TECHNISCHE UNIVERSITÄT MÜNCHEN
Lehrstuhl für Informatik VII

Verification of Discrete- and Continuous-Time Non-Deterministic Markovian Systems

Jan Křetínský

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Helmut Seidl

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Dr. h.c. Francisco Javier Esparza Estaun
2. Univ.-Prof. Dr. ir. Joost-Pieter Katoen
   Rheinisch-Westflische Technische Hochschule Aachen

Die Dissertation wurde am 10.07.2013 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 19.10.2013 angenommen.

Abstract

This thesis deals with verification of probabilistic systems both in the setting of discrete time (e.g. Markov decision processes) and of continuous time (e.g. continuous-time stochastic games, interactive Markov chains).

In the discrete-time setting, we design a novel method to translate formulae of linear temporal logic to automata and show how this speeds up the verification process. This enables us, for instance, to verify network protocols with more participants in situations where this was previously hopelessly infeasible.

In the continuous-time setting, we give algorithms to compute or approximate probabilities to reach a given state within a given time bound. Moreover, we also consider the setting where the system under verification operates in an unknown environment or in an environment conforming to a given specification given in our new formalism. This enables us, for instance, to verify a component of a system without knowing the code of other components of the system, only with the knowledge of some guarantees on their behaviour. This is the first compositional approach to verification and assume-guarantee reasoning method in the probabilistic continuous-time setting.

Attached to this thesis, there are papers published at conferences CAV 2012, ATVA 2012, FSTTCS 2012, CAV 2013, ATVA 2013 and CONCUR 2013 and in the journal Information and Computation in 2013.

Zusammenfassung

Diese Arbeit beschäftigt sich mit Verifikation von probabilistischen Systemen sowohl in diskreter Zeit (z.B. Markov Entscheidungsprozesse) als auch in kontinuierlicher Zeit (z.B. stochastische Spiele in kontinuierlichen Zeit, interaktive Markov Ketten).

Bezüglich diskreter Zeit entwerfen wir eine neuartige Methode, um Formeln der linearen zeitlichen Logik in Automaten übersetzen, und zeigen, wie diese den Verifikationsprozess beschleunigt. Dies ermöglicht es uns zum Beispiel, Netzwerk-Protokolle mit mehr Teilnehmer zu überprüfen, auch in Situationen in denen dies bisher hoffnungslos unmöglich war.

Bezüglich kontinuierlicher Zeit geben wir Algorithmen die Wahrscheinlichkeiten von der zeitgebundenen Erreichbarkeit berechnen oder annähern. Diese sind Wahrscheinlichkeiten, dass ein gegebene Zustand innerhalb gegebener Zeit erreicht ist. Darüber hinaus betrachten wir den Fall, dass das System zum Verifizieren in einer unbekannten Umgebung oder in einer Umgebung, die eine gegebenen Spezifikation in unserem neuen Formalismus erfüllt, laüft. Dies ermöglicht es uns zum Beispiel, eine Komponente eines Systems, ohne den Code der anderen Komponenten des Systems zu kennen, sondern nur mit einiger Garantien für ihr Verhalten, zu verifizieren. Dies ist das erste kompositorische Verifikationsverfahren und assume-guarantee Argumentationsverfahren für probabilistische Systemen in kontinuierlicher Zeit.

Eingebunden in dieser Dissertationsarbeit befinden sich Papiere die in den Konferenzen CAV 2012, ATVA 2012, FSTTCS 2012, CAV 2013, ATVA 2013 und CONCUR 2013 und in der Zeitschrift Information and Computation im Jahr 2013 veröffentlicht wurden.

Acknowledgements

First, I would like to thank all co-authors of the papers appended to this thesis. These are the following amazing people: Javier, my advisor here in Munich, always helpful, cheerful and friendly, I am glad I could learn from this wise man; Tony and Tomáš, my consultants in Brno, who taught me my first steps, tried to teach me that more is always more and were always in for good food; Krish, my future advisor in Vienna, smiling, efficient and always in for a Greek fish; my dear office mates and friends Andreas, Ruslán and Jeňa in Brno, each of whom was always ready to talk with me and help me, they made me survive and enjoy my stay in a foreign country and I cherish the moments spent with them, luckily too many to be listed; the two Vojta's, 48, who abandoned any Botswanian ambitions and grew content with Oxford, and Řehák, who likes it the harder way in Brno; Holger, the best homeless dean ever, who knows where to spend the winter and how to shelter his guests properly.

I also want to thank all my colleagues from Lehrstuhl VII for the atmosphere they created here, the Mensa lunches where I have learnt German and the nice 4 years at TUM. Last, but not least, I want to thank my beloved parents for their unconditional support and care, and my dearest Zuzka for her love and for making Garching a rustic paradise for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cell phones, electric networks, water pumps, medical equipment, airplanes, intelligent houses and autonomous cars are all examples of *cyber-physical systems* surrounding us with an increasing intensity. They are systems whose operation in a physical environment is controlled by an embedded computational core. They have arisen from *service-oriented systems*, large and distributed systems providing services supporting machine-to-machine interaction over network, by incorporating a number of *embedded systems*, small and closed systems autonomously controlling complex physical systems [mtl11]. Studying these complex systems is a challenging and important task also for computer science.

Since cyber-physical systems operate in physical environment, it is vital to guarantee they operate safely and efficiently in real-time. How can safe behaviour ("the steering system never crashes the car") and acceptable performance ("the internet service is available 98 % of time") be ensured?

- In the first step, one can build and validate *models* of these systems. Due to the systems' interaction with the physical environment, appropriate modelling languages need to feature *probabilities* to cope with uncertainties, noisy inputs or failing components; and often also *continuous-time* to capture random delays in real time such as customers' arrivals, message transmission time or time to failure. These features are often called *quantitative* due to their use of real values as opposed to the traditional discrete setting based on truth values.
- In the second step, one can then *verify*, possibly automatically, the desired properties (called *specifications*) of the models using formal methods.

Cyber-physical systems have been identified as a key area of research by many funding organisations such as the US National Science Foundation (see [Wol07]). Many current projects develop and improve theories and methods for modelling and verification of these complex systems. As to the recent projects in the

Germanic region, there are, for instance, AVACS (Automatic Verification And Analysis of Complex Systems; universities of Oldenburg, Freiburg, Saarland et al. [ava04]), ROCKS (RigorOus dependability analysis using model ChecKing techniques for Stochastic systems; universities of Aachen, Dresden, Saarland, Munich, Nijmegen, Twente [roc09]), QUASIMODO (Quantitative System Properties in Model-Driven-Design of Embedded Systems; universities and institutes in Aalborg, Eindhoven, Nijmegen, Twente, Cachan, Aachen, Saarland, Brussels, and industrial partners [qua08]), MT-LAB (Modelling of Information Technology; universities of Copenhagen and Aalborg [mtl09]), IDEA4CPS (Foundations fro cyber-physical systems; universities of Copenhagen and Aalborg and in China [ide11]), to name just a few.

Further, many top conferences deal with various aspects of cyber-physical systems in general using methods of control theory, robotics and applied mathematics like ICCPS (International Conference on Cyber-Physical Systems), HSCC (Hybrid Systems: Computation and Control), CDC (Conference on Decision and Control), EMSOFT (International Conference on Embedded Software), or RTAS (Real-Time and Embedded Technology and Applications Symposium). Moreover, theoretical computer science can also offer theory and techniques for specification and verification of such systems. Indeed, many renown conferences extend their scope to analysis of quantitative models, e.g. LICS, CONCUR, CAV, ATVA, POPL, FSTTCS, or focus purely on this area, e.g. QEST (Conference on Quantitative Evaluation of SysTems), FORMATS (Formal Modelling and Analysis of Timed Systems).

In this thesis, we improve theory and tools for specification and verification of such quantitative systems. As foreshadowed, we have to deal with probabilities and often with continuous-time, too. Nevertheless, we also deal with models where time is not continuous, but simply advances in discrete steps. The reason is twofold. Firstly, one can often reduce the continuous-time case to the discrete one and then more easily solve the latter. Secondly, in many applications such as randomised protocols or distributed randomised algorithms, the precise timing is irrelevant for checking many properties such as mutual exclusion property or eventual election of the leader.

We focus on some of the most common modelling frameworks featuring probabilities and (continuous or discrete) time, namely on *Markov chains* and their extensions such as *Markov decision processes* and *stochastic games*. We improve some existing methods and provide new ones to cope with the demanding task of analysis of these models.

Markov chains [Mar06, Nor98, MT09] are the most successful class of stochastic processes and are for decades heavily used not only in computer science, but also in biology (population models, genetics), operations research (queueing theory), chemistry (kinetics of reactions), physics (thermodynamics), economics (price models), or social sciences (regime-switching models) and are used in many applications as diverse as internet searching, speech recognition, baseball analysis, gambling and algorithmic music composition.

Markov decision processes [Bel57, How60, Put94, FV96] are an extension of Markov chains, where the process can also be controlled. Apart from computer science, they are used in robotics, automated control, economics, and manufacturing to solve optimisation problems using methods of linear programming, dynamic programming or reinforcement learning. Stochastic games [Sha53, FV96, NS03] extend Markov decision processes to a competitive case where the control is divided between two antagonistic players. Their applications range from economics over evolutionary biology to cyber-physical systems.

1.1 State of the art

Research on Markov chains, Markov decision processes and their extension has traditionally been focused on *long run properties*, which describe how the system performs on average when its behaviour "stabilises" after running "long enough". An example of a typical property is "the internet service is available 98 % of time". Efficient techniques to compute or approximate this kind of performance have been developed, see e.g. [Put94, Bré99].

However, many properties remain out of scope of this most traditional framework. For instance, questions like "How long is the service continuously unavailable?" or "How likely is it that the service fails today?" require to inspect the behaviour of the system until a *finite time horizon*, i.e. to analyse what the system does in a given time. Such analysis is often also called *transient* because it describes what the system goes through at a given time point). Since satisfaction of these properties is vital in safety-critical systems, this topic has attracted a lot of recent attention triggered by seminal papers [ASSB96] and [BHHK04], for more details see Chapter 4. However, decidability of many optimisation problems remains open and approximation techniques only start to mature. Furthermore, the analysis is often limited to systems that operate in a known environment despite the effort to transfer compositionality principles to this setting. In this thesis, we establish decidability and approximability of several optimisation problems and provide algorithms also for the setting where the environment is unknown or only some of its properties are known.

Another type of properties interesting from the verification point of view are *temporal properties*, which describe ordering of events in the system rather than their frequency. A typical example is "whenever a service is requested it is eventually almost surely provided without any idling inbetween". These properties—often expressed in *temporal logics*—have a long tradition in computer science [Pnu77] and proved appropriate for describing systems' desired behaviour. Methods for verification of these properties are very mature, see e.g. [BK08]. However, in the probabilistic setting, the running times of the methods are often impractical. Furthermore, in the continuous-time setting, the current logics are often inappropriate for description of interesting properties. In this thesis, we improve on the existing methods in the probabilistic setting

and provide a new formalism for specifying properties in the continuous-time setting.

1.2 Contribution of the thesis

Here we only give a very brief and high-level account on the contribution of the thesis. For a technical summary, we refer the reader to Chapter 5.

In the discrete-time setting, we have designed a novel method to process temporal properties [KE12, GKE12, KG13] and show how this speeds up the verification process [CGK13]. This enables us, for instance, to verify network protocols with more participants in situations where this was previously hopelessly infeasible. For more details on these results, please see Chapter 3. Part I of Appendix then provides the co-authored papers cited above as Paper A, B, D, and C as published at conferences CAV 2012, ATVA 2012, ATVA 2013, and CAV 2013, respectively.

In the continuous-time setting, we give algorithms to compute or approximate probabilities on which the transient analysis depends [BFK$^+$13, BHK$^+$12, HKK13]. Moreover, we also consider the setting where the system under verification operates in an unknown environment or in an environment conforming to a given specification given in our new formalism. This enables us, for instance, to verify components of a system without knowing the code of other components of the system, only with the knowledge of some guarantees on the their behaviour. This is the first compositional verification and assume-guarantee reasoning in the probabilistic continuous-time setting. For more details on these results, see Chapter 4. Part II of Appendix then provides the co-authored papers cited above as Paper E, F, and G as published in journal Information and Computation (2013) and at conferences FSTTCS 2012, and CONCUR 2013, respectively.

1.3 Publication summary

In Part I of Appendix, we present the following papers:

A Jan Křetínský and Javier Esparza. Deterministic automata for the (F,G)-fragment of LTL. *CAV*, 2012. [KE12]

B Andreas Gaiser, Jan Křetínský, and Javier Esparza. Rabinizer: Small deterministic automata for LTL($\mathbf{F}, \mathbf{G}$). *ATVA*, 2012. [GKE12]

C Krishnendu Chatterjee, Andreas Gaiser, and Jan Křetínský. Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis. *CAV*, 2013. [CGK13]

D Jan Křetínský and Ruslán Ledesma Garza. Rabinizer 2: Small deterministic automata for $\mathrm{LTL}_{\setminus \mathbf{GU}}$. *ATVA*, 2013. [KG13]

In Part II of Appendix, we present the following papers:

E Tomáš Brázdil, Vojtěch Forejt, Jan Krčál, Jan Křetínský, and Antonín Kučera. Continuous-time stochastic games with time-bounded reachability. *Information and Computation*, 2013. [BFK+13]

F Tomáš Brázdil, Holger Hermanns, Jan Krčál, Jan Křetínský, and Vojtěch Řehák. Verification of open interactive Markov chains. *FSTTCS*, 2012. [BHK+12]

G Holger Hermanns, Jan Krčál, and Jan Křetínský. Compositional verification and optimisation of interactive Markov chains. *CONCUR*, 2013. [HKK13]

1.3.1 Other co-authored papers

For the sake of completeness, apart from the presented papers we also list other co-authored papers, which are, however, not a part of this thesis.

- Tomáš Brázdil, Vojtěch Forejt, Jan Křetínský, and Antonín Kučera. The satisfiability problem for probabilistic CTL. *LICS*, 2008. [BFKK08]

Papers on non-Markovian continuous-time systems

- Tomáš Brázdil, Jan Krčál, Jan Křetínský, Antonín Kučera, and Vojtěch Řehák. Stochastic real-time games with qualitative timed automata objectives. *CONCUR*, 2010. [BKK+10]
- Tomáš Brázdil, Jan Krčál, Jan Křetínský, Antonín Kučera, and Vojtěch Řehák. Measuring performance of continuous-time stochastic processes using timed automata. *HSCC*, 2011. [BKK+11]
- Tomáš Brázdil, Jan Krčál, Jan Křetínský, and Vojtěch Řehák. Fixed-delay events in generalized semi-Markov processes revisited. *CONCUR*, 2011. [BKKŘ11]
- Tomáš Brázdil, Luboš Korenčiak, Jan Krčál, Jan Křetínský, and Vojtěch Řehák. On time-average limits in deterministic and stochastic Petri nets. *ICPE*, 2013. [BKK+13] (poster paper)

Papers on modal transition systems

- Nikola Beneš, Jan Křetínský, Kim G. Larsen, and Jiri Srba. EXPTIME-completeness of thorough refinement on modal transition systems. *Information and Computation*, 2012. [BKLS12]. Extended journal version of an *ICTAC 2009* conference paper. [BKLS09a]

- Nikola Beneš, Jan Křetínský, Kim Guldstrand Larsen, and Jiri Srba. On determinism in modal transition systems. *Theoretical Computer Science*, 2009. [BKLS09b]
- Nikola Beneš and Jan Křetínský. Process algebra for modal transition systemses. *MEMICS*, 2010. [BK10]
- Nikola Beneš, Ivana Černá, and Jan Křetínský. Modal transition systems: Composition and LTL model checking. *ATVA*, 2011. [BčK11]
- Nikola Beneš, Jan Křetínský, Kim G. Larsen, Mikael H. Moller, and Jiri Srba. Parametric modal transition systems. *ATVA*, 2011. [BKL$^+$11]
- Nikola Beneš, Jan Křetínský, Kim Guldstrand Larsen, Mikael H. Moller, and Jiri Srba. Dual-priced modal transition systems with time durations. *LPAR*, 2012. [BKL$^+$12]
- Nikola Beneš and Jan Křetínský. Modal process rewrite systems. *ICTAC*, 2012. [BK12]
- Jan Křetínský and Salomon Sickert. On refinements of boolean and parametric modal transition systems. *ICTAC*, 2013. [KS13a]
- Nikola Beneš, Benoît Delahaye, Uli Fahrenberg, Jan Křetínský, and Axel Legay. Hennessy-Milner logic with greatest fixed points as a complete behavioural specification theory. *CONCUR*, 2013. [BDF$^+$13]
- Jan Křetínský and Salomon Sickert. MoTraS: A tool for modal transition systems and their extensions. *ATVA*, 2013. [KS13b] (tool paper)

1.3.2 Summary

Altogether Jan Křetínský has published 3 journal papers (2×Information and Computation, Theoretical Computer Science), 16 conference regular papers (2×ATVA, 2×CAV, 4×CONCUR, 2×FSTTCS, HSCC, 3×ICTAC, LICS, LPAR), 3 conference tool papers (3×ATVA), 1 workshop paper (MEMICS) and 1 conference poster paper (ICPE), altogether 24 papers. The current number of citations according to Google Scholar is 164.

1.4 Outline of the thesis

Chapter 2 introduces the investigated models, properties and problems. We first introduce the stochastic processes in Section 2.1 and then their decision and game extensions in Section 2.2. We proceed with the properties of interest in Section 2.3. We only give a semi-formal account; for technical definitions, please consult the respective papers in Appendix. Section 2.4 closes with an overview of the verification problems for the introduced models and properties, and their

complexities. The state of the art as well as our contribution are described in more detail (1) in the discrete-time setting in Chapter 3 and (2) in the continuous-time setting in Chapter 4. Chapter 5 summarises the contribution of the thesis and new results of the respective papers. The appendix has three parts. In the first and the second part, we present preprints of Papers A-D and E-G, respectively. The third part then provides several supplementary materials on the context of our research and permissions to publish the preprints of the papers within this thesis as they were sent to the publisher.

The reader is assumed to have some familiarity with labelled transition systems, probability theory, logic, complexity and automata over infinite words to the extent of standard basic courses.

Chapter 2

Quantitative models, properties and analysis

In this chapter, we introduce the models, properties and analysis problems we are interested in. Our results are concerned with different models, e.g. continuous-time stochastic systems, discrete-time stochastic systems, non-stochastic games, and with different properties, e.g. untimed linear time properties, time-bounded properties. Despite this variety of topics, we want to present a unifying view on our results. Therefore, we present quite a general framework unifying discrete-time and continuous-time systems in Section 2.1 and extend it with non-determinism to decision processes and games in Section 2.2. Section 2.3 then gives a unifying view on the properties of interest. Section 2.4 provides an overview of the analysis problems for the described systems and properties.

As we deal with many different objects, we mostly refrain from defining them completely formally and stick to semi-formal descriptions. For technical definitions, we refer to the respective papers in Appendix; for more extensive introductions to the topics, please consult the respectively cited sources.

2.1 Stochastic processes and Markov chains

Systems that we are considering do not evolve deterministically, but randomly. Hence, in order to describe them we make use of their formalisation through stochastic processes. A stochastic process is at each time point in a particular state and can change its state at any time point to any other state according to a probability distribution, which may depend on the previously visited states. Formally, given

- a set S of *states* equipped with a measurable space (S, Σ),

- a set $\mathbb{T}$ of *time* points equipped with a total ordering $<$ and its smallest element 0,
- a set Ω of *behaviours* equipped with a probability space $(\Omega, \mathcal{F}, \mathbf{P})$,

a *stochastic process* is a collection $\{X_t \mid t \in \mathbb{T}\}$ of random variables $X_t : \Omega \to S$.

In this thesis, we mainly focus on stochastic processes corresponding to finite transition systems, which makes them easier to analyse than infinite systems. To ensure that, we require that

(F) the set S of states be *finite*,

(M) the process satisfy the *Markov property*, i.e. its current state distribution depends only on its most recent configuration (its state and time), formally

$$\mathbf{P}[X_{t_n} = x_{t_n} \mid X_{t_{n-1}} = x_{t_{n-1}}, \ldots, X_{t_0} = x_{t_0}] = \mathbf{P}[X_{t_n} = x_{t_n} \mid X_{t_{n-1}} = x_{t_{n-1}}]$$

for any $t_n > t_{n-1} > \cdots > t_0$,

(H) time $\mathbb{T}$ be equipped with (associative and commutative) addition $+$ and the process be *time homogeneous*, i.e. probabilities to change the state from one to another do not change over time, formally

$$\mathbf{P}[X_{t+d} = x \mid X_t = y] = \mathbf{P}[X_{t'+d} = x \mid X_{t'} = y]$$

for any $t, t', d \in \mathbb{T}$.

We call such stochastic processes *Markov chains* [Nor98]. They can be equivalently represented by the set S of states with an initial distribution μ on S and a transition function P assigning to each state a probability measure over timed transitions to the subsequent states, written $P : S \to \mathfrak{P}(\mathbb{T} \times S)$. Behaviours of the chain then correspond to *runs* starting in a state randomly chosen according to μ and then changing states according to P. This representation is thus much closer to the *labelled transition systems* (LTS), a de facto standard in computer science.

In this thesis, the stochastic processes considered are Markov chains. The only deviations can be be found in Paper E, where we sometimes relax Requirement (F) and only demand countability of S. A comment how realistic this is and what the limitations are can be found in Appendix H.*

2.1.1 Discrete-time Markov chains

By setting the time domain $\mathbb{T}$ to the discrete set $\mathbb{N}_0$ of natural numbers, we obtain *discrete-time Markov chains* (DTMC). The time spent in a state is always 1 and then a transition is taken. The transition function P is thus just a probability transition matrix and the chain can be drawn as a finite directed

*For author's papers on non-Markovian systems, see Section 1.3.1.

graph with edges (i, j) labelled by $P(i, j)$. In the following, we will mostly use this representation.

The respective probability space $(\Omega, \mathcal{F}, \mathbf{P})$ over the set $\Omega := S^{\mathbb{T}}$ of runs is then generated by

$$\mathbf{P}[X_0 = i_0, X_1 = i_1 \dots X_n = i_n] = \mu(i_0)P(i_0, i_1)P(i_1, i_2) \cdots P(i_{n-1}, i_n)$$

and the cylindrical construction using Carathéodory's extension theorem [Bil12].

2.1.2 Continuous-time Markov chains

By setting the time domain $\mathbb{T}$ to the continuous set $\mathbb{R}_{\geq 0}$ of non-negative real numbers, we obtain *continuous-time Markov chains* (CTMC). They can be represented by a discrete-time Markov chain together with a function $R : S \to \mathbb{R}_{\geq 0}$ assigning a *rate* to each state. Alternatively, we also use the representation without R where edges (i, j) are labelled by $P(i, j)R(i)$, which clearly contains the same information.

The chain then evolves in a manner similar to the discrete-time Markov chain. The only difference is that a *random* time is spent in a state before a transition to the next one is taken. Due to the Markov property (M), the time spent in state s is given by a negative exponential distribution. More precisely, the probability that s is left within time t is $1 - e^{-R(s)\cdot t}$. For more details, see Paper E.

2.2 Non-deterministic systems

Markov chains have been a successful modelling framework not only in computer science, but also in operation research, telecommunication or biology. Huge effort has been invested to deal with Markov chains. There are many tools for analysis of Markov chains, e.g. PEPA [TDG09], PRISM [KNP11], MRMC [KZH+09] to name a few commonly used in computer science, as well as toolboxes for computing environments such as Matlab [MAT13] or Mathematica [Mat12], or countless simulation tools. These tools have been applied in real practice to analyse e.g. network, embedded, or chemical and biological systems.

Using Markov chains, one can model closed systems (with no inputs from the environment, but stochastic ones) and answer questions such as "What is the expected response time of a service?" or "What is the probability of failure before the next scheduled maintenance?". However, as cyber-physical systems often communicate with the user and are distributed, such features as communication with the surrounding physical environment, I/O data flow, synchronisation between components, resource contention and hierarchical composition must be

present in the framework. These features are added uniformly by a new kind of transitions labelled by letters of an *alphabet* A, which is in our case again required to be finite. Presence of such an outgoing transition under a letter $a \in A$ denotes the system is ready to synchronise on the *action* a. We say a is *available* or *enabled*. If all participating systems have a enabled they can jointly perform it and synchronously change their states to the targets of their respective a transitions. We call the resulting system a *composition* of the original systems. Apart from these new "synchronising" actions, the systems still may have "internal" (non-synchronising) action, which are performed by each system alone without any interaction.

Whenever two actions can be performed at the same time, a non-deterministic decision must be made. This means that some external entity resolves this conflict. This entity is usually called a *scheduler* or a *policy* or in the game-theoretical context a *strategy*. Based on what happened so far, the scheduler decides which of the enabled actions shall be performed. Since Markov chains are discrete-event systems, the current history of what has happened is just a finite sequence of configurations of the system. Formally, the scheduler is then a (measurable) function $\sigma : (S \times \mathbb{T})^* \to \mathfrak{P}(A)$ returning a distribution on actions as we allow the scheduler to choose an action randomly, too. The semantics of the non-deterministic process M is then a set of Markov chains M^σ generated by applying the decisions of each scheduler σ.

These non-deterministic processes are often called *decision processes*. The non-determinism in the decision processes can model many useful features:

- If the distribution on the environmental inputs is *unknown* or properties need to be guaranteed while not relying on the knowledge of the distribution, the interaction with the environment can be modelled non-deterministically.
- In many plants, there are more options how to react to the input. The optimisation task is then to choose among the possible responses so that the plant is *controlled* in the most efficient way with respect to given criteria.
- Systems and algorithms are often *underspecified* and leave more options open, e.g. how to search a tree or pop from a set.
- Whenever several components are composed and they run *concurrently*, it may be out of control in which order these independent components change their states. They can even run on a single core and then a scheduling policy must determine in which order they are executed.

Now that we know that the semantics of a decision process M is the set $\{M^\sigma \mid \sigma \text{ is a scheduler}\}$, how do we interpret that? When does a decision process satisfy a property? For systems *without* non-determinism, whenever we have a property ϕ of e.g. a probabilistic logic, then the stochastic process (Markov chain) either satisfies the property (denoted $M \models \phi$) or not. The *model checking*

problem is then to decide which of the two is the case. For *non-deterministic* systems either all chains from the set satisfy ϕ, or none of them, or some do and some do not. The *generalised model checking* problem is then to decide which of the three happens and if the last option is the case then also to synthesise witnessing schedulers. Furthermore, we are mostly interested in one of the two specialised questions:

1. $\forall\sigma : M^\sigma \models \phi$?
2. $\exists\sigma : M^\sigma \models \phi$?

We ask the first question if the non-determinism is *demonic* (uncontrollable, adversarial). This means we cannot affect the choices and to be safe we have to count on the worst. An example is the first bullet with the unknown inputs from the user where we cannot assume anything unless we want to end up in a court.

The second question is posed in the case with *angelic* (controllable) non-determinism. Here we assume we can control the choices. If the answer is yes we also want to obtain a scheduler that achieves the goal ϕ so that we can implement it into the plant. An example is the second bullet.

