implies that computing the probability of a set of infinite paths (LHS) can be reduced to computing the probability of a set of finite paths (RHS). The latter has been solved in Section 7.3 with the characterization of a system of integral equations and also the approximation by a system of PDEs. The case of a single clock DTA$^\omega$, due to this reduction, can also be solved as a system of ODEs (as in Section 7.3.2).

7.5 Summary

We addressed the quantitative verification of a CTMC $\mathcal{C}$ against a DTA$^\diamond$ $\mathcal{A}$ (DTA$^\omega$ $\mathcal{A}^\omega$). As a key result, we showed that the set of the accepting paths in $\mathcal{C}$ by DTA is measurable and the probability of $\mathcal{C} \models \mathcal{A}$ can be reduced to computing reachability probabilities in the embedded DTMP of a PDP. The probabilities can be characterized by a system of Volterra integral equations of the second type and can be approximated by a system of PDEs. For single-clock DTA$^\diamond$, this reduces to solving a system of linear equations whose coefficients are a system of ODEs. The probability of $\mathcal{C} \models \mathcal{A}^\omega$ is reducible to computing the reachability probabilities to the accepting BSCCs in the region graph and the thus obtained PDP.

Our results partially complete Table 1.1 in the Introduction as in Table 7.1. For the detailed explanation of the table, please refer to page 1.1. Note that the time complexity of solving the system of integral equation is open.

7.5.1 Related Work

We summarize some of the most relevant work here:

Checking Markov Chains with Actions and State Labels. Baier et al. introduced the logic asCSL [BCH+07] which provides a powerful means to characterize execution paths of Markov chains with both actions and state labels. asCSL can be regarded as an extension of the purely state-based logic CSL. In asCSL, path properties are characterized by regular expressions over actions and state formulae. Thus, the truth value of a path formula depends not only on the available actions in a given time interval, but also on the validity of certain state formulae in intermediate states. Using

	branching time		linear time	
discrete-time (DTMC $\mathcal{D}$)	PCTL		LTL	
	linear equations [HJ94] $(\star)$		automata-based [Var85][CSS03] $(\star\star)$	tableau-based [CY95b]
	PTIME		PSPACE-C	
continuous-time (CTMC $\mathcal{C}$)	untimed PCTL	real-time CSL	untimed LTL	real-time DTA$^\diamondsuit$/DTA$^\omega$
	$emb(\mathcal{C})$ cf. $(\star)$	integral equations [ASSB00][BHHK03]	$emb(\mathcal{C})$ cf. $(\star\star)$	integral equations [CHKM09a]/this thesis
	PTIME	PTIME	PSPACE-C	?

Table 7.1: An overview of verifying Markov chains

an automaton-based technique, an asCSL formula and a Markov chain with actions and state labels are combined into a product Markov chain. For time intervals starting at zero (asCSL$^{\leqslant t}$), a reduction of the model checking problem for asCSL to CSL model checking on this product Markov chain is established.

To compare the expressiveness of the specifications, DTA as a linear-time specification is *incomparable* with branching-time specifications like asCSL, or CSL$^{\mathrm{TA}}$ (see below). The following comparison is based on a CSL-like logic where the path formulae are characterized by DTAs. It is shown in [DHS09] how a nondeterministic program automaton (NPA) for a regular expression path formula can be systematically encoded into an equivalent DTA. Actually, the timing feature in the DTA is not relevant here. This implies that DTA is strictly more expressive than a regular expression as a path formula in asCSL.

Model Checking Timed and Stochastic Properties with CSL$^{\mathbf{TA}}$. Donatelli, Haddad and Sproston presented the stochastic logic CSL$^{\mathrm{TA}}$ [DHS09], which is more expressive than CSL and at least as expressive as asCSL, and in which path formulae are specified using automata (more precisely, *single-clock* DTA$^\diamondsuit$). A model-checking algorithm for CSL$^{\mathrm{TA}}$ is provided. In [DHS09], $\mathcal{C}\otimes\mathcal{A}$ is interpreted as a Markov renewal processes and model checking CSL$^{\mathrm{TA}}$ is reduced to computing reachability probabilities in a DTMC whose transition probabilities are given by subordinate CTMCs. This technique cannot be generalized to multiple clocks.

As before, we make the comparison of DTA and CSL$^{\mathrm{TA}}$ when considering DTA as a path formula in a CSL-like logic. Our approach does not restrict the number of clocks and supports more specifications than CSL$^{\mathrm{TA}}$. For the single-clock DTA$^\diamondsuit$, our approach produces the same result as [DHS09], but yields a conceptually simpler formulation whose correctness can be derived from the simplification of the system of integral equations obtained in the general case. Our work essentially provides a proof of the procedure proposed in [DHS09]. Moreover, measurability and the specification of properties referring to the probability of an infinite sequence of timed events has not been addressed in [DHS09].

Probabilistic Semantics of Timed Automata. Bertrand et al. provided a quantitative interpretation to timed automata where delays and discrete choices are interpreted probabilistically [BBB$^+$07][BBB$^+$08][BBBM08]. In this approach, delays of unbounded clocks are governed by exponential distributions like in CTMCs. Decidability results have been obtained for almost-sure properties [BBB$^+$08] and quantitative verification [BBBM08] for (a subclass of) single-clock timed automata.

Chapter 8

Conclusion

In this dissertation, we have addressed three aspects concerning probabilistic model checking: *diagnosis* — generating counterexamples, *synthesis* — providing valid parameter values and *analysis* — verifying linear real-time properties. The three aspects are relatively independent while all contribute to developing new theory and algorithms in the research field of probabilistic model checking. We summarize our contributions briefly as follows:

- We gave a formal definition as well as an algorithmic framework of generating counterexamples in the probabilistic setting;
- We made first important steps to the problem of synthesizing values for parametric probabilistic models;
- We provided approximate algorithms for checking linear real-time properties on CTMCs.

All three problems ave received scant attention in the literature so far. Below we provide our specific contributions together with some future research directions in each aspect.

Diagnosis. We introduce a formal definition of counterexamples in the setting of probabilistic model checking. The counterexample is represented either by enumerating paths or by regular expressions; for the former, finding (informative) counterexamples is cast as shortest paths problems, while for the latter, it is reducible to finding short regular expressions. An algorithmic framework is proposed that can be adapted to various combinations of models and logics in the probabilistic setting, such as PCTL and LTL model checking on DTMCs.

In the future, we expect to apply our definition of counterexamples in the CEGAR-framework for other abstraction techniques such as [KKLW07][KNP06]. Some more work can be done concerning the regular expression counterexamples, e.g., generating more "traceable" counterexamples or the efficient computation of the value of a constrained regular expression.

Synthesis. We drew the attention to the problem of synthesizing values for parametric CTMCs wrt. time-bounded reachability specifications. We presented two algorithms that yield an approximation of the region of values that can guarantee the satisfaction of the specification. We specified the algorithm, error bound and time complexity for both the symbolic approach (by polynomial solving) and the non-symbolic approach (by pCTMC instantiation), accompanied by a case study.

In the future, we foresee the extension of the current work on time-bounded reachability properties to full CSL formulae. We also expect that synthesizing parameters in the probabilistic models with rewards, nondeterminism or hybrid features may also profit from the results in this dissertation.

Analysis. We focus on verifying CTMCs against linear real-time properties specified by deterministic timed automata (DTAs). The model checking problem aims at computing the probability of the set of paths in CTMC $\mathcal{C}$ that can be accepted by DTA $\mathcal{A}$, denoted $Paths^{\mathcal{C}}(\mathcal{A})$. We consider DTAs with reachability (finite, DTA$^{\Diamond}$) and Muller (infinite, DTA$^{\omega}$) acceptance conditions, respectively. It is shown that $Paths^{\mathcal{C}}(\mathcal{A})$ is measurable and computing its probability for DTA$^{\Diamond}$ can be reduced to computing the reachability probability in a PDP. The reachability probability is characterized as the least solution of a system of integral equations and is shown to be approximated by solving a system of PDEs. Furthermore, we show that the special case of single-clock DTA$^{\Diamond}$ can be simplified to solving a system of linear equations. We also deal with DTA$^{\omega}$ specifications, where the problem is proven to be reducible to the reachability problem as in the DTA$^{\Diamond}$ case.

In the future, several other forms of the linear real-time properties can be considered: those specified by a logic, e.g., M(I)TL, or dropping the constraint of the determinism in the timed automaton. Other more advanced models, e.g. CTMDPs or continuous-time MRMs are also of great interest.

Appendix A

Proofs in Chapter 7

A.1 Proof of Theorem 7.7

Theorem 7.7 For a CTMC $\mathcal{C}$ and DTA $\mathcal{A}$, $\mathit{Paths}^{\mathcal{C}}(\mathcal{A})$ is measurable.
Proof: We first deal with the case that $\mathcal{A}$ only contains *strict* inequality. Since $\mathit{Paths}^{\mathcal{C}}(\mathcal{A})$ is a set of *finite* paths, $\mathit{Paths}^{\mathcal{C}}(\mathcal{A}) = \bigcup_{n\in\mathbb{N}} \mathit{Paths}^{\mathcal{C}}_n(\mathcal{A})$, where $\mathit{Paths}^{\mathcal{C}}_n(\mathcal{A})$ is the set of accepting paths by $\mathcal{A}$ of length n. For any path $\rho := s_0 \xrightarrow{t_0} s_1 \cdots s_{n-1} \xrightarrow{t_{n-1}} s_n \in \mathit{Paths}^{\mathcal{C}}_n(\mathcal{A})$, we can associate ρ with a path $\theta := q_0 \xrightarrow{L(s_0),t_0} q_1 \cdots q_{n-1} \xrightarrow{L(s_{n-1}),t_{n-1}} q_n$ of $\mathcal{A}$ induced by the location sequence:

$$q_0 \xrightarrow{L(s_0),g_0,X_0} q_1 \cdots q_{n-1} \xrightarrow{L(s_{n-1}),g_{n-1},X_{n-1}} q_n,$$

such that $q_n \in Q_F$ and there exist $\{\eta_i\}_{1\leqslant i<n}$ with 1) $\eta_0 = \vec{0}$; 2) $(\eta_i + t_i) \models g_i$; and 3) $\eta_{i+1} = (\eta_i + t_i)[X_i := 0]$, where η_i is the clock valuation on entering q_i.

To prove the measurability of $\mathit{Paths}^{\mathcal{C}}_n(\mathcal{A})$, it suffices to show that for each path $\rho := s_0 \xrightarrow{t_0} \cdots \xrightarrow{t_{n-1}} s_n \in \mathit{Paths}^{\mathcal{C}}_n(\mathcal{A})$, there exists a cylinder set $C(s_0, I_0, \ldots, I_{n-1}, s_n)$ (C_ρ for short) that contains ρ and that each path in C_ρ is accepted by $\mathcal{A}$. The interval I_i is constructed according to t_i as $I_i = [t_i^-, t_i^+]$ such that

- If $t_i \in \mathbb{Q}$, then $t_i^- = t_i^+ := t_i$;
- else if $t_i \in \mathbb{R} \setminus \mathbb{Q}$, then let $t_i^-, t_i^+ \in \mathbb{Q}$ such that

  $t_i^- \leqslant t_i \leqslant t_i^+$ and $\lfloor t_i^- \rfloor = \lfloor t_i \rfloor$ and $\lceil t_i^+ \rceil = \lceil t_i \rceil$;

  $t_i^+ - t_i^- < \dfrac{\Delta}{2\cdot n}$, where (with $\{\cdot\}$ denoting the fractional part) $\Delta = \min_{0\leqslant j<n,\ x\in\mathcal{X}} \Big\{ \{\eta_j(x)+t_j\}, 1-\{\eta_j(x)+t_j\} \;\Big|\; \{\eta_j(x)+t_j\} \neq 0 \Big\}$[1].

[1]Note that we are considering open timed automata. Hence for any i with $\eta_i + t_i \models g_i$, it must be the case that $\{\eta_i(x)+t_i\} \neq 0$.

To show that $\rho' := s_0 \xrightarrow{t'_0} \cdots \xrightarrow{t'_{n-1}} s_n \in C_\rho$ is accepted by $\mathcal{A}$, let $\eta'_0 := \vec{0}$ and $\eta'_{i+1} := (\eta'_i + t'_i)[X_i := 0]$. We will show that $\eta'_i + t'_i \models g_i$. To this end, it suffices to observe that $\eta'_0 = \eta_0$, and for any $i > 0$ and any clock variable x,

$$\left|\eta'_i(x) - \eta_i(x)\right| \leqslant \sum_{j=0}^{i-1} \left|t'_j - t_j\right| \leqslant \sum_{j=0}^{i-1} t^+_j - t^-_j \leqslant n \cdot (t^+_j - t^-_j) \leqslant \frac{\Delta}{2}.$$

We claim that since DTA $\mathcal{A}$ is open, it must be the case that $\eta'_i + t'_i \models g_i$. To see this, suppose g_i is of the form $x > K$ for some integer K. We have that $|\eta'_i(x) - \eta_i(x)| \leqslant \frac{\Delta}{2}$ and $|t'_i - t_i| < \frac{\Delta}{2}$, therefore $|(\eta'_i(x) + t'_i) - (\eta_i(x) + t_i)| < \Delta$. Note that $\eta_i(x) + t_i > K$, and thus $\eta_i(x) + t_i - \{\eta_i(x) + t_i\} = \lceil \eta_i(x) + t_i \rceil \geq K$. Hence $\eta_i(x) + t_i - \Delta \geq K$ since $\Delta \leqslant \{\eta_i(x) + t_i\}$. It follows that $\eta'_i(x) + t'_i > K$. A similar argument applies to the case $x < K$ and can be extended to any constraint g_i. Thus, $\eta'_i + t'_i \models g_i$.

It follows that C_ρ is a cylinder set of $\mathcal{C}$ and each path in this cylinder set is accepted by $\mathcal{A}$, namely, $\rho \in C_\rho$ and $C_\rho \subseteq \mathit{Paths}^{\mathcal{C}}_n(\mathcal{A})$ with $|\rho| = n$. Together with the fact that $\mathit{Paths}^{\mathcal{C}}_n(\mathcal{A}) \subseteq \bigcup_{\rho \in \mathit{Paths}^{\mathcal{C}}_n(\mathcal{A})} C_\rho$, we have:

$$\mathit{Paths}^{\mathcal{C}}_n(\mathcal{A}) = \bigcup_{\rho \in \mathit{Paths}^{\mathcal{C}}_n(\mathcal{A})} C_\rho \quad \text{and} \quad \mathit{Paths}^{\mathcal{C}}(\mathcal{A}) = \bigcup_{n \in \mathbb{N}} \bigcup_{\rho \in \mathit{Paths}^{\mathcal{C}}_n(\mathcal{A})} C_\rho.$$

We note that each interval in the cylinder set C_ρ has rational bounds, thus C_ρ is measurable. It follows that $\mathit{Paths}^{\mathcal{C}}(\mathcal{A})$ is a union of *countably many* cylinder sets, and hence is measurable.

We then deal with $\mathcal{A}$ with equalities of the form $x = n$ for $n \in \mathbb{N}$. We show the measurability by induction on the number of equalities appearing in $\mathcal{A}$. We have shown the base case (DTA with only strict inequalities). Now suppose there exists a transition $\iota = q \xrightarrow{a,g,X} q'$ where g contains $x = n$. We first consider a DTA $\mathcal{A}_\iota$ obtained from $\mathcal{A}$ by deleting the transitions from q other than ι. We then consider three DTA $\bar{\mathcal{A}}_\iota$, $\mathcal{A}^>_\iota$ and $\mathcal{A}^<_\iota$ where $\bar{\mathcal{A}}_\iota$ is obtained from $\mathcal{A}_\iota$ by replacing $x = n$ by true; $\mathcal{A}^>_\iota$ is obtained from $\mathcal{A}_\iota$ by replacing $x = n$ by $x > n$ and $\mathcal{A}^<_\iota$ is obtained from $\mathcal{A}_\iota$ by replacing $x = n$ by $x < n$. It is not difficult to see that

$$\mathit{Paths}^{\mathcal{C}}(\mathcal{A}_\iota) = \mathit{Paths}^{\mathcal{C}}(\bar{\mathcal{A}}_\iota) \setminus (\mathit{Paths}^{\mathcal{C}}(\mathcal{A}^>_\iota) \cup \mathit{Paths}^{\mathcal{C}}(\mathcal{A}^<_\iota)).$$

Note that this holds since $\mathcal{A}$ is deterministic. By induction hypothesis, $\mathit{Paths}^{\mathcal{C}}(\bar{\mathcal{A}}_\iota)$, $\mathit{Paths}^{\mathcal{C}}(\mathcal{A}^>_\iota)$ and $\mathit{Paths}^{\mathcal{C}}(\mathcal{A}^<_\iota)$ are measurable. Hence $\mathit{Paths}^{\mathcal{C}}(\mathcal{A}_\iota)$ is measurable. Furthermore, we note that

$$\mathit{Paths}^{\mathcal{C}}(\mathcal{A}) = \bigcup_{\iota = q \xrightarrow{a,g,X} q'} \mathit{Paths}^{\mathcal{C}}(\mathcal{A}_\iota),$$

therefore $\mathit{Paths}^{\mathcal{C}}(\mathcal{A})$ is measurable as well.

For arbitrary $\mathcal{A}$ with time constraints of the form $x \bowtie n$ where $\bowtie \in \{\geq, \leq\}$, we consider two DTA $\mathcal{A}_=$ and $\mathcal{A}_\bowtie$. Clearly $\mathit{Paths}^{\mathcal{C}}(\mathcal{A}) = \mathit{Paths}^{\mathcal{C}}(\mathcal{A}_=) \cup \mathit{Paths}^{\mathcal{C}}(\mathcal{A}_{\bar{\bowtie}})$, where $\bar{\bowtie} => $ if $\bowtie = \geq$; $<$ otherwise. Clearly $\mathit{Paths}^{\mathcal{C}}(\mathcal{A})$ is measurable. ■

A.2 Proof of Lemma 7.15

Lemma 7.15 For any CTMC $\mathcal{C}$ and DTA$^{\Diamond}$ $\mathcal{A}$,

$$\mathit{Paths}^{\mathcal{C}}(\mathcal{A}) = \mathit{Paths}^{\mathcal{C}\otimes\mathcal{A}}(\Diamond\, \mathit{Loc}_F)\!\downharpoonright_1.$$

Proof: ($\Longrightarrow$) It is to prove that for any path $\rho \in \mathit{Paths}^{\mathcal{C}}(\mathcal{A})$, there exists a path $\tau \in \mathit{Paths}^{\mathcal{C}\otimes\mathcal{A}}(\Diamond\, \mathit{Loc}_F)$ such that $\tau\!\downharpoonright_1 = \rho$.

We assume w.l.o.g. that $\rho = s_0 \xrightarrow{t_0} s_1 \cdots s_{n-1} \xrightarrow{t_{n-1}} s_n \in \mathit{Paths}^{\mathcal{C}}$ is accepted by $\mathcal{A}$, i.e., $s_n \in Q_F$ and for $0 \leqslant i < n$, $\eta_0 \models \vec{0}$ and $\eta_i + t_i \models g_i$ and $\eta_{i+1} = (\eta_i + t_i)[X_i := 0]$, where η_i is the time valuation on entering state s_i. We can then construct a path $\theta \in \mathit{Paths}^{\mathcal{A}}$ from ρ such that $\theta = q_0 \xrightarrow{L(s_0),t_0} q_1 \cdots q_{n-1} \xrightarrow{L(s_{n-1}),t_{n-1}} q_n$, where s_i and q_i have the same entering clock valuation. From ρ and θ, we can construct the path

$$\tau = \langle s_0, q_0\rangle \xrightarrow{t_0} \langle s_1, q_1\rangle \cdots \langle s_{n-1}, q_{n-1}\rangle \xrightarrow{t_{n-1}} \langle s_n, q_n\rangle,$$

where $\langle s_n, q_n\rangle \in \mathit{Loc}_F$. It follows from the definition of an accepting path in a DTA$^{\omega}$ that $\tau \in \mathit{Paths}^{\mathcal{C}\otimes\mathcal{A}}(\Diamond\, \mathit{Loc}_F)$ and $\tau\!\downharpoonright_1 = \rho$.

($\Longleftarrow$) It is to prove that for any path $\tau \in \mathit{Paths}^{\mathcal{C}\otimes\mathcal{A}}(\Diamond\, \mathit{Loc}_F)$, $\tau\!\downharpoonright_1 \in \mathit{Paths}^{\mathcal{C}}(\mathcal{A})$.

We assume w.l.o.g. that path

$$\tau = \langle s_0, q_0\rangle \xrightarrow{t_0} \cdots \xrightarrow{t_{n-1}} \langle s_n, q_n\rangle \in \mathit{Paths}^{\mathcal{C}\otimes\mathcal{A}}(\Diamond\, \mathit{Loc}_F),$$

it holds that $\langle s_n, q_n\rangle \in \mathit{Loc}_F$ and for $0 \leqslant i < n$, $\eta_0 \models \vec{0}$ and $\eta_i + t_i \models g_i$ and $\eta_{i+1} = (\eta_i + t_i)[X_i := 0]$, where η_i is the time valuation on entering state $\langle s_i, q_i\rangle$. It then directly follows that $q_n \in Q_F$ and $\tau\!\downharpoonright_1 \in \mathit{Paths}^{\mathcal{C}}(\mathcal{A})$, given η_i the entering clock valuation of state s_i. ■

A.3 Proof of Theorem 7.16

Theorem 7.16 For any CTMC $\mathcal{C}$ and DTA$^{\Diamond}$ $\mathcal{A}$:

$$\Pr^{\mathcal{C}}\left(\mathit{Paths}^{\mathcal{C}}(\mathcal{A})\right) = \Pr^{\mathcal{C}\otimes\mathcal{A}}\left(\mathit{Paths}^{\mathcal{C}\otimes\mathcal{A}}(\Diamond \mathit{Loc}_F)\right).$$

Proof: According to Theorem 7.7, $\mathit{Paths}^{\mathcal{C}}(\mathcal{A})$ can be rewritten as the combination of cylinder sets of the form $C(s_0, I_0, \ldots, I_{n-1}, s_n)$ which are all accepted by DTA $\mathcal{A}$[1]. By Lemma 7.15, namely by path lifting, we can establish exactly the same combination of cylinder sets $C(\ell_0, I_0, \ldots, I_{n-1}, \ell_0)$ for $\mathit{Paths}^{\mathcal{C}\otimes\mathcal{A}}(\Diamond \mathit{Loc}_F)$, where $s_i = \ell_i\!\downharpoonright_1$. It then suffices to show that for each cylinder set $C(s_0, I_0, \ldots, I_{n-1}, s_n)$ which is accepted by $\mathcal{A}$, $\Pr^{\mathcal{C}}$ and $\Pr^{\mathcal{C}\otimes\mathcal{A}}$ yield the same probabilities. Note that a cylinder set C is accepted by a DTA $\mathcal{A}$, if each path that C generates can be accepted by $\mathcal{A}$.

[1]Note that this means each path in the cylinder set is accepted by $\mathcal{A}$.

For the measure $\Pr^{\mathcal{C}}$, according to Eq. (2.1) (page 12),

$$\Pr^{\mathcal{C}}\big(C(s_0, I_0, \ldots, I_{n-1}, s_n)\big) = \prod_{0 \leqslant i < n} \int_{I_i} \mathbf{P}(s_i, s_{i+1}) \cdot E(s_i) \cdot e^{-E(s_i)\tau} d\tau.$$

For the measure $\Pr_{\vec{0}}^{\mathcal{C}\otimes\mathcal{A}}$, according to Section 7.2.1, it is given by $\mathbb{P}_0^{\mathcal{C}\otimes\mathcal{A}}(\vec{0})$ where $\mathbb{P}_n^{\mathcal{C}\otimes\mathcal{A}}(\eta) = 1$ for any clock valuation η and

$$\mathbb{P}_i^{\mathcal{C}\otimes\mathcal{A}}(\eta_i) = \int_{I_i} \mathbf{1}_{g_i}(\eta_i + \tau_i) \cdot p_i \cdot E(\ell_i) \cdot e^{-E(\ell_i)\tau_i} \cdot \mathbb{P}_{i+1}^{\mathcal{C}\otimes\mathcal{A}}(\eta_{i+1})\, d\tau_i,$$

where $\eta_{i+1} = (\eta_i + \tau_i)[X_i := 0]$ and $\mathbf{1}_{g_i}(\eta_i + \tau_i) = 1$, if $\eta_i + \tau_i \models g_i$; 0, otherwise.