The third and fourth bullets are examples of non-determinisms that are angelic or demonic depending on the setting. Furthermore, it may be necessary to have both angelic and demonic non-determinisms in our system if there are both controllable and uncontrollable components. Such a system is then called a *game*. In order to obtain a Markov chain, we need to apply a pair (σ, π) of schedulers: the first being angelic and the latter demonic. We usually ask the question whether we can control the controllable parts so that no matter what happens in the uncontrollable part, the property is satisfied. Formally we ask

3. $\exists\sigma : \forall\pi : M^{\sigma,\pi} \models \phi$?

If the answer is positive, we again want to *synthesise* the respective scheduler σ. For this reason, the task to decide the model checking question and compute the respective scheduler is often called *synthesis*. In the discrete-time setting, this question has been posed already in [Chu57]. The alternative question with quantifiers swapped is not that much interesting as the scheduler of the environment rarely becomes known to the scheduler of the plant. In this thesis we deal with decision processes as well as games. For an overview of the relationships among the discussed systems, see Figure 2.1 and 2.2.

2.2.1 Markov decision processes and games

There are traditional models (see e.g. [Put94, FV96]) of *discrete-time Markov decision processes* (MDP) and their game extension as well as *continuous-time Markov decision processes* (CTMDP) and their new game extension of ours. In all these models, the synchronisation and random steps strictly alternate and

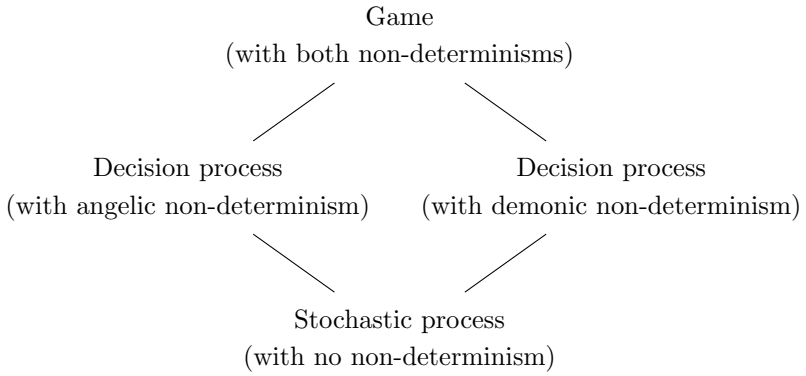


Figure 2.1: Processes with zero, one and two non-determinisms

hence are merged into one step, which often simplifies argumentation. Formally, they consist of the set S of states and the initial distribution μ similarly to Markov chain, but the transition function is now $P : S \times A \to \mathfrak{P}(S)$. After visiting states $s_0 s_1 \cdots s_n$, a transition to a state s is taken with probability

$$\sum_{a \in A} \sigma((s_0, 0)(s_1, 1) \cdots (s_n, n))(a) \cdot P(s_n, a)(s)$$

The same holds for games where we, moreover, distinguish which state belongs to which player and depending on that we take decision of σ or π into account. In the discrete-time case, we call them *stochastic games* (SG) and in the continuous-time case *continuous-time stochastic games* (CTSG).

2.2.2 Interactive Markov chains and games

CTMDP are a simple and useful framework as long as composition is not involved. Further, the process algebras designed for description of Markov chains, such as Hillston's PEPA[Hil96], are compositional, but the result often does not faithfully reflect the real behaviour. Indeed, in PEPA the rate of a synchronisation is set to be the minimum of the synchronising rates in order to model that the faster process waits for the slower one. However, this is imprecise if both run at a similar speed and thus not finish in a fixed order, see e.g. [Her02].

To bridge the gap between CTMDP and compositional methodology, *interactive Markov chains* (IMC) [HHK02] with their corresponding process algebra have been designed. This allows for building complex Markov models in a compositional, hierarchical way. Technically, it is sufficient to strictly separate the synchronising immediate transitions from the random timed transitions and *not* require alternation. Formally, the states are partitioned into Markovian states with transitions as in CTMC, and immediate states with the transition function

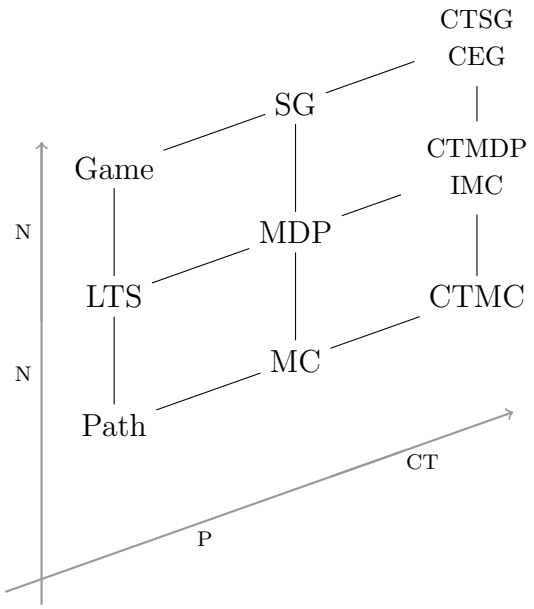


Figure 2.2: Processes with non-determinism (N), Markovian probabilities (P), and continuous time (CT).

$S \times A \to S$.[†]

In the same way, we can extend this concept to games. However, the reader might ask whether timing should not at least partially be under control of the schedulers. For instance, the user may delay the input as long as he likes; the plant might also take its time to choose its response. This is partially developed in Papers F and G in the form of *controller-environment games* (CEG) where timing is partially under control of the demonic non-determinism. However, the model where time is governed by both non-determinisms remains out of the scope of this thesis. Nevertheless, let us at least note that problems over such games become very easily undecidable [BF09].

2.3 Quantitative properties

The properties to be checked or optimised range from verification questions such as "What is the probability of visiting an unsafe state?" to performance questions "What is the average throughput?". However, as argued e.g. in [BHHK10], verification and performance evaluation have become very close in the setting of

[†]Here we employ the maximum progress assumption and thus ignore any stochastic continuous-time transitions if internal transitions are enabled, see Paper F. Therefore, no "mixed" states are present.

stochastic systems. Indeed, consider a simple system repeatedly sending messages. When asking "What is the probability that a message gets lost on the way?" it is both a question of quantitative verification as well as a question on average performance of the protocol. Further, liveness-like question as "something good eventually happens" are often bounded by a finite horizon under consideration, which turns them into safety questions. The desirable properties of safety, reliability, dependability and performance thus go hand in hand here.

Although these are all temporal properties and thus are evaluated based on monitoring the system over time, we may still fuzzily distinguish two kinds of properties:

- *Cumulative* properties are closer to performance measures. They stress the total portion of time (discrete or continuous) spent in states more than the order of visits. Typical properties are concerned with averages, sums or discounted sums of rewards collected during the run. (Note that this includes the reachability property, i.e. that a given state of the system will eventually be reached with some probability.)
- *Structural* properties are closer to verification. The ordering of events happening is crucial, such as in "A packet is being resubmitted and no other action is taken until its reception is acknowledged". These properties are typically expressed using temporal logics. (Again note that reachability is an example of such properties.)

While the properties of the first kind usually evaluate a run to a real number, in the second case the range is often 0 and 1. The overall result is then a weighted sum or integral taken over the whole system. One can also examine the whole distribution function of the values. While this gives no more information in the second setting, a lot more additional information is yielded in the first case. (Indeed, consider a system where every other message of each user is lost compared to one where half of the users communicate perfectly and the other half not at all.)

While the former class has traditionally been in focus of stochastic analysis, in this thesis, we focus more on the latter class. However, as we investigate also the fundamental reachability questions, the results have impact for both classes.[‡]

2.3.1 Properties of linear and branching time

One of the most successful ways to express properties of systems are *temporal logics*. As opposed to predicate logic, they are tailored to this task and thus easier to use and easier to analyse. The main feature of these logics is the ability to talk about events subsequently happening over time in the system, and to do so using a couple of built-in specialised constructs. Let us present as an example the *linear temporal logic* (LTL) [Pnu77]. For us, this will be the

[‡]For author's papers on analysis of the properties of the former kind, see Section 1.3.1.

most important example for the discrete-time systems (or the continuous-time systems where precise timing is not important). A formula of this logic is given by the following syntax:

$$\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid \mathbf{X}\phi \mid \phi_1 \mathbf{U} \phi_2$$

where a ranges over propositions that can be atomically evaluated in a system's state to be true or false, e.g. "this is an error state" or "the system is ready to receive a request". As usual with logics, we have Boolean connectives. The most interesting part are temporal operators $\mathbf{X}$ and $\mathbf{U}$. A formula $\mathbf{X}\phi$ holds if φ in the next step of the computation (recall the time being discrete here). A formula $\phi_1 \mathbf{U} \phi_2$ holds if eventually ϕ_2 will hold for a moment and up to this moment ϕ_1 holds. This way we can formalise e.g. the property "A packet is being resubmitted and no other action is taken until it is acknowledged" mentioned above as $(\mathit{resubmit} \wedge \mathit{noother})\mathbf{U}\mathit{ack}$ using the appropriate atomic propositions. Further, we often use derived operators $\mathbf{F}$ and $\mathbf{G}$. Here $\mathbf{F}\phi$ holds iff $\mathbf{U}\phi$ holds, meaning ϕ will eventually hold at some point of time; and $\mathbf{G}\phi$ holds iff $\neg\mathbf{F}\neg\phi$ holds, meaning ϕ will hold from now forever.

A formula ϕ of LTL can thus be evaluated on a run to be true or false. Further, ϕ is said to hold in a (non-probabilistic) system such as LTS if *all* runs of the system satisfy ϕ. LTL is a logic of *linear time* as it expresses properties of runs. However, one may demand that the fact that "all runs from a state satisfy a property" is a first-order citizen of the logic. This leads to logics of *branching time* where the the most common logic used is computation tree logic (CTL) [CE81]. The syntax is very similar, but every operator is preceded by either $\forall$ meaning that all paths from the current state satisfy the property, or $\exists$ meaning there is a path from the current state satisfying the property. An example of a property expressible by CTL is "at every time point there is an action enabled leading us to a state where emergency holds" written as $\forall\mathbf{G}(\exists\mathbf{X}\mathit{emergency})$.

2.3.2 Probabilistic and timed extensions

As the logics discussed above are not able to cope with many quantitative properties of interest, they have been extended to handle probabilistic behaviour and precise timing.

Various temporal logics have been proposed capturing properties of probabilistic systems. The main idea is to interpret the same logics as for non-probabilistic systems now over probabilistic systems by replacing "all runs" with "almost all runs", i.e. "with probability 1". This probabilistic interpretation was introduced already in [LS83] for CTL and in [Var85] for LTL. However, the probabilistic quantification is only *qualitative* here. Using the probabilistic quantification and negations we can only express that something happens with probability $= 1$, > 0, < 1, or $= 0$. The full *quantitative* probabilistic extensions have been

considered only later in PCTL [HJ94] and probabilistic LTL model checking as in PRISM [KNP11] (often also called quantitative model checking). Here the probabilistic operator expresses that a property holds true with a probability in some interval. The syntax of probabilistic LTL formula is then

$$\mathbf{P}_{\in[a,b]}\ \phi$$

where $[a, b] \subseteq [0, 1]$ is an interval and ϕ is an LTL formula. The model checking problem for Markov chains is then to decide whether the probability $p := \mathbf{P}[\phi]$ that ϕ holds lies in the interval $[a, b]$. Furthermore, we are often rather interested in computing or approximating p. For a decision processes M, p is given as

$$\sup_{\sigma} \mathbf{P}^{M^\sigma}[\phi]$$

where σ ranges over all (or a subclass of) schedulers and $\mathbf{P}^{M^\sigma}$ is the probability measure $\mathbf{P}$ of the stochastic process M^σ. Of course, infimum can be considered, too. Further, the problem is generalised to games similarly as the generalised model checking is further generalised to question 3. in Section 2.2 and p is then

$$\sup_{\sigma}\inf_{\pi} \mathbf{P}^{M^{\sigma,\pi}}[\phi] \tag{2.1}$$

where π ranges over the adversarial schedulers as in Section 2.2. As usual in game theory, if a strategy achieves the extremum, it is called optimal; if it achieves an ε-neighbourhood of the extremum, it is called ε-optimal.

The same probabilistic extensions happen with PCTL where every operator is parametrised by an interval. Orthogonally, logics have been extended with time. CTL has been extended with time to TCTL [ACD90], while MTL [Koy90] is a timed extension of LTL. Here essentially the *time bounded reachability* (reachability until a given time) has been substituted for the standard reachability. More formally, the operators are now parametrised by a time interval. For instance,

$$\mathit{resubmit}\ \mathbf{U}_{[0,5]}\ \mathit{ack}$$

says that the acknowledgement must arrive within 5 time units (and be preceded by resubmissions up to that point).

Both probabilistic and timed extension of CTL is then covered in continuous stochastic logic (CSL) [BHHK00]. The operators are then parametrised by an interval (or inequality) both for probability and for time. A simple example is then a formula

$$\mathbf{P}_{\geq 0.9}\ \mathbf{F}_{[0,5]}\ \mathit{goal}$$

which expresses that one of the goal states is reached within 5 time units with probability at least 90 per cent. This property is called *time-bounded reachability*. In general, we want to decide whether for a given time T and set of goal states *goal*,

$$\mathbf{P}[\mathbf{F}_{[0,T]}\ \mathit{goal}]$$

lies in a given interval. We also consider the same approximation problem and game extension

$$\sup_{\sigma} \inf_{\pi} \mathbf{P}^{M^{\sigma,\pi}}[\mathbf{F}_{[0,T]}\ goal] \tag{2.2}$$

as above in (2.1). Computing (or at least approximating) time-bounded reachability is the core of checking whether any CSL formula holds and as such will be one of the topics of the thesis.

As CSL is a logic of branching time, when interpreting it over non-deterministic systems every formula is evaluated as infimum (or supremum) over all non-deterministic resolutions. This sometimes leads to counterintuitive results due to nested quantification over the schedulers. One solution is to perform the quantification *once* for the whole formula only, or in particular places only such as in stochastic game logic (SGL) [BBGK07]. This probabilistic logic is based on ATL [AHK97], an extension of CTL where strategy quantification is made explicit. Further, one could also consider a probabilistic version of timed extensions of linear time logics. To the best of our knowledge, MTL has not been considered in the probabilistic interpretation yet. However, deterministic timed automata can serve as the automata antipode of MTL and can be used to specify linear time properties. The probabilistic interpretation has been considered in [CHKM09, BKK+10, BKK+11]. The violation of the property "every *req* must be answered by a *resp* within 5 time units" can be expressed as a deterministic timed automaton (DTA) in Figure 2.3.2. A deterministic timed automaton is like a standard finite automaton equipped with clocks that keep on running, can be reset or checked against a constant when taking a transition. A run is accepted by a DTA if it reaches a final state at least once [BKK+10] or more generally infinitely often [CHKM09, BKK+10, BKK+11].

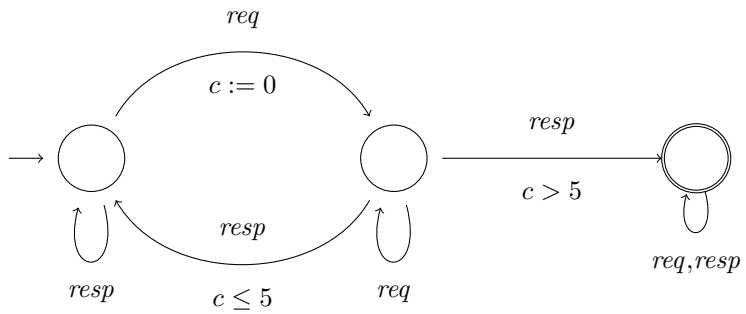


Figure 2.3: A deterministic timed automaton specification

2.4 Quantitative analysis

We now give a very brief overview of results on model checking the systems and properties introduced above. The purpose of this section is to sketch the

landscape of the verification and synthesis discipline and the positions of our results therein.

We consider the systems of Figure 2.2. These are non-stochastic, discrete-time stochastic and continuous-time stochastic systems each of which is considered in its deterministic, non-deterministic and game form. In the tables below, we denote them by 0, 1, and 2, respectively, according to the number of players. The problems we consider are model checking (1) non-stochastic systems with respect to reachability, LTL and CTL, (2) stochastic systems with respect to their probabilistic (both qualitative and quantitative) extensions, and finally we consider (3) quantitative continuous-time specifications. These are time-bounded reachability (TBR), deterministic timed automata (DTA), and continuous stochastic logic (CSL). For better readability, we display the complexity of the respective algorithms in the tables below. For more details on where and how these results were obtained, please see Appendix I.

Table 2.1: Non-stochastic analysis

	0	1	2
reachability	NL	NL	P
LTL	P	PSPACE	2-EXP
CTL	P	P	P

Observe the higher complexity of solving the LTL games. The problem is 2-EXP-complete [PR89] since one needs to construct a *deterministic ω-automaton* for the given LTL property ϕ (the reason for this will be explained in Chapter 3), which may require space doubly exponential in $|\phi|$. The very same difficulty occurs when LTL model checking stochastic decision processes (for MDP the problem is again 2-EXP-complete) and in quantitative model checking Markov chains.

Table 2.2: Qualitative stochastic analysis

	0	1	2
reachability	P	P	P
PLTL	PSPACE	2-EXP	3-NEXP$\cap$co-3-NEXP
PCTL	P	EXP	?

Table 2.3: Quantitative stochastic analysis

	0	1	2
reachability	P	P	NP$\cap$co-NP
PLTL	2-EXP	2-EXP	3-NEXP$\cap$co-3-NEXP
PCTL	P	undecidable	undecidable

Note that stochastic analysis also applies to continuous-time systems by simply

ignoring the timing. This is done by ignoring the rates R. For continuous-time properties, the qualitative analysis can safely ignore the timing aspect as the exponential distribution is supported on $[0, \infty)$. However, the results in quantitative analysis are much less encouraging. The main difficulty with computing exact probabilities is caused by the irrationality and transcendence of e, which is omnipresent due to the exponential distribution on the waiting times. Time-bounded reachability probability in a CTMC can be expressed and computed as an expression of the form $\sum q_i \cdot e^{r_i}$ where the sum is finite and q_i, r_i are rational [ASSB96]. The comparison of such a sum to a rational number is decidable using Lindemann-Weierstrass theorem, hence quantitative reachability and CSL model checking is decidable (ibid.). However, apart from that not much is known to be decidable. As a result, *approximations* are usually considered instead.

Table 2.4: Continuous-time analysis

	0	1	2
TBR	decidable	approximable	approximable
DTA	approximable	?	?
CSL	decidable	?	?

The focus of this thesis will be on

(1) quantitative probabilistic LTL model checking MDP and LTL model checking (non-stochastic) games, and

(2) quantitative time-bounded reachability in CTSG and IMC (which provides a base for CSL model checking as there we iteratively check reachability for the subformulae of the formula [ZN10]).

Since in (1) we are interested in optimising (closed or open) systems, we will also discuss how to obtain optimal schedulers. We improve the complexity both theoretically and practically. To achieve that, in Chapter 3 and in papers of Part I, we propose a new type of a deterministic ω-automaton for representing LTL properties that allows for more efficient analysis.

As for (2), we focus on optimisation of open systems. There are three different settings dealt with in the three papers of Part II. Namely, they are the case where the structure of the environment is known (only its decisions are unknown), the case with completely unknown environment, and the case where the environment is unknown but conforms to a known specification. We provide analysis techniques for all the cases and a specification formalism for the last one. As mentioned above, the area of continuous-time systems mostly relies on approximations. Therefore, although we give optimal schedulers in some cases, we mostly focus on synthesising ε-optimal schedulers.

Chapter 3

Discrete-time systems

In this chapter, we focus on the analysis of time-unbounded properties. The properties of interest are expressed as formulae of LTL. We consider both discrete-time and continuous-time systems. However, note that the precise waiting times are irrelevant for LTL properties. Therefore, instead of CTMDP we work with the "embedded" discrete-time MDP instead. The same holds for games. Here we thus focus on analysis of MDP and (non-stochastic) games.

3.1 State of the art

While model checking purely branching logics such as CTL can be done inductively, where each step amounts to optimising reachability probabilities, the case with LTL is more complex. The model checking task cannot be decomposed into checking subformulae and has to deal with the whole formula at once. Along the lines of the automata theoretic approach to verification [VW86], the negation of the formula ϕ to be verified is first transformed to an ω-automaton $\mathcal{A}$ to be multiplied with the system $\mathcal{S}$ and then checked for emptiness as described in Algorithm `AllRunsConform`$(\mathcal{S}, \phi)$ in Figure 3.1.

Algorithm `AllRunsConform`$(\mathcal{S}, \phi)$
1. $\mathcal{A} \leftarrow \omega$-automaton for $\neg\phi$
2. return $L(\mathcal{S} \otimes \mathcal{A}) = \emptyset$

Figure 3.1: Algorithm for non-probabilistic verification of LTL over transition systems

For systems without probabilities and without the second non-determinism, $\mathcal{A}$ is usually a (non-deterministic) Büchi automaton. Firstly, it is rich enough

to express LTL.* Secondly, it is guaranteed to be at most exponentially larger than ϕ. In contrast, deterministic Büchi automata cannot express the whole LTL, e.g. $\mathbf{FG}a$, and singly exponential procedure cannot exist for deterministic Muller or deterministic Rabin automata [KR10]. This approach is graphically summarised in Figure 3.2.

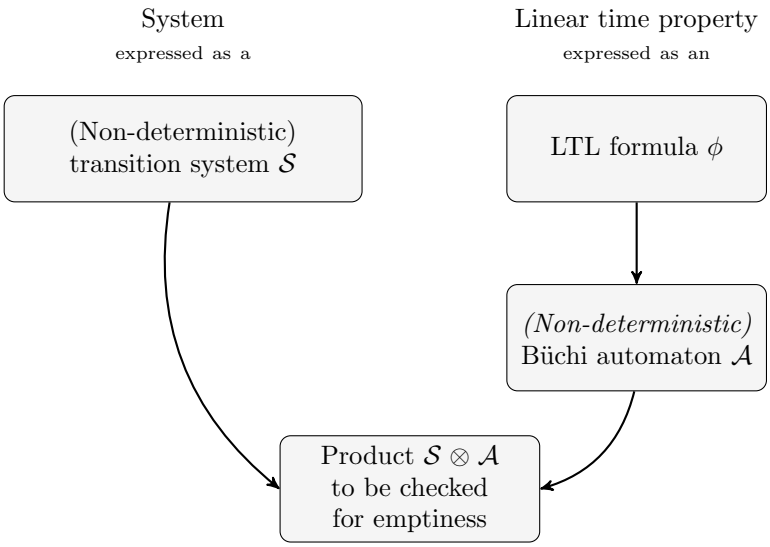


Figure 3.2: Automata-theoretic approach to non-probabilistic verification

However, checking systems with probabilities or with two players requires $\mathcal{A}$ be *deterministic*. Intuitively, the non-determinism of $\mathcal{A}$ is neither angelic nor demonic as it is not resolved during the run, but only after the infinite run of the system is created. Therefore, when a Markov chain is multiplied with a Büchi automaton, the resulting MDP cannot be played as a "pebble" game, where non-determinism is always resolved in each step. Let us illustrate this on a concrete example. Consider the Markov chain on the left and the Büchi automaton on the right of Figure 3.3.

*It can express LTL, i.e. first order properties, as it can even express all ω-regular properties, i.e. monadic second order properties.

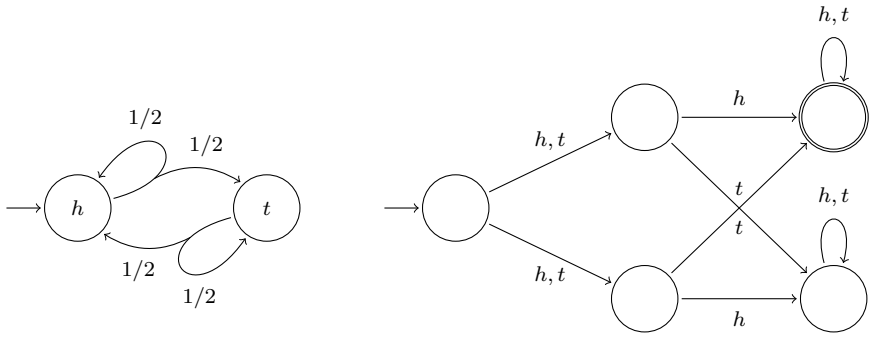


Figure 3.3: An example showing why non-deterministic automata cannot be used in Algorithm `AllRunsConform`$(\mathcal{S}, \phi)$ for probabilistic systems. The example is independent of whether a state or action based logic is used. For simplicity, we use state-based notation where h is assumed to hold in the left state and t in the right one.

The Markov chain produces a run by repeating coin tosses and announcing head or tail. The Büchi automaton recognises the whole $\{h, t\}^\omega$, i.e. any run. However, in the product MDP, half of the runs are rejected. Indeed, after reading the first h (no matter which transition is taken in the automaton), in the next step either h only or t only will be available. No matter whether we are in the upper or the lower branch of the automaton, with 50 % chance, the only available transition will lead to the trap state. The trouble is that a decision had to be made in the first step, which was not informed about the future.

The same holds for games even without probabilities. Consider the following game in Figure 3.4, actually with only the player whose task is to play so that the run is not in the language of the automaton on the right.

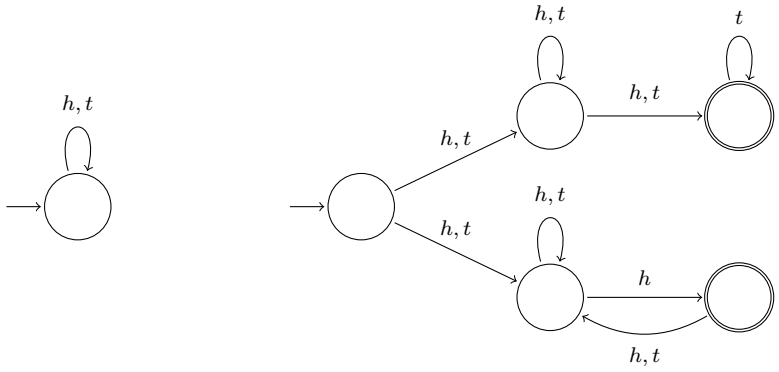


Figure 3.4: An example showing why non-deterministic automata cannot be used in Algorithm `AllRunsConform`$(\mathcal{S}, \phi)$ for games.

Consider now the product of the one-player game and the non-deterministic automaton. This gives rise to a two-player game with the game player and the automaton "player". The game player always has a strategy how to make the run rejected no matter what the automaton player does. Indeed, although the language is again the whole $\{h,t\}^\omega$, the very first step resolves whether there are finitely (in the upper branch) or infinitely (in the lower branch) many h's and the game player can adapt his strategy to then play only or no h's, respectively. Therefore, the problem again cannot be checked by a "pebble" game on the product of $\mathcal{S}$ and $\mathcal{A}$. Note that if there is only non-determinism trying to find a conforming run, this problem does not arise as both players can cooperate. Similarly, with only the other non-determinism (as in our example), we could negate the formula first and switch the role of the player. However, when both players are present, one will always be of the second kind causing the problem above.

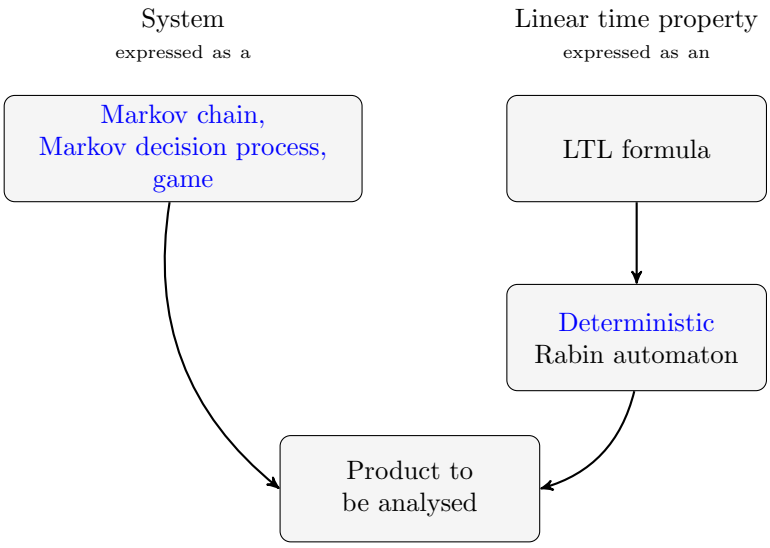


Figure 3.5: Automata-theoretic approach to synthesis and probabilistic verification

As a result, $\mathcal{A}$ must be deterministic if the approach of analysing the product as in Algorithm `AllRunsConform`$(\mathcal{S},\phi)$ is taken. We illustrate this in Figure 3.5. Note that there are alternatives, ranging from considering tree automata instead [KV05] over "in-the-limit-deterministic" automata [CY88] to restricting to fragments allowing for deterministic Büchi automata (or "generators") [AT04]. However, in the approach of Algorithm `AllRunsConform`$(\mathcal{S},\phi)$, Step 1. must yield a deterministic ω-automaton. Deterministic Muller automaton would be an option, but an expensive one. Although the state space is guaranteed to be at most doubly exponentially larger than ϕ, the representation of the acceptance condition can require triply exponential space. Thus both the construction in

Step 1. and checking in Step 2. would often be infeasible. Therefore, deterministic Rabin (or also Streett) automata are used as they combine reasonable state space size (can require more than Muller automata, but still fit in doubly exponential space) with a compact representation of the acceptance condition.

As to Step 1., the standard way to produce Rabin automata from LTL formulae is to first obtain a non-deterministic Büchi automaton and then determinise it using the procedure of Safra [Saf88] or its variants, extensions and optimisations [MS95, Pit06, KB06]. Both steps are in the worst case exponential, see Figure 3.6. Unfortunately, this is unavoidable as the problems of LTL model checking MDP and LTL game solving are 2-EXP-complete. Note that the most widespread probabilistic model checker `PRISM` [KNP11] uses this approach by chaining `LTL2BA` [GO01]—transforming LTL to Büchi automata—and `ltl2dstar` [Kle]—determinising Büchi automata to deterministic Rabin automata. We note that the approach of `LTL2BA` has been developed because of poor performance of the tableaux method for fairness constraints. However, despite `LTL2BA` yielding smaller automata for fairness constraints, the approach of `PRISM` often fails here as the determinisation then blows the automata up to sizes beyond practical use.

Step 2. then for MDP requires to compute maximal end-components satisfying the Rabin condition and then to optimise the probability to reach them. Step 2. for games consists in deciding the existence of winning strategies in the Rabin games.

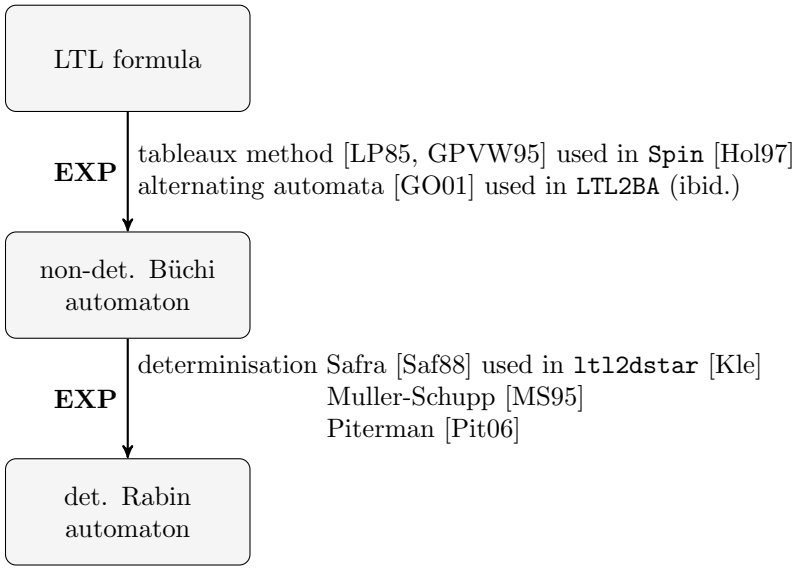


Figure 3.6: The standard way of translating LTL to deterministic ω-automata. All algorithms are also implemented in the academic tool `GOAL` [TCT+08].

3.2 New results

In Part I of the thesis, we show how Algorithm `AllRunsConform`$(\mathcal{S}, \phi)$ of the previous section can be significantly improved:

- Firstly, we give a *direct way to compute Rabin automata* from formulae of some LTL fragment without using non-deterministic automata. We avoid the determinisation procedures, which are designed to cope with general Büchi automata, i.e. automata stemming from formulae not only of LTL, but also of (monadic) second order logic. This results in a more specialised procedure yielding (1) often smaller automata, (2) state space with clear logical structure as opposed to the results of Safra's construction (oriented trees over subsets of states with binary flags are generally regarded as "messy" [Kup12]), and (3) is considerably simpler than the general Safra's construction (together with the translation of formulae to Büchi automata).
- Secondly, we introduce *a new kind of deterministic ω-automaton* which we call either *generalised Rabin automaton* (DGRA) or *automaton with a generalised Rabin pairs acceptance condition* as an intermediate step of the construction, see Figure 3.7. This deterministic automaton is often a much more compact representation of the corresponding de-generalised Rabin automaton (cf. generalised Büchi automata and Büchi automata, for details see below). For more complex formulae, the difference grows fast in orders of magnitude, see Table 3.2. The only price we have to pay is a slightly more complex acceptance condition. However, we show it is basically as easy to handle as the Rabin acceptance condition, both for MDP and games analysis (as required by Step 2. of the approach of Algorithm `AllRunsConform`$(\mathcal{S}, \phi)$). This yields a significant speed-up, see Table 3.3 and 3.4.

The drawback of this method is that it currently does not cover the whole LTL. We first give a translation for LTL$(\mathbf{F}, \mathbf{G})$ only and then for LTL$(\mathbf{X}, \mathbf{F}, \mathbf{G}, \mathbf{U})$ where $\mathbf{U}$ does not appear inside the scope of any $\mathbf{G}$ in the negation (also called positive) normal form. The translation of the whole LTL is a subject of our current research and we conjecture this method to be extensible to the whole LTL.

Similarly to [WVS83], our translation (1) deals with "finitary" properties using an automaton that checks local consistency after one transition is taken and (2) uses a separate automata mechanism that checks "infinitary" properties such as a satisfaction of a subformula infinitely often (or almost always in our case, too). The idea of the translation relies on two observations. Firstly, we can cope with "finitary" properties by *lazily unfolding* the formula.

Example 1. *The formula $\phi = \mathbf{G}a \vee \mathbf{F}b$ is equivalent to $(a \wedge \mathbf{XG}a) \vee b \vee \mathbf{XF}b$, which provides a way to determine a formula that is to be satisfied in the next step if we know the current valuation. For example, if a holds and b does not*

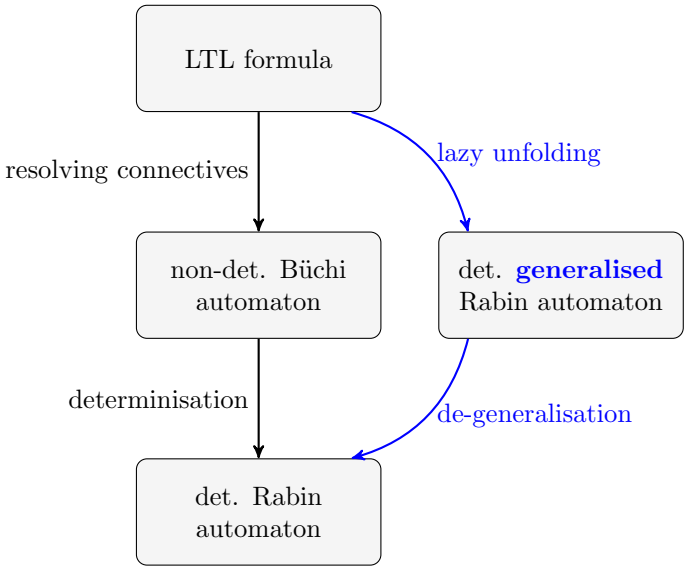


Figure 3.7: A new way of translating LTL to deterministic ω-automata

then the formula to be satisfied in the next step remains the same ϕ as opposed to the other translation methods, which non-deterministically pick which disjunct will hold, see Figure 3.8.

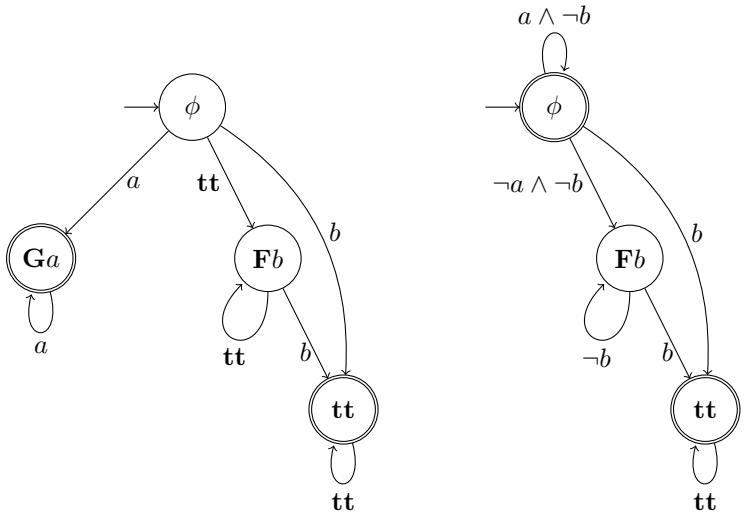


Figure 3.8: Automata produced by the traditional approaches (on the left) and the new approach (on the right) for formula $\phi = \mathbf{G}a \vee \mathbf{F}b$

Secondly, when there are more sets of states to be visited infinitely often, we capture that in the acceptance condition similarly as in translations via generalised Büchi automata. Similarly as generalised Büchi can be further *de-generalised*, see e.g. [BK08], we can also de-generalise DGRA into Rabin automata.

Example 2. *For the formula* $\phi = \mathbf{GF}a \wedge \mathbf{GF}b$ *we can have the following automaton, which simply keeps the current valuation in its state. The acceptance condition corresponding to* ϕ *is to visit some of the right states* and *some of the lower states infinitely often. As Rabin conditions cannot express this (without blowing up the state space), we introduce a more powerful condition.*

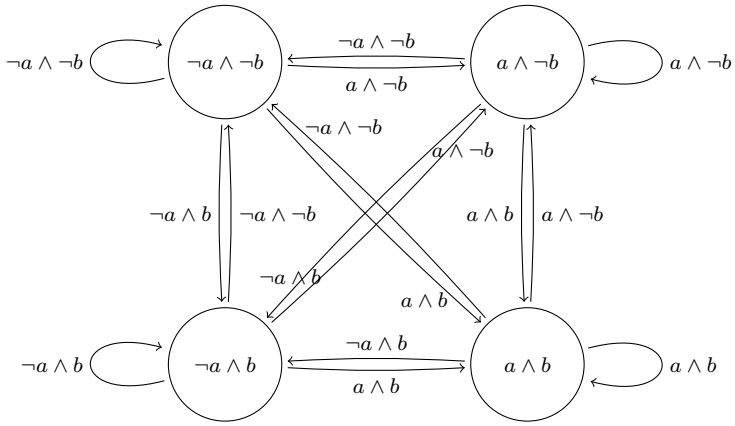


Figure 3.9: Automaton for $\phi = \mathbf{GF}a \wedge \mathbf{GF}b$

Generalised Rabin pairs acceptance condition

All standard types of ω-automata share the same structure, and differ only in their acceptance condition. We recall some of the most used conditions. All sets below are subsets of the set of states. Further, we abuse the notation and use "conjunctions" and "disjunctions" of sets. They are purely symbolic and serve to ease reading of the conditions as logic-like formulae.

- A **Büchi** condition is given by a set I. A run is accepting if it visits I infinitely often.
- A **generalised Büchi** condition is a conjunction

  $$\bigwedge_{i=1..n} I^i$$

  of Büchi conditions; thus we need to visit **each** I^i infinitely often.
- A **Rabin** condition is a disjunction

  $$\bigvee_{j=1..k} (F_j, I_j)$$

of *Rabin pairs*; for some j we must visit F_j finitely often and I_j infinitely often.

Inspired by Example 2, we introduce our generalisation of the Rabin condition.

- A **generalised Rabin** pairs condition is a Rabin condition generalised in the same way the Büchi condition is generalized, i.e. it is a disjunction

$$\bigvee_{j=1..k} (F_j, \bigwedge_{i=1..n} I_j^i)$$

of *generalised Rabin pairs*; for some j we must visit F_j finitely often and **each** I_j^i infinitely often. Note that there is no need to use conjunctions on the first components as F^1 *and* F^2 are visited finitely often if and only if $F^1 \cup F^2$ is visited finitely often.[†]

Example 3. *Consider the conjunction*

$$\bigwedge_{i\in\{1,\ldots,n\}} \mathbf{GF}a_i \Rightarrow \mathbf{GF}b_i$$

of strong fairness constraints. While the traditional approach of Figure 3.6 is hardly able to handle conjunction of three conditions (it takes more than a day to compute the automaton), our new approach generates automata of sizes comparable to non-deterministic automata. We display experimental results in Table 3.1. Our approach is represented by our implementations `Rabinizer` *[GKE12] (Paper B) generating Rabin automata and* `Rabinizer 2` *[KG13] (Paper D) generating generalised Rabin automata.*

Table 3.1: Comparison of the methods on simple fairness constraints.

Automaton type	NBA	DRA	DRA	DGRA	DTGRA
Tool	`LTL2BA`	`ltl2dstar`	`Rabinizer`	`Rabinizer 2`	
$n = 1$	4	4	4	4	1
$n = 2$	14	11 324	18	16	1
$n = 3$	40	$> 10^6$	462	64	1

We also consider deterministic transition-based generalised Rabin automata *(DTGRA) where the sets are sets of transitions to be taken finitely/infinitely often. Their size is* 1 *for the following reason. The DGRA remembers here exactly the last letter read as in Example 2. Therefore, we can easily encode DGRA as a one-state TDGRA as follows. Whenever a state of a DGRA remembering letter ℓ is in a set of the state-based condition, the incoming transition under ℓ in the TDGRA is in the set of the transition-based condition. In general, we store*

[†]Furthermore, we could consider arbitrary Boolean combinations. However, our algorithm generates the condition already in the disjunctive normal form and, moreover, we can only provide efficient analysis for conditions in the disjunctive normal form.

more information in the DGRA states, but TDGRA can still help to reduce the size. However, since the systems under verification have valuations on states, the information about the current valuation is present in the product anyway. Hence the size of the product is not affected by this transition-based acceptance optimisation. Therefore, we do not consider it here any more.

How can we use DGRA in the verification process, namely for probabilistic LTL model checking and LTL synthesis? Consider an automaton $\mathcal{A}$ with the condition $\bigvee_{j=1..k} (F_j, \mathfrak{I}_j)$ where $\mathfrak{I}_j = \bigwedge_{i=1..n_j} I_j^i$. There are two ways:

- **De-generalise** $\mathcal{A}$ into a Rabin automaton. Similarly to the de-generalisation of generalised Büchi automata, we create copies of $\mathcal{A}$ to track which I_j^i's are now awaited for each j. The number of copies is thus

$$\mathcal{D} := |\prod_{j=1}^{k} \mathfrak{I}^j| = n_1 \cdot \ldots \cdot n_k$$

  which we call the *de-generalisation index*. This determines the ratio of a de-generalised automaton (not employing any optimisations) to the original automaton.

  Example 4. *Again for the example of fairness constraints, we compare the sizes of DGRA, the optimised and the naive de-generalisation and the old approach in Table 3.2.*

Table 3.2: Blow up caused by de-generalisation.

n	$\mathcal{D}$	DGRA `Rabinizer 2`	DRA `Rabinizer`	DRA naive de-gen.	DRA `ltl2dstar`
1	1	4	4	4	4
2	2	16	18	32	11 324
3	24	64	462	29568	$> 10^6$
4	20736	128	?	$> 10^6$	?

  Note that even with the naive de-generalisation the results are by orders of magnitude smaller than those generated by the old approach.

- **Directly** use $\mathcal{A}$ as input to model checking/synthesis algorithms. Here we need to extend these algorithms from the DRA setting to DGRA. Fortunately, the complexity for model checking remains almost the same and for synthesis is only $\mathcal{D}$ times slower [CGK13] (Paper C). We thus obtain the following theoretical bounds for the speed up of our method, where the speed up factor is defined as the ratio of the upper bounds for the old and the new algorithm.

  Theorem 1. *The speed up factor for probabilistic LTL model checking is at least $\mathcal{D}^{5/3}$ and for LTL synthesis $\mathcal{D}^{k+1}$.*

Note that the speed up even for probabilistic model checking is exponential in the number of non-trivially generalised pairs, and is doubly exponential in the number of fairness constraints in Example 3 and 4.

Probabilistic LTL model checking

For probabilistic model checking, the idea of the extension is very simple. We take the traditional algorithm implemented in `PRISM` and at the point where we check non-emptiness of the intersection of maximal end-components (MECs) with the set to be visited infinitely often, we replace this by a *conjunction over all* sets to be visited infinitely often. Our algorithm is depicted in Figure 3.10. The box there replaces "I_j" of the traditional algorithm.

Algorithm `MaxConformingRuns`$(\mathcal{S}, \mathcal{A})$

1. For $j = 1..k$
   (a) Remove $\mathcal{S} \times F_j$ from $\mathcal{S} \times \mathcal{A}$
   (b) Compute the maximal end-component (MEC) decomposition
   (c) If a MEC intersects $\boxed{\text{each } I_j^i, \text{ for } i = 1, 2, \ldots, n_j}$
   then include it as a winning MEC
   (d) $Win_j := \bigcup$ winning MECs (for the jth pair).
2. $Win := \bigcup_{j=1}^{k} Win_j$.
3. Return the maximal probability to reach W

Figure 3.10: Algorithm for the analysis of the product $\mathcal{S} \times \mathcal{A}$

Example 5. *We show experimental results on the Pnueli-Zuck randomised mutual exclusion protocol [PZ86], which has* 2 368 *states for 3 participants,* 27 600 *for 4 participants, and* 308 800 *for 5 participants. We consider running times of* `ltl2dstar` *(**L**), optimised de-generalisation of DGRA into DRA by* `Rabinizer` *(**R**), and the direct use of DGRA (**GR**). Table 3.3 further displays the de-generalisation index $\mathcal{D}$ and also the speed up **R**/**GR** of our method against the de-generalisation method; this displays practical consequences of Theorem 1. Finally, the speed up **L**/**GR** against the traditional method is shown.*

LTL synthesis

The algorithm for games with generalised Rabin pairs winning condition is more complex. It is an extension of progress mesasure style algorithms for Rabin games and Streett games. Further, we also give a symbolic algorithm.

Example 6. *The theoretically predicted speed up according to Theorem 1 for 2 fairness constraints (with $\mathcal{D} = 2, k = 4$) is by factor $2^4 = 16$, while for 3 fairness constraints (with $\mathcal{D} = 24, k = 8$) it is by factor $24^8 \approx 10^{11}$. Below we*

Table 3.3: Experimental results for model checking (time-out after 1800 s)

Formula	#	**L**	**R**	**GR**	$\frac{\mathbf{R}}{\mathbf{GR}}$	$\mathcal{D}$	$\frac{\mathbf{L}}{\mathbf{GR}}$
P_{max} =?[$\mathbf{GF}p_1{=}10$	3	1.2	0.4	0.2	2.2	3	6.8
$\wedge\ \mathbf{GF}p_2{=}10$	4	17.4	1.8	0.3	6.4	3	60.8
$\wedge\ \mathbf{GF}p_3{=}10$]	5	257.5	15.2	0.6	26.7	3	447.9
P_{min} =?[	3	289.7	12.6	3.4	3.7	12	84.3
$(\mathbf{FG}p_1{\neq}0 \vee \mathbf{FG}p_2{\neq}0 \vee \mathbf{GF}p_3{=}0)\vee$	4	–	194.5	33.2	5.9	12	–
$(\mathbf{FG}p_1{\neq}10 \wedge \mathbf{GF}p_2 = 10 \wedge \mathbf{GF}p_3 = 10)$]	5	–	–	543	–	12	–
P_{max} =?[$(\mathbf{GF}p_1{=}0 \vee \mathbf{FG}p_2{\neq}0)$	3	–	122.1	7.1	17.2	24	–
$\wedge(\mathbf{GF}p_2{=}0 \vee \mathbf{FG}p_3{\neq}0)$	4	–	–	75.6	–	24	–
$\wedge(\mathbf{GF}p_3{=}0 \vee \mathbf{FG}p_1{\neq}0)$]	5	–	–	1219.5	–	24	–
P_{max} =?[$(\mathbf{GF}p_1{=}0 \vee \mathbf{FG}p_1{\neq}0)$	3	–	76.3	7.2	12	24	–
$\wedge(\mathbf{GF}p_2{=}0 \vee \mathbf{FG}p_2{\neq}0)$	4	–	1335.6	78.9	19.6	24	–
$\wedge(\mathbf{GF}p_3{=}0 \vee \mathbf{FG}p_3{\neq}0)$]	5	–	–	1267.6	–	24	–

show an experimental example for $\mathcal{D} = 6, k = 3$. Apart from the running time, we show columns "Size factor" displaying the ratio of the product size $|\mathcal{S} \otimes \mathcal{A}|$ to the system size $|\mathcal{S}|$.

Table 3.4: Experimental results for synthesis

Formula	$\lvert\mathcal{S}\rvert$	Size factor			Time		
		L	**R**	**GR**	**L**	**R**	**GR**
$(\mathbf{GF}a \vee \mathbf{FG}b)$	3	21.1	10.1	4.0	–	117.5	12.3
$\wedge(\mathbf{GF}c \vee \mathbf{GF}\neg a)$	6	16.2	9.2	3.7	–	–	196.7
$\wedge(\mathbf{GF}c \vee \mathbf{GF}\neg b)$	9	17.6	9.2	3.6	–	–	1017.8

Chapter 4

Continuous-time systems

In this chapter, we focus on time-bounded reachability in continuous-time systems. We optimise the control in open systems, i.e. we deal with games where both the controller of the plant and the inputs from the environment take turns.

4.1 State of the art

The environments here are modelled by processes of the same kind as the system under verification. In this thesis, we consider CTMDP (in Paper E) and IMC (in Paper F and G). We distinguish three different cases of the analysis:

- The structure is of the environment is **known**. This means that at the verification time, we know the available actions of the environment and its distributions on the waiting times at each moment. The only unknown part are the decisions of the environmental scheduler, i.e. which action will be chosen. This case is the most often considered one and has appeared in various timed settings, e.g. [BF09, Spr11, HNP$^+$11]. For continuous-time systems, this problem naturally translates to CTSG. We consider two types of strategies in CTSG: time-abstract, which do not base their decisions on the time waited (for example because they do not have access to it) and time-dependent, which know the precise times.

  In the time-abstract setting, the first algorithm to approximate time-bounded reachability probability (2.2) appeared in [BHHK04] for one-player game (CTMDP) with one rate. Further, one can employ the method of *uniformisation*—a transformation of systems with more rates to systems with a single rate—to extend the result to systems with more rates [RS10].

  In the time-dependent setting, analytic methods are too complex and numerical approaches are employed instead. The basic method is *discreti-*

sation, applied to IMC in [ZN10] and thus directly applicable to CTMDP [NZ10]. The complexity has been further improved in [FRSZ11] and [HH13]. A similar more involved approach closer to uniformisation has been proposed in [BS11]. The idea of the discretisation method is to chop time into small chunks and assume that at most one stochastic transition is taken in each chunk. Thus we obtain a discrete-time Markovian system. Moreover, the probability that more than one transition happens is quadratically smaller than that of at most one happening, hence we obtain quadratically small error, which becomes negligible for smaller chunks. The optimisations then relax the assumption that only one transition happens in one chunk.

- The environment is completely **unknown**. This means there are no limitations on what the environment can do except that it is again an IMC/CTMDP, i.e. an unknown component of the same kind as our system. In the discrete setting this case is close to the classical synthesis problem [Chu57]. In the continuous-time setting, this problem has to the best of our knowledge not been considered. For other forms of timed systems see the third case below.

  Example 7. *Consider the upper system of Figure 4.1, where* outdated *is an action that takes place when both the system and the environment have this action available in the current states. What are the guarantees to reach the goal state (double circled) after at most 0.5 time units? The worst case that can happen is the following. The environment first waits, which yields high probability that the faulty branch happens. Shortly before the deadline (at* $\approx$*0.31) if this transition has not been taken yet, the environment enables the* outdated *action. It is executed and we are taken to a state where we are quite likely stuck till the deadline. The resulting guaranteed lower bound is then* ≈ 0.52*.*

- The environment **is unknown, but conforms to a known specification**. This means that we do not know its structure, but we have some information about its behaviour. Proving properties based on this information is called *assume-guarantee reasoning*. Iterating this approach on components of the system gives basis for a compositional method for verification. This approach has been widely used in the discrete setting [MC81, AH96] as well as the real-time setting [TAKB96, AH97, HMP01].

  Example 8. *Consider the lower system of Figure 4.1. As opposed to the previous example, we can derive no guarantees on reaching the target state without any knowledge of the environment. Indeed, the environment may enforce* outdated *and then block* update *forever. Therefore, in order to obtain better guarantees, we would need to assume that* update *will be available some time after executing* outdated*.*

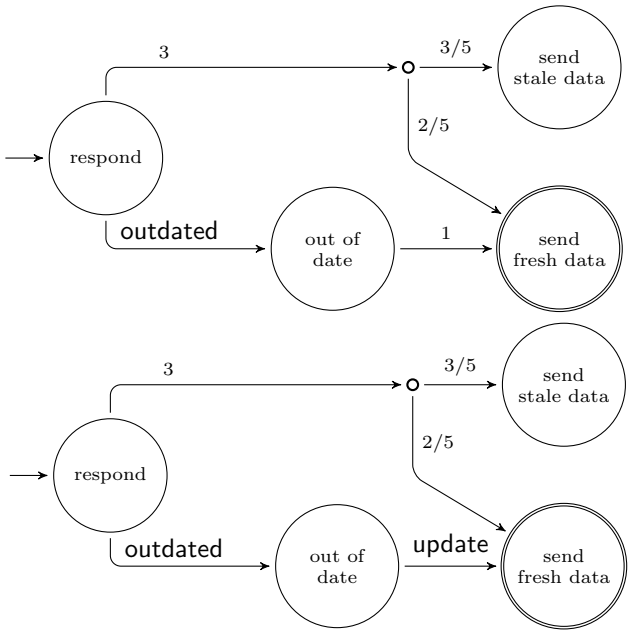


Figure 4.1: Two examples of open IMC

4.2 New results

We now present our results for each of the three settings.