We will show, by induction, that $\mathbb{P}_i^{\mathcal{C}\otimes\mathcal{A}}(\eta_i)$ is a constant, i.e., is independent of η_i, if the cylinder set $C(\ell_0, I_0, \ldots, I_{n-1}, \ell_n)$ is accepted by $\mathcal{C} \otimes \mathcal{A}$. Firstly let us note that for $C(\ell_0, I_0, \ldots, I_{n-1}, \ell_n)$, there must exist some sequence of transitions

$$\ell_0 \xmapsto[p_0]{g_0, X_0} \ell_1 \cdots \ell_{n-1} \xmapsto[p_{n-1}]{g_{n-1}, X_{n-1}} \ell_n$$

with $\eta_0 = \vec{0}$ and $\forall t_i \in I_i$ with $0 \leqslant i < n$, $\eta_i + t_i \models g_i$ and $\eta_{i+1} := (\eta_i + t_i)[X_i := 0]$. Moreover, according to Def. 7.11, we have:

$$p_i = \mathbf{P}(s_i, s_{i+1}) \qquad \text{and} \qquad E(\ell_i) = E(s_i). \tag{A.1}$$

We apply a backward induction on n down to 0. The base case is trivial since $\mathbb{P}_n^{\mathcal{C}\otimes\mathcal{A}}(\eta) = 1$. By I.H., $\mathbb{P}_{i+1}^{\mathcal{C}\otimes\mathcal{A}}(\eta)$ is a constant. For the induction step, consider $i < n$. For any $\tau_i \in I_i$, since $\eta_i + \tau_i \models g_i$, $\mathbf{1}_{g_i}(\eta_i + \tau_i) = 1$, it follows that

$$\begin{aligned}
\mathbb{P}_i^{\mathcal{C}\otimes\mathcal{A}}(\eta_i) &= \int_{I_i} \mathbf{1}_{g_i}(\eta_i + \tau_i) \cdot p_i \cdot E(\ell_i) \cdot e^{-E(\ell_i)\tau_i} \cdot \mathbb{P}_{i+1}^{\mathcal{C}\otimes\mathcal{A}}(\eta_{i+1})\, d\tau_i \\
&\overset{\text{I.H.}}{=} \int_{I_i} p_i \cdot E(\ell_i) \cdot e^{-E(\ell_i)\tau_i} d\tau_i \cdot \mathbb{P}_{i+1}^{\mathcal{C}\otimes\mathcal{A}}(\eta_{i+1}) \\
&\overset{\text{Eq.(A.1)}}{=} \int_{I_i} \mathbf{P}(s_i, s_{i+1}) \cdot E(s_i) \cdot e^{-E(s_i)\tau_i} d\tau_i \cdot \mathbb{P}_{i+1}^{\mathcal{C}\otimes\mathcal{A}}(\eta_{i+1}).
\end{aligned}$$

Clearly, this is a constant. It is thus easy to see that

$$\Pr_{\vec{0}}^{\mathcal{C}\otimes\mathcal{A}}\big(C(\ell_0, I_0, \ldots, I_{n-1}, \ell_n)\big) := \mathbb{P}_0^{\mathcal{C}\otimes\mathcal{A}}(\vec{0}) = \prod_{0 \leqslant i < n} \int_{I_i} \mathbf{P}(s_i, s_{i+1}) \cdot E(s_i) \cdot e^{-E(s_i)\tau} d\tau,$$

which completes the proof. ■

A.4 Proof of Theorem 7.23

Theorem 7.23 For any CTMC $\mathcal{C}$ and DTA$^\diamondsuit$ $\mathcal{A}$, $\Pr_{\vec{0}}^{\mathcal{C}\otimes\mathcal{A}}\big(Paths^{\mathcal{C}\otimes\mathcal{A}}(\diamondsuit\, Loc_F)\big)$ is the least solution of $Prob_{v_0}^{\mathcal{D}}(\cdot)$, where $\mathcal{D}$ is the embedded DTMP of $\mathcal{C}\otimes\mathcal{A}$.
Proof: We can express the set of all finite paths in $\mathcal{C}\otimes\mathcal{A}$ ending in some accepting location $\ell_n \in Loc_F$ for $n \in \mathbb{N}$ as the union over all location sequences i.e.,

$$\begin{aligned}\Pi^{\mathcal{C}\otimes\mathcal{A}} &= \bigcup_{n\in\mathbb{N}} \bigcup_{(\ell_0,\ldots,\ell_n)\in Loc^{n+1}} C(\ell_0, I_0, \ldots, I_{n-1}, \ell_n)\\ &= Paths^{\mathcal{C}\otimes\mathcal{A}}(\diamondsuit\, Loc_F) \cup \overline{Paths^{\mathcal{C}\otimes\mathcal{A}}(\diamondsuit\, Loc_F)}.\end{aligned}$$

where $C(\ell_0, I_0, \ldots, I_{n-1}, \ell_n)$ is a cylinder set, $I_i = [0,\infty[$ and $\overline{Paths^{\mathcal{C}\otimes\mathcal{A}}(\diamondsuit\, Loc_F)}{\downarrow_1}$ are the set of paths which are not accepted by the DTA $\mathcal{A}$. Notice that we can easily extend the measure $\Pr_{\vec{0}}^{\mathcal{C}\otimes\mathcal{A}}$ to $\Pi^{\mathcal{C}\otimes\mathcal{A}}$ such that

$$\Pr_{\vec{0}}^{\mathcal{C}\otimes\mathcal{A}}\left(\Pi^{\mathcal{C}\otimes\mathcal{A}}\right) = \Pr_{\vec{0}}^{\mathcal{C}\otimes\mathcal{A}}\big(Paths^{\mathcal{C}\otimes\mathcal{A}}(\diamondsuit\, Loc_F)\big).$$

This means that in order to prove the theorem we need to show that

$$\Pr_{\vec{0}}^{\mathcal{C}\otimes\mathcal{A}}\left(\Pi^{\mathcal{C}\otimes\mathcal{A}}\right) = Prob_{v_0}^{\mathcal{D}}(\hat{\vec{0}}), \tag{A.2}$$

where $Prob_{v_0}^{\mathcal{D}}(\hat{\vec{0}})$ is the short form of $Prob^{\mathcal{D}}\big((v_0,\hat{\vec{0}}),(V_F,\cdot)\big)$, i.e., the reachability probability from state $(v_0,\hat{\vec{0}})$ to $(V_F,\cdot)$. Note that for better readability, we indicate clock valuations in $\mathcal{D}$ by adding a "^".

Eq. (A.2) is to be shown on cylinder sets. Note that each cylinder set $C(\ell_0, I_0, \ldots, I_{n-1}, \ell_n) \subseteq \Pi^{\mathcal{C}\otimes\mathcal{A}}$ (C_n for short) induces a region graph $\mathcal{G}(C_n) = (V, v_0, V_F, \Lambda, \hookrightarrow)$, where its underlying PDP and embedded DTMP is $\mathcal{Z}(C_n)$ and $\mathcal{D}(C_n)$, respectively. To prove Eq. (A.2), it suffices to show that for each C_n,

$$\Pr_{\vec{0}}^{\mathcal{C}\otimes\mathcal{A}}(C_n) = Prob_{v_0}^{\mathcal{D}(C_n)}(\hat{\vec{0}}),$$

since $\Pi^{\mathcal{C}\otimes\mathcal{A}} = \bigcup_{n\in\mathbb{N}}\bigcup_{(\ell_0,\ldots,\ell_n)\in Loc^{n+1}} C_n$ and $\mathcal{D} = \bigcup_{n\in\mathbb{N}}\bigcup_{(\ell_0,\ldots,\ell_n)\in Loc^{n+1}} \mathcal{D}(C_n)$.

We will prove it by induction on the length n of the cylinder set $C_n \subseteq \Pi^{\mathcal{C}\otimes\mathcal{A}}$.

- By B.C. of $n = 0$, i.e. $C_0 = C(\ell_i)$ and $\ell_i \in Loc_F$, it holds that $\Pr_{\eta_i}^{\mathcal{C}\otimes\mathcal{A}}(C_0) = 1$; while in the embedded DTMP $\mathcal{D}(C_0)$, since the initial vertex of $\mathcal{G}(C_0)$ is $v_0 = (\ell_i, \Theta_0)$, where $\eta_i \in \Theta_0$ and v_0 is consequently the initial location of $\mathcal{Z}(C_0)$ as well as $\mathcal{D}(C_0)$ which is accepting, $Prob_{v_0}^{\mathcal{D}(C_0)}(\hat{\eta}_i) = 1$. Note $\ell_i \in Loc$ is not necessarily the initial location ℓ_0.

- By I.H., we have that for $n = k-1$, $\Pr_{\eta_{i+1}}^{\mathcal{C}\otimes\mathcal{A}}(C_{k-1}) = Prob_{v_{i+1}}^{\mathcal{D}(C_{k-1})}(\hat{\eta}_{i+1})$, where $C_{k-1} = C(\ell_{i+1}, I_{i+1}, \ldots, I_{i+k-1}, \ell_{i+k})$ and $\ell_{i+k} \in Loc_F$. Note $\ell_{i+1} \in Loc$ is not necessarily the initial location ℓ_0.

- For $n = k$, let $C_k = C(\ell_i, I_i, \ell_{i+1}, I_{i+1}, \ldots, I_{i+k-1}, \ell_{i+k})$. As a result, there exists a transition $\ell_i \xmapsto[p_i]{g_i, X_i} \ell_{i+1}$ where $\eta_i + \tau_i \models g_i$ for every $\tau_i \in]t_1, t_2[$. $t_1, t_2 \in \mathbb{Q}_{\geqslant 0} \cup \{\infty\}$ can be obtained from g_i, such that $\tau_j \in]t_1, t_2[$ iff $\eta_i + \tau_j \models g_i$. According to the semantics of MTA we have
$$\mathrm{Pr}^{\mathcal{C}\otimes\mathcal{A}}_{\eta_i}(C_k) = \int_{t_1}^{t_2} p_i \cdot E(\ell_i) \cdot e^{-E(\ell_i)\tau_i} \cdot \mathrm{Pr}^{\mathcal{C}\otimes\mathcal{A}}_{\eta_{i+1}}(C_{k-1})\, d\tau_i, \tag{A.3}$$
where $\eta_{i+1} = (\eta_i + \tau_i)[X_i := 0]$.

$$\begin{tikzcd}
{} \arrow[r] & \boxed{\begin{array}{c} v_0^i{=}(\ell_i,\Theta_0) \\ \flat(v_0^i,\hat{\eta}_0^i){\leqslant}1 \end{array}} \arrow[r, "\delta"] & \cdots \arrow[r, "\delta"] & \boxed{\begin{array}{c} v_{m-1}^i{=}(\ell_i,\Theta_{m-1}) \\ \flat(v_{m-1}^i,\hat{\eta}_{m-1}^i){=}1 \end{array}} \arrow[r, "\delta"] & \boxed{\begin{array}{c} v_m^i{=}(\ell_i,\Theta_m) \\ \flat(v_m^i,\hat{\eta}_m^i){=}1 \end{array}} \arrow[r, "\delta"] \arrow[d, "p_i"] & \cdots \arrow[r, "\delta"] & \boxed{\begin{array}{c} v_{m'}^i{=}(\ell_i,\Theta_{m'}) \\ \flat(v_{m'}^i,\hat{\eta}_{m'}^i){=}1 \end{array}} \arrow[d, "p_i"] \\
 & & & & \boxed{\begin{array}{c} v_m^{i+1}{=}(\ell_{i+1},\Theta_m) \\ \flat(v_m^{i+1},\hat{\eta}_m^{i+1}){\leqslant}1 \end{array}} & & \boxed{\begin{array}{c} v_{m'}^{i+1}{=}(\ell_{i+1},\Theta_{m'}) \\ \flat(v_{m'}^{i+1},\hat{\eta}_{m'}^{i+1}){\leqslant}1 \end{array}}
\end{tikzcd}$$

Now we deal with the inductive step for $\mathcal{D}(C_k)$. Let us assume that C_k induces the region graph $\mathcal{G}(C_k)$ whose subgraph corresponding to transition $\ell_i \xmapsto[p_i]{g_i, X_i} \ell_{i+1}$ is depicted in the figure above. For simplicity we consider that location ℓ_i induces the vertices $\{v_j^i = (\ell_i, \Theta_j) \mid 0 \leqslant j \leqslant m'\}$ and location ℓ_{i+1} induces the vertices $\{v_j^{i+1} = (\ell_{i+1}, \Theta_j) \mid m \leqslant j \leqslant m'\}$, respectively. Note that for Markovian transitions, the regions stay the same. We denote $\hat{\eta}_j^i$ (resp. $\hat{\eta}_j^{i+1}$) as the entering clock valuation on vertex v_j^i (resp. $\hat{\eta}_j^{i+1}$), for j the indices of the regions. For any $\hat{\eta} \in \bigcup_{j=0}^{m-1} \Theta_j \cup \bigcup_{j>m'} \Theta_j$, $\hat{\eta} \not\models g_i$; or more specifically,
$$t_1 = \sum_{j=0}^{m-1} \flat(v_j^i, \hat{\eta}_j^i) \qquad \text{and} \qquad t_2 = \sum_{j=0}^{m'} \flat(v_j^i, \hat{\eta}_j^i).$$

Recall that $\hat{\eta}_i$ (in the I.H.) is the clock valuation to first hit a region with ℓ_i and $\hat{\eta}_i$. Given the fact that from v_0^i the process can only execute a delay transition before time t_1, it holds that
$$\begin{aligned}
\mathit{Prob}^{\mathcal{D}(C_k)}_{v_0^i}(\hat{\eta}_i) &= e^{-t_1\Lambda(v_i)} \cdot \mathit{Prob}^{\mathcal{D}(C_k)}_{v_m^i}(\hat{\eta}_m^i) \\
\mathit{Prob}^{\mathcal{D}(C_k)}_{v_m^i}(\hat{\eta}_m^i) &= \mathit{Prob}^{\mathcal{D}(C_k)}_{v_m^i,\delta}(\hat{\eta}_m^i) + \mathit{Prob}^{\mathcal{D}(C_k)}_{v_m^i, v_m^{i+1}}(\hat{\eta}_{i+1}).
\end{aligned}$$
Therefore, we get by substitution of variables:
$$\begin{aligned}
\mathit{Prob}^{\mathcal{D}}_{v_0^i}(\hat{\eta}_i) &= e^{-t_1\Lambda(v_i)} \cdot \mathit{Prob}^{\mathcal{D}(C_k)}_{v_m^i,\delta}(\hat{\eta}_m^i) + e^{-t_1\Lambda(v_i)} \cdot \mathit{Prob}^{\mathcal{D}(C_k)}_{v_m^i, v_m^{i+1}}(\hat{\eta}_{i+1}) \\
&= e^{-t_1\Lambda(v_i)} \cdot \mathit{Prob}^{\mathcal{D}(C_k)}_{v_m^i,\delta}(\hat{\eta}_m^i) \\
&+ e^{-t_1\Lambda(v_i)} \cdot \int_0^{\flat(v_m^i, \hat{\eta}_m^i)} p_i \Lambda(v_i) e^{-\Lambda(v_i)\tau} \cdot \mathit{Prob}^{\mathcal{D}(C_{k-1})}_{v_m^{i+1}}\big((\hat{\eta}_m^i + \tau)[X_i := 0]\big) d\tau \\
&= e^{-t_1\Lambda(v_i)} \cdot \mathit{Prob}^{\mathcal{D}(C_k)}_{v_m^i,\delta}(\hat{\eta}_m^i) \\
&+ \int_{t_1}^{t_1 + \flat(v_m^i, \hat{\eta}_m^i)} p_i \Lambda(v_i) e^{-\Lambda(v_i)\tau} \cdot \mathit{Prob}^{\mathcal{D}(C_{k-1})}_{v_m^{i+1}}\big((\hat{\eta}_m^i + \tau - t_1)[X_i := 0]\big) d\tau.
\end{aligned}$$