Known environment

Here we focus on continuous-time stochastic games, which we introduced in [BFK$^+$09] (Paper E is an extended journal version [BFK$^+$13]). We consider time-abstract schedulers and show the following:

- Even with countably infinite arenas these games are determined, i.e. (2.2) equals to value where the supremum and the infimum are swapped.
- Optimal strategies may not exist in countably infinite arenas. However, they exist for finitely branching games with rates bounded from above, thus also in particular in finite games.
- We give an algorithm to compute optimal strategies in finite games with one rate; moreover, we also describe their structure, which yields a finite description of optimal strategies:

Theorem 2. *For every finite CTSG, there are computable optimal time-abstract strategies and $n \in \mathbb{N}$ such that after n steps, the decisions of the strategies depend only on the current state.*

This algorithm on CTSG is also the first algorithm computing optimal strategies in CTMDP. It has been further generalised to CTSG with more rates in [RS10] using uniformisation.

The idea of our result is that after long enough time has most likely elapsed (recall the strategies are time-abstract), the optimal strategy tries to reach the goal in as few discrete steps as possible no matter what the probabilities are. Our result reduces the problem to solving inequalities of the form $\sum_{i=1}^{k} q_i \cdot e^{r_i} > 0$ for rational q_i, r_i similarly as [ASSB96]. Although the latter problem is decidable, there is no upper bound known.*

- Therefore, we also give an approximative algorithm:

Theorem 3. *For every finite game with n states, m actions, r rates, maximum rate λ, bit length b of transition probabilities (considered as fractions of integers), and $\varepsilon > 0$, ε-optimal time-abstract strategies for time-bounded reachability till time t are computable in time*

$$n^2 \cdot m \cdot b^2 \cdot \left(\lambda \cdot t + \ln \frac{1}{\varepsilon}\right)^{2r+\mathcal{O}(1)}$$

This extends the result [BHHK04] for CTMDP.

Note that the complexity depends on the logarithm of the error in contrast to the results for time-dependent strategies. There the discretisation approaches lead to dependencies on roots of the error at best [FRSZ11].

Unknown environment

Here we want to give guarantees on a system S no matter which environment E it is composed with (recall the composition of Section 2.2 denoted here by $S \parallel E$). Furthermore, we want to compute the respective ε-optimal (time-dependent) scheduler of S that yields this guarantee. Formally,† we want to compute

$$\sup_{\sigma} \inf_{\substack{E \\ \pi}} \mathbf{P}^{(S \| E)^{\sigma,\pi}} [\mathbf{F}_{[0,t]}\, goal] \qquad (*)$$

*That is why there is no upper bound known on CSL model checking CTMC.

†Technically, π ranges over schedulers of the composition "respecting" σ. For details, see Paper F.

- We transform the problem to a game played on S where the second player chooses some of the waiting times non-deterministically. We solve this game using the discretisation approach. However, this game has the same value only under some conditions as we show in [BHK+12] (Paper F), under which the complexity stays the same as for the one player case (closed IMC):

> **Theorem 4.** *For every $\varepsilon > 0$, $t \geq 0$ and IMC S with n states and maximum rate λ, where internal (non-synchronising) actions and synchronising action are not available in the same state, an ε-approximation of $(*)$ and the respective ε-optimal scheduler can be computed in polynomial time $\mathcal{O}(n^2\lambda^2t^2/\varepsilon)$.*

- In [HKK13], (Paper G) we lift the assumption and give a solution to the general case. In the previous case, the unknown environment E interferes only with stochastic timed transitions which leads to games with stochastic and non-deterministic time. Now E interferes with internal transitions as well. This introduces incomplete information and leads to games with partial observation yielding worse complexity:

> **Theorem 5.** *For every $\varepsilon > 0$, $t \geq 0$ and IMC S, an ε-approximation of $(*)$ and the respective ε-optimal scheduler can be computed in exponential time.*

Specified environment

In compositional assume-guarantee analysis, we ask what guarantees we can get for our system S when it works in composition with an unknown environment E *satisfying a specification* φ. Formally, we want to compute

$$\sup_{\sigma} \inf_{\substack{E \models \varphi \\ \pi}} \mathbf{P}^{(S \| E)^{\sigma,\pi}} [\mathbf{F}_{[0,t]}\, goal] \qquad (**)$$

In Paper G, we design a specification formalism for expressing assumptions and an algorithm for computing guarantees:

- Firstly, we provide a specification formalism to express assumptions on continuous-time stochastic systems called *modal continuous-time automata* (MCA). The novel feature of the formalism are *continuous time constraints*. They are like guards in time automata [AD94] only not using constants, but distributions. This is a crucial step for getting guarantees with respect to time-bounded reachability in IMC. Indeed, hard bounds cannot be applied in the setting where waiting is always positively distributed on $[0, \infty)$. Furthermore, to allow underspecification, we use modal extension of automata following [LT88]. Further, as usual, we require determinism.

Example 9. *Recall Example 8 with the lower system of Figure 4.1. We could not derive any guarantees unless we assume that* update *will be available some time after executing* outdated*. This is exactly what the MCA in Figure 4.2 describes. Indeed, if* update *comes nothing changes;* outdated *may not be available, but if it is and is taken then we go to another state. We stay there for at most the time distributed according to the exponential distribution with rate 3, and then move to a state where* update *must be present. However, if it is not taken (because S was not ready) then the time flow (denoted by $\top$) leads us to the starting situation. Assuming this, one can derive the same guarantee of ≈ 0.52 as in Example 7 for the upper system of the same figure.*

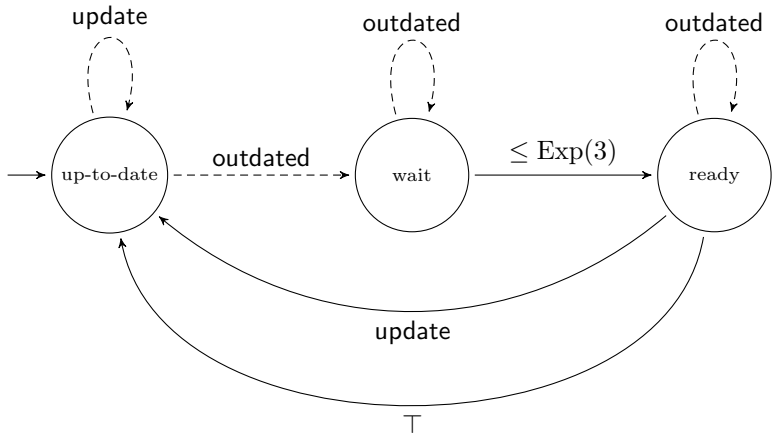


Figure 4.2: A modal continuous-time automaton

- Secondly, we integrate the assume-guarantee reasoning to the IMC framework. We show how to synthesise ε-optimal schedulers for IMC in an unknown environment satisfying a given specification and approximate the respective guarantee (∗∗). The approach is to compute some kind of a product $S \times \varphi$ of the system and the specification and reduce this problem to the previous one, namely computing

$$\sup_{\sigma} \inf_{\substack{E \\ \pi}} \mathbf{P}^{(S\times\varphi)\|E\ \sigma,\pi} [\mathbf{F}_{[0,t]} goal]$$

The product construction introduces further incomplete information to the game, but the complexity class stays the same:

Theorem 6. *For every $\varepsilon > 0$, $t \geq 0$, IMC S and MCA φ, we can ε-approximate (∗∗) and compute an ε-optimal scheduler in exponential time.*

Chapter 5

Summary of the results and future work

The thesis contributes in two areas:

Firstly, we have provided a novel translation of LTL formulae to ω-automata. It introduces a new type of deterministic ω-automata with *generalised Rabin pairs* acceptance condition, which allows for

- much more *compact representation* of many LTL formulae than Rabin automata [KE12];
- almost as *efficient analysis* as Rabin automata both for probabilistic LTL model checking and LTL synthesis [CGK13];
- *speed up in orders of magnitude* even for very small, but more complex formulae such as fairness constraints, at least by factor of $\mathcal{D}^{5/3}$ for probabilistic model checking and $\mathcal{D}^{k+1}$ for synthesis, where $\mathcal{D}$ is the *de-generalisation index* and k the number of pairs.

We have provided tools for translating formulae of LTL fragments to deterministic generalised Rabin automata. The tool `Rabinizer` [GKE12] covers LTL($\mathbf{F}, \mathbf{G}$) and has been experimentally incorporated to `PRISM` [CGK13]. Secondly, `Rabinizer 2` [KG13] covers LTL($\mathbf{X}, \mathbf{F}, \mathbf{G}, \mathbf{U}$) $\setminus$ $\mathbf{GU}$ where $\mathbf{U}$ operators are not in the scope of $\mathbf{G}$ operators.

In future work, we plan to extend the translation algorithms to make use of the generalised condition for the *whole LTL*. Further, we plan to release an implementation of our method, which would be downloadable as a *plug-in* for `PRISM` and thus would facilitate practical use of our method.

Secondly, we have also studied optimisation of open continuous-time systems with respect to the fundamental problem of time-bounded reachability. We have shown how to

- compute optimal and ε-optimal time-abstract strategies in [BFK$^+$13];
- compute ε-optimal strategies in IMC which operate in an unknown environment [BHK$^+$12, HKK13] introducing a new concept of *controller–environment* game, which is an asymmetric mixture of CTMDP and timed automata;
- specify properties using a new formalism of *modal continuous-time automata* [HKK13] using *continuous time constraints*;
- compute ε-optimal strategies in an IMC operating in an unknown environment conforming to an MCA specification [HKK13]

In future work, we will focus on identifying structural subclasses of IMC allowing for polynomial analysis. Further, we are aiming at designing a temporal logic matching the needs of the assume-guarantee verification in this context and its translation to MCA. Finally, we plan to implement our approach and employ heuristics to cope with the higher complexity of incomplete information games.

5.1 Summary of the papers

In Part I of Appendix, we present the following papers:

A *Deterministic automata for the (F,G)-fragment of LTL.* (CAV 2012)

The paper provides a novel translation of LTL($\mathbf{F}, \mathbf{G}$) to Rabin automata and introduces the generalised Rabin pairs acceptance condition.

B *Rabinizer: Small deterministic automata for LTL($\mathbf{F}, \mathbf{G}$).* (ATVA 2012)

This tool paper provides an optimised implementation of the construction of Paper A.

C *Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis.* (CAV 2013)

This paper extends verification algorithms from Rabin to the generalised Rabin setting and shows both theoretical and experimental speed ups.

D *Rabinizer 2: Small deterministic automata for LTL$_{\setminus \mathbf{GU}}$.* (ATVA 2013)

This tool paper extends the translation of Paper A to the fragment of LTL($\mathbf{X}, \mathbf{F}, \mathbf{G}, \mathbf{U}$) where $\mathbf{U}$ does not appear in the scope of any $\mathbf{G}$, and provides an implementation thereof.

In Part II of Appendix, we present the following papers:

E *Continuous-time stochastic games with time-bounded reachability.* (Information and Computation 2013)

This paper is an extended journal version of an *FSTTCS 2009* conference paper [BFK+09]. It defines continuous-time stochastic games, investigates their basic properties and gives algorithms to compute optimal and ε-optimal time-abstract strategies.

F *Verification of open interactive Markov chains.* (FSTTCS 2012)

This paper identifies and solves the problem of synthesising ε-optimal schedulers of IMC operating in an unknown environment.

G *Compositional verification and optimization of interactive Markov chains.* (CONCUR 2013)

This paper introduces the specification formalism of modal continuous-time automata and solves the problem of synthesising ε-optimal schedulers of IMC operating in an unknown environment satisfying a given specification.

In Appendix, each paper is again summarised and the author's contribution is listed. The percentage indicating the author's contribution has been approved by the respective co-authors.

Bibliography

[ACD90] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking for real-time systems. In *LICS*, pages 414–425. IEEE Computer Society, 1990.

[ACHH92] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 1992.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[AH96] Rajeev Alur and Thomas A. Henzinger. Reactive modules. In *LICS*, pages 207–218. IEEE Computer Society, 1996.

[AH97] Rajeev Alur and Thomas A. Henzinger. Modularity for timed and hybrid systems. In Antoni W. Mazurkiewicz and Józef Winkowski, editors, *CONCUR*, volume 1243 of *Lecture Notes in Computer Science*, pages 74–88. Springer, 1997.

[AHK97] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. In *FOCS*, pages 100–109, 1997.

[ASSB96] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert K. Brayton. Verifying continuous time Markov chains. In Rajeev Alur and Thomas A. Henzinger, editors, *CAV*, volume 1102 of *Lecture Notes in Computer Science*, pages 269–276. Springer, 1996.

[AT04] Rajeev Alur and Salvatore La Torre. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Log.*, 5(1):1–25, 2004.

[ava04] AVACS. http://www.avacs.org, 2004.

[BBFK06] Tomáš Brázdil, Václav Brožek, Vojtěch Forejt, and Antonín Kučera. Stochastic games with branching-time winning objectives. In *LICS* [DBL06], pages 349–358.

[BBGK07] Christel Baier, Tomáš Brázdil, Marcus Größer, and Antonín Kučera. Stochastic game logic. In *QEST*, pages 227–236. IEEE Computer Society, 2007.

[BčK11] Nikola Beneš, Ivana černá, and Jan Křetínský. Modal transition systems: Composition and LTL model checking. In Bultan and Hsiung [BH11], pages 228–242.

[BDF+13] Nikola Beneš, Benoît Delahaye, Uli Fahrenberg, Jan Křetínský, and Axel Legay. Hennessy-milner logic with greatest fixed points as a complete behavioural specification theory. In D'Argenio and Melgratti [DM13], pages 76–90.

[Bel57] Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6, 1957.

[BF09] Patricia Bouyer and Vojtěch Forejt. Reachability in stochastic timed games. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *ICALP (2)*, volume 5556 of *Lecture Notes in Computer Science*, pages 103–114. Springer, 2009.

[BFK08] Tomáš Brázdil, Vojtěch Forejt, and Antonín Kučera. Controller synthesis and verification for Markov decision processes with qualitative branching time objectives. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 148–159. Springer, 2008.

[BFK+09] Tomáš Brázdil, Vojtěch Forejt, Jan Krčál, Jan Křetínský, and Antonín Kučera. Continuous-time stochastic games with time-bounded reachability. In Ravi Kannan and K. Narayan Kumar, editors, *FSTTCS*, volume 4 of *LIPIcs*, pages 61–72. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2009.

[BFK+13] Tomáš Brázdil, Vojtech Forejt, Jan Krčál, Jan Křetínský, and Antonín Kucera. Continuous-time stochastic games with time-bounded reachability. *Inf. Comput.*, 224:46–70, 2013.

[BFKK08] Tomáš Brázdil, Vojtěch Forejt, Jan Křetínský, and Antonín Kučera. The satisfiability problem for probabilistic CTL. In *LICS*, pages 391–402. IEEE Computer Society, 2008.

[BGL+04] Christel Baier, Marcus Größer, Martin Leucker, Benedikt Bollig, and Frank Ciesinski. Controller synthesis for probabilistic systems. In Jean-Jacques Lévy, Ernst W. Mayr, and John C. Mitchell, editors, *IFIP TCS*, pages 493–506. Kluwer, 2004.

[BH11] Tevfik Bultan and Pao-Ann Hsiung, editors. *Automated Technology for Verification and Analysis, 9th International Symposium, ATVA*

2011, Taipei, Taiwan, October 11-14, 2011. Proceedings, volume 6996 of *Lecture Notes in Computer Science*. Springer, 2011.

[BHHK00] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model checking continuous-time Markov chains by transient analysis. In E. Allen Emerson and A. Prasad Sistla, editors, *CAV*, volume 1855 of *Lecture Notes in Computer Science*, pages 358–372. Springer, 2000.

[BHHK04] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. In Kurt Jensen and Andreas Podelski, editors, *TACAS*, volume 2988 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2004.

[BHHK10] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Performance evaluation and model checking join forces. *Commun. ACM*, 53(9):76–85, 2010.

[BHK+12] Tomáš Brázdil, Holger Hermanns, Jan Krčál, Jan Křetínský, and Vojtech Řehák. Verification of open interactive Markov chains. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *FSTTCS*, volume 18 of *LIPIcs*, pages 474–485. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[Bil12] Patrick Billingsley. *Probability and Measure*. Series in Probability and Statistics. John Wiley, 2012.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[BK10] Nikola Beneš and Jan Křetínský. Process algebra for modal transition systemses. In Ludek Matyska, Michal Kozubek, Tomas Vojnar, Pavel Zemcik, and David Antos, editors, *MEMICS*, volume 16 of *OASICS*, pages 9–18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2010.

[BK12] Nikola Beneš and Jan Křetínský. Modal process rewrite systems. In Abhik Roychoudhury and Meenakshi D'Souza, editors, *ICTAC*, volume 7521 of *Lecture Notes in Computer Science*, pages 120–135. Springer, 2012.

[BKK+10] Tomáš Brázdil, Jan Krčál, Jan Křetínský, Antonín Kučera, and Vojtěch řehák. Stochastic real-time games with qualitative timed automata objectives. In Paul Gastin and François Laroussinie, editors, *CONCUR*, volume 6269 of *Lecture Notes in Computer Science*, pages 207–221. Springer, 2010.

[BKK+11] Tomáš Brázdil, Jan Krčál, Jan Křetínský, Antonín Kučera, and Vojtěch řehák. Measuring performance of continuous-time stochastic

processes using timed automata. In Marco Caccamo, Emilio Frazzoli, and Radu Grosu, editors, *HSCC*, pages 33–42. ACM, 2011.

[BKK+13] Tomáš Brázdil, Lubos Korenciak, Jan Krčál, Jan Křetínský, and Vojtech Řehák. On time-average limits in deterministic and stochastic Petri nets. In Seetharami Seelam, Petr Tuma, Giuliano Casale, Tony Field, and José Nelson Amaral, editors, *ICPE*, pages 421–422. ACM, 2013.

[BKKŘ11] Tomáš Brázdil, Jan Krčál, Jan Křetínský, and Vojtech Řehák. Fixed-delay events in generalized semi-Markov processes revisited. In Joost-Pieter Katoen and Barbara König, editors, *CONCUR*, volume 6901 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 2011.

[BKL+11] Nikola Beneš, Jan Křetínský, Kim G. Larsen, Mikael H. Moller, and Jiri Srba. Parametric modal transition systems. In Bultan and Hsiung [BH11], pages 275–289.

[BKL+12] Nikola Beneš, Jan Křetínský, Kim Guldstrand Larsen, Mikael H. Moller, and Jiri Srba. Dual-priced modal transition systems with time durations. In Nikolaj Bjørner and Andrei Voronkov, editors, *LPAR*, volume 7180 of *Lecture Notes in Computer Science*, pages 122–137. Springer, 2012.

[BKLS09a] Nikola Beneš, Jan Křetínský, Kim Guldstrand Larsen, and Jiri Srba. Checking thorough refinement on modal transition systems is exptime-complete. In Martin Leucker and Carroll Morgan, editors, *ICTAC*, volume 5684 of *Lecture Notes in Computer Science*, pages 112–126. Springer, 2009.

[BKLS09b] Nikola Beneš, Jan Křetínský, Kim Guldstrand Larsen, and Jiri Srba. On determinism in modal transition systems. *Theor. Comput. Sci.*, 410(41):4026–4043, 2009.

[BKLS12] Nikola Beneš, Jan Křetínský, Kim G. Larsen, and Jiri Srba. Exptime-completeness of thorough refinement on modal transition systems. *Inf. Comput.*, 218:54–68, 2012.

[Bré99] Pierre Brémaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer, 1999.

[BS11] Peter Buchholz and Ingo Schulz. Numerical Analysis of Continuous Time Markov Decision processes over Finite Horizons. *Computers and Operations Research*, 38:651–659, 2011.

[CE81] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In Dexter Kozen, editor, *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.

[CES83] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. In John R. Wright, Larry Landweber, Alan J. Demers, and Tim Teitelbaum, editors, *POPL*, pages 117–126. ACM Press, 1983.

[CGK13] Krishnendu Chatterjee, Andreas Gaiser, and Jan Křetínský. Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis. In Helmut Veith and Natasha Sharygina, editors, *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 559–575, 2013.

[CHKM09] Taolue Chen, Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. Quantitative model checking of continuous-time Markov chains against timed automata specifications. In *LICS*, pages 309–318. IEEE Computer Society, 2009.

[Chu57] Alonzo Church. Applications of recursive arithmetic to the problem of circuit synthesis. In *Summaries of the Summer Institute of Symbolic Logic*, volume I, pages 3–50. Cornell Univ., Ithaca, N.Y., 1957.

[CJH04] Krishnendu Chatterjee, Marcin Jurdzinski, and Thomas A. Henzinger. Quantitative stochastic parity games. In J. Ian Munro, editor, *SODA*, pages 121–130. SIAM, 2004.

[CY88] Costas Courcoubetis and Mihalis Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In *FOCS* [DBL88], pages 338–345.

[DBL88] *29th Annual Symposium on Foundations of Computer Science, 24-26 October 1988, White Plains, New York, USA*. IEEE, 1988.

[DBL06] *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*. IEEE Computer Society, 2006.

[DBL10] *QEST 2010, Seventh International Conference on the Quantitative Evaluation of Systems, Williamsburg, Viginia, USA, 15-18 September 2010*. IEEE Computer Society, 2010.

[DBL11] *Eighth International Conference on Quantitative Evaluation of Systems, QEST 2011, Aachen, Germany, 5-8 September, 2011*. IEEE Computer Society, 2011.

[DM13] Pedro R. D'Argenio and Hernán Melgratti, editors. *CONCUR 2013 - Concurrency Theory, 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, volume 8052 of *Lecture Notes in Computer Science*. Springer, 2013.

[EH82] E. Allen Emerson and Joseph Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. In Harry R.

Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *STOC*, pages 169–180. ACM, 1982.

[FRSZ11] John Fearnley, Markus Rabe, Sven Schewe, and Lijun Zhang. Efficient approximation of optimal control for continuous-time Markov games. In Supratik Chakraborty and Amit Kumar, editors, *FSTTCS*, volume 13 of *LIPIcs*, pages 399–410. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[FV96] Jerzy Filar and Koos Vrieze. *Competitive Markov Decision Processes*. Springer, 1996.

[GKE12] Andreas Gaiser, Jan Křetínský, and Javier Esparza. Rabinizer: Small deterministic automata for LTL(F, G). In Supratik Chakraborty and Madhavan Mukund, editors, *ATVA*, volume 7561 of *Lecture Notes in Computer Science*, pages 72–76. Springer, 2012.

[GO01] Paul Gastin and Denis Oddoux. Fast ltl to büchi automata translation. In Gérard Berry, Hubert Comon, and Alain Finkel, editors, *CAV*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2001.

[GPVW95] Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In Piotr Dembinski and Marek Sredniawa, editors, *PSTV*, volume 38 of *IFIP Conference Proceedings*, pages 3–18. Chapman & Hall, 1995.

[Her02] Holger Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, volume 2428 of *Lecture Notes in Computer Science*. Springer, 2002.

[HH13] Hassan Hatefi and Holger Hermanns. Improving time bounded computations in interactive Markov chain. In Farhad Arbab and Marjan Sirjani, editors, *Fundamentals of Software Engineering, 5th IPM International Conference, FSEN 2013*, Lecture Notes in Computer Science. Springer, 2013. to appear.

[HHK02] Holger Hermanns, Ulrich Herzog, and Joost-Pieter Katoen. Process algebra for performance evaluation. *Theor. Comput. Sci.*, 274(1-2):43–87, 2002.

[Hil96] Jane Hilston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.

[HJ94] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *FAC*, 6:512–535, 1994.

[HKK13] Holger Hermanns, Jan Krčál, and Jan Křetínský. Compositional verification and optimization of interactive Markov chains. In D'Argenio and Melgratti [DM13], pages 364–379.

[HMP01] Thomas A. Henzinger, Marius Minea, and Vinayak S. Prabhu. Assume-guarantee reasoning for hierarchical hybrid systems. In Maria Domenica Di Benedetto and Alberto L. Sangiovanni-Vincentelli, editors, *HSCC*, volume 2034 of *Lecture Notes in Computer Science*, pages 275–290. Springer, 2001.

[HNP+11] Ernst Moritz Hahn, Gethin Norman, David Parker, Björn Wachter, and Lijun Zhang. Game-based abstraction and controller synthesis for probabilistic hybrid systems. In *QEST* [DBL11], pages 69–78.

[HO13] Dang Van Hung and Mizuhito Ogawa, editors. *Automated Technology for Verification and Analysis, 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15 - 18, 2013. Proceedings*, Lecture Notes in Computer Science. Springer, 2013.

[Hol97] Gerard J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23:279–295, 1997.

[How60] Ronald A. Howard. *Dynamic Programming and Markov Processes*. M.I.T. Press, 1960.

[HPW10] Michael Huth, Nir Piterman, and Daniel Wagner. p-automata: New foundations for discrete-time probabilistic verification. In *QEST* [DBL10], pages 161–170.

[HS84] Sergiu Hart and Micha Sharir. Probabilistic temporal logics for finite and bounded models. In *STOC*, pages 1–13. ACM, 1984.

[ide11] IDEA4CPS. http://www.idea4cps.dk, 2011.

[KB06] Joachim Klein and Christel Baier. Experiments with deterministic *omega*-automata for formulas of linear temporal logic. *Theor. Comput. Sci.*, 363(2):182–195, 2006.

[KE12] Jan Křetínský and Javier Esparza. Deterministic automata for the (F,G)-fragment of LTL. In P. Madhusudan and Sanjit A. Seshia, editors, *CAV*, volume 7358 of *Lecture Notes in Computer Science*, pages 7–22. Springer, 2012.

[KG13] Jan Křetínský and Ruslán Ledesma Garza. Rabinizer 2: Small deterministic automata for LTL\GU. In Hung and Ogawa [HO13]. To appear.

[Kle] Joachim Klein. ltl2dstar - LTL to deterministic Streett and Rabin automata. http://www.ltl2dstar.de/.

[KNP11] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[Kom12] Zuzana Komárková. Phase-type approximation techniques. Bachelor's thesis, Masaryk University, Faculty of Informatics, 2012.

[Koy90] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[KR10] Orna Kupferman and Adin Rosenberg. The blowup in translating LTL to deterministic automata. In *MoChArt*, volume 6572 of *LNCS*, pages 85–94. Springer, 2010.

[KS13a] Jan Křetínský and Salomon Sickert. On refinements of boolean and parametric modal transition systems. In Zhiming Liu, Jim Woodcock, and Huibiao Zhu, editors, *ICTAC*, volume 8049 of *Lecture Notes in Computer Science*, pages 213–230, 2013.

[KS13b] Jan Křetínský and Salomon Sickert. MoTraS: A tool for modal transition systems and their extensions. In Hung and Ogawa [HO13]. To appear.

[Kup12] Orna Kupferman. Recent challenges and ideas in temporal synthesis. In *SOFSEM*, volume 7147 of *LNCS*, pages 88–98. Springer, 2012.

[KV05] Orna Kupferman and Moshe Y. Vardi. Safraless decision procedures. In *FOCS*, pages 531–542. IEEE Computer Society, 2005.

[KZH+09] Joost-Pieter Katoen, Ivan S. Zapreev, Ernst Moritz Hahn, Holger Hermanns, and David N. Jansen. The ins and outs of the probabilistic model checker MRMC. In *Quantitative Evaluation of Systems (QEST)*, pages 167–176. IEEE Computer Society, 2009. www.mrmc-tool.org.

[LP85] Orna Lichtenstein and Amir Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In Mary S. Van Deusen, Zvi Galil, and Brian K. Reid, editors, *POPL*, pages 97–107. ACM Press, 1985.

[LS83] Daniel J. Lehmann and Saharon Shelah. Reasoning with time and chance (extended abstract). In Josep Díaz, editor, *ICALP*, volume 154 of *Lecture Notes in Computer Science*, pages 445–457. Springer, 1983.

[LT88] Kim G. Larsen and Bent Thomsen. A modal process logic. In *LICS*, pages 203–210. IEEE Computer Society, 1988.

[Mar06] Andrey A. Markov. Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga. *Izvestiya Fiziko-matematicheskogo obschestva pri Kazanskom universitete*, 15, 2-ya seriya:135–156, 1906.

[Mat12] Mathematica. *version 9*. Wolfram Research, Inc., Champaign, Illinois, 2012.

[MAT13] MATLAB. *version 8.1 (R2013a)*. The MathWorks Inc., Natick, Massachusetts, 2013.

[MC81] Jayadev Misra and K. Mani Chandy. Proofs of networks of processes. *IEEE Trans. Software Eng.*, 7(4):417–426, 1981.

[MS95] David E. Muller and Paul E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *Theor. Comput. Sci.*, 141(1&2):69–107, 1995.

[MT09] Sean P. Meyn and Richard L. Tweedie. *Markov chains and stochastic stability*. Cambridge University Press, 2009.

[mtl09] MT-LAB. http://www.mt-lab.dk/en/research/research.htm, 2009.

[mtl11] MT-LAB: Midterm status report. http://www.mt-lab.dk/download/research/midterm_report.pdf, 2011.

[Nor98] James R. Norris. *Markov Chains*. Cambridge University Press, 1998.

[NS03] Abraham Neyman and Sylvain Sorin. *Stochastic Games and Applications*. Dordrecht: Kluwer Academic Press, 2003.

[NZ10] Martin R. Neuhäußer and Lijun Zhang. Time-bounded reachability probabilities in continuous-time Markov decision processes. In *QEST* [DBL10], pages 209–218.

[Pit06] Nir Piterman. From nondeterministic Buchi and Streett automata to deterministic parity automata. In *LICS* [DBL06], pages 255–264.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.

[PR89] Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190. ACM Press, 1989.

[Put94] Martin L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[PZ86] Amir Pnueli and Lenore Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1(1):53–72, 1986.

[qua08] QUASIMODO. http://www.quasimodo.aau.dk, 2008.

[roc09] ROCKS. http://rocks.w3.rz.unibw-muenchen.de, 2009.

[RS10] Markus Rabe and Sven Schewe. Optimal time-abstract schedulers for ctmdps and Markov games. In Alessandra Di Pierro and Gethin Norman, editors, *QAPL*, volume 28 of *EPTCS*, pages 144–158, 2010.

[Saf88] Shmuel Safra. On the complexity of ω-automata. In *FOCS* [DBL88], pages 319–327.

[SC85] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985.

[Sha53] Lloyd S. Shapley. Stochastic games. *PNAS*, 39 (10):1095–110, 1953.

[Spr11] Jeremy Sproston. Discrete-time verification and control for probabilistic rectangular hybrid automata. In *QEST* [DBL11], pages 79–88.

[TAKB96] Serdar Tasiran, Rajeev Alur, Robert P. Kurshan, and Robert K. Brayton. Verifying abstractions of timed systems. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR*, volume 1119 of *Lecture Notes in Computer Science*, pages 546–562. Springer, 1996.

[TCT+08] Yih-Kuen Tsay, Yu-Fang Chen, Ming-Hsien Tsai, Wen-Chin Chan, and Chi-Jian Luo. GOAL extended: Towards a research tool for omega automata and temporal logic. In C. R. Ramakrishnan and Jakob Rehof, editors, *TACAS*, volume 4963 of *Lecture Notes in Computer Science*, pages 346–350. Springer, 2008.

[TDG09] Mirco Tribastone, Adam Duguid, and Stephen Gilmore. The PEPA eclipse plugin. *SIGMETRICS Performance Evaluation Review*, 36(4):28–33, 2009.

[Var85] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS*, pages 327–338. IEEE, 1985.

[VW86] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *LICS*, pages 332–344. IEEE Computer Society, 1986.

[Wol07] Wayne Wolf. The good news and the bad news (embedded computing column). *IEEE Computer*, 40 (11):104, 2007.

[WVS83] Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. Reasoning about infinite computation paths (extended abstract). In *FOCS*, pages 185–194. IEEE Computer Society, 1983.

[ZN10] Lijun Zhang and Martin R. Neuhäußer. Model checking interactive Markov chains. In Javier Esparza and Rupak Majumdar, editors, *TACAS*, volume 6015 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 2010.

Appendix

Part I

Papers on discrete-time Markovian systems

Paper A:

Deterministic Automata for the (F,G)-Fragment of LTL

Jan Křetínský and Javier Esparza

This paper has been published in P. Madhusudan and Sanjit A. Seshia (eds.): Proceedings of Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012. Lecture Notes in Computer Science, vol. 7358, pages 7-22. Springer, 2012. [KE12]

Summary

Methods for probabilistic LTL model checking and LTL synthesis mostly require to construct a deterministic ω-automaton from a given LTL formula. The standard way is to first translate the formula into a non-deterministic Büchi automaton and then determinise it using Safra's construction or its variants. As this construction is very general and covers more than automata arising from LTL, we design a more specialised translation tailored to the goal mentioned above. We give a translation of an LTL fragment with only the **F** and **G** operators directly to a new kind of a deterministic ω-automaton called generalised Rabin automaton. This automaton can be further de-generalised into a deterministic Rabin automaton. Preliminary experimental results show huge improvements for more complex formulae such as e.g. fairness constraints.

Author's contribution: 85 %

- design of the method,
- proof of correctness,

- experimental implementation and evaluation,
- writing the paper from Section 2 onwards.

Deterministic Automata for the (F,G)-fragment of LTL

Jan Křetínský[1,2⋆] and Javier Esparza[1]

[1] Fakultät für Informatik, Technische Universität München, Germany
{jan.kretinsky,esparza}@in.tum.de
[2] Faculty of Informatics, Masaryk University, Brno, Czech Republic

Abstract. When dealing with linear temporal logic properties in the setting of e.g. games or probabilistic systems, one often needs to express them as deterministic omega-automata. In order to translate LTL to deterministic omega-automata, the traditional approach first translates the formula to a non-deterministic Büchi automaton. Then a determinization procedure such as of Safra is performed yielding a deterministic ω-automaton. We present a direct translation of the (**F**,**G**)-fragment of LTL into deterministic ω-automata with no determinization procedure involved. Since our approach is tailored to LTL, we often avoid the typically unnecessarily large blowup caused by general determinization algorithms. We investigate the complexity of this translation and provide experimental results and compare them to the traditional method.

1 Introduction

The ω-regular languages play a crucial role in formal verification of linear time properties, both from a theoretical and a practical point of view. For model-checking purposes one can comfortably represent them using nondeterministic Büchi automata (NBW), since one only needs to check emptiness of the intersection of two NBWs corresponding to the system and the negation of the property, and NBWs are closed under intersection. However, two increasingly important problems require to represent ω-regular languages by means of *deterministic* automata. The first one is synthesis of reactive modules for LTL specifications, which was theoretically solved by Pnueli and Rosner more than 20 years ago [PR88], but is recently receiving a lot of attention (see the references below). The second one is model checking Markov decision processes (see e.g. [BK08]), where impressive advances in algorithmic development and tool support are quickly extending the range of applications.

It is well known that NBWs are strictly more expressive then their deterministic counterpart, and so cannot be determinized. The standard theoretical solution to this problem is to translate NBW into deterministic Rabin automata (DRW) using Safra's construction [Saf88] or a recent improvement by Piterman

⋆ The author is a holder of Brno PhD Talent Financial Aid and is supported by the Czech Science Foundation, grant No. P202/12/G061.

[Pit06]. However, it is commonly accepted that Safra's construction is difficult to handle algorithmically due to its "messy state space" [Kup12]. Many possible strategies for solving this problem have been investigated. A first one is to avoid Safra's construction altogether. A Safraless approach that reduces the synthesis problem to emptiness of nondeterministic Büchi tree automata has been proposed in [KV05,KPV06]. The approach has had considerable success, and has been implemented in [JB06]. Another strategy is to use heuristics to improve Safra's construction, a path that has been followed in [KB06,KB07] and has produced the ltl2dstar tool [Kle]. Finally, a third strategy is to search for more efficient or simpler algorithms for subclasses of ω-regular languages. A natural choice is to investigate classes of LTL formulas. While LTL is not as expressive as NBW, the complexity of the translation of LTL to DRW still has $2^{2^{\Theta(n)}}$ complexity [KR10]. However, the structure of NBWs for LTL formulas can be exploited to construct a symbolic description of a deterministic parity automaton [MS08]. Fragments of LTL have also been studied. In [AT04], single exponential translations for some simple fragments are presented. Piterman et al. propose in [PPS06] a construction for reactivity(1) formulas that produces in cubic time a symbolic representation of the automaton. The construction has been implemented in the ANZU tool [JGWB07].

Despite this impressive body of work, the problem cannot yet be considered solved. This is particularly so for applications to probabilistic model checking. Since probabilistic model checkers need to deal with linear arithmetic, they profit much less from sophisticated symbolic representations like those used in [PPS06,MS08], or from the Safraless approach which requires to use tree automata. In fact, to the best of our knowledge no work has been done so far in this direction. The most successful approach so far is the one followed by the ltl2dstar tool, which explicitly constructs a reduced DRW. In particular, the ltl2dstar has been reimplemented in PRISM [KNP11], the leading probabilistic model checker.

However, the work carried in [KB06,KB07] has not considered the development of specific algorithms for fragments of LTL. This is the question we investigate in this paper: is it possible to improve on the results of ltl2dstar by restricting attention to a subset of LTL? We give an affirmative answer by providing a very simple construction for the ($\mathbf{F}$,$\mathbf{G}$)-fragment of LTL, i.e., the fragment generated by boolean operations and the temporal operators $\mathbf{F}$ and $\mathbf{G}$. Our construction is still double exponential in the worst case, but is algorithmically very simple. We construct a deterministic Muller automaton for a formula φ of the fragment with a very simple state space: boolean combinations of formulas of the closure of φ. This makes the construction very suitable for applying reductions based on logical equivalences: whenever some logical rule shows that two states are logically equivalent, they can be merged. (This fact is also crucial for the success of the constructions from LTL to NBW.) Since the number of Muller accepting sets can be very large, we also show that the Muller condition of our automata admits a compact representation as a generalized Rabin acceptance condition. We also show how to efficiently transform

this automaton to a standard Rabin automaton. Finally, we report on an implementation of the construction, and present a comparison with ltl2dstar. We show that our construction leads to substantially smaller automata for formulas expressing typical fairness conditions, which play a very important rôle in probabilistic model checking. For instance, while ltl2dstar produces an automaton with over one million states for the formula $\bigwedge_{i=1}^{3}(\mathbf{GF}a_i \rightarrow \mathbf{GF}b_i)$, our construction delivers an automaton with 1560 states.

2 Linear Temporal Logic

This section recalls the notion of linear temporal logic (LTL) [Pnu77].

Definition 1 (LTL Syntax). *The formulae of the (**F**,**G**)-fragment of linear temporal logic are given by the following syntax:*

$$\varphi ::= a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi$$

where a ranges over a finite fixed set Ap of atomic propositions.

We use the standard abbreviations $\mathbf{tt} := a \vee \neg a$, $\mathbf{ff}; = a \wedge \neg a$. We only have negations of atomic propositions, as negations can be pushed inside due to the equivalence of $\mathbf{F}\varphi$ and $\neg\mathbf{G}\neg\varphi$.

Definition 2 (LTL Semantics). *Let $w \in (2^{Ap})^\omega$ be a word. The ith letter of w is denoted $w[i]$, i.e. $w = w[0]w[1]\cdots$. Further, we define the ith suffix of w as $w_i = w[i]w[i+1]\cdots$. The semantics of a formula on w is then defined inductively as follows:*

$$\begin{aligned}
w &\models a && \iff a \in w[0]\\
w &\models \neg a && \iff a \notin w[0]\\
w &\models \varphi \wedge \psi && \iff w \models \varphi \text{ and } w \models \psi\\
w &\models \varphi \vee \psi && \iff w \models \varphi \text{ or } w \models \psi\\
w &\models \mathbf{F}\varphi && \iff \exists\, k \in \mathbb{N} : w_k \models \varphi\\
w &\models \mathbf{G}\varphi && \iff \forall\, k \in \mathbb{N} : w_k \models \varphi
\end{aligned}$$

We define a symbolic one-step unfolding $\mathfrak{U}$ of a formula inductively by the following rules, where the symbol $\mathbf{X}$ intuitively corresponds to the meaning of the standard next operator.

$$\begin{aligned}
\mathfrak{U}(a) &= a\\
\mathfrak{U}(\neg a) &= \neg a\\
\mathfrak{U}(\varphi \wedge \psi) &= \mathfrak{U}(\varphi) \wedge \mathfrak{U}(\psi)\\
\mathfrak{U}(\varphi \vee \psi) &= \mathfrak{U}(\varphi) \vee \mathfrak{U}(\psi)\\
\mathfrak{U}(\mathbf{F}\varphi) &= \mathfrak{U}(\varphi) \vee \mathbf{XF}\varphi\\
\mathfrak{U}(\mathbf{G}\varphi) &= \mathfrak{U}(\varphi) \wedge \mathbf{XG}\varphi
\end{aligned}$$

Example 3. Consider $\varphi = \mathbf{F}a \wedge \mathbf{GF}b$. Then $\mathfrak{U}(\varphi) = (a \vee \mathbf{XF}a) \wedge (b \vee \mathbf{XF}b) \wedge \mathbf{XGF}b$.

3 Deterministic Automaton for the (F,G)-fragment

Let φ be an arbitrary but fixed formula. In the following, we construct a deterministic finite ω-automaton that recognizes the words satisfying φ. The definition of the acceptance condition and its variants follow in the subsequent sections. We start with a construction of the state space. The idea is that a state corresponds to a formula that needs to be satisfied when coming into this state. After evaluating the formulae on the propositions currently read, the next state will be given by what remains in the one-step unfold of the formula. E.g. for Example 3 and reading a, the successor state needs to satisfy $\mathbf{F}b \wedge \mathbf{GF}b$.

In the classical syntactic model constructions, the states are usually given by sets of subformulae of φ. This corresponds to the conjunction of these subformulae. The main difference in our approach is the use of both conjunctions and also disjunctions that allow us to dispose of non-determinism in the corresponding transition function. In order to formalize this, we need some notation.

Let $\mathbb{F}$ and $\mathbb{G}$ denote the set of all subformulae of φ of the form $\mathbf{F}\psi$ and $\mathbf{G}\psi$, respectively. Further, all temporal subformulae are denoted by a shorthand $\mathbb{T} := \mathbb{F} \cup \mathbb{G}$. Finally, for a set of formulae Ψ, we denote $\mathbf{X}\Psi := \{\mathbf{X}\psi \mid \psi \in \Psi\}$.

We denote the *closure* of φ by $\mathbb{C}(\varphi) := Ap \cup \{\neg a \mid a \in Ap\} \cup \mathbf{X}\mathbb{T}$. Then $\mathfrak{U}(\varphi)$ is a positive Boolean combination over $\mathbb{C}(\varphi)$. By states(φ) we denote the set $2^{2^{\mathbb{C}(\varphi)}}$. Each element of states(φ) is a positive Boolean function over $\mathbb{C}(\varphi)$ and we often use a positive Boolean formula as its representative. For instance, the definition of $\mathfrak{U}$ is clearly independent of the choice of representative, hence we abuse the notation and apply $\mathfrak{U}$ to elements of states(φ). Note that $|\text{states}(\varphi)| \in \mathcal{O}(2^{2^{|\varphi|}})$ where $|\varphi|$ denotes the length of φ.

Our state space has two components. Beside the logical component, we also keep track of one-step history of the word read. We usually use letters ψ, χ when speaking about the former component and α, β for the latter one.

Definition 4. *Given a formula φ, we define $\mathcal{A}(\varphi) = (Q, i, \delta)$ to be a deterministic finite automaton over $\Sigma = 2^{Ap}$ given by*

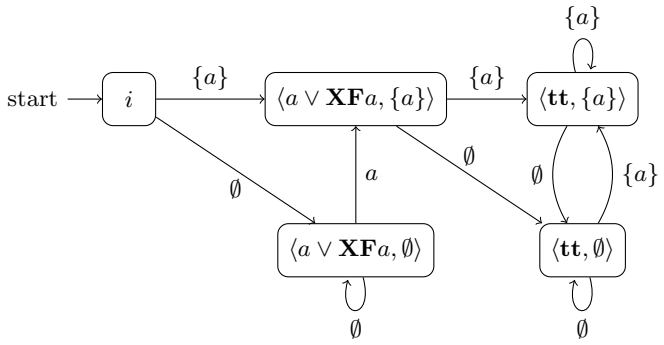
- *the set of states* $Q = \{i\} \cup \Big(\text{states}(\varphi) \times 2^{Ap}\Big)$
- *the initial state i;*
- *the transition function*

$$\delta = \big\{\big(i, \alpha, \langle \mathfrak{U}(\varphi), \alpha\rangle\big) \mid \alpha \in \Sigma\big\} \cup \big\{\big(\langle \psi, \alpha\rangle, \beta, \langle \text{succ}(\psi, \alpha), \beta\rangle\big) \mid \langle \psi, \alpha\rangle \in Q, \beta \in \Sigma\big\}$$

where $\text{succ}(\psi, \alpha) = \mathfrak{U}(\text{next}(\psi[\alpha \mapsto \mathbf{tt}, Ap \setminus \alpha \mapsto \mathbf{ff}])$ *where* $\text{next}(\psi')$ *removes* $\mathbf{X}$*'s from* ψ' *and* $\psi[T \mapsto \mathbf{tt}, F \mapsto \mathbf{ff}]$ *denotes the equivalence class of formulae where in ψ we substitute* $\mathbf{tt}$ *for all elements of T and* $\mathbf{ff}$ *for all elements of F.*

Intuitively, a state $\langle \psi, \alpha\rangle$ of Q corresponds to the situation where ψ needs to be satisfied and α is being read.

Example 5. The automaton for $\mathbf{F}a$ with $Ap = \{a\}$ is depicted in the following figure. The automaton is obviously unnecessarily large, one can expect to merge e.g. the two states bearing the requirement $\mathbf{tt}$ as the proposition a is irrelevant for satisfaction of $\mathbf{tt}$ that does not even contain it. For the sake of simplicity, we leave all possible combinations here and comment on this in Section 8.



The reader might be surprised or even annoyed by the fact that the logical structure of the state space is not sufficient to keep enough information to decide whether a run ρ is accepting. In order to ensure this, we remember one-step history in the state. Why is that? Consider $\varphi = \mathbf{GF}(a \wedge \mathbf{F}b)$. Its unfold is then

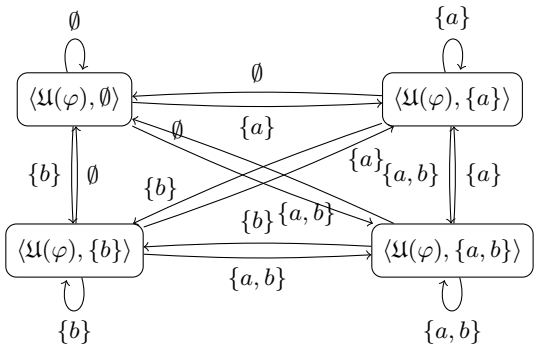
$$\mathbf{XGF}(a \wedge \mathbf{F}b) \wedge \Big(\mathbf{XF}(a \wedge \mathbf{F}b) \vee \big(a \wedge (b \vee \mathbf{XF}b)\big)\Big) \qquad (*)$$

Then moving under $\{a\}$ results into the requirement $\mathbf{GF}(a \wedge \mathbf{F}b) \wedge \big(\mathbf{F}(a \wedge \mathbf{F}b) \vee \mathbf{F}b\big)$ for the next step where the alternative of pure $\mathbf{F}b$ signals progress made by not having to wait for an a. Nevertheless, the unfold of this formula is propositionally equivalent to $(*)$. This is indeed correct as the two formulae are temporally equivalent (i.e. in LTL semantics). Thus, the information about the read a is not kept in the state and the information about this partial progress is lost! And now the next step under both $\{b\}$ and $\emptyset$ again lead to the same requirement $\mathbf{GF}(a \wedge \mathbf{F}b) \wedge \mathbf{F}(a \wedge \mathbf{F}b)$. Therefore, there is no information that if b is read, then it can be matched with the previous a and we already have one satisfaction of (infinitely many required satisfactions of) $\mathbf{F}(a \wedge \mathbf{F}b)$ compared to reading $\emptyset$. Hence, the runs on $(\{a\}\{b\})^\omega$ and $(\{a\}\emptyset)^\omega$ are the same while the former should be accepting and the latter rejecting. However, this can be fixed by remembering the one-step history and using the acceptance condition defined in the following section.

4 Muller Acceptance Condition

In this section, we introduce a Muller acceptance condition. In general, the number of sets in a Muller condition can be exponentially larger than the size of the automaton. Therefore, we investigate the particular structure of the condition. In the next section, we provide a much more compact whilst still useful description of the condition. Before giving the formal definition, let us show an example.

Example 6. Let $\varphi = \mathbf{F}(\mathbf{G}a \vee \mathbf{G}b)$. The corresponding automaton is depicted below, for clarity, we omit the initial state. Observe that the formula stays the same and the only part that changes is the letter currently read that we remember in the state. The reason why is that φ can neither fail in finite time (there is always time to fulfill it), nor can be partially satisfied (no progress counts in this formula, only the infinite suffix). However, at some finite time the argument of $\mathbf{F}$ needs to be satisfied. Although we cannot know when and whether due to $\mathbf{G}a$ or $\mathbf{G}b$, we know it is due to one of these (or both) happening. Thus we may shift the non-determinism to the acceptance condition, which says here: accept if the states where a holds are ultimately never left, or the same happens for b. The commitment to e.g. ultimately satisfying $\mathbf{G}a$ can then be proved by checking that all infinitely often visited states read a.



We now formalize this idea. Let φ be a formula and $\mathcal{A}(\varphi) = (Q, i, \delta)$ its corresponding automaton. Consider a formula χ as a Boolean function over elements of $\mathbb{C}(\varphi)$. For sets $T, F \subseteq \mathbb{C}(\varphi)$, let $\chi[T \mapsto \mathbf{tt}, F \mapsto \mathbf{ff}]$ denote the formula where $\mathbf{tt}$ is substituted for elements of T, and $\mathbf{ff}$ for F. As elements of $\mathbb{C}(\varphi)$ are considered to be atomic expressions here, the substitution is only done on the propositional level and does not go through the modality, e.g. $(a \vee \mathbf{XG}a)[a \to \mathbf{ff}] = \mathbf{ff} \vee \mathbf{XG}a$, which is equivalent to $\mathbf{XG}a$ in the propositional semantics.

Further, for a formula χ and $\alpha \in \Sigma$ and $I \subseteq \mathbb{T}$, we put $I \models_\alpha \chi$ to denote that

$$\chi[\alpha \cup I \mapsto \mathbf{tt}, Ap \setminus \alpha \mapsto \mathbf{ff}]$$

is equivalent to $\mathbf{tt}$ in the propositional semantics. We use this notation to describe that we rely on a commitment to satisfy all formulae of I.

Definition 7 (Muller acceptance). *A set $M \subseteq Q$ is* Muller accepting for a set $I \subseteq \mathbb{T}$ *if the following is satisfied:*

1. *for each $(\chi, \alpha) \in M$, we have $\mathbf{X}I \models_\alpha \chi$,*
2. *for each $\mathbf{F}\psi \in I$ there is $(\chi, \alpha) \in M$ with $I \models_\alpha \psi$,*
3. *for each $\mathbf{G}\psi \in I$ and for each $(\chi, \alpha) \in M$ we have $I \models_\alpha \psi$.*

A set $F \subseteq Q$ is Muller accepting (for φ) *if it is Muller accepting for some $I \subseteq \mathbb{T}$.*

The first condition ensures that the commitment to formulae in I being ultimately satisfied infinitely often is enough to satisfy the requirements. The second one guarantees that each **F**-formula is unfolded only finitely often and then satisfied, while the third one guarantees that **G**-formulae indeed ultimately hold. Note that it may be impossible to see the satisfaction of a formula directly and one must rely on further promises, formulae of smaller size. In the end, promising the atomic proposition is not necessary and is proven directly from the second component of the state space.

4.1 Correctness

Given a formula φ, we have defined a Muller automaton $\mathcal{A}(\varphi)$ and we let the acceptance condition $\mathcal{M}(\varphi) = \{M_1, \ldots, M_k\}$ be given by all the Muller accepting sets M_i for φ. Every word $w : \mathbb{N} \to 2^{Ap}$ induces a run $\rho = \mathcal{A}(\varphi)(w) : \mathbb{N} \to Q$ starting in i and following δ. The run is thus accepting and the word is accepted if the set of states visited infinitely often $\mathrm{Inf}(\rho)$ is Muller accepting for φ. Vice versa, a run $\rho = i(\chi_1, \alpha_1)(\chi_2, \alpha_2)\cdots$ induces a word $Ap(\rho) = \alpha_1\alpha_2\cdots$. We now prove that this acceptance condition is sound and complete.

Theorem 8. *Let φ be a formula and w a word. Then w is accepted by the deterministic automaton $\mathcal{A}(\varphi)$ with the Muller condition $\mathcal{M}(\varphi)$ if and only if $w \models \varphi$.*

We start by proving that the first component of the state space takes care of all progress or failure in finite time.

Proposition 9 (Local (finitary) correctness). *Let w be a word and $\mathcal{A}(\varphi)(w) = i(\chi_0, \alpha_0)(\chi_1, \alpha_1)\cdots$ the corresponding run. Then for all $n \in \mathbb{N}$, we have $w \models \varphi$ if and only if $w_n \models \chi_n$.*

Proof (Sketch). The one-step unfold produces a temporally equivalent (w.r.t. LTL satisfaction) formula. The unfold is a Boolean function over atomic propositions and elements of $\mathbf{X}\mathbb{T}$. Therefore, this unfold is satisfied if and only if the next state satisfies $\mathrm{next}(\psi)$ where ψ is the result of partial application of the Boolean function to the currently read letter of the word. We conclude by induction. □

Further, each occurrence of satisfaction of **F** must happen in finite time. As a consequence, a run with $\chi_i \not\equiv \mathbf{ff}$ is rejecting if and only if satisfaction of some $\mathbf{F}\psi$ is always postponed.