Evaluating each term $Prob^{\mathcal{D}(C_k)}_{v^i_m,\delta}(\hat{\eta}^i_m)$ we get the following sum of integrals:

$$\begin{aligned} Prob^{\mathcal{D}(C_k)}_{v^i_0}(\hat{\eta}_i) &= \sum_{j=0}^{m'-m} \int_{t_1+\sum_{h=0}^{j-1}\flat(v^i_{m+h},\hat{\eta}^i_{m+h})}^{t_1+\sum_{h=0}^{j}\flat(v^i_{m+h},\hat{\eta}^i_{m+h})} p_i\Lambda(v_i)e^{-\Lambda(v_i)\tau} \\ &\quad \cdot \quad Prob^{\mathcal{D}(C_{k-1})}_{v^{i+1}_{m+j}}\big((\hat{\eta}^i_{m+j}+\tau-t_1-\sum_{h=0}^{j-1}\flat(v^i_{m+h},\hat{\eta}^i_{m+h}))[X_i{:=}0]\big)d\tau. \end{aligned}$$

Now we define the function $F^{\mathcal{D}(C_{k-1})}(t) : [t_1, t_2] \to [0,1]$, such that when $t \in [t_1 + \sum_{h=0}^{j-1}\flat(v^i_{m+h},\hat{\eta}^i_{m+h}), t_1+\sum_{h=0}^{j}\flat(v^i_{m+h},\hat{\eta}^i_{m+h})]$ for $j \leqslant m'-m$ then $F^{\mathcal{D}(C_{k-1})}(t) = Prob^{\mathcal{D}(C_{k-1})}_{v^{i+1}_{m+j}}\big((\hat{\eta}^i_{m+j} + t - t_1 - \sum_{h=0}^{j-1}\flat(v^i_{m+h},\hat{\eta}^i_{m+h}))[X_i := 0]\big)$. Using $F^{\mathcal{D}(C_{k-1})}(t)$ we can rewrite $Prob^{\mathcal{D}(C_k)}_{v^i_0}(\hat{\eta}_i)$ to an equivalent form as:

$$\begin{aligned} Prob^{\mathcal{D}(C_k)}_{v^i_0}(\hat{\eta}_i) &= \sum_{j=0}^{m'-m} \int_{t_1+\sum_{h=0}^{j-1}\flat(v^i_{m+h},\hat{\eta}^i_{m+h})}^{t_1+\sum_{h=0}^{j}\flat(v^i_{m+h},\hat{\eta}^i_{m+h})} p_i\Lambda(v_i)e^{-\Lambda(v_i)\tau}F^{\mathcal{D}(C_{k-1})}(\tau)d\tau \\ &= \int_{t_1}^{t_2} p_i\Lambda(v_i)e^{-\Lambda(v_i)\tau}F^{\mathcal{D}(C_{k-1})}(\tau)d\tau. \end{aligned}$$

By the I.H. we now have that for every $t \in [t_1 + \sum_{h=0}^{j-1}\flat(v^i_{m+h},\hat{\eta}^i_{m+h}), t_1 + \sum_{h=0}^{j}\flat(v^i_{m+h},\hat{\eta}^i_{m+h})]$ for $j \leqslant m'-m$ we have that:

$$\mathrm{Pr}^{\mathcal{C}\otimes\mathcal{A}}_{\eta_{i+1}}(C_{k-1}) = Prob^{\mathcal{D}(C_{k-1})}_{v^{i+1}_{m+j}}\big((\hat{\eta}^i_{m+j}+t-t_1-\sum_{h=0}^{j-1}\flat(v^i_{m+h},\hat{\eta}^i_{m+h}))[X_i{:=}0]\big) = F^{\mathcal{D}(C_{k-1})}(t),$$

where $\eta_{i+1} = (\eta_i + t)[X_i := 0]$ and $\hat{\eta}^i_{m+j} = \hat{\eta}_i + t_1 + \sum_{h=0}^{j-1}\flat(v^i_{m+h},\hat{\eta}^i_{m+h})$. This shows that $\mathrm{Pr}^{\mathcal{C}\otimes\mathcal{A}}_{\eta_i}(C_k) = Prob^{\mathcal{D}(C_k)}_{v_i}(\hat{\eta}_i)$ which proves the theorem. ■

A.5 Proof of Proposition 7.26

Proposition 7.26 Given a CTMC $\mathcal{C}$, an augmented DTA$^\Diamond$ $\mathcal{A}[t_f]$ and the underlying PDP $\mathcal{Z}(\mathcal{C}\otimes\mathcal{A}[t_f]) = (V, \mathcal{X}, Inv, \phi, \Lambda, \mu)$, $\mathrm{Pr}^{\mathcal{C}}\left(Paths^{\mathcal{C}}(\mathcal{A}[t_f])\right) = \hbar_{v_0}(0,\vec{0})$ (which is the probability to reach the final states in $\mathcal{Z}$ starting from initial state $(v_0, \vec{0}_{\mathcal{X}\cup\{y\}}$[1]$)$) is the unique solution of the following system of PDEs:

$$\frac{\partial\hbar_v(y,\eta)}{\partial y} + \sum_i \frac{\partial\hbar_v(y,\eta)}{\partial\eta^{(i)}} + \Lambda(v)\cdot \sum_{v\overset{p,X}{\hookrightarrow}v'} p\cdot(\hbar_{v'}(y,\eta[X:=0]) - \hbar_v(y,\eta)) = 0,$$

[1]denoting the valuation η with $\eta(x) = 0$ for $x \in \mathcal{X}\cup\{y\}$

where $v \in V \setminus V_F$, $\eta \in \mathit{Inv}(v)$, $\eta^{(i)}$ is the i'th clock variable and $y \in [0, t_f[$. For every $\eta \in \partial \mathit{Inv}(v)$ and transition $v \overset{\delta}{\hookrightarrow} v'$, the boundary conditions take the form: $\hbar_v(y,\eta) = \hbar_{v'}(y,\eta)$. For every vertex $v \in V_F$, $\eta \in \mathit{Inv}(v)$ and $y \in [0, t_f[$, we have the following PDE:

$$\frac{\partial \hbar_v(y,\eta)}{\partial y} + \sum_i \frac{\partial \hbar_v(y,\eta)}{\partial \eta^{(i)}} + 1 = 0.$$

The final boundary conditions are that for every vertex $v \in V$ and $\eta \in \mathit{Inv}(v) \cup \partial \mathit{Inv}(v)$, $\hbar_v(t_f,\eta) = 0$.

Proof: For any set of clocks $\mathcal{X}$ (n clocks) of the PDP $\mathcal{Z} = (Z, \mathcal{X}, \mathit{Inv}, \phi, \Lambda, \mu)$ we define a system of ODEs:

$$\frac{d\eta(y)}{dy} = \vec{1}, \eta(y_0) = \eta_0 \in \mathbb{R}^n_{\geqslant 0}, \tag{A.4}$$

which describe the evolution of clock values $\eta(y)$ at time y given the initial value η_0 of all clocks at time y_0. Notice that contrary to our DTA notation, Eq. (A.4) describes a system of ODEs where $\eta(y)$ is a vector of clock valuations at time y and $\frac{d\eta^{(i)}(y)}{dy}$ gives the timed evolution of clock $\eta^{(i)}$. Given a continuous differentiable functional $f : Z \times \mathbb{R}^n_{\geqslant 0} \to \mathbb{R}_{\geqslant 0}$, for every $z \in Z$ let:

$$\frac{df(z,\eta(y))}{dy} = \sum_{i=1}^{n} \frac{\partial f(z,\eta(y))}{\partial \eta^{(i)}} \cdot \frac{d\eta^{(i)}(y)}{dy} \overset{\text{Eq.(A.4)}}{=} \sum_{i=1}^{n} \frac{\partial f(z,\eta(y))}{\partial \eta^{(i)}}.$$

For notation simplicity we define the vector field from Eq. (A.5) as the operator Ξ which acts on functional $f(z,\eta(y))$ i.e., $\Xi f(z,\eta(y)) = \sum_{i=1}^{n} \frac{\partial f(z,\eta(y))}{\partial \eta^{(i)}}$. We also define the equivalent notation $\Xi f(\xi)$ for the state $\xi = (z, \eta(y))$ and any $y \in \mathbb{R}_{\geqslant 0}$.

We define the value of $\Pr^{\mathcal{C}}\left(\mathit{Paths}^{\mathcal{C}}(\mathcal{A})\right)$ as the expectation $\hbar(0,\xi_0)$ on PDP $\mathcal{Z}$ as follows:

$$\hbar(0,\xi_0) = \mathbb{E}\left[\int_0^{t_f} \mathbf{1}_{\mathcal{Z}}(X_\tau)d\tau \mid X_0 = \xi_0\right] = \mathbb{E}_{(0,\xi_0)}\left[\int_0^{t_f} \mathbf{1}_{\mathcal{Z}}(X_\tau)d\tau\right],$$

where the initial starting time is 0 the starting state is $\xi_0 = (z_0, \vec{0})$, X_τ is the underlying stochastic process of $\mathcal{Z}$ defined on the state space $\mathbb{S}$ and $\mathbf{1}_{\mathcal{Z}}(X_\tau) = 1$ when $X_\tau \in \{(z,\eta(\tau)) \mid z \in V_F, \eta(\tau) \in \mathit{Inv}(z)\}$, $\mathbf{1}_{\mathcal{Z}}(X_\tau) = 0$, otherwise. Notice that we can also define the expectation in Eq. (A.5) for any starting time $y < t_f$ and state ξ as $\mathbb{E}_{(y,\xi)}\left[\int_y^{t_f} \mathbf{1}_{\mathcal{Z}}(X_\tau)d\tau\right]$.

We can obtain the expectation $\hbar(0,\xi_0)$ by following the construction in [Dav93]. For this we form the new state space $\hat{\mathbb{S}} = ([0,t_f] \times \mathbb{S}) \cup \{\Delta\}$ where Δ is the sink state and the boundary is $\partial\hat{\mathbb{S}} := ([0,t_f] \times \partial\mathbb{S}) \cup (\{t_f\} \times \mathbb{S})$. We define the following functions: $\hat{\Lambda}(y,\xi) = \Lambda(\xi)$, $\hat{\mu}((y,\xi), \{y\} \times A) = \mu(\xi, A)$ and $\hat{\mu}((t_f,\xi),\{\Delta\}) = 1$ for $y \in [0,t_f[$, $A \subseteq \mathbb{S}$ and $\xi \in \mathbb{S}$.

Given the construction we obtain an equivalent form for the expectation (A.5) i.e.,:

$$\hbar(0,\xi_0) = \mathbb{E}_{(0,\xi_0)}\left[\int_0^\infty \vec{\hat{1}}_{\mathcal{Z}}(\tau, X_\tau)d\tau\right], \tag{A.5}$$

where $\vec{\hat{1}}_{\mathcal{Z}} : \hat{\mathbb{S}} \to \{0,1\}$, $\vec{\hat{1}}_{\mathcal{Z}}(\tau, X_\tau) = 1$ when $X_\tau \in \{(z,\eta(\tau)) \mid z \in V_F, \eta(\tau) \in Inv(z)\}$ and $\tau \in [0, t_f[$, $\vec{\hat{1}}_{\mathcal{Z}}(\tau, X_\tau) = 0$, otherwise. We also define $\vec{\hat{1}}_{\mathcal{Z}}(\Delta)$ to be zero. Notice that we introduce the sink state Δ in order to ensure that $\lim_{y\to\infty} \mathbb{E}_{(0,\xi)}\hbar(y, X_y) = 0$, which is a crucial condition in order to obtain a unique value for the expectation $\hbar(0,\xi_0)$.