Proposition 10 (Completeness). *If $w \models \varphi$ then $\mathrm{Inf}(\mathcal{A}(\varphi)(w))$ is a Muller accepting set.*

Proof. Let us show that $M := \mathrm{Inf}(\mathcal{A}(\varphi)(w))$ is Muller accepting for

$$I := \{\psi \in \mathbb{F} \mid w \models \mathbf{G}\psi\} \cup \{\psi \in \mathbb{G} \mid w \models \mathbf{F}\psi\}$$

As a technical device we use the following. For every finite Boolean combination ψ of elements of the closure $\mathbb{C}$, there are only finitely many options to satisfy

it, each corresponding to a subset of $\mathbb{C}$. Therefore, if $w_i \models \psi$ for infinitely many $i \in \mathbb{N}$ then at least one of the options has to recur. More precisely, for some subset $\alpha \subseteq Ap$ there are infinitely many $i \in \mathbb{N}$ with $w_i \models \psi \cup \alpha \cup \{\neg a \mid a \in Ap \setminus \alpha\}$. For each such α we pick one subset $I_{\chi,\alpha} \subseteq \mathbb{T}$ such that for infinitely many i, after reading $w^i = w[0] \cdots w[i]$ we are in state (χ, α) and $w_i \models \psi \cup \mathbf{X} I_{\chi,\alpha}$, and $I_{\chi,\alpha} \models_\alpha \psi$. We say that we have a *recurring set $I_{\chi,\alpha}$ modelling ψ (for a state (χ, α))*. Obviously, the recurring sets for all states are included in I, i.e. $I_{\chi,\alpha} \subseteq I$ for every $(\chi, \alpha) \in Q$.

Let us now proceed with proving the three conditions of Definition 7 for M and I.

Condition 1. Let $(\chi, \alpha) \in M$. Since $w \models \varphi$, by Proposition 9 $w_i \models \chi$ whenever we enter (χ, α) after reading w^i, which happens for infinitely many $i \in \mathbb{N}$. Hence we have a recurring set $I_{\chi,\alpha}$ modelling χ. Since $I_{\chi,\alpha} \models_\alpha \chi$, we get also $I \models_\alpha \chi$ by $I_{\chi,\alpha} \subseteq I$.

Condition 2. Let $\mathbf{F}\psi \in I$, then $w \models \mathbf{GF}\psi$. Since there are finitely many states, there is $(\chi, \alpha) \in M$ for which after infinitely many entrances by w^i it holds $w_i \models \psi$ by Proposition 9, hence we have a recurring set $I_{\chi,\alpha}$ modelling ψ and conclude as above.

Condition 3. Let $\mathbf{G}\psi \in I$, then $w \models \mathbf{FG}\psi$. Hence for every $(\chi, \alpha) \in M$ infinitely many w^i leading to (χ, α) satisfy $w_i \models \psi$ by Proposition 9, hence we have a recurring set $I_{\chi,\alpha}$ modelling ψ and conclude as above. □

Before proving the opposite direction of the theorem, we provide a property of Muller accepting sets opposite to the previous proposition.

Lemma 11. *Let ρ be a run. If $\mathrm{Inf}(\rho)$ is Muller accepting for I then $Ap(\rho) \models \mathbf{G}\psi$ for each $\psi \in I \cap \mathbb{F}$ and $Ap(\rho) \models \mathbf{F}\psi$ for each $\psi \in I \cap \mathbb{G}$.*

Proof. Denote $w = Ap(\rho)$. Let us first assume $\psi \in I \cap \mathbb{F}$ and $w_j \not\models \psi$ for all $j \geq i \in \mathbb{N}$. Since $\psi \in I \cap \mathbb{F}$, for infinitely many j, ρ passes through some $(\chi, \alpha) \in \mathrm{Inf}(\rho)$ for which $I \models_\alpha \psi$. Hence, there is $\psi_1 \in I$ which is a subformula of ψ such that for infinitely many i, $w_i \not\models \psi_1$. If $\psi_1 \in \mathbb{F}$, we proceed as above; similarly for $\psi_1 \in \mathbb{G}$. Since we always get a smaller subformula, at some point we obtain either $\psi_n = \mathbf{F}\beta$ or $\psi_n = \mathbf{G}\beta$ with β a Boolean combination over Ap and we get a contradiction with the second or the third point of Definition 7, respectively. □

In other words, if we have a Muller accepting set for I then all elements of I hold true in w_i for almost all i.

Proposition 12 (Soundness). *If $\mathrm{Inf}(\mathcal{A}(\varphi)(w))$ is a Muller accepting set then $w \models \varphi$.*

Proof. Let $M := \mathrm{Inf}(\mathcal{A}(\varphi)(w))$ be a Muller accepting set for some I. There is $i \in \mathbb{N}$ such that after reading w^i we come to (χ, α) and stay in $\mathrm{Inf}(\mathcal{A}(\varphi)(w))$ from now on and, moreover, $w_i \models \psi$ for all $\psi \in I$ by Lemma 11. For a contradiction, let $w \not\models \varphi$. By Proposition 9 we thus get $w_i \not\models \chi$. By the first condition of Definition 7, we get $I \models_\alpha \chi$. Therefore, there is $\psi \in I$ such that $w_i \not\models \psi$, a contradiction. □

5 Generalized Rabin Condition

In this section, we investigate the structure of the previously defined Muller condition and propose a new type of acceptance condition that compactly, yet reasonably explicitly captures the accepting sets.

Let us first consider a fixed $I \subseteq \mathbb{T}$ and examine all Muller accepting sets for I. The first condition of Definition 7 requires not to leave the set of states $\{(\chi, \alpha \mid I \models_\alpha \chi)\}$. Similarly, the third condition is a conjunction of $|I \cap \mathbb{G}|$ conditions not to leave sets $\{(\chi, \alpha) \mid I \models_\alpha \psi\}$ for each $\mathbf{G}\psi \in I$. Both conditions thus together require that certain set (complement of the intersection of the above sets) is visited only finitely often. On the other hand, the second condition requires to visit certain sets infinitely often. Indeed, for each $\mathbf{F}\psi$ the set $\{(\chi, \alpha) \mid I \models_\alpha \psi\}$ must be visited infinitely often.

Furthermore, a set is accepting if the conditions above hold for *some* set I. Hence, the acceptance condition can now be expressed as a positive Boolean combination over Rabin pairs in a similar way as the standard Rabin condition is a disjunction of Rabin pairs.

Example 13. Let us consider the (strong) fairness constraint $\varphi = \mathbf{FG}a \vee \mathbf{GF}b$. Since each atomic proposition has both $\mathbf{F}$ and $\mathbf{G}$ as ancestors in the syntactic tree, it is easy to see that there is only one reachable element of states(φ) and the state space of $\mathcal{A}$ is $\{i\} \cup 2^{\{a,b\}}$, i.e. of size $1 + 2^2 = 5$. Furthermore, the syntactic tree of $\mathfrak{U}(\varphi) = \mathbf{XFG}a \vee (\mathbf{XG}a \wedge a) \vee (\mathbf{XGF}b \wedge (\mathbf{XF}b \vee b))$ immediately determines possible sets I. These either contain $\mathbf{G}a$ (possibly with also $\mathbf{FG}a$ or some other elements) or $\mathbf{GF}b, \mathbf{F}b$. The first option generates the requirement to visit states with $\neg a$ only finitely often, the second one to visit b infinitely often. Thus the condition can be written as

$$(\{q \mid q \models \neg a\}, Q) \vee (\emptyset, \{q \mid q \models b\})$$

and is in fact a Rabin acceptance condition.

We formalize this new type of acceptance condition as follows.

Definition 14 (Generalized Rabin Automaton). *A* generalized Rabin automaton *is a (deterministic) ω-automaton $\mathcal{A} = (Q, i, \delta)$ over some alphabet Σ, where Q is a set of states, i is the initial state, $\delta : Q \times \Sigma \to Q$ is a transition function, together with a* generalized Rabin condition $\mathcal{GR} \in \mathcal{B}^+(2^Q \times 2^Q)$. *A run ρ of $\mathcal{A}$ is accepting if* $\mathrm{Inf}(\rho) \models \mathcal{GR}$, *which is defined inductively as follows:*

$$\begin{aligned}
\mathrm{Inf}(\rho) &\models \varphi \wedge \psi && \iff \mathrm{Inf}(\rho) \models \varphi \text{ and } \mathrm{Inf}(\rho) \models \psi \\
\mathrm{Inf}(\rho) &\models \varphi \vee \psi && \iff \mathrm{Inf}(\rho) \models \varphi \text{ or } \mathrm{Inf}(\rho) \models \psi \\
\mathrm{Inf}(\rho) &\models (F, I) && \iff F \cap \mathrm{Inf}(\rho) = \emptyset \text{ and } I \cap \mathrm{Inf}(\rho) \neq \emptyset
\end{aligned}$$

The generalized Rabin condition corresponding to the previously defined Muller condition $\mathcal{M}$ can now be formalized as follows.

Definition 15 (Generalized Rabin Acceptance). *Let φ be a formula. The* generalized Rabin condition $\mathcal{GR}(\varphi)$ *is*

$$\bigvee_{I \subseteq \mathbb{T}} \left(\Big(\{(\chi, \alpha) \mid I \not\models_\alpha \chi \wedge \bigwedge_{\mathbf{G}\psi \in I} \psi\}, Q \Big) \wedge \bigwedge_{\mathbf{F}\omega \in I} \Big(\emptyset, \{(\chi, \alpha) \mid I \models_\alpha \omega\} \Big) \right)$$

By the argumentation above, we get the equivalence of the Muller and the generalized Rabin conditions for φ and thus the following.

Proposition 16. *Let φ be a formula and w a word. Then w is accepted by the deterministic automaton $\mathcal{A}(\varphi)$ with the generalized Rabin condition $\mathcal{GR}(\varphi)$ if and only if $w \models \varphi$.*

Example 17. Let us consider a conjunction of two (strong) fairness constraints $\varphi = (\mathbf{FG}a \vee \mathbf{GF}b) \wedge (\mathbf{FG}c \vee \mathbf{GF}d)$. Since each atomic proposition is wrapped in either **FG** or **GF**, there is again only one relevant element of states(φ) and the state space of $\mathcal{A}$ is $\{i\} \cup 2^{\{a,b,c,d\}}$, i.e. of size $1 + 2^4 = 17$. From the previous example, we already know the disjunctions correspond to $(\neg a, Q) \vee (\emptyset, b)$ and $(\neg c, Q) \vee (\emptyset, d)$. Thus for the whole conjunction, we get a generalized Rabin condition

$$\Big((\neg a, Q) \vee (\emptyset, b) \Big) \wedge \Big((\neg c, Q) \vee (\emptyset, d) \Big)$$

6 Rabin Condition

In this section, we briefly describe how to obtain a Rabin automaton from $\mathcal{A}(\varphi)$ and the generalized Rabin condition $\mathcal{GR}(\varphi)$ of Definition 15. For a fixed I, the whole conjunction of Definition 15 corresponds to the intersection of automata with different Rabin conditions. In order to obtain the intersection, one has first to construct the product of the automata, which in this case is still the original automaton with the state space Q, as they are all the same. Further, satisfying

$$(G, Q) \wedge \bigwedge_{f \in \mathcal{F} := I \cap \mathbb{F}} (\emptyset, F_f)$$

amounts to visiting G only finitely often and each F_f infinitely often. To check the latter (for a non-empty conjunction), it is sufficient to multiply the state space by $\mathcal{F}$ with the standard trick that we leave the fth copy once we visit F_f and immediately go to the next copy. The resulting Rabin pair is thus

$$\Big(G \times \mathcal{F}, F_{\bar{f}} \times \{\bar{f}\} \Big)$$

for an arbitrary fixed $\bar{f} \in \mathcal{F}$.

As for the disjunction, Rabin condition is closed under it as it simply takes the union of the pairs when the two automata have the same state space. In our case, one can multiply the state space of each disjunct corresponding to I by all

$J \cap \mathbb{F}$ for each $J \in 2^{\mathbb{T}} \setminus \{I\}$ to get the same state space for all of them. We thus get a bound for the state space

$$\prod_{I \subseteq \mathbb{T}} |I \cap \mathbb{F}| \cdot |Q|$$

Example 18. The construction of Definition 15 for the two fairness constraints Example 17 yields

$$(\neg a \vee \neg c, Q) \vee (\neg a, d) \vee (\neg c, b) \vee \big((\emptyset, b) \wedge (\emptyset, d)\big)$$

where we omitted all pairs (F, I) for which we already have a pair (F', I') with $F \subseteq F'$ and $I \supseteq I'$. One can eliminate the conjunction as described above at the cost of multiplying the state space by two. The corresponding Rabin automaton thus has $2 \cdot 1 \cdot |\{i\} \cup 2^{Ap}| = 34$ states. (Of course, for instance the initial state need not be duplicated, but for the sake of simplicity of the construction we avoid any optimizations.)

For a conjunction of three conditions, $\varphi = (\mathbf{FG}a \vee \mathbf{GF}b) \wedge (\mathbf{FG}c \vee \mathbf{GF}d) \wedge (\mathbf{FG}e \vee \mathbf{GF}f)$, the right components of the Rabin pairs correspond to $\mathbf{tt}, b, d, f, b \wedge d, b \wedge f, d \wedge f, b \wedge d \wedge f$. The multiplication factor to obtain a Rabin automaton is thus $2 \cdot 2 \cdot 2 \cdot 3 = 24$ and the state space is of the size $24 \cdot 1 \cdot (1 + 2^6) = 1560$.

7 Complexity

In this section, we summarize the theoretical complexity bounds we have obtained.

The traditional approach first translates the formula φ of length n into a non-deterministic automaton of size $\mathcal{O}(2^n)$. Then the determinization follows. The construction of Safra has the complexity $m^{\mathcal{O}(m)}$ where m is the size of the input automaton [Saf88]. This is in general optimal. The overall complexity is thus

$$2^{n \cdot \mathcal{O}(2^n)} = 2^{\mathcal{O}(2^{n + \log n})}$$

The recent lower bound for the whole LTL is $2^{2^{\Omega(n)}}$ [KR10]. However, to be more precise, the example is of size less than $2^{\mathcal{O}(2^n)}$. Hence, there is a small gap. To the authors' best knowledge, there is no better upper bound when restricting to automata arising from LTL formulae or from the full (**F**,**G**)-fragment. (There are results on smaller fragments [AT04] though.) We tighten this gap slightly as shown below. Further, note that the number of Rabin pairs is $\mathcal{O}(m) = \mathcal{O}(2^n)$.

Our construction first produces a Muller automaton of size

$$\mathcal{O}(2^{2^{|\mathbb{T}|}} \cdot 2^{|Ap|}) = \mathcal{O}(2^{2^n + n}) \subseteq 2^{\mathcal{O}(2^n)}$$

which is strictly less than in the traditional approach. Moreover, as already discussed in Example 13, one can consider an "infinitary" fragment where every atomic proposition has in the syntactic tree both **F**and **G**as some ancestors. In this fragment, the state space of the Muller/generalized Rabin automaton

is simply 2^{Ap} (when omitting the initial state) as for all $\alpha \subseteq Ap$, we have $\mathrm{succ}(\varphi, \alpha) = \varphi$. This is useful, since for e.g. fairness constraints our procedure yields exponentially smaller automaton.

Although the size of the Muller acceptance condition can be potentially exponentially larger than the state space, we have shown it can be compactly written as a disjunction of up to 2^n of conjunctions each of size at most n.

Moreover, using the intersection procedure we obtain a Rabin automaton with the upper bound on the state space

$$|\mathbb{F}|^{2^{|\mathbb{T}|}} \cdot |Q| \in n^{2^n} \cdot 2^{\mathcal{O}(2^n)} = 2^{\mathcal{O}(\log n \cdot 2^n)} = 2^{\mathcal{O}(2^{n+\log\log n})} \subsetneq 2^{\mathcal{O}(2^{n+\log n})}$$

thus slightly improving the upper bound. Further, each conjunction is transformed into one pair, we are thus left with at most $2^{|\mathbb{T}|} \in \mathcal{O}(2^n)$ Rabin pairs.

8 Experimental Results and Evaluation

We have implemented the construction of the state space of $\mathcal{A}(\varphi)$ described above. Further, Definition 15 then provides a way to compute the multiplication factor needed in order to get the Rabin automaton. We compare the sizes of this generalized Rabin automaton and Rabin automaton with the Rabin automaton produced by ltl2dstar. Ltl2dstar first calls an external translator from LTL to non-deterministic Büchi automata. In our experiments, it is LTL2BA [GO01] recommended by the authors of ltl2dstar. Then it performs Safra's determinization. Ltl2dstar implements several optimizations of Safra's construction. The optimizations shrink the state space by factor of 5 (saving 79.7% on average on the formulae considered here) to 10 (89.7% on random formulae) [KB06]. Our implementation does not perform any ad hoc optimization, since we want to evaluate whether the basic idea of the Safraless construction is already competitive. The only optimizations done are the following.

- Only the reachable part of the state space is generated.
- Only atomic propositions relevant in each state are considered. In a state (χ, α), a is not relevant if $\chi[a \mapsto \mathbf{tt}] \equiv \chi[a \mapsto \mathbf{ff}]$, i.e. if for every valuation, χ has the same value no matter which value a takes. For instance, let $Ap = \{a, b\}$ and consider $\chi = \mathfrak{U}(\mathbf{F}a) = \mathbf{F}a \vee a$. Then instead of having four copies (for $\emptyset, \{a\}, \{b\}, \{a, b\}$), there are only two for the *sets* of valuations $\{\emptyset, \{b\}\}$ and $\{\{a\}, \{a, b\}\}$. For its successor $\mathbf{tt}$, we only have one copy standing for the whole set $\{\emptyset, \{a\}, \{b\}, \{a, b\}\}$.
- Definition 15 takes a disjunction over $I \in 2^{\mathbb{T}}$. If $I \subseteq I'$ but the set of states (χ, α) with $I \models_\alpha \chi$ and $I' \models_\alpha \chi$ are the same, it is enough to consider the disjunct for I only. E.g. for $\mathfrak{U}(\mathbf{G}(\mathbf{F}a \vee \mathbf{F}b))$, we only consider I either $\{\mathbf{G}(\mathbf{F}a \vee \mathbf{F}b), \mathbf{F}a\}$ or $\{\mathbf{G}(\mathbf{F}a \vee \mathbf{F}b), \mathbf{F}b\}$, but not their union.
  This is an instance of a more general simplification. For a conjunction of pairs $(F_1, I_1) \wedge (F_2, I_2)$ with $I_1 \subseteq I_2$, there is a single equivalent condition $(F_1 \cup F_2, I_1)$.

Table 1 shows the results on formulae from BEEM (BEnchmarks for Explicit Model checkers)[Pel07] and formulae from [SB00] on which ltl2dstar was originally tested [KB06]. In both cases, we only take formulae of the (**F**,**G**)-fragment. In the first case this is 11 out of 20, in the second 12 out of 28. There is a slight overlap between the two sets. Further, we add conjunctions of strong fairness conditions and a few other formulae. For each formula φ, we give the number $|\text{states}(\varphi)|$ of distinct states w.r.t. the first (logical) component. The overall number of states of the Muller or generalized Rabin automaton follows. The respective runtimes are not listed as they were less than a second for all listed formulae, with the exception of the fifth formula from the bottom where it needed 3 minutes (here ltl2dstar needed more than one day to compute the Rabin automaton). In the column $\mathcal{GR}$-factor, we describe the complexity of the generalized Rabin condition, i.e. the number of copies of the state space that are created to obtain an equivalent Rabin automaton, whose size is thus bounded from above by the column Rabin. The last column states the size of the state space of the Rabin automaton generated by ltl2dstar using LTL2BA.

Table 1. Experimental comparison to ltl2dstar on formulae of [Pel07], [SB00], fairness constraints and some other examples of formulae of the "infinitary" fragment

Formula	states	Muller/GR	$\mathcal{GR}$-factor	Rabin	ltl2dstar
$\mathbf{G}(a \vee \mathbf{F}b)$	2	5	1	5	4
$\mathbf{FG}a \vee \mathbf{FG}b \vee \mathbf{GF}c$	1	9	1	9	36
$\mathbf{F}(a \vee b)$	2	4	1	4	2
$\mathbf{GF}(a \vee b)$	1	3	1	3	4
$\mathbf{G}(a \vee b \vee c)$	2	4	1	4	3
$\mathbf{G}(a \vee \mathbf{F}b)$	2	5	1	5	4
$\mathbf{G}(a \vee \mathbf{F}(b \vee c))$	2	5	1	5	4
$\mathbf{F}a \vee \mathbf{G}b$	3	7	1	7	5
$\mathbf{G}(a \vee \mathbf{F}(b \wedge c))$	2	5	1	5	4
$(\mathbf{FG}a \vee \mathbf{GF}b)$	1	5	1	5	12
$\mathbf{GF}(a \vee b) \wedge \mathbf{GF}(b \vee c)$	1	5	2	10	12
$(\mathbf{FF}a \wedge \mathbf{G}\neg a) \vee (\mathbf{GG}\neg a \wedge \mathbf{F}a)$	2	4	1	4	1
$(\mathbf{GF}a) \wedge \mathbf{FG}b$	1	5	1	5	7
$(\mathbf{GF}a \wedge \mathbf{FG}b) \vee (\mathbf{FG}\neg a \wedge \neg b)$	1	5	1	5	14
$\mathbf{FG}a \wedge \mathbf{GF}a$	1	3	1	3	3
$\mathbf{G}(\mathbf{F}a \wedge \mathbf{F}b)$	1	5	2	10	5
$\mathbf{F}a \wedge \mathbf{F}b$	4	8	1	8	4
$(\mathbf{G}(b \vee \mathbf{GF}a) \wedge \mathbf{G}(c \vee \mathbf{GF}\neg a)) \vee \mathbf{G}b \vee \mathbf{G}c$	4	18	2	36	26
$(\mathbf{G}(b \vee \mathbf{FG}a) \wedge \mathbf{G}(c \vee \mathbf{FG}\neg a)) \vee \mathbf{G}b \vee \mathbf{G}c$	4	18	1	18	29
$(\mathbf{F}(b \wedge \mathbf{FG}a) \vee \mathbf{F}(c \wedge \mathbf{FG}\neg a)) \wedge \mathbf{F}b \wedge \mathbf{F}c$	4	18	1	18	8
$(\mathbf{F}(b \wedge \mathbf{GF}a) \vee \mathbf{F}(c \wedge \mathbf{GF}\neg a)) \wedge \mathbf{F}b \wedge \mathbf{F}c$	4	18	1	18	45
$(\mathbf{FG}a \vee \mathbf{GF}b)$	1	5	1	5	12
$(\mathbf{FG}a \vee \mathbf{GF}b) \wedge (\mathbf{FG}c \vee \mathbf{GF}d)$	1	17	2	34	17527
$\bigwedge_{i=1}^{3}(\mathbf{GF}a_i \rightarrow \mathbf{GF}b_i)$	1	65	24	1 560	1 304 706
$(\bigwedge_{i=1}^{5} \mathbf{GF}a_i) \rightarrow \mathbf{GF}b$	1	65	1	65	972
$\mathbf{GF}(\mathbf{F}a\mathbf{GF}b\mathbf{FG}(a \vee b))$	1	5	1	5	159
$\mathbf{FG}(\mathbf{F}a \vee \mathbf{GF}b \vee \mathbf{FG}(a \vee b))$	1	5	1	5	2918
$\mathbf{FG}(\mathbf{F}a \vee \mathbf{GF}b \vee \mathbf{FG}(a \vee b) \vee \mathbf{FG}b)$	1	5	1	5	4516

While the advantages of our approach over the general determinization are clear for the infinitary fragment, there seem to be some drawbacks when "finitary" behaviour is present, i.e. behaviour that can be satisfied or disproved after finitely many steps. The reason and the patch for this are the following. Consider the formula $\mathbf{F}a$ and its automaton from Example 5. Observe that one can easily collapse the automaton to the size of only 2. The problem is that some states such as $\langle a \vee \mathbf{XF}a, \{a\}\rangle$ are only "passed through" and are equivalent to some of their successors, here $\langle \mathbf{tt}, \{a\}\rangle$. However, we may safely perform the following collapse. Whenever two states $(\chi, \alpha), (\chi', \alpha)$ satisfy that $\chi[\alpha \mapsto \mathbf{tt}, Ap \setminus \alpha \mapsto \mathbf{ff}]$ is propositionally equivalent to $\chi'[\alpha \mapsto \mathbf{tt}, Ap \setminus \alpha \mapsto \mathbf{ff}]$ we may safely merge the states as they have the same properties: they are bisimilar with the same set of atomic propositions satisfied. Using these optimizations, e.g. the automaton for $\mathbf{F}a \wedge \mathbf{F}b$ has size 4 as the one produced by ltl2dstar.

Next important observation is that the blow-up from generalized Rabin to Rabin automaton (see the column $\mathcal{GR}$-factor) corresponds to the number of elements of $\mathbb{F}$ that have a descendant or an ancestor in $\mathbb{G}$ and are combined with conjunction. This follows directly from the transformation described in Section 6 and is illustrated in the table.

Thus, we may conclude that our approach is competitive to the determinization approach and for some classes of useful properties such as fairness constraints or generally the infinitary properties it shows significant advantages. Firstly, the state space of the Rabin automaton is noticeably smaller. Secondly, compact generalized Rabin automata tend to be small even for more complex formulae. Thirdly, the state spaces of our automata have a clear structure to be exploited for further possible optimizations, which is more difficult in the case of determinization. In short, the state space is less "messy".

9 Discussion on Extensions

Our approach seems to be extensible to the (**X**,**F**,**G**)-fragment. In this setting, instead of remembering the one-step history one needs to remember n last steps (or have a n-step look-ahead) in order to deal with formulae such as $\mathbf{GF}(a \wedge \mathbf{X}b)$. Indeed, the acceptance condition requires to visit infinitely often a state provably satisfying $a \wedge \mathbf{X}b$. This can be done by remembering the last n symbols read, where n can be chosen to be the nesting depth of **X**s. We have not presented this extension mainly for the sake of clarity of the construction.

Further, one could handle the positive (**X**,**U**)-fragment, where only atomic propositions may be negated as defined above. These formulae are purely "finitary" and the logical component of the state space is sufficient. Indeed, the automaton simply accepts if and only if **tt** is reached and there is no need to check any formulae that we had committed to.

For the (**U**,**G**)-fragment or the whole LTL, our approach would need to be significantly enriched as the state space (and last n symbols read) is not sufficient to keep enough information to decide whether a run ρ is accepting only based on $\mathrm{Inf}(\rho)$. Indeed, consider a formula $\varphi = \mathbf{GF}(a \wedge b\mathbf{U}c)$. Then reading $\{a, b\}$ results

in the requirement $\mathbf{GF}(a \wedge b\mathbf{U}c) \wedge \big(\mathbf{F}(a \wedge b\mathbf{U}c) \vee (b\mathbf{U}c)\big)$ which is, however, temporally equivalent to φ (their unfolds are propositionally equivalent). Thus, runs on $(\{a,b\}\{c\}\emptyset)^\omega$ and $(\{a,b\}\emptyset\{c\})^\omega$ have the same set of infinitely often visited states. Hence, the order of visiting the states matters and one needs the history. However, words such as $(\{a,b\}\{b\}^n\{c\})^\omega$ vs. $(\{b\}^n\{c\})^\omega$ show that more complicated structure is needed than last n letters. The conjecture that this approach is extensible to the whole LTL is left open and considered for future work.

10 Conclusions

We have shown a direct translation of the LTL fragment with operators **F** and **G** to deterministic automata. This translation has several advantages compared to the traditional way that goes via non-deterministic Büchi automata and then performs determinization. First of all, in our opinion it is a lot simpler than the determinization and its various non-trivial optimizations. Secondly, the state space has a clear logical structure. Therefore, any work with the automata or further optimizations seem to be conceptually easier. Moreover, many optimizations are actually done by the logic itself. Indeed, logical equivalence of the formulae helps to shrink the state space with no further effort. In a sense, the logical part of a state contains precisely the information that the semantics of LTL dictates, see Proposition 9. Thirdly, the state space is—according to the experiments—not much bigger even when compared to already optimized determinization. Moreover, very often it is considerably smaller, especially for the "infinitary" formulae; in particular, for fairness conditions. Furthermore, we have also given a very compact deterministic ω-automaton with a small and in our opinion reasonably simple generalized Rabin acceptance condition.