For the expectation (A.5) [Dav93] defines the following integro-differential equations (for any $y \in [0, t_f[$):

$$\mathcal{U}\hbar(y,\xi) = \Xi\hbar(y,\xi) + \hat{\Lambda}(y,\xi)\cdot\int_{\mathbb{S}}\left(\hbar(y,\xi') - \hbar(y,\xi)\right)\hat{\mu}((y,\xi),(y,d\xi')), \xi \in \mathbb{S} \tag{A.6}$$

$$\hbar(y,\xi) = \int_{\mathbb{S}}\hbar(y,\xi')\hat{\mu}((y,\xi),(y,d\xi')), \xi \in \partial\mathbb{S} \tag{A.7}$$

$$\mathcal{U}\hbar(y,\xi) + \vec{\hat{1}}_{\mathcal{Z}}(y,\xi) = 0, \xi \in \mathbb{S} \tag{A.8}$$

Equation (A.6) denotes the generator of the stochastic process X_y and Eq. (A.7) states the boundary conditions for Eq. (A.8). We can rewrite the integro-differential equations (A.6),(A.7) and (A.8) into a system of PDEs with boundary conditions given the fact that the measure $\hat{\mu}$ is not uniform. For each vertex $v \notin V_F$, $\eta \in Inv(v)$ and $y \in [0, t_f[$ of the region graph $\mathcal{G}$ we write the PDE as follows (here we define $\hbar_v(y,\eta) := \hbar(y,\xi)$ for $\xi = (v,\eta)$):

$$\frac{\partial\hbar_v(y,\eta)}{\partial y} + \sum_i \frac{\partial\hbar_v(y,\eta)}{\partial\eta^{(i)}} + \Lambda(v)\sum_{v \overset{p,X}{\hookrightarrow} v'} p\cdot\left(\hbar_{v'}(y,\eta[X:=0]) - \hbar_v(y,\eta)\right) = 0,$$

Notice that for any edge $v \overset{p,X}{\hookrightarrow} v'$ in the region graph $\mathcal{G}$, $\hat{\mu}((y,(v,\eta)),(y,(v',\eta'))) = p$. For every $\eta \in \partial Inv(v)$ and transition $v \overset{\delta}{\hookrightarrow} v'$ the boundary conditions take the form: $\hbar_v(y,\eta) = \hbar_{v'}(y,\eta)$. For every vertex $v \in V_F$, $\eta \in Inv(v)$ and $y \in [0, t_f[$ we get:

$$\frac{\partial\hbar_v(y,\eta)}{\partial y} + \sum_i \frac{\partial\hbar_v(y,\eta)}{\partial\eta^{(i)}} + 1 = 0$$

Notice that all final states are made absorbing. The final boundary conditions are that for every vertex $v \in Z$ and $\eta \in Inv(v) \cup \partial Inv(v)$, $\hbar_v(t_f,\eta)$=0. ■

A.6 Proof of Proposition 7.28

Proposition 7.28 Given a CTMC $\mathcal{C}$ with initial distribution α, rate matrix $\mathbf{P}\cdot\mathbf{E}$ and $\mathbf{\Pi}(t)$, $\vec{\wp}(t)$ satisfies the following two equations:

$$\vec{\wp}(t) = \alpha\cdot\mathbf{\Pi}(t), \tag{A.9}$$

$$\frac{d\vec{\wp}(t)}{dt} = \vec{\wp}(t)\cdot\mathbf{Q}, \tag{A.10}$$

where $\mathbf{Q} = \mathbf{P}{\cdot}\mathbf{E} - \mathbf{E}$ is the infinitesimal generator.
Proof: The transition probability matrix $\mathbf{\Pi}(t)$ for a CTMC $\mathcal{C}$ with state space S is denoted by the following system of integral equations:

$$\mathbf{\Pi}(t) = \int_0^t \mathbf{M}(\tau)\mathbf{\Pi}(t-\tau)d\tau + \mathbf{D}(t), \tag{A.11}$$

where $\mathbf{M}(\tau) = \mathbf{P}{\cdot}\mathbf{E}{\cdot}\mathbf{D}(\tau)$. Now we define for the CTMC $\mathcal{C}$ a stochastic process $X(t)$. The probability $\Pr(X(t+\Delta t) = s_j)$ to be in state s_j at time $t+\Delta t$ can be defined as:

$$\Pr(X(t+\Delta t) = s_j) = \sum_{s_i \in S} \Pr(X(t) = s_i) \cdot \Pr(X(t+\Delta t) = s_j | X(t) = s_i)$$

We can define $\Pr(X(t+\Delta t) = s_j)$ in the vector form as follows:

$$\vec{\wp}(t+\Delta t) = \vec{\wp}(t)\mathbf{\Phi}(t, t+\Delta t),$$

where $\vec{\wp}(t) = [\Pr(X(t) = s_1), \ldots, \Pr(X(t) = s_n)]$ and $\mathbf{\Phi}(t, t+\Delta t)[i,j] = \Pr(X(t+\Delta t) = s_j | X(t) = s_i)$.

As the stochastic process $X(t)$ is time-homogeneous we have that

$$\Pr(X(t+\Delta t) = s_j | X(t) = s_i) = \Pr(X(\Delta t) = s_j | X(0) = s_i),$$

which means that $\mathbf{\Phi}(t, t+\Delta t) = \mathbf{\Phi}(0, \Delta t)$. As $\Pr(X(\Delta t) = s_j | X(0) = s_i)$ denotes the transition probability to go from state s_i to state s_j at time Δt we have that $\mathbf{\Phi}(0, \Delta t) = \mathbf{\Pi}(\Delta t)$, which results in the equation:

$$\vec{\wp}(t+\Delta t) = \vec{\wp}(t)\mathbf{\Pi}(\Delta t). \tag{A.12}$$

Now we transform Eq. (A.12) as follows:

$$\begin{aligned}
& \vec{\wp}(t+\Delta t) = \vec{\wp}(t)\mathbf{\Pi}(\Delta t) \\
\Longrightarrow \quad & \vec{\wp}(t+\Delta t) - \vec{\wp}(t) = \vec{\wp}(t)\mathbf{\Pi}(\Delta t) - \vec{\wp}(t) \\
\Longrightarrow \quad & \vec{\wp}(t+\Delta t) - \vec{\wp}(t) = \vec{\wp}(t)(\mathbf{\Pi}(\Delta t) - \mathbf{I}) \\
\Longrightarrow \quad & \frac{d\vec{\wp}(t)}{dt} = \lim_{\Delta t \to 0} \frac{\vec{\wp}(t+\Delta t) - \vec{\wp}(t)}{\Delta t} = \vec{\wp}(t) \lim_{\Delta t \to 0} \frac{\mathbf{\Pi}(\Delta t) - \mathbf{I}}{\Delta t}.
\end{aligned}$$

Now it is to compute $\lim_{\Delta t \to 0} \frac{\mathbf{\Pi}(\Delta t) - \mathbf{I}}{\Delta t}$. For this we rewrite the right hand limit as:

$$\lim_{\Delta t \to 0} \frac{1}{\Delta t} \int_0^{\Delta t} \mathbf{M}(\tau)\mathbf{\Pi}(\Delta t - \tau)d\tau + \lim_{\Delta t \to 0} \frac{1}{\Delta t} \left(\mathbf{D}(\Delta t) - \mathbf{I}\right).$$

The $\lim_{\Delta t \to 0} \frac{1}{\Delta t} \int_0^{\Delta t} \mathbf{M}(\tau)\mathbf{\Pi}(\Delta t - \tau)d\tau$ is of the type $\frac{0}{0}$, which means we have to use l'Hospital rule:

$$\begin{aligned}
\frac{d(\Delta t)}{d\Delta t} &= 1, \\
\frac{d}{d\Delta t}\left(\int_0^{\Delta t} \mathbf{M}(\tau)\mathbf{\Pi}(\Delta t - \tau)d\tau\right) &= \mathbf{M}(\Delta t)\mathbf{\Pi}(0) + \int_0^{\Delta t} \mathbf{M}(\tau)\frac{\partial}{\partial \Delta t}\mathbf{\Pi}(\Delta t - \tau)d\tau.
\end{aligned}$$

Notice that $\mathbf{\Pi}(0) = \mathbf{I}$ and we obtain:

$$\begin{aligned}
&\lim_{\Delta t \to 0} \frac{1}{\Delta t} \int_0^{\Delta t} \mathbf{M}(\tau)\mathbf{\Pi}(\Delta t - \tau) d\tau \\
= \; &\lim_{\Delta t \to 0} \left(\mathbf{M}(\Delta t)\mathbf{\Pi}(0) + \int_0^{\Delta t} \mathbf{M}(\tau) \frac{\partial}{\partial \Delta t} \mathbf{\Pi}(\Delta t - \tau) d\tau \right) = \mathbf{M}(0)\mathbf{\Pi}(0) = \mathbf{P}{\cdot}\mathbf{E}.
\end{aligned}$$

The $\lim_{\Delta t \to 0} \frac{1}{\Delta t} (\mathbf{D}(\Delta t) - \mathbf{I})$ is of the type $\frac{0}{0}$, which means the use of l'Hospital rule:

$$\begin{aligned}
\frac{d(\Delta t)}{d\Delta t} &= 1 \\
\frac{d}{d\Delta t} (\mathbf{D}(\Delta t) - \mathbf{I}) &= -\mathbf{E}\mathbf{D}(\Delta t)
\end{aligned}$$

Therefore, we obtain $\lim_{\Delta t \to 0} \frac{1}{\Delta t} (\mathbf{D}(\Delta t) - \mathbf{I}) = -\mathbf{E}$ and

$$\lim_{\Delta t \to 0} \frac{\mathbf{\Pi}(\Delta t) - \mathbf{I}}{\Delta t} = \mathbf{P}{\cdot}\mathbf{E} - \mathbf{E} = \mathbf{Q},$$

where $\mathbf{Q}$ is the infinitesimal generator of the CTMC $\mathcal{C}$. As a result we obtain:

$$\frac{d\vec{\wp}(t)}{dt} = \vec{\wp}(t) \lim_{\Delta t \to 0} \frac{\mathbf{\Pi}(\Delta t) - \mathbf{I}}{\Delta t} = \vec{\wp}(t)\mathbf{Q}.$$

Combining with Eq. (A.12) we get:

$$\begin{aligned}
\vec{\wp}(t) =& \alpha \cdot \mathbf{\Pi}(t), \\
\frac{d\vec{\wp}(t)}{dt} =& \vec{\wp}(t) \cdot \mathbf{Q}.
\end{aligned}$$ ■

A.7 Proof of Theorem 7.29

Theorem 7.29 For subgraph $\mathcal{G}_i$ of $\mathcal{G}$ with k_i states, it holds for $0 \leqslant i < m$ that:

$$\vec{U}_i(0) = \mathbf{\Pi}_i(\Delta c_i) \cdot \mathbf{F}_i \vec{U}_{i+1}(0) + \bar{\mathbf{\Pi}}_i^a(\Delta c_i) \cdot \vec{U}_0(0), \tag{A.13}$$

where $\mathbf{\Pi}_i(\Delta c_i)$ and $\bar{\mathbf{\Pi}}_i^a(\Delta c_i)$ are for CTMC $\mathcal{C}_i$ and the augmented CTMC $\mathcal{C}_i^a$, respectively. For case $i = m$, it holds that:

$$\vec{U}_m(0) = \hat{\mathbf{P}}_i \cdot \vec{U}_m(0) + \vec{1}_F + \hat{\mathbf{B}}_m \cdot \vec{U}_0(0), \tag{A.14}$$

where $\hat{\mathbf{P}}_i(v, v') = \mathbf{P}(v, v')$ if $v \notin V_F$; 0 otherwise and $\hat{\mathbf{B}}_m = \int_0^\infty \mathbf{B}_m(\tau) d\tau$.
Proof: We first deal with the case $i < m$. If in $\mathcal{G}_i$, there exists some backward edge, namely, for some j, j', $\mathbf{B}_i(x)[j, j'] \neq 0$, then we shall consider the *augmented* CTMC

$\mathcal{C}_i^a$ with $k_i^a = k_i + k_0$ states. In view of this, the augmented integral equation $\vec{U}_i^a(x)$ is defined as:

$$\vec{U}_i^a(x) = \int_0^{\Delta c_i - x} \mathbf{M}_i^a(\tau)\vec{U}_i^a(x+\tau)d\tau + \mathbf{D}_i^a(\Delta c_i - x)\cdot \mathbf{F}_i^a \vec{\hat{U}}_i(0)$$

where $\vec{U}_i^a(x) = \left(\frac{\vec{U}_i(x)}{\vec{U}_i'(x)}\right) \in \mathbb{R}^{k_i^a \times 1}$, $\vec{U}_i'(x) \in \mathbb{R}^{k_0 \times 1}$ is the vector representing reachability probability for the augmented states in $\mathcal{G}_i$, $\mathbf{F}_i^a = \left(\ \mathbf{F}_i' \mid \mathbf{B}_i'\ \right) \in \mathbb{R}^{k_i^a \times (k_{i+1}+k_0)}$ such that $\mathbf{F}_i' = \left(\frac{\mathbf{F}_i}{\mathbf{0}}\right) \in \mathbb{R}^{k_i^a \times k_{i+1}}$ is the incidence matrix for delay edges and $\mathbf{B}_i' = \left(\frac{\mathbf{0}}{\mathbf{I}}\right) \in \mathbb{R}^{k_i^a \times k_0}$, $\vec{\hat{U}}_i(0) = \left(\frac{\vec{U}_{i+1}(0)}{\vec{U}_0(0)}\right) \in \mathbb{R}^{(k_{i+1}+k_0)\times 1}$.

First, we prove the following equation:

$$\vec{U}_i^a(x) = \mathbf{\Pi}_i^a(\Delta c_i - x)\cdot \mathbf{F}_i^a \vec{\hat{U}}_i(0),$$

where

$$\mathbf{\Pi}_i^a(x) = \int_0^x \mathbf{M}_i^a(\tau)\mathbf{\Pi}_i^a(x-\tau)d\tau + \mathbf{D}_i^a(x). \tag{A.15}$$

We consider the iterations of the solution of the following system of integral equations: set $c_{i,x} = \Delta c_i - x$.

$$\begin{aligned} \vec{U}_i^{a,(0)}(x) &= \vec{0} \\ \vec{U}_i^{a,(j+1)}(x) &= \int_0^{c_{i,x}} \mathbf{M}_i^a(\tau)\vec{U}_i^{a,(j)}(x+\tau)d\tau + \mathbf{D}_i^a(c_{i,x})\cdot \mathbf{F}_i^a\vec{\hat{U}}_i(0). \end{aligned}$$

and

$$\begin{aligned} \mathbf{\Pi}_i^{a,(0)}(c_{i,x}) &= \mathbf{0} \\ \mathbf{\Pi}_i^{a,(j+1)}(c_{i,x}) &= \int_0^{c_{i,x}} \mathbf{M}_i^a(\tau)\mathbf{\Pi}_i^{a,(j)}(c_{i,x}-\tau)d\tau + \mathbf{D}_i^a(c_{i,x}). \end{aligned}$$

By induction on j, we prove the following relation:

$$\vec{U}_i^{a,(j)}(x) = \mathbf{\Pi}_i^{a,(j)}(c_{i,x})\cdot \mathbf{F}_i^a\vec{\hat{U}}_i(0).$$

- Base case: $\vec{U}_i^{a,(0)}(x) = \vec{0}$ and $\mathbf{\Pi}_i^{a,(0)}(c_{i,x}) = \mathbf{0}$.
- Induction hypothesis: $\vec{U}_i^{a,(j)}(x) = \mathbf{\Pi}_i^{a,(j)}(c_{i,x})\cdot \mathbf{F}_i^a\vec{\hat{U}}_i(0)$.
- Induction step $j \to j+1$:

$$\vec{U}_i^{a,(j+1)}(x) = \int_0^{c_{i,x}} \mathbf{M}_i^a(\tau)\vec{U}_i^{a,(j)}(x+\tau)d\tau + \mathbf{D}_i^a(c_{i,x})\cdot \mathbf{F}_i^a\vec{\hat{U}}_i(0).$$

By induction hypothesis we have

$$\begin{aligned}
\vec{U}_i^{a,(j+1)}(x) &= \int_0^{c_{i,x}} \mathbf{M}_i^a(\tau)\vec{U}_i^{a,(j)}(x+\tau)d\tau + \mathbf{D}_i^a(c_{i,x})\cdot \mathbf{F}_i^a\vec{\hat{U}}_i(0)\\
&= \int_0^{c_{i,x}} \mathbf{M}_i^a(\tau)\mathbf{\Pi}_i^{a,(j)}(c_{i,x}-\tau)\cdot \mathbf{F}_i^a\vec{\hat{U}}_i(0)d\tau + \mathbf{D}_i^a(c_{i,x})\cdot \mathbf{F}_i^a\vec{\hat{U}}_i(0)\\
&= \left(\int_0^{c_{i,x}} \mathbf{M}_i^a(\tau)\mathbf{\Pi}_i^{a,(j)}(c_{i,x}-\tau)d\tau + \mathbf{D}_i^a(c_{i,x})\right)\cdot \mathbf{F}_i^a\vec{\hat{U}}_i(0)\\
&= \mathbf{\Pi}_i^{a,(j+1)}(c_{i,x})\cdot \mathbf{F}_i\vec{\hat{U}}_i(0).
\end{aligned}$$

Clearly, $\mathbf{\Pi}_i^a(c_{i,x}) = \lim_{j\to\infty}\mathbf{\Pi}_i^{a,(j+1)}(c_{i,x})$ and $\vec{U}_i^a(x) = \lim_{j\to\infty}\vec{U}_i^{a,(j+1)}(x)$.

Let $x=0$ and we obtain

$$\vec{U}_i^a(0) = \mathbf{\Pi}_i^a(c_{i,0})\cdot \mathbf{F}_i^a\vec{\hat{U}}_i(0).$$

We can also write the above relation for $x=0$ as:

$$\begin{aligned}
\left(\frac{\vec{U}_i(0)}{\vec{U}_i'(0)}\right) &= \mathbf{\Pi}_i^a(\Delta c_i)\left(\ \mathbf{F}_i' \mid \mathbf{B}_i'\ \right)\left(\frac{\vec{U}_{i+1}(0)}{\vec{U}_0(0)}\right)\\
&= \left(\begin{array}{c|c}\mathbf{\Pi}_i(\Delta c_i) & \bar{\mathbf{\Pi}}_i^a(\Delta c_i)\\ \hline \mathbf{0} & \mathbf{I}\end{array}\right)\left(\begin{array}{c|c}\mathbf{F}_i & \mathbf{0}\\ \hline \mathbf{0} & \mathbf{I}\end{array}\right)\left(\frac{\vec{U}_{i+1}(0)}{\vec{U}_0(0)}\right)\\
&= \left(\begin{array}{c|c}\mathbf{\Pi}_i(\Delta c_i)\mathbf{F}_i & \bar{\mathbf{\Pi}}_i^a(\Delta c_i)\\ \hline \mathbf{0} & \mathbf{I}\end{array}\right)\left(\frac{\vec{U}_{i+1}(0)}{\vec{U}_0(0)}\right)\\
&= \left(\frac{\mathbf{\Pi}_i(\Delta c_i)\mathbf{F}_i\vec{U}_{i+1}(0)+\bar{\mathbf{\Pi}}_i^a(\Delta c_i)\vec{U}_0(0)}{\vec{U}_0(0)}\right).
\end{aligned}$$

As a result we can represent $\vec{U}_i(0)$ in the following matrix form

$$\vec{U}_i(0) = \mathbf{\Pi}_i(\Delta c_i)\mathbf{F}_i\vec{U}_{i+1}(0) + \bar{\mathbf{\Pi}}_i^a(\Delta c_i)\vec{U}_0(0)$$

by noting that $\mathbf{\Pi}_i$ is formed by the first k_i rows and columns of matrix $\mathbf{\Pi}_i^a$ and $\bar{\mathbf{\Pi}}_i^a$ is formed by the first k_i rows and the last $k_i^a - k_i$ columns of $\mathbf{\Pi}_i^a$.

For $i=m$, i.e., the last graph $\mathcal{G}_m$, the region size is infinite, therefore delay transitions do not exist. The vector $\vec{U}_m(x+\tau)$ in $\int_0^\infty \hat{\mathbf{M}}_m(\tau)\vec{U}_m(x+\tau)d\tau$ does not depend on entering time x, therefore we can take it out of the integral. As a result we obtain $\int_0^\infty \hat{\mathbf{M}}_m(\tau)d\tau\cdot\vec{U}_m(0)$. More than that $\int_0^\infty \hat{\mathbf{M}}_m(\tau)d\tau$ boils down to $\hat{\mathbf{P}}_m$ and $\int_0^\infty \mathbf{B}_m(\tau)d\tau$ to $\hat{\mathbf{B}}_m$. Also we add the vector $\vec{1}_F$ to ensure that the probability to start from a state in V_F is one (see (7.6)). ■

A.8 Proof of Theorem 7.34

Theorem 7.34 For any CTMC $\mathcal{C}$, DTA$^\omega$ $\mathcal{A}^\omega$, and their region graph $\mathcal{G}^\omega = (V, v_0, V_F^\omega, \Lambda, \hookrightarrow)$ of the product, it holds that:

$$Prob^{\mathcal{C}}(\mathcal{A}^\omega) = Prob^{\mathcal{G}^\omega}(v_0, \Diamond V_F^\omega).$$

Proof: We show the theorem by the following three steps:

1. $\mathit{Prob}^{\mathcal{C}}(\mathcal{A}^\omega) = \mathit{Prob}^{\mathcal{C}\otimes\mathcal{A}}(\mathit{Loc}_{\mathcal{F}})$, where $\mathit{Prob}^{\mathcal{C}\otimes\mathcal{A}}(\mathit{Loc}_{\mathcal{F}})$ denotes the probability of accepting paths of DMTA $\mathcal{C}\otimes\mathcal{A}$ w.r.t. Muller accepting conditions;
2. $\mathit{Prob}^{\mathcal{C}\otimes\mathcal{A}}(\mathit{Loc}_{\mathcal{F}}) = \mathit{Prob}^{\mathcal{G}^\omega}(v_0, \mathit{Loc}_{\mathcal{F}})$;
3. $\mathit{Prob}^{\mathcal{G}^\omega}(v_0, \mathit{Loc}_{\mathcal{F}}) = \mathit{Prob}^{\mathcal{G}^\omega}(v_0, \Diamond V_F^\omega)$.

For the first step, we note that $\mathit{Paths}^{\mathcal{C}}(\mathcal{A}^\omega) = \bigcap_{1\leqslant i\leqslant k} \mathit{Paths}^i$ where

$$\mathit{Paths}^i = \bigcap_{n\geqslant 0}\bigcup_{m\geqslant n}\bigcup_{s_0,\ldots,s_n,s_{n+1}\ldots,s_m} C(s_0, I_0, \ldots, I_{n-1}, s_n, \ldots, I_{m-1}, s_m), \qquad \text{where}$$

- $\{s_{n+1}, \ldots, s_m\} = L_{F_i}$;
- $C(s_0, I_0, \ldots, I_{n-1}, s_n, \ldots, I_{m-1}, s_m)$ is the cylinder set such that each timed path of the cylinder set of the form $s_0 \xrightarrow{t_0} \cdots \xrightarrow{t_{n-1}} s_n \cdots \xrightarrow{t_{m-1}} s_m$ is a prefix of an accepting path of $\mathcal{A}$.

Similar to Lemma 7.15, one can easily see that each path of CTMC $\mathcal{C}$ can be lifted to a unique path of DMTA$^\omega$ $\mathcal{C}\otimes\mathcal{A}^\omega$. Following the same argument as in Theorem 7.16, one can obtain that for each cylinder set of the form $C(s_0, I_0, \ldots, I_{n-1}, s_n, \ldots, I_{m-1}, s_m)$, $\mathcal{C}$ and $\mathcal{C}\otimes\mathcal{A}^\omega$ give rise to the same probability. Hence $\mathit{Prob}^{\mathcal{C}}(\mathcal{A}^\omega) = \mathit{Prob}^{\mathcal{C}\otimes\mathcal{A}^\omega}(\mathit{Loc}_{\mathcal{F}})$.

For the second step, we need to define a timed path of $\mathcal{G}^\omega$, which is of the form $v_0 \xrightarrow{t_0} v_1 \xrightarrow{t_2} \cdots$ such that given the initial valuation η_0, one can construct a sequence $\{\eta_i\}$ such that

- $\eta_{i+1} = (\eta_i + t_i)[X_i := 0]$ if $\eta_i + t_i \models \mathit{Inv}(v_i)$ (namely, the transition from v_i to v_{i+1} is via a Markovian transition); and
- $\eta_{i+1} = \eta_i + t_i$ if $\eta_i + t_i \in \partial \mathit{Inv}(v_i)$ (namely, the transition from v_i to v_{i+1} is via a forced boundary jump).

A path of $\mathcal{G}^\omega$ is accepted if the discrete part of the path, namely $v_0 v_1 \cdots$ meets the Muller condition.

Following the standard region construction, one can lift a timed path of DMTA$^\omega$ $\mathcal{C}\otimes\mathcal{A}^\omega$ to a unique timed path of the corresponding region graph $\mathcal{G}^\omega$. Moreover, following the same argument of Theorem 7.23, one can show that $\mathcal{C}\otimes\mathcal{A}^\omega$ and $\mathcal{G}^\omega$ give rise to the same probability to the accepted paths.