Although we presented a possible direction to extend the approach to the whole LTL, we leave this problem open and will focus on this in future work. Further, since only the obvious optimizations mentioned in Section 8 have been implemented so far, there is space for further performance improvements in this new approach.

Acknowledgement

Thanks to Andreas Gaiser for pointing out to us that ltl2dstar constructs surprisingly large automata for fairness constraints and the anonymous reviewers for their valuable comments.

References

[AT04] Rajeev Alur and Salvatore La Torre. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Log.*, 5(1):1–25, 2004.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking.* MIT Press, 2008.

[GO01] Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In *CAV*, volume 2102 of *LNCS*, pages 53–65. Springer, 2001. Tool accessible at http://www.lsv.ens-cachan.fr/ gastin/ltl2ba/.

[JB06] Barbara Jobstmann and Roderick Bloem. Optimizations for LTL synthesis. In *FMCAD*, pages 117–124. IEEE Computer Society, 2006.

[JGWB07] Barbara Jobstmann, Stefan J. Galler, Martin Weiglhofer, and Roderick Bloem. Anzu: A tool for property synthesis. In *CAV*, volume 4590 of *LNCS*, pages 258–262. Springer, 2007.

[KB06] Joachim Klein and Christel Baier. Experiments with deterministic *omega*-automata for formulas of linear temporal logic. *Theor. Comput. Sci.*, 363(2):182–195, 2006.

[KB07] Joachim Klein and Christel Baier. On-the-fly stuttering in the construction of deterministic *omega* -automata. In *CIAA*, volume 4783 of *LNCS*, pages 51–61. Springer, 2007.

[Kle] Joachim Klein. ltl2dstar - LTL to deterministic Streett and Rabin automata. http://www.ltl2dstar.de/.

[KNP11] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[KPV06] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. Safraless compositional synthesis. In *CAV*, volume 4144 of *LNCS*, pages 31–44. Springer, 2006.

[KR10] Orna Kupferman and Adin Rosenberg. The blowup in translating LTL to deterministic automata. In *MoChArt*, volume 6572 of *LNCS*, pages 85–94. Springer, 2010.

[Kup12] Orna Kupferman. Recent challenges and ideas in temporal synthesis. In *SOFSEM*, volume 7147 of *LNCS*, pages 88–98. Springer, 2012.

[KV05] Orna Kupferman and Moshe Y. Vardi. Safraless decision procedures. In *FOCS*, pages 531–542. IEEE Computer Society, 2005.

[MS08] Andreas Morgenstern and Klaus Schneider. From LTL to symbolically represented deterministic automata. In *VMCAI*, volume 4905 of *LNCS*, pages 279–293. Springer, 2008.

[Pel07] Radek Pelánek. Beem: Benchmarks for explicit model checkers. In *Proc. of SPIN Workshop*, volume 4595 of *LNCS*, pages 263–267. Springer, 2007.

[Pit06] Nir Piterman. From nondeterministic Buchi and Streett automata to deterministic parity automata. In *LICS*, pages 255–264. IEEE Computer Society, 2006.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.

[PPS06] Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. In *VMCAI*, volume 3855 of *LNCS*, pages 364–380. Springer, 2006.

[PR88] Amir Pnueli and Roni Rosner. A framework for the synthesis of reactive modules. In *Concurrency*, volume 335 of *LNCS*, pages 4–17. Springer, 1988.

[Saf88] Shmuel Safra. On the complexity of ω-automata. In *FOCS*, pages 319–327. IEEE Computer Society, 1988.

[SB00] Fabio Somenzi and Roderick Bloem. Efficient Büchi automata from LTL formulae. In *CAV*, volume 1855 of *LNCS*, pages 248–263. Springer, 2000.

Paper B:

Rabinizer:
Small Deterministic Automata for LTL(F,G)

Andreas Gaiser, Jan Křetínský, and Javier Esparza

This tool paper has been published in Supratik Chakraborty and Madhavan Mukund (eds.): Proceedings of Automated Technology for Verification and Analysis - 10th International Symposium, ATVA 2012, Thiruvananthapuram, India, October 3-6, 2012. Lecture Notes in Computer Science, vol. 7561, pages 72-76. Springer, 2012. [GKE12]

Summary

Methods for probabilistic LTL model checking and LTL synthesis mostly require to construct a deterministic ω-automaton from a given LTL formula. We implement the approach of [KE12] (Paper A) for the LTL fragment with **F** and **G** operators and produce the respective Rabin automata. Compared to [KE12], where only a very preliminary implementation was used, we implement the whole process and optimise it significantly. Firstly, the information carried in the states is modified in order to avoid constructing many redundant transient states. Secondly, the intermediate generalised Rabin acceptance condition is simplified resulting in huge savings in the size of the de-generalised automaton. Thirdly, the de-generalisation is optimised so that the redundant intermediately accepting states are not created. Fourthly, each copy in the de-generalisation is factorised with respect to the congruence induced by the acceptance condition of that copy. Finally, we provide not only command line interface, but also a graphical web interface.

Author's contribution: 60 %

- design and definition of the optimisations,
- the proofs of correctness,
- experimental evaluation,
- writing the paper.

Rabinizer: Small Deterministic Automata for LTL(F,G)

Andreas Gaiser[1⋆], Jan Křetínský[1,2⋆⋆], and Javier Esparza[1]

[1] Institut für Informatik, Technische Universität München, Germany
{gaiser, jan.kretinsky, esparza}@model.in.tum.de
[2] Faculty of Informatics, Masaryk University, Brno, Czech Republic

Abstract. We present Rabinizer, a tool for translating formulae of the fragment of linear temporal logic with the operators **F** (eventually) and **G** (globally) into deterministic Rabin automata. Contrary to tools like ltl2dstar, which translate the formula into a Büchi automaton and apply Safra's determinization procedure, Rabinizer uses a direct construction based on the logical structure of the formulae. We describe a number of optimizations of the basic procedure, crucial for the good performance of Rabinizer, and present an experimental comparison.

1 Introduction

The automata-theoretic approach to model checking is one of the most important successes of theoretical computer science in the last decades. It has led to many tools of industrial strength, like Holzmann's SPIN. In its linear-time version, the approach translates the negation of a specification, formalized as a formula of Linear Time Temporal logic (LTL), into a non-deterministic ω-automaton accepting the possible behaviours of the system that violate the specification. Then the product of the automaton with the state space of the system is constructed, and the resulting ω-automaton is checked for emptiness. Since the state space can be very large (medium-size systems can easily have tens of millions of states) and the size of a product of automata is equal to the product of their sizes, it is crucial to transform the formula into a small ω-automaton: saving one state in the ω-automaton may amount to saving tens of millions of states in the product. For this reason, cutting down the number of states has been studied in large depth, and very efficient tools like `LTL2BA` [4] have been developed.

In recent years the theory of the automata-theoretic approach has been successfully extended to probabilistic systems and to synthesis problems. However, these applications pose a new challenge: they require to translate formulas into *deterministic* ω-automata (loosely speaking, the applications require a game-theoretical setting with $1\frac{1}{2}$ or 2 players, whose arenas are only closed under product with deterministic automata) [3]. For this, deterministic Rabin, Streett, or parity automata can be used. The standard approach is to first transform LTL formulae into non-deterministic Büchi automata and then translate these into deterministic Rabin automata by means of Safra's construction [10]. (The determinization procedure of Muller-Schupp is known to produce larger automata [1].)

⋆ The author is supported by the DFG Graduiertenkolleg 1480 (PUMA).
⋆⋆ The author is supported by the Czech Science Foundation, grant No. P202/12/G061.

In particular, this is the procedure followed by `ltl2dstar` [5], the tool used in PRISM [8], the leading probabilistic model checker. However, the approach has two disadvantages: first, since Safra's construction is not tailored for Büchi automata derived from LTL formulae, it often produces (much) larger automata than necessary; second, there are no efficient ways to minimize Rabin automata.

We have recently presented a procedure to directly transform LTL formulae into deterministic automata [7]. The procedure, currently applicable to the (**F**,**G**)-fragment of the logic, heavily exploits formula structure to yield much smaller automata for important formulae, in particular for formulae describing fairness. For instance, a conjunction of three fairness constraints requiring more than one million states with `ltl2dstar` only required 1560 states in [7].

While the experiments of [7] are promising, they were conducted using a primitive implementation. In this paper we report on subsequent work that has transformed the prototype of [7] into `Rabinizer`, a tool incorporating several non-trivial optimizations, and mature enough to be offered to the community. For example, for the formula above, `Rabinizer` returns an automaton with only 462 states.

2 Rabinizer and Optimizations

We assume the reader is familiar with LTL and ω-automata. The idea of the construction of [7] is the following. The states of the Rabin automaton consist of two components. The first component is the LTL formula that, loosely speaking, remains to be satisfied. For example, if a state has $\varphi = \mathbf{F}a \wedge \mathbf{G}b$ as first component, then reading the label $\{a, b\}$ leads to another state with $\mathbf{G}b$ as first component. The second component remembers the last label read (one-step history). To see why this is necessary, observe that for $\varphi = \mathbf{GF}a$ the first component of all states is the same. The second component allows one to check whether a is read infinitely often. Finally, while the Rabin acceptance condition is a disjunction of Rabin pairs, the construction yields a disjunction of *conjunctions* of Rabin pairs, cf. e.g. $\varphi = \mathbf{GF}a \wedge \mathbf{GF}b$, and so in a final step the "generalized Rabin" automaton is expanded into a Rabin automaton.

The only optimizations of implementation used for the experiments of [7] are the following. Firstly, only the reachable state space is constructed. Secondly, the one-step history only records letters appearing in the first component of each state. Thirdly, a simple subsumption of generalized Rabin conditions is considered and the stronger (redundant) conditions are removed. Nevertheless, no algorithm to do this has been presented and manual computation had to be done to obtain the optimized results.

`Rabinizer` is a mature implementation of the procedure of [7] with several additional non-trivial optimizations. `Rabinizer` is written in Java, and uses BDDs to construct the state space of the automata and generate Rabin pairs. While for "easy" formulas `ltl2dstar` would often generate slightly smaller automata than the implementation of [7], `Rabinizer` only generates a larger automaton in 1 out of 27 benchmarks. Moreover, for "difficult" formulas, `Rabinizer` considerably outperforms the previous implementation. We list the most important optimizations performed.

- The evolution of the first component containing the formula to be satisfied has been altered. In the original approach, in order to obtain the acceptance condition easily, not all known information has been reflected immediately in the state space thus resulting in redundant "intermediate" states.
- The generalized Rabin condition is now subject to several optimizations. Firstly, conjunctions of "compatible" Rabin pairs are merged into single pairs thus reducing the blowup from generalized Rabin to Rabin automaton. Secondly, some subformulae, such as outer **F** subformulae, are no more considered in the acceptance condition generation.
- The one-step history now does not contain full information about the letters, but only equivalence classes of letters. The quotienting is done in the coarsest way to still reflect the acceptance condition. A simple example is a formula $\varphi = \mathbf{GF}(a \vee b)$ where we only distinguish between reading any of $\{\{a\}, \{b\}, \{a, b\}\}$ and reading $\emptyset$.
- The blow-up of the generalized Rabin automaton into a Rabin automaton has been improved. Namely, the copies of the original automaton are now quotiented one by one according to the criterion above, but only the conjuncts corresponding to a particular copy are taken into account. Thus we obtain smaller (and different) copies.
  Further, linking of the copies is now made more efficient. Namely, the final states in all but one copy have been removed completely.
- No special state is dedicated to be initial without any other use. Although this results only in a decrease by one, it plays a role in tiny automata.

For further details, correctness proofs, and a detailed input/output description, see `Rabinizer`'s web page `http://www.model.in.tum.de/tools/rabinizer/`.

3 Experimental Results

The following table shows the results on formulae from BEEM (BEnchmarks for Explicit Model checkers)[9] and formulae from [11] on which `ltl2dstar` was originally tested [6]. In both cases, we only take formulae of the (**F**,**G**)-fragment. In the first case this is 11 out of 21, in the second 12 out of 28. There is a slight overlap between the two sets. Further, we add conjunctions of strong fairness conditions and a few other formulae.

For each formula φ, we give the size of the Rabin automaton generated by `ltl2dstar` (using the recommended configuration with `LTL2BA`), the prototype of [7], and `Rabinizer`. For reader's convenience, we also include the size of *nondeterministic* Büchi automata generated by `LTL2BA` [4] and its recent improvement `LTL3BA` [2] whenever they differ. The last two columns state the number of Rabin pairs for automata generated by `ltl2dstar` and `Rabinizer`, respectively.

In all the cases but one, `Rabinizer` generates automata of the same size as `ltl2dstar` or often considerably smaller. Further, while in some cases `Rabinizer` generates one additional pair, it generates less pairs when the number of pairs is high. Runtimes have not been included, as `Rabinizer` transforms all formulae within a second except for the conjunction of three fairness constraints. This one took 13 seconds on an Intel i7 with 8 GB RAM, whereas `ltl2dstar` crashes here and needs more than one day on a machine with 64 GB of RAM.

Formula	ltl2dstar	[7]	Rabinizer	LTL2(3)BA	$*$-pairs	R-pairs
$\mathbf{G}(a \vee \mathbf{F}b)$	4	5	4	2	**1**	2
$\mathbf{FG}a \vee \mathbf{FG}b \vee \mathbf{GF}c$	8	9	8	6	3	3
$\mathbf{F}(a \vee b)$	2	4	2	2	1	1
$\mathbf{GF}(a \vee b)$	2	3	2	2	1	1
$\mathbf{G}(a \vee \mathbf{F}a)$	4	3	**2**	2	**1**	2
$\mathbf{G}(a \vee b \vee c)$	3	4	**2**	1	1	1
$\mathbf{G}(a \vee \mathbf{F}(b \vee c))$	4	5	4	2	**1**	2
$\mathbf{F}a \vee \mathbf{G}b$	4	7	**3**	4	2	2
$\mathbf{G}(a \vee \mathbf{F}(b \wedge c))$	4	5	4	5 (2)	**1**	2
$\mathbf{FG}a \vee \mathbf{GF}b$	4	5	4	5	2	2
$\mathbf{GF}(a \vee b) \wedge \mathbf{GF}(b \vee c)$	7	10	**3**	3	2	**1**
$(\mathbf{FF}a \wedge \mathbf{G}\neg a) \vee (\mathbf{GG}\neg a \wedge \mathbf{F}a)^3$	1	4	1	1	0	0
$\mathbf{GF}a \wedge \mathbf{FG}b$	3	5	3	3	1	1
$(\mathbf{GF}a \wedge \mathbf{FG}b) \vee (\mathbf{FG}\neg a \wedge \mathbf{GF}\neg b)$	5	5	**4**	7	2	2
$\mathbf{FG}a \wedge \mathbf{GF}a$	2	3	2	2 (3)	1	1
$\mathbf{G}(\mathbf{F}a \wedge \mathbf{F}b)$	5	10	**3**	3	1	1
$\mathbf{F}a \wedge \mathbf{F}\neg a$	4	8	4	4	1	1
$(\mathbf{G}(b \vee \mathbf{GF}a) \wedge \mathbf{G}(c \vee \mathbf{GF}\neg a)) \vee \mathbf{G}b \vee \mathbf{G}c$	**13**	36	18	11	**3**	4
$(\mathbf{G}(b \vee \mathbf{FG}a) \wedge \mathbf{G}(c \vee \mathbf{FG}\neg a)) \vee \mathbf{G}b \vee \mathbf{G}c$	14	18	**6**	12 (8)	4	**3**
$(\mathbf{F}(b \wedge \mathbf{FG}a) \vee \mathbf{F}(c \wedge \mathbf{FG}\neg a)) \wedge \mathbf{F}b \wedge \mathbf{F}c$	7	18	**5**	15 (10)	**1**	2
$(\mathbf{F}(b \wedge \mathbf{GF}a) \vee \mathbf{F}(c \wedge \mathbf{GF}\neg a)) \wedge \mathbf{F}b \wedge \mathbf{F}c$	7	18	**5**	13 (10)	2	2
$(\mathbf{GF}a \rightarrow \mathbf{GF}b)$	4	5	4	5	2	2
$(\mathbf{GF}a \rightarrow \mathbf{GF}b) \wedge (\mathbf{GF}c \rightarrow \mathbf{GF}d)$	11324	34	**18**	14	8	**4**
$\bigwedge_{i=1}^{3}(\mathbf{GF}a_i \rightarrow \mathbf{GF}b_i)$	1 304 707	1 560	**462**	40	10	**8**
$\mathbf{GF}(\mathbf{F}a \vee \mathbf{GF}b \vee \mathbf{FG}(a \vee b))$	14	5	**4**	25 (6)	4	**3**
$\mathbf{FG}(\mathbf{F}a \vee \mathbf{GF}b \vee \mathbf{FG}(a \vee b))$	145	5	**4**	24 (6)	9	**3**
$\mathbf{FG}(\mathbf{F}a \vee \mathbf{GF}b \vee \mathbf{FG}(a \vee b) \vee \mathbf{FG}b)$	181	5	**4**	24 (6)	9	**3**

References

1. C. S. Althoff, W. Thomas, and N. Wallmeier. Observations on determinization of Büchi automata. *Theor. Comput. Sci.*, 363(2):224–233, 2006.
2. T. Babiak, M. Křetínský, V. Řehák, and J. Strejček. LTL to Büchi automata translation: Fast and more deterministic. In *TACAS*, pages 95–109, 2012.
3. C. Baier and J.-P. Katoen. *Principles of model checking*. MIT Press, 2008.
4. P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *CAV*, volume 2102 of *LNCS*, pages 53–65, 2001. Tool accessible at http://www.lsv.ens-cachan.fr/~gastin/ltl2ba/.
5. J. Klein. ltl2dstar - LTL to deterministic Streett and Rabin automata. http://www.ltl2dstar.de/.
6. J. Klein and C. Baier. Experiments with deterministic ω-automata for formulas of linear temporal logic. *Theor. Comput. Sci.*, 363(2):182–195, 2006.
7. J. Křetínský and J. Esparza. Deterministic automata for the (F,G)-fragment of LTL. In *CAV*, 2012. To appear.
8. M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591, 2011.
9. R. Pelánek. BEEM: Benchmarks for explicit model checkers. In *Proc. of SPIN Workshop*, volume 4595 of *LNCS*, pages 263–267, 2007.
10. S. Safra. On the complexity of ω-automata. In *FOCS*, pages 319–327, 1988.
11. F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *CAV*, volume 1855 of *LNCS*, pages 248–263, 2000.

Paper C:

Automata with Generalized Rabin Pairs for Probabilistic Model Checking and LTL Synthesis

Krishnendu Chatterjee, Andreas Gaiser, and Jan Křetínský

This paper has been published in Helmut Veith and Natasha Sharygina (eds.): Proceedings of Computer Aided Verification - 25th International Conference, CAV 2012, St. Petersburg, Russia, July 13-19, 2013. Lecture Notes in Computer Science, vol. 8044. Springer, 2013. [CGK13]

Summary

Methods for probabilistic LTL model checking and LTL synthesis mostly require to construct a deterministic ω-automaton from a given LTL formula. The automata used by the state-of-the-art tools such as PRISM are Rabin automata. Theoretical algorithms sometimes also use Streett automata, too. Here we present extensions of these algorithms to the setting of generalised Rabin automata of [KE12] (Paper A). Firstly, we deal with the analysis of the product of Markov decision processes and the respective ω-automata, which is needed for probabilistic LTL model checking. Secondly, we solve LTL games with the generalised winning condition, which are needed for LTL synthesis. Our theoretical results predict speed ups in orders of magnitude even for small, but more complex formulae, such as e.g. fairness constraints. These predictions have been confirmed by our implementations of the algorithms.

Author's contribution: 65 %

- the extension of the algorithms,
- the proofs of correctness,

- analysis of the speed ups,
- parts of experimental evaluation,
- writing most of the paper (from Section 2 onwards with some exceptions in Subsection 3.1).

Automata with Generalized Rabin Pairs for Probabilistic Model Checking and LTL Synthesis

Krishnendu Chatterjee[1⋆], Andreas Gaiser[2⋆⋆], and Jan Křetínský[2,3⋆⋆⋆]

[1] IST Austria
[2] Fakultät für Informatik, Technische Universität München, Germany
[3] Faculty of Informatics, Masaryk University, Brno, Czech Republic

Abstract. The model-checking problem for probabilistic systems crucially relies on the translation of LTL to deterministic Rabin automata (DRW). Our recent Safraless translation [KE12,GKE12] for the LTL(**F**,**G**) fragment produces smaller automata as compared to the traditional approach. In this work, instead of DRW we consider deterministic automata with acceptance condition given as disjunction of generalized Rabin pairs (DGRW). The Safraless translation of LTL(**F**,**G**) formulas to DGRW results in smaller automata as compared to DRW. We present algorithms for probabilistic model-checking as well as game solving for DGRW conditions. Our new algorithms lead to improvement both in terms of theoretical bounds as well as practical evaluation. We compare PRISM with and without our new translation, and show that the new translation leads to significant improvements.

1 Introduction

Logic for ω-regular properties. The class of ω-regular languages generalizes regular languages to infinite strings and provides a robust specification language to express all properties used in verification and synthesis. The most convenient way to describe specifications is through logic, as logics provide a concise and intuitive formalism to express properties with very precise semantics. The linear-time temporal logic (LTL) [Pnu77] is the de-facto logic to express linear time ω-regular properties in verification and synthesis.

Deterministic ω-automata. For model-checking purposes, LTL formulas can be converted to nondeterministic Büchi automata (NBW) [VW86], and then the problem reduces to checking emptiness of the intersection of two NBWs (representing the system and the negation of the specification, respectively). However, for two very important problems deterministic automata are used, namely, (1) the synthesis problem [Chu62,PR89]; and (2) the model-checking problem

⋆ The author is supported by Austrian Science Fund (FWF) Grant No P 23499-N23, FWF NFN Grant No S11407-N23 (RiSE), ERC Start grant (279307: Graph Games), and Microsoft faculty fellows award.
⋆⋆ The author is supported by the DFG Graduiertenkolleg 1480 (PUMA).
⋆⋆⋆ The author is supported by the Czech Science Foundation, project No. P202/10/1469

for probabilistic systems or Markov decision processes (MDPs) [BK08] which has a wide range of applications from randomized communication, to security protocols, to biological systems. The standard approach is to translate LTL to NBW [VW86], and then convert the NBW to a deterministic automata with Rabin acceptance condition (DRW) using Safra's determinization procedure [Saf88] (or using a recent improvement of Piterman [Pit06]).

Avoiding Safra's construction. The key bottleneck of the standard approach in practice is Safra's determinization procedure which is difficult to implement due to the complicated state space and data structures associated with the construction [Kup12]. As a consequence several alternative approaches have been proposed, and the most prominent ones are as follows. The first approach is the Safraless approach. One can reduce the synthesis problem to emptiness of nondeterministic Büchi tree automata [KV05]; it has been implemented with considerable success in [JB06]. For probabilistic model checking other constructions can be also used, however, all of them are exponential [Var85,CY95]. The second approach is to use heuristic to improve Safra's determinization procedure [KB06,KB07] which has led to the tool ltl2dstar [Kle]. The third approach is to consider fragments of LTL. In [AT04] several simple fragments of LTL were proposed that allow much simpler (single exponential as compared to the general double exponential) translations to deterministic automata. The generalized reactivity(1) fragment of LTL (called GR(1)) was introduced in [PPS06] and a cubic time symbolic representation of an equivalent automaton was presented. The approach has been implemented in the ANZU tool [JGWB07]. Recently, the $(\mathbf{F}, \mathbf{G})$-fragment of LTL, that uses boolean operations and only $\mathbf{F}$ (eventually or in future) and $\mathbf{G}$ (always or globally) as temporal operators, was considered and a simple and direct translation to deterministic Rabin automata (DRW) was presented [KE12]. Not only it covers all fragments of [AT04], but it can also express all complex fairness constraints, which are widely used in verification.

Probabilistic model-checking. Despite several approaches to avoid Safra's determinization, for probabilistic model-checking the deterministic automata are still necessary. Since probabilistic model-checkers handle linear arithmetic, they do not benefit from the symbolic methods of [PPS06,MS08] or from the tree automata approach. The approach for probabilistic model-checking has been to explicitly construct a DRW from the LTL formula. The most prominent probabilistic model-checker PRISM [KNP11] implements the ltl2dstar approach.

Our results. In this work, we focus on the $(\mathbf{F}, \mathbf{G})$-fragment of LTL. Instead of the traditional approach of translation to DRW we propose a translation to deterministic automata with *generalized Rabin pairs*. We present probabilistic model-checking as well as symbolic game solving algorithms for the new class of conditions which lead to both theoretical as well as significant practical improvements. The details of our contributions are as follows.

1. A Rabin pair consists of the conjunction of a Büchi (always eventually) and a coBüchi (eventually always) condition, and a Rabin condition is a disjunction of Rabin pairs. A generalized Rabin pair is the conjunction of conjunctions

of Büchi conditions and conjunctions of coBüchi conditions. However, as conjunctions of coBüchi conditions is again a coBüchi condition, a generalized Rabin pair is the conjunction of a coBüchi condition and conjunction of Büchi conditions.† We consider deterministic automata where the acceptance condition is a disjunction of generalized Rabin pairs (and call them DGRW). The $(\mathbf{F}, \mathbf{G})$-fragment of LTL admits a direct and algorithmically simple translation to DGRW [KE12] and we consider DGRW for probabilistic model-checking and synthesis. The direct translation of LTL($\mathbf{F}$,$\mathbf{G}$) could be done to a compact deterministic automaton with a Muller condition, however, the explicit representation of the Muller condition is typically huge and not algorithmically efficient, and thus reduction to deterministic Rabin automata was performed (with a blow-up) since Rabin conditions admit efficient algorithmic analysis. We show that DGRW allow both for a very compact translation of the $(\mathbf{F}, \mathbf{G})$-fragment of LTL as well as efficient algorithmic analysis. The direct translation of LTL($\mathbf{F}$,$\mathbf{G}$) to DGRW has the same number of states as for a general Muller condition. For many formulae expressing e.g. fairness-like conditions the translation to DGRW is significantly more compact than the previous ltl2dstar approach. For example, for a conjunction of three strong fairness constraints, ltl2dstar produces a DRW with more than a million states, translation to DRW via DGRW requires 469 states, and the corresponding DGRW has only 64 states.

2. One approach for probabilistic model-checking and synthesis for DGRW would be to first convert them to DRW, and then use the standard algorithms. Instead we present direct algorithms for DGRW that avoids the translation to DRW both for probabilistic model-checking and game solving. The direct algorithms lead to both theoretical and practical improvements. For example, consider the disjunctions of k generalized Rabin pairs such that in each pair there is a conjunction of a coBüchi condition and conjunctions of j Büchi conditions. Our direct algorithms for probabilistic model-checking as well as game solving is more efficient by a multiplicative factor of j^k and j^{k^2+k} as compared to the approach of translation to DRW for probabilistic model checking and game solving, respectively. Moreover, we also present symbolic algorithms for game solving for DGRW conditions.
3. We have implemented our approach for probabilistic model checking in PRISM, and the experimental results show that as compared to the existing implementation of PRISM with ltl2dstar our approach results in improvement of order of magnitude. Moreover, the results for games confirm that the speed up is even greater than for probabilistic model checking.