For the third step, we note that according to the ergodicity of PDP (region graph), for each path of $\mathcal{G}^\omega$, with probability 1 the states visited infinitely often constitute a BSCC. It follows that

$$\mathit{Prob}^{\mathcal{G}^\omega}(v_0, \mathit{Loc}_{\mathcal{F}}) = \sum_{B\in a\mathcal{B}} \mathit{Prob}\{\rho \mid \mathit{inf}(\rho) = B\}.$$

We note that for each note v in an accepting BSCC, $\mathit{Prob}\{\mathit{Paths}^{\mathcal{G}^\omega}(v)\} = 1$. Hence

$$\mathit{Prob}^{\mathcal{G}^\omega}(v_0, \mathit{Loc}_{\mathcal{F}}) = \mathit{Prob}^{\mathcal{G}^\omega}(v_0, \Diamond V_F^\omega).$$

■

Bibliography

[ABW90] Helmut Alt, Johannes Blömer, and Hubert Wagener. Approximation of convex polygons. In *ICALP*, pages 703–716, 1990. 103

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994. 3, 131, 132, 136, 142

[ADvR08] Miguel E. Andrés, Pedro D'Argenio, and Peter van Rossum. Significant diagnostic counterexamples in probabilistic model checking. In *HVC*, LNCS 5394, pages 129–148, 2008. 71, 92

[AFH96] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996. 3

[AFHL04] Christer Andersson, Simone Fischer-Hübner, and Reine Lundin. Enabling anonymity for the mobile Internet using the mCrowds system. In *IFIP WG 9.2, 9.6/11.7 Summer School on Risks and Challenges of the Network Society*, page 35, 2004. 55

[AHH96] Rajeev Alur, Thomas A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. Software Eng.*, 22(3):181–201, 1996. 129

[AHK03] Suzana Andova, Holger Hermanns, and Joost-Pieter Katoen. Discrete-time rewards model-checked. In *FORMATS*, LNCS 2719, pages 88–104, 2003. 82

[AHL05] Husain Aljazzar, Holger Hermanns, and Stefan Leue. Counterexamples for timed probabilistic reachability. In *FORMATS*, LNCS 3829, pages 177–195, 2005. 91

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *STOC*, pages 592–601, 1993. 128

[AL06] Husain Aljazzar and Stefan Leue. Extended directed search for probabilistic timed reachability. In *FORMATS*, LNCS 4202, pages 33–51, 2006. 91

[AL08a] Husain Aljazzar and Stefan Leue. Debugging of dependability models using interactive visualization of counterexamples. In *QEST*, pages 189–198, 2008. 93

[AL08b] Husain Aljazzar and Stefan Leue. K*: A directed on-the-fly algorithm for finding the k shortest paths. Technical report, soft-08-03, Chair for Software Engineering, University of Konstanz, 2008. 92

[AL09] Husain Aljazzar and Stefan Leue. Generation of counterexamples for model checking of Markov decision processes. In *QEST*, to appear, 2009. 92

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, Inc., 1993. 31

[ASSB00] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert K. Brayton. Model-checking continous-time Markov chains. *ACM Trans. Comput. Log.*, 1(1):162–170, 2000. 3, 20, 85, 159

[Avr03] Mordecai Avriel. *Nonlinear Programming: Analysis and Methods*. Dover Publishing, 2003. 98

[AvR08] Miguel E. Andrés and Peter van Rossum. Conditional probabilities over probabilistic and nondeterministic systems. In *TACAS*, LNCS 4963, pages 157–172, 2008. 91

[AW95] George B. Arfken and Hans J. Weber. *Mathematical Methods for Physicists (4th ed.)*. Academic Press, 1995. 148

[BBB$^+$07] Christel Baier, Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Marcus Größer. Probabilistic and topological semantics for timed automata. In *FSTTCS*, pages 179–191, 2007. 160

[BBB$^+$08] Christel Baier, Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Marcus Grösser. Almost-sure model checking of infinite paths in one-clock timed automata. In *LICS*, pages 217–226, 2008. 160

[BBBM08] Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Nicolas Markey. Quantitative model-checking of one-clock timed automata under probabilistic semantics. In *QEST*, pages 55–64, 2008. 160

[BCH$^+$07] Christel Baier, Lucia Cloth, Boudewijn R. Haverkort, Matthias Kuntz, and Markus Siegle. Model checking Markov chains with actions and state labels. *IEEE Trans. Software Eng.*, 33(4):209–224, 2007. 132, 158

[BCvdP09] Jiři Barnat, Jakub Chaloupka, and Jaco van de Pol. Distributed algorithms for SCC decomposition. *J. Log. Comput.*, to appear, 2009. 71

[BdA95] Andrea Bianco and Luca de Alfaro. Model checking of probabilistic and nondeterministic systems. In *FSTTCS*, LNCS 1026, pages 499–513, 1995. 16, 19

[BDR03] Véronique Bruyère, Emmanuel Dall'Olio, and Jean-François Raskin. Durations, parametric model-checking in timed automata with Presburger arithmetic. In *STACS*, LNCS 2607, pages 687–698, 2003. 128

[Bel58] Richard Bellman. On a routing problem. *Quarterly of Appl. Math.*, 16(1):87–90, 1958. 31, 32

[BHHK03] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003. 3, 12, 20, 85, 87, 97, 132, 159

[BHKvdP08] Stefan Blom, Boudewijn R. Haverkort, Matthias Kuntz, and Jaco van de Pol. Distributed Markovian bisimulation reduction aimed at CSL model checking. *Electr. Notes Theor. Comput. Sci.*, 220(2):35–50, 2008. 71

[BK98] Christel Baier and Marta Z. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11:125–155, 1998. 16, 84

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008. 15, 16, 20, 81, 82

[BLR05] Gerd Behrmann, Kim G. Larsen, and Jacob I. Rasmussen. Optimal scheduling using priced timed automata. *ACM SIGMETRICS Perf. Ev. Review*, 32(4):34–40, 2005. 4

[BNR03] Thomas Ball, Mayur Naik, and Sriram K. Rajamani. From symptom to cause: localizing errors in counterexample traces. In *POPL*, pages 97–105, 2003. 4

[BPDG98] Béatrice Bérard, Antoine Petit, Volker Diekert, and Paul Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundam. Inform.*, 36(2-3):145–182, 1998. 135

[BR07] Véronique Bruyère and Jean-François Raskin. Real-time model-checking: Parameters everywhere. *CoRR*, abs/cs/0701138, 2007. 129

[Brz64] Janusz A. Brzozowski. Derivatives of regular expressions. *J. ACM*, 11(4):481–494, 1964. 63

[BS86] Gérard Berry and Ravi Sethi. From regular expressions to deterministic automata. *Theor. Comput. Sci.*, 48(3):117–126, 1986. 63

[CD88] Oswaldo L.V. Costa and Mark H.A. Davis. Approximations for optimal stopping of a piecewise-deterministic process. *Math. Control Signals Systems*, 1(2):123–146, 1988. 143

[CDF$^+$08] Najla Chamseddine, Marie Duflot, Laurent Fribourg, Claudine Picaronny, and Jeremy Sproston. Computing expected absorption times for parametric determinate probabilistic timed automata. In *QEST*, pages 254–263, 2008. 127

[CGJ$^+$00] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In *CAV*, LNCS 1855, pages 154–169, 2000. 4

[CHK08] Taolue Chen, Tingting Han, and Joost-Pieter Katoen. Time-abstracting bisimulation for probabilistic timed automata. In *TASE*, pages 177–184, 2008. 8

[CHKM09a] Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Quantitative model checking of continuous-time Markov chains against timed automata specification. In *LICS*, pages 309–318, 2009. 7, 8, 159

[CHKM09b] Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. LTL model checking of time-inhomogeneous Markov chains. In *ATVA*, to appear, 2009. 8

[CJLV02] Edmund M. Clarke, Somesh Jha, Yuan Lu, and Helmut Veith. Tree-like counterexamples in model checking. In *LICS*, pages 19–29, 2002. 4

[Cla08] Edmund M. Clarke. The birth of model checking. In *25 Years of Model Checking*, pages 1–26, 2008. 4

[CLRS01a] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, chapter 24.1 The Bellman-Ford algorithm, pp. 588-592. In [CLRS01b], 2001. 31, 32

[CLRS01b] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, 2001. 99, 181

[Com74] Louis Comtet. *Advanced Combinatorics: The Art of Finite and Infinite Expansion*. D.Reidel Publishing Company, 1974. 58

[Cor91] C. Corduneanu. *Integral Equations and Applications*. Cambridge University Press, 1991. 148

[CSS03] Jean-Michel Couvreur, Nasser Saheb, and Grégoire Sutre. An optimal automata approach to LTL model checking of probabilistic systems. In *LPAR*, LNCS 2850, pages 361–375, 2003. 3, 159

[CV08] Rohit Chadha and Mahesh Viswanathan. A counterexample guided abstraction-refinement framework for Markov decision processes. *CoRR*, abs/0807.1173, 2008. 93

[CY95a] Ing-Ray Chen and I-Ling Yen. Analysis of probabilistic error checking procedures on storage systems. *Comput. J.*, 38(5):348–354, 1995. 95, 115

[CY95b] Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995. 2, 3, 21, 159

[dAHM00] Luca de Alfaro, Thomas A. Henzinger, and Freddy Y.C. Mang. Detecting errors before reaching them. In *CAV*, LNCS 1855, pages 186–201, 2000. 4

[Dam08] Berteun Damman. Representing PCTL Counterexamples. Master's thesis, FMT, University of Twente, 2008. 56

[Dav84] Mark H. A. Davis. Piecewise-deterministic Markov processes: A general class of non-diffusion stochastic models. *Journal of the Royal Statistical Society (B)*, 46(3):353–388, 1984. 143

[Dav93] Mark H. A. Davis. *Markov Models and Optimization*. Chapman and Hall, 1993. 131, 143, 144, 145, 170, 171

[Daw04] Conrado Daws. Symbolic and parametric model checking of discrete-time Markov chains. In *ICTAC*, LNCS 3407, pages 280–294, 2004. 55, 59, 60, 61, 127

[DHK08] Berteun Damman, Tingting Han, and Joost-Pieter Katoen. Regular expressions for PCTL counterexamples. In *QEST*, pages 179–188, 2008. 7

[DHS09] Susanna Donatelli, Serge Haddad, and Jeremy Sproston. Model checking timed and stochastic properties with CSL$^{\mathrm{TA}}$. *IEEE Trans. Software Eng.*, 35(2):224–240, 2009. 132, 133, 159, 160

[Dij59] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. 31

[DK01] Ding-Shu Du and Ker-I Ko. *Problem Solving in Automata, Languages, and Complexity*. John Wiley and Sons, 2001. 63

[DM04] Manuel Delgado and José Morais. Approximation to the smallest regular expression for a given regular language. In *CIAA*, LNCS 3317, pages 312–314, 2004. 55, 64, 66, 67

[EKSW05] Keith Ellul, Bryan Krawetz, Jeffrey Shallit, and Mingwei Wang. Regular expressions: New results and open problems. *Journal of Automata, Languages and Combinatorics*, 10(4):407–437, 2005. 66

[Epp98] David Eppstein. Finding the k shortest paths. *SIAM J. Comput.*, 28(2):652–673, 1998. 41

[FG88] Bennett L. Fox and Peter W. Glynn. Computing Poisson probabilities. *Comm. ACM*, 31(4):440–445, 1988. 14

[For73] G. David Forney. The Viterbi algorithm. *Proc. of the IEEE*, 61(3):268–278, 1973. 33

[Fre08] Goran Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. *STTT*, 10(3):263–279, 2008. 129

[Fro98] Ralf Froeberg. *An Introduction to Gröbner Bases*. John Wiley & Sons, 1998. 99

[GC03] Arie Gurfinkel and Marsha Chechik. Proof-like counter-examples. In *TACAS*, LNCS 2619, pages 160–175, 2003. 4

[GH08] Hermann Gruber and Markus Holzer. Provably shorter regular expressions from deterministic finite automata. In *Developments in Language Theory*, pages 383–395, 2008. 66

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979. 31

[Gru08] Lars Grunske. Specification patterns for probabilistic quality properties. In *ICSE*, pages 31–40, 2008. 66

[GS07] Gregor Gramlich and Georg Schnitger. Minimizing NFA's and regular expressions. *J. Comput. Syst. Sci.*, 73(6):908–923, 2007. 64

[Ham73] Richard W. Hamming. *Numerical Methods for Scientists and Engineers*. McGraw-Hill, 1973. 99, 105

[Han92] Eldon Hansen. *Global Optimization Using Interval Analysis*. Dekker, New York, 1992. 99

[HHWT97] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Hytech: A model checker for hybrid systems. *STTT*, 1(1-2):110–122, 1997. 129

[HHZ09] Ernst Moritz Hahn, Holger Hermanns, and Lijun Zhang. Probabilistic reachability for parametric Markov models. In *SPIN*, pages 88–106, 2009. 127

[HJ94] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994. 2, 3, 16, 19, 159

[HK88] Peter Henrici and William R. Kenan. *Applied & Computational Complex Analysis: Power Series Integration Conformal Mapping Location of Zero*. John Wiley & Sons, 1988. 99, 105

[HK07a] Tingting Han and Joost-Pieter Katoen. Counterexamples in probabilistic model checking. In *TACAS*, LNCS 4424, pages 72–86, 2007. 7

[HK07b] Tingting Han and Joost-Pieter Katoen. Providing evidence of likely being on time: Counterexample generation for CTMC model checking. In *ATVA*, LNCS 4762, pages 331–346, 2007. 7, 90

[HKD09] Tingting Han, Joost-Pieter Katoen, and Berteun Damman. Counterexample generation in probabilistic model checking. *IEEE Trans. Software Eng.*, 35(2):241–257, 2009. 7

[HKM08a] Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Approximate parameter synthesis for probabilistic time-bounded reachability. In *RTSS*, pages 173–182, 2008. 7

[HKM08b] Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Compositional modeling and minimization of time-inhomogeneous Markov chains. In *HSCC*, pages 244–258, 2008. 8

[HRSV02] Thomas Hune, Judi Romijn, Mariëlle Stoelinga, and Frits W. Vaandrager. Linear parametric model checking of timed automata. *J. Log. Algebr. Program.*, 52-53:183–220, 2002. 128

[HSP83] Sergiu Hart, Micha Sharir, and Amir Pnueli. Termination of probabilistic concurrent programs. *ACM Trans. Program. Lang. Syst.*, 5(3):356–380, 1983. 83

[HW07] Yo-Sub Han and Derick Wood. Obtaining shorter regular expressions from finite-state automata. *Theor. Comput. Sci.*, 370(1-3):110–120, 2007. 55, 64, 65, 66, 67

[HWZ08] Holger Hermanns, Björn Wachter, and Lijun Zhang. Probabilistic CEGAR. In *CAV*, LNCS 5123, pages 162–175, 2008. 92

[IR90] Alon Itai and Michael Rodeh. Symmetry breaking in distributed networks. *Inf. Comput.*, 88(1):60–87, 1990. 55, 56

[Jaf84] Jeffery M. Jaffe. Algorithms for finding paths with multiple constraints. *IEEE Network*, 14:95–116, 1984. 83

[Jen53] Arne Jensen. Markoff chains as an aid in the study of Markoff processes. *Skand. Aktuarietidskrift*, 36:87–91, 1953. 13, 85

[JM99] Víctor M. Jiménez and Andrés Marzal. Computing the k shortest paths: A new algorithm and an experimental comparison. In *WAE*, LNCS 1668, pages 15–29, 1999. 41, 43, 45, 51

[JRS04] HoonSang Jin, Kavita Ravi, and Fabio Somenzi. Fate and free will in error traces. *STTT*, 6(2):102–116, 2004. 4

[JTW07] Seema H. Jadhav, B. S. Tarle, and L. M. Waghmare. Polygonal approximation of 2-d binary images. In *ITNG*, pages 219–223, 2007. 103

[KKKvM04] Fernando A. Kuipers, Turgay Korkmaz, Marwan Krunz, and Piet van Mieghem. Performance evaluation of constraint-based path selection algorithms. *IEEE Network*, 18(5):16–23, 2004. 83

[KKLW07] Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf. Three-valued abstraction for continuous-time Markov chains. In *CAV*, LNCS 4580, pages 311–324, 2007. 161

[KKZ05] Joost-Pieter Katoen, M. Khattri, and Ivan S. Zapreev. A Markov reward model checker. In *QEST*, pages 243–244, 2005. 2, 56, 88

[KKZJ07] Joost-Pieter Katoen, Tim Kemna, Ivan Zapreev, and David Jansen. Bisimulation minimisation mostly speeds up probabilistic model checking. In *TACAS*, LNCS 4424, pages 87–101, 2007. 71

[Kle56] Stephen Cole Kleene. *Representation of Events in Nerve Nets and Finite Automata*, pages 3–42. Princeton Univ. Press, 1956. 61, 63

[KNP04] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Probabilistic symbolic model checking with PRISM: a hybrid approach. *STTT*, 6(2):128–142, 2004. 2, 56, 88

[KNP06] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Game-based abstraction for Markov decision processes. In *QEST*, pages 157–166, 2006. 161

[KNSS02] Marta Z. Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theor. Comput. Sci.*, 282(1):101–150, 2002. 148

[Koy90] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990. 3

[Lag83] Edmond Laguerre. Sur la théorie des équations numériques. *J. Math. Pures Appl. (3e série)*, 9:99–146, 1883. 99

[Law76] Eugene Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Reinhart and Winston, 1976. 32

[lGM02] Hélène le Guen and Raymond A. Marie. Visiting probabilities in non-irreducible Markov chains with strongly connected components. In *ESM*, pages 548–552, 2002. 55, 71, 92

[Lin01] Peter Linz. *An Introduction to Formal Languages and Automata*. Jones and Bartless Publishers, Sudbury, MA, 2001. 63

[LL85] Suzanne M. Lenhart and Yu-Chung Liao. Integro-differential equations associated with optimal stopping time of a piecewise-deterministic process. *Stochastics*, 15(3):183–207, 1985. 143

[LMST07] Ruggero Lanotte, Andrea Maggiolo-Schettini, and Angelo Troina. Parametric probabilistic transition systems for system design and analysis. *Formal Asp. Comput.*, 19(1):93–109, 2007. 127

[LR01] Gang Liu and K. G. Ramakrishnan. A*prune: An algorithm for finding K shortest paths subject to multiple constraints. In *INFOCOM*, pages 743–749, 2001. 83

[LS91] Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991. 55, 71

[LY91] Suzanne M. Lenhart and Naoki Yamada. Perron's method for viscosity solutions associated with piecewise-deterministic processes. *Funkcialaj Ekvacioj*, 34:173–186, 1991. 143

[MMG08] Annabelle McIver, Carroll Morgan, and Carlos Gonzalia. Proofs and refutations for probabilistic systems. In *FM*, LNCS 5014, pages 100–115, 2008. 93

[MMR05] José João Morais, Nelma Moreira, and Rogério Reis. Acyclic automata with easy-to-find short regular expressions. In *CIAA*, pages 349–350, 2005. 66

[Mon70] Ugo Montanari. A note on minimal length polygonal approximation to a digitized contour. *Commun. ACM*, 13(1):41–47, 1970. 103

[MRT87] Raymond A. Marie, Andrew L. Reibman, and Kishor S. Trivedi. Transient analysis of acyclic Markov chains. *Perform. Eval.*, 7(3):175–194, 1987. 85, 88

[MT91] Manish Malhotra and Kishor S. Trivedi. Higher-order methods for transient analysis of stiff markov chains. In *Proc. Int. Conf. on the Performance of Distributed Systems and Integrated Communication Networks*, 1991. 126

[MZ00] Kurt Mehlhorn and Mark Ziegelmann. Resource constrained shortest paths. In *ESA*, LNCS 1879, pages 326–337, 2000. 31

[Neu81] Marcel F. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach.* The Johns Hopkins University Press, 1981. 87

[Neu05] Christoph Neumann. Converting deterministic finite automata to regular expressions. `http://neumannhaus.com/christoph/papers/2005-03-16.DFA_to_RegEx.pdf`, 2005. 63

[NW99] Jorge Nocedal and Stephen Wright. *Numerical Optimization.* Springer, 1999. 99

[Pan00] Victor Y. Pan. Approximating complex polynomial zeros: Modified Weyl's quadtree construction and improved Newton's iteration. *J. Complexity*, 16(1):213–264, 2000. 105, 114

[Pnu77] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977. 20

[Pri] PRISM website. `http://www.prismmodelchecker.org`. 56, 73

[Pur99] Anuj Puri. An undecidable problem for timed automata. *Discrete Event Dynamic Systems*, 9(2):135–146, 1999. 128

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM TISSEC*, 1(1):66–92, 1998. 55, 73

[RST89] Andrew L. Reibman, Roger Smith, and Kishor S. Trivedi. Markov and Markov reward models transient analysis: An overview of numerical approaches. *European J. OR*, 4:257–267, 1989. 126

[RT88] Andrew L. Reibman and Kishor S. Trivedi. Numerical transient analysis of Markov models. *Computers & OR*, 15(1):19–36, 1988. 126

[SBFT03] Gary A. Sitton, C. Sidney Burrus, James W. Fox, and Sven Treitel. Factoring very-high-degree polynomials. *Signal Processing Magazine, IEEE*, 20(6):27–42, 2003. 105, 119

[Sch98] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998. 99

[SG03] Sharon Shoham and Orna Grumberg. A game-based framework for CTL counterexamples and 3-valued abstraction-refinement. In *CAV*, LNCS 2725, pages 275–287, 2003. 4, 82

[Shm04] Vitaly Shmatikov. Probabilistic analysis of an anonymity system. *Journal of Computer Security*, 12(3-4):355–377, 2004. 73

[Ste73] H.J. Stetter. *Analysis of Discretization Methods for Ordinary Differential Equations*. Springer-Verlag, 1973. 111

[Ste89] Ian Stewart. *Galois Theory*. Chapman and Hall, 1989. 99

[SVV09] Matthias Schmalz, Daniele Varacca, and Hagen Völzer. Counterexamples in probabilistic LTL model checking for Markov chains. In *CONCUR*, to appear, 2009. 92

[TBT06] Axel Thümmler, Peter Buchholz, and Miklós Telek. A novel approach for phase-type fitting with the EM algorithm. *IEEE Trans. Dependable Sec. Comput.*, 3(3):245–258, 2006. 5

[Var85] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS*, pages 327–338, 1985. 2, 3, 12, 21, 159

[Vit67] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. on Inf. Theory*, 13(2):260–269, 1967. 33

[VSV05] Antti Vaha-Sipila and Teemupekka Virtanen. BT-Crowds: Crowds-style anonymity with Bluetooth and Java. In *HICSS*, 2005. 55

[VTdlH$^+$05] Enrique Vidal, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco:. Probabilistic finite-state machines-part I. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7):1013–1025, 2005. 33

[WBB09] Ralf Wimmer, Bettina Braitling, and Bernd Becker. Counterexample generation for discrete-time Markov chains using bounded model checking. In *VMCAI*, pages 366–380, 2009. 93

[Woo87] Derick Wood. *Theory of Computation*. John Wiley, 1987. 66

[ZC05] Dezhuang Zhang and Rance Cleaveland. Fast on-the-fly parametric real-time model checking. In *RTSS*, pages 157–166, 2005. 128

Curriculum Vitae

Tingting Han was born on December 27th, 1980 in Hangzhou, Zhejiang province, China. In 2003, she graduated with a bachelor degree in computer science from Nanjing University in Nanjing, China. In 2006 she received her master degree in computer science from the same university with a thesis titled "Research on the Software Coordination Based on Mobile Process Algebra", under the supervision of prof. dr. Jian Lu.

In November 2005, Han started her doctor's study in the Formal Methods and Tools (FMT) group in University of Twente and in the Software Modeling and Verification (MOVES) group in RWTH Aachen University. Under the supervision of prof. dr. ir. Joost-Pieter Katoen, she worked on probabilistic verification. Her work was carried out under the auspices of the QUPES (Verifying Quantitative Properties of Embedded Software) project supported by the Netherlands Organization for Scientific Research (NWO). The present dissertation contains the results of this work.

Titles in the IPA Dissertation Series since 2005

E. Ábrahám. *An Assertional Proof System for Multithreaded Java -Theory and Tool Support-* . Faculty of Mathematics and Natural Sciences, UL. 2005-01

R. Ruimerman. *Modeling and Remodeling in Bone Tissue*. Faculty of Biomedical Engineering, TU/e. 2005-02

C.N. Chong. *Experiments in Rights Control - Expression and Enforcement*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-03

H. Gao. *Design and Verification of Lock-free Parallel Algorithms*. Faculty of Mathematics and Computing Sciences, RUG. 2005-04

H.M.A. van Beek. *Specification and Analysis of Internet Applications*. Faculty of Mathematics and Computer Science, TU/e. 2005-05

M.T. Ionita. *Scenario-Based System Architecting - A Systematic Approach to Developing Future-Proof System Architectures*. Faculty of Mathematics and Computing Sciences, TU/e. 2005-06

G. Lenzini. *Integration of Analysis Techniques in Security and Fault-Tolerance*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-07

I. Kurtev. *Adaptability of Model Transformations*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2005-08

T. Wolle. *Computational Aspects of Treewidth - Lower Bounds and Network Reliability*. Faculty of Science, UU. 2005-09

O. Tveretina. *Decision Procedures for Equality Logic with Uninterpreted Functions*. Faculty of Mathematics and Computer Science, TU/e. 2005-10

A.M.L. Liekens. *Evolution of Finite Populations in Dynamic Environments*. Faculty of Biomedical Engineering, TU/e. 2005-11

J. Eggermont. *Data Mining using Genetic Programming: Classification and Symbolic Regression*. Faculty of Mathematics and Natural Sciences, UL. 2005-12

B.J. Heeren. *Top Quality Type Error Messages*. Faculty of Science, UU. 2005-13

G.F. Frehse. *Compositional Verification of Hybrid Systems using Simulation Relations*. Faculty of Science, Mathematics and Computer Science, RU. 2005-14

M.R. Mousavi. *Structuring Structural Operational Semantics*. Faculty of Mathematics and Computer Science, TU/e. 2005-15

A. Sokolova. *Coalgebraic Analysis of Probabilistic Systems*. Faculty of Mathematics and Computer Science, TU/e. 2005-16

M.W.A. Streppel. *Multifunctional Geometric Data Structures*. Faculty of Mathematics and Computer Science, TU/e. 2007-07

N. Trčka. *Silent Steps in Transition Systems and Markov Chains*. Faculty of Mathematics and Computer Science, TU/e. 2007-08

R. Brinkman. *Searching in encrypted data*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-09

A. van Weelden. *Putting types to good use*. Faculty of Science, Mathematics and Computer Science, RU. 2007-10

J.A.R. Noppen. *Imperfect Information in Software Development Processes*. Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2007-11

R. Boumen. *Integration and Test plans for Complex Manufacturing Systems*. Faculty of Mechanical Engineering, TU/e. 2007-12

A.J. Wijs. *What to do Next?: Analysing and Optimising System Behaviour in Time*. Faculty of Sciences, Division of Mathematics and Computer Science, VUA. 2007-13

C.F.J. Lange. *Assessing and Improving the Quality of Modeling: A Series of Empirical Studies about the UML*. Faculty of Mathematics and Computer Science, TU/e. 2007-14

T. van der Storm. *Component-based Configuration, Integration and Delivery*. Faculty of Natural Sciences, Mathematics, and Computer Science,UvA. 2007-15

B.S. Graaf. *Model-Driven Evolution of Software Architectures*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2007-16

A.H.J. Mathijssen. *Logical Calculi for Reasoning with Binding*. Faculty of Mathematics and Computer Science, TU/e. 2007-17

D. Jarnikov. *QoS framework for Video Streaming in Home Networks*. Faculty of Mathematics and Computer Science, TU/e. 2007-18

M. A. Abam. *New Data Structures and Algorithms for Mobile Data*. Faculty of Mathematics and Computer Science, TU/e. 2007-19

W. Pieters. *La Volonté Machinale: Understanding the Electronic Voting Controversy*. Faculty of Science, Mathematics and Computer Science, RU. 2008-01

A.L. de Groot. *Practical Automaton Proofs in PVS*. Faculty of Science, Mathematics and Computer Science, RU. 2008-02

M. Bruntink. *Renovation of Idiomatic Crosscutting Concerns in Embedded Systems*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2008-03