† Note that our condition (disjunction of generalized Rabin pairs) is very different from both generalized Rabin conditions (conjunction of Rabin conditions) and the generalized Rabin(1) condition of [Ehl11], which considers a set of assumptions and guarantees where each assumption and guarantee consists of *one* Rabin pair. Syntactically, disjunction of generalized Rabin pairs condition is $\bigvee_i(\mathbf{FG}a_i \wedge \bigwedge_j \mathbf{GF}b_{ij})$, whereas generalized Rabin condition is $\bigwedge_j(\bigvee_i(\mathbf{FG}a_{ij} \wedge \mathbf{GF}b_{ij}))$, and generalized Rabin(1) condition is $(\bigwedge_i(\mathbf{FG}a_i \wedge \mathbf{GF}b_i) \Rightarrow \bigwedge_j(\mathbf{FG}a_j \wedge \mathbf{GF}b_j))$.

2 Preliminaries

In this section, we recall the notion of linear temporal logic (LTL) and illustrate the recent translation of its (**F**,**G**)-fragment to DRW [KE12,GKE12] through the intermediate formalism of DGRW. Finally, we define an index that is important for characterizing the savings the new formalism of DGRW brings as shown in the subsequent sections.

2.1 Linear temporal logic

We start by recalling the fragment of linear temporal logic with *future* (**F**) and *globally* (**G**) modalities.

Definition 1 (LTL(F,G) syntax). *The formulae of the* (**F**,**G**)-fragment of linear temporal logic *are given by the following syntax:*

$$\varphi ::= a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi$$

where a ranges over a finite fixed set Ap of atomic propositions.

We use the standard abbreviations $\mathbf{tt} := a \vee \neg a$ and $\mathbf{ff} := a \wedge \neg a$. Note that we use the negation normal form, as negations can be pushed inside to atomic propositions due to the equivalence of $\mathbf{F}\varphi$ and $\neg\mathbf{G}\neg\varphi$.

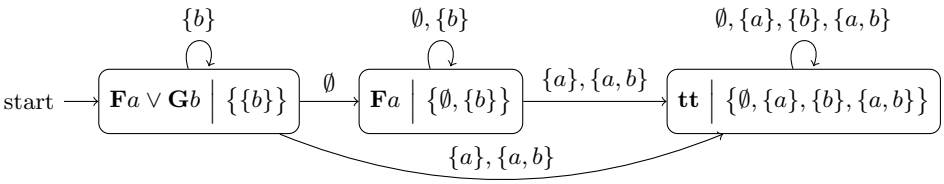
Definition 2 (LTL(F,G) semantics). *Let $w \in (2^{Ap})^\omega$ be a word. The ith letter of w is denoted $w[i]$, i.e. $w = w[0]w[1]\cdots$. Further, we define the ith suffix of w as $w_i = w[i]w[i+1]\cdots$. The semantics of a formula on w is then defined inductively as follows: $w \models a \iff a \in w[0]$; $w \models \neg a \iff a \notin w[0]$; $w \models \varphi \wedge \psi \iff w \models \varphi$ and $w \models \psi$; $w \models \varphi \vee \psi \iff w \models \varphi$ or $w \models \psi$; and*

$$\begin{aligned} w \models \mathbf{F}\varphi &\iff \exists\, k \in \mathbb{N}_0 : w_k \models \varphi \\ w \models \mathbf{G}\varphi &\iff \forall\, k \in \mathbb{N}_0 : w_k \models \varphi \end{aligned}$$

2.2 Translating LTL(F,G) into deterministic ω-automata

Recently, in [KE12,GKE12], a new translation of LTL(**F**,**G**) to deterministic automata has been proposed. This construction avoids Safra's determinization and makes direct use of the structure of the formula. We illustrate the construction in the following examples.

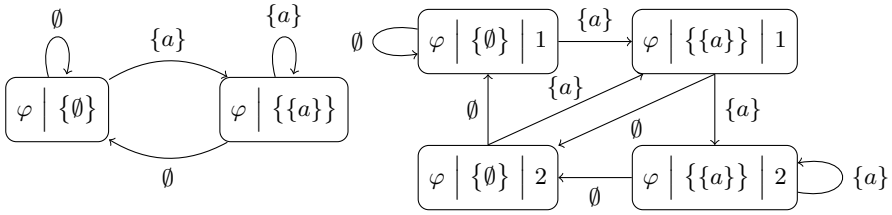
Example 3. Consider a formula $\mathbf{F}a \vee \mathbf{G}b$. The construction results in the following automaton. The state space of the automaton has two components. The first component stores the current formula to be satisfied. Whenever a letter is read, the formula is updated accordingly. For example, when reading a letter with no b, the option to satisfy the formula due to satisfaction of $\mathbf{G}b$ is lost and is thus reflected in changing the current formula to $\mathbf{F}a$ only.



The second component stores the last letter read (actually, an equivalence class thereof). The purpose of this component is explained in the next example. For formulae with no mutual nesting of **F** and **G** this component is redundant.

The formula $\mathbf{F}a \vee \mathbf{G}b$ is satisfied either due to $\mathbf{F}a$ or $\mathbf{G}b$. Therefore, when viewed as a Rabin automaton, there are two Rabin pairs. One forcing infinitely many visits of the third state (a in $\mathbf{F}a$ must be eventually satisfied) and the other prohibiting infinitely many visits of the second and third states (b in $\mathbf{G}b$ must never be violated). The acceptance condition is a disjunction of these pairs.

Example 4. Consider now the formula $\varphi = \mathbf{GF}a \wedge \mathbf{GF}\neg a$. Satisfaction of this formula does not depend on any finite prefix of the word and reading $\{a\}$ or $\emptyset$ does not change the first component of the state. This infinitary behaviour requires the state space to record which letters have been seen infinitely often and the acceptance condition to deal with that. In this case, satisfaction requires visiting the second state infinitely often *and* visiting the first state infinitely often.



However, such a conjunction cannot be written as a Rabin condition. In order to get a Rabin automaton, we would duplicate the state space. In the first copy, we wait for reading $\{a\}$. Once this happens we move to the second copy, where we wait for reading $\emptyset$. Once we succeed we move back to the first copy and start again. This bigger automaton now allows for a Rabin condition. Indeed, it is sufficient to infinitely often visit the "successful" state of the last copy as this forces infinite visits of "successful" states of all copies.

In order to obtain a DRW from an LTL formula, [KE12,GKE12] first constructs an automaton similar to DGRW (like the one on the left) and then the state space is blown-up and a DRW (like the one on the right) is obtained. However, we shall argue that this blow-up is unnecessary for application in probabilistic model checking and in synthesis. This will result in much more efficient algorithms for complex formulae. In order to avoid the blow-up we define and use DGRW, an automaton with more complex acceptance condition, yet as we show algorithmically easy to work with and efficient as opposed to e.g. the general Muller condition.

2.3 Automata with generalized Rabin pairs

In the previous example, the cause of the blow-up was the conjunction of Rabin conditions. In [KE12], a generalized version of Rabin condition is defined that allows for capturing conjunction. It is defined as a positive Boolean combination of Rabin pairs. Whether a set $\mathrm{Inf}(\rho)$ of states visited infinitely often on a run ρ is accepting or not is then defined inductively as follows:

$$\begin{aligned}
\mathrm{Inf}(\rho) &\models \varphi \wedge \psi && \iff \mathrm{Inf}(\rho) \models \varphi \text{ and } \mathrm{Inf}(\rho) \models \psi \\
\mathrm{Inf}(\rho) &\models \varphi \vee \psi && \iff \mathrm{Inf}(\rho) \models \varphi \text{ or } \mathrm{Inf}(\rho) \models \psi \\
\mathrm{Inf}(\rho) &\models (F, I) && \iff F \cap \mathrm{Inf}(\rho) = \emptyset \text{ and } I \cap \mathrm{Inf}(\rho) \neq \emptyset
\end{aligned}$$

Denoting Q as the set of all states, (F, I) is then equivalent to $(F, Q) \wedge (\emptyset, I)$. Further, $(F_1, Q) \wedge (F_2, Q)$ is equivalent to $(F_1 \cup F_2, Q)$. Therefore, one can transform any such condition into a disjunctive normal form and obtain a condition of the following form:

$$\bigvee_{i=1}^{k} \left(\left(F_i, Q\right) \wedge \bigwedge_{j=1}^{\ell_i} \left(\emptyset, I_i^j\right) \right) \qquad (*)$$

Therefore, in this paper we define the following new class of ω-automata:

Definition 5 (DGRW). *An* automaton with generalized Rabin pairs (DGRW) *is a (deterministic) ω-automaton $\mathcal{A} = (Q, q_0, \delta)$ over an alphabet Σ, where Q is a set of states, q_0 is the initial state, $\delta : Q \times \Sigma \to Q$ is a transition function, together with a* generalized Rabin pairs (GRP) acceptance condition $\mathcal{GR} \subseteq 2^{2^Q \times 2^{2^Q}}$. *A run ρ of $\mathcal{A}$ is accepting for $\mathcal{GR} = \left\{ \left(F_i, \{I_i^1, \ldots, I_i^{\ell_i}\}\right) \;\middle|\; i \in \{1, \ldots, k\} \right\}$ if there is $i \in \{1, \ldots, k\}$ such that*

$$\begin{aligned}
&F_i \cap \mathrm{Inf}(\rho) = \emptyset \text{ and} \\
&I_i^j \cap \mathrm{Inf}(\rho) \neq \emptyset \text{ for every } j \in \{1, \ldots, \ell_i\}
\end{aligned}$$

Each $(F_i, \mathcal{I}_i) = \left(F_i, \{I_i^1, \ldots, I_i^{\ell_i}\}\right)$ is called a generalized Rabin pair (GRP)*, and the* GRP condition *is thus a disjunction of generalized Rabin pairs..*

W.l.o.g. we assume $k > 0$ and $\ell_i > 0$ for each $i \in \{1, \ldots, k\}$ (whenever $\ell_i = 0$ we could set $\mathcal{I}_i = \{Q\}$). Although the type of the condition allows for huge instances of the condition, the construction of [KE12] (producing this disjunctive normal form) guarantees efficiency not worse than that of the traditional determinization approach. For a formula of size n, it is guaranteed that $k \leq 2^n$ and $\ell_i \leq n$ for each $i \in \{1, \ldots, k\}$. Further, the size of the state space is at most $2^{\mathcal{O}(2^n)}$. Moreover, consider *"infinitary"* formulae, where each atomic proposition has both **F** and **G** as ancestors in the syntactic tree of the formula. Since the first component of the state space is always the same, the size of the state space is bounded by $2^{|Ap|}$ as the automaton only remembers the last letter read. We will make use of this fact later.

2.4 Degeneralization

As already discussed, one can blow up any automaton with generalized Rabin pairs and obtain a Rabin automaton. We need the following notation. For any $n \in \mathbb{N}$, let $[1..n]$ denote the set $\{1, \ldots, n\}$ equipped with the operation $\oplus$ of cyclic addition, i.e. $m \oplus 1 = m + 1$ for $m < n$ and $n \oplus 1 = 1$.

The DGRW defined above can now be degeneralized as follows. For each $i \in \{1, \ldots, k\}$, multiply the state space by $[1..\ell_i]$ to keep track for which I_i^j we are currently waiting for. Further, adjust the transition function so that we leave the jth copy once we visit I_i^j and immediately go to the next copy. Formally, for $\sigma \in \Sigma$ set $(q, w_1, \ldots, w_k) \xrightarrow{\sigma} (r, w_1', \ldots, w_k')$ if $q \xrightarrow{\sigma} r$ and $w_i' = w_i$ for all i with $q \notin I_i^{w_i}$ and $w_i' = w_i \oplus 1$ otherwise.

The resulting blow-up factor is then the following:

Definition 6 (Degeneralization index). *For a GRP condition* $\mathcal{GR} = \{(F_i, \mathcal{I}_i) \mid i \in [1..k]\}$*, we define the* degeneralization domain $B := \prod_{i=1}^{k} [1..|\mathcal{I}_i|]$ *and the* degeneralization index of $\mathcal{GR}$ *to be* $|B| = \prod_{i=1}^{k} |\mathcal{I}_i|$.

The state space of the resulting Rabin automaton is thus $|B|$-times bigger and the number of pairs stays the same. Indeed, for each $i \in \{1, \ldots, k\}$ we have a Rabin pair

$$\left(F_i \times B, I_i^{\ell_i} \times \{b \in B \mid b(i) = \ell_i\}\right)$$

Example 7. In Example 3 there is one pair and the degeneralization index is 2.

Example 8. For a conjunction of three fairness constraints $\varphi = (\mathbf{FG}a \vee \mathbf{GF}b) \wedge (\mathbf{FG}c \vee \mathbf{GF}d) \wedge (\mathbf{FG}e \vee \mathbf{GF}f)$, the Büchi components $\mathcal{I}_i$'s of the equivalent GRP condition correspond to $\mathbf{tt}, b, d, f, b \wedge d, b \wedge f, d \wedge f, b \wedge d \wedge f$. The degeneralization index is thus $|B| = 1 \cdot 1 \cdot 1 \cdot 1 \cdot 2 \cdot 2 \cdot 2 \cdot 3 = 24$. For four constraints, it is $1 \cdot 1^4 \cdot 2^6 \cdot 3^4 \cdot 4 = 20736$. One can easily see the index grows doubly exponentially.

3 Probabilistic Model Checking

In this section, we show how automata with generalized Rabin pairs can significantly speed up model checking of Markov decision processes (i.e., probabilistic model checking). For example, for the fairness constraints of the type mentioned in Example 8 the speed-up is by a factor that is doubly exponential. Although there are specialized algorithms for checking properties under strong fairness constraints (implemented in PRISM), our approach is general and speeds up for a wide class of constraints. The combinations (conjunctions, disjunctions) of properties not expressible by small Rabin automata (and/or Streett automata) are infeasible for the traditional approach, while we show that automata with generalized Rabin pairs often allow for efficient model checking. First, we present the theoretical model-checking algorithm for the new type of automata and the theoretical bounds for savings. Second, we illustrate the effectiveness of the approach experimentally.

3.1 Model checking using generalized Rabin pairs

We start with the definitions of Markov decision processes (MDPs), and present the model-checking algorithms. For a finite set V, let $\mathrm{Distr}(V)$ denote the set of probability distributions on V.

Definition 9 (MDP and MEC). *A* Markov decision process (MDP) $\mathcal{M} = (V, E, (V_0, V_P), \delta)$ *consists of a finite directed* MDP graph (V, E)*, a partition* (V_0, V_P) *of the* finite *set* V *of vertices into player-0 vertices* (V_0) *and probabilistic vertices* (V_P)*, and a probabilistic transition function* $\delta\colon V_P \to \mathrm{Distr}(V)$ *such that for all vertices* $u \in V_P$ *and* $v \in V$ *we have* $(u, v) \in E$ *iff* $\delta(u)(v) > 0$.

An end-component U *of an MDP is a set of its vertices such that (i) the subgraph induced by* U *is strongly connected and (ii) for each edge* $(u, v) \in E$*, if* $u \in U \cap V_P$*, then* $v \in U$ *(i.e., no probabilistic edge leaves* U*).*

A maximal end-component (MEC) *is an end-component that is maximal w.r.t. to the inclusion ordering.*

If U_1 and U_2 are two end-components and $U_1 \cap U_2 \neq \emptyset$, then $U_1 \cup U_2$ is also an end-component. Therefore, every MDP induces a unique set of its MECs, called *MEC decomposition*.

For precise definition of semantics of MDPs we refer to [Put94]. Note that MDPs are also defined in an equivalent way in literature with a set of actions such that every vertex and choice of action determines the probability distribution over the successor states; the choice of actions corresponds to the choice of edges at player-0 vertices of our definition.

The standard model-checking algorithm for MDPs proceeds in several steps. Given an MDP $\mathcal{M}$ and an LTL formula φ

1. compute a deterministic automaton $\mathcal{A}$ recognizing the language of φ,
2. compute the product $\overline{\mathcal{M}} = \mathcal{M} \times \mathcal{A}$,
3. solve the product MDP $\overline{\mathcal{M}}$.

The algorithm is generic for all types of deterministic ω-automata $\mathcal{A}$. The leading probabilistic model checker PRISM [KNP11] re-implements ltl2dstar [Kle] that transforms φ into a deterministic *Rabin* automaton. This approach employs Safra's determinization and thus despite many optimization often results in an unnecessarily big automaton.

There are two ways to fight the problem. Firstly, one can strive for smaller Rabin automata. Secondly, one can employ other types of ω-automata. As to the former, we have plugged our implementation Rabinizer [GKE12] of the approach [KE12] into PRISM, which already results in considerable improvement. For the latter, Example 4 shows that Muller automata can be smaller than Rabin automata. However, explicit representation of Muller acceptance conditions is typically huge. Hence the third step to solve the product MDP would be too expensive. Therefore, we propose to use automata with generalized Rabin pairs.

On the one hand, DGRW often have small state space after translation. Actually, it is the same as the state space of the intermediate Muller automaton

of [KE12]. Compared to the corresponding naively degeneralized DRW it is $|B|$ times smaller (one can still perform some optimizations in the degeneralization process, see the experimental results).

On the other hand, as we show below the acceptance condition is still algorithmically efficient to handle. We now present the steps to solve the product MDP for a GRP acceptance condition, i.e. a disjunction of generalized Rabin pairs. Consider an MDP with k generalized Rabin pairs $(F_i, \{I_i^1, \ldots, I_i^{\ell_i}\})$, for $i = 1, 2, \ldots, k$. The steps of the computation are as follows:

1. For $i = 1, 2, \ldots, k$;
   (a) Remove the set of states F_i from the MDP.
   (b) Compute the MEC decomposition.
   (c) If a MEC C has a non-empty intersection with each I_i^j, for $j = 1, 2, \ldots, \ell_i$, then include C as a winning MEC.
   (d) let W_i be the union of winning MECs (for the ith pair).
2. Let W be the union of W_i, i.e. $W = \bigcup_{i=1}^{k} W_i$.
3. The solution (or optimal value of the product MDP) is the maximal probability to reach the set W.

Given an MDP with n vertices and m edges, let $\mathsf{MEC}(n, m)$ denote the complexity of computing the MEC decomposition; and $\mathsf{LP}(n, m)$ denotes the complexity to solve linear-programming solution with m constraints over n variables.

Theorem 10. *Given an MDP with n vertices and m edges with k generalized Rabin pairs $(F_i, \{I_i^1, \ldots, I_i^{\ell_i}\})$, for $i = 1, 2, \ldots, k$, the solution can be achieved in time $\mathcal{O}(k \cdot \mathsf{MEC}(n, m) + n \cdot \sum_{i=1}^{k} \ell_i) + \mathcal{O}(\mathsf{LP}(n, m))$.*

Remark 11. The best known complexity to solve MDPs with Rabin conditions of k pairs require time $\mathcal{O}(k \cdot \mathsf{MEC}(n, m)) + \mathcal{O}(\mathsf{LP}(n, m))$ time [dA97]. Thus degeneralization of generalized Rabin pairs to Rabin conditions and solving MDPs would require time $\mathcal{O}(k \cdot \mathsf{MEC}(|B| \cdot n, |B| \cdot m)) + \mathcal{O}(\mathsf{LP}(|B| \cdot n, |B| \cdot m))$ time. The current best known algorithms for maximal end-component decomposition require at least $\mathcal{O}(m \cdot n^{2/3})$ time [CH11], and the simplest algorithms that are typically implemented require $\mathcal{O}(n \cdot m)$ time. Thus our approach is more efficient at least by a factor of $B^{5/3}$ (given the current best known algorithms), and even if both maximal end-component decomposition and linear-programming can be solved in linear time, our approach leads to a speed-up by a factor of $|B|$, i.e. exponential in $\mathcal{O}(k)$ the number of non-trivially generalized Rabin pairs. In general if $\beta \geq 1$ is the sum of the exponents required to solve the MEC decomposition (resp. linear-programming), then our approach is better by a factor of $|B|^{\beta}$.

Example 12. A Rabin automaton for n constraints of Example 8 is of doubly exponential size, which is also the factor by which the product and thus the running time grows. However, as the formula is "infinitary" (see end of Section 2.3), the state space of the generalized automaton is 2^{Ap} and the product is of the very same size as the original system since the automaton only monitors the current labelling of the state.

3.2 Experimental results

In this section, we compare the performance of

L the original PRISM with its implementation of ltl2dstar producing Rabin automata,
R PRISM with Rabinizer [GKE12] (our implementation of [KE12]) producing DRW via *optimized* degeneralization of DGRW, and
GR PRISM with Rabinizer producing DGRW and with the modified MEC checking step.

We have performed a case study on the Pnueli-Zuck randomized mutual exclusion protocol [PZ86] implemented as a PRISM benchmark. We consider the protocol with 3, 4, and 5 participants. The sizes of the respective models are $s_3 = 2\,368$, $s_4 = 27\,600$, and $s_5 = 308\,800$ states. We have checked these models against several formulae illustrating the effect of the degeneralization index on the speed up of our method; see Table 1.

In the first column, there are the formulae in the form of a PRISM query. We ask for a maximal/minimal value over all schedulers. Therefore, in the P_{max} case, we create an automaton for the formula, whereas in the case of P_{min} we create an automaton for its negation. The second column then states the number i of participants, thus inducing the respective size s_i of the model.

The next three columns depict the size of the product of the system and the automaton, for each of the **L**, **R**, **GR** variants. The size is given as the ratio of the actual size and the respective s_i. The number then describes also the "effective" size of the automaton when taking the product. The next three columns display the total running times for model checking in each variant.

The last three columns illustrate the efficiency of our approach. The first column $t_{\mathbf{R}}/t_{\mathbf{GR}}$ states the time speed-up of the DGRW approach when compared to the corresponding degeneralization. The second column states the degeneralization index $|B|$. The last column $t_{\mathbf{L}}/t_{\mathbf{GR}}$ then displays the overall speed-up of our approach to the original PRISM.

In the formulae, an atomic proposition $p_i = j$ denotes that the ith participant is in its state j. The processes start in state 0. In state 1 they want to enter the critical section. State 10 stands for being in the critical section. After leaving the critical section, the process re-enters state 0 again.

Formulae 1 to 3 illustrate the effect of $|B|$ on the ratio of sizes of the product in the **R** and **GR** cases, see $\frac{s_{\mathbf{R}}}{s_i}$, and ratio of the required times. The theoretical prediction is that $s_{\mathbf{R}}/s_{\mathbf{GR}} = |B|$. Nevertheless, due to optimizations done in the degeneralization process, the first is often slightly smaller than the second one, see columns $\frac{s_{\mathbf{R}}}{s_i}$ and B. (Note that $s_{\mathbf{GR}}/s_i$ is 1 for "infinitary" formulae.) For the same reason, $\frac{t_{\mathbf{R}}}{t_{\mathbf{GR}}}$ is often smaller than $|B|$. However, with the growing size of the systems it gets bigger hence the saving factor is larger for larger systems, as discussed in the previous section.

Formulae 4 to 7 illustrate the doubly exponential growth of $|B|$ and its impact on systems of different sizes. The DGRW approach (**GR** method) is often the only way to create the product at all.

Table 1. Experimental comparison of **L**, **R**, and **GR** methods. All measurements performed on Intel i7 with 8 GB RAM. The sign "−" denotes either crash, out-of-memory, time-out after 30 minutes, or a ratio where one operand is −.

Formula	#	$\frac{s_{\mathbf{L}}}{s_i}$	$\frac{s_{\mathbf{R}}}{s_i}$	$\frac{s_{\mathbf{GR}}}{s_i}$	$t_{\mathbf{L}}$	$t_{\mathbf{R}}$	$t_{\mathbf{GR}}$	$\frac{t_{\mathbf{R}}}{t_{\mathbf{GR}}}$	$\lvert B\rvert$	$\frac{t_{\mathbf{L}}}{t_{\mathbf{GR}}}$
P_{max} =?[$\mathbf{GF}p_1{=}10$	3	4.1	2.6	1	1.2	0.4	0.2	2.2	3	6.8
$\wedge\ \mathbf{GF}p_2{=}10$	4	4.3	2.7	1	17.4	1.8	0.3	6.4	3	60.8
$\wedge\ \mathbf{GF}p_3{=}10]$	5	4.4	2.7	1	257.5	15.2	0.6	26.7	3	447.9
P_{max} =?[$\mathbf{GF}p_1{=}10 \wedge \mathbf{GF}p_2{=}10$	4	6	3.5	1	27.3	2.5	0.9	2.8	4	32.1
$\wedge\ \mathbf{GF}p_3{=}10 \wedge \mathbf{GF}p_4{=}10]$	5	6.2	3.6	1	408.5	17.8	0.9	20.4	4	471.2
P_{min} =?[$\mathbf{GF}p_1{=}10 \wedge \mathbf{GF}p_2{=}10$	4	−	1	1	−	36.5	36.3	1	1	−
$\wedge\ \mathbf{GF}p_3{=}10 \wedge \mathbf{GF}p_4{=}10]$	5	−	1	1	−	610.6	607.2	1	1	−
P_{max} =?[($\mathbf{GF}p_1{=}0 \vee \mathbf{FG}p_2{\neq}0$)	3	79.7	1.9	1	225.5	4.1	2.2	1.8	2	101.8
$\wedge(\mathbf{GF}p_2{=}0 \vee \mathbf{FG}p_3{\neq}0)]$	4	−	1.9	1	−	61.7	29.2	2.1	2	−
	5	−	1.9	1	−	1007	479	2.1	2	−
P_{max} =?[($\mathbf{GF}p_1{=}0 \vee \mathbf{FG}p_1{\neq}0$)	3	23.3	1.9	1	66.4	3.92	2.2	1.8	2	30.7
$\wedge(\mathbf{GF}p_2{=}0 \vee \mathbf{FG}p_2{\neq}0)]$	4	23.3	1.9	1	551.5	61	28.2	2.2	2	19.6
	5	−	1.9	1	−	1002.7	463	2.2	2	−
P_{max} =?[($\mathbf{GF}p_1{=}0 \vee \mathbf{FG}p_2{\neq}0$)	3	−	16.3	1	−	122.1	7.1	17.2	24	−
$\wedge(\mathbf{GF}p_2{=}0 \vee \mathbf{FG}p_3{\neq}0)$	4	−	−	1	−	−	75.6	−	24	−
$\wedge(\mathbf{GF}p_3{=}0 \vee \mathbf{FG}p_1{\neq}0)]$	5	−	−	1	−	−	1219.5	−	24	−
P_{max} =?[($\mathbf{GF}p_1{=}0 \vee \mathbf{FG}p_1{\neq}0$)	3	−	12	1	−	76.3	7.2	12	24	−
$\wedge(\mathbf{GF}p_2{=}0 \vee \mathbf{FG}p_2{\neq}0)$	4	−	12.1	1	−	1335.6	78.9	19.6	24	−
$\wedge(\mathbf{GF}p_3{=}0 \vee \mathbf{FG}p_3{\neq}0)]$	5	−	−	1	−	−	1267.6	−	24	−
P_{min} =?[($\mathbf{GF}p_1{\neq}10 \vee \mathbf{GF}p_1{=}0 \vee \mathbf{FG}p_1{=}1$)	3	2.1	1	1	1.2	0.9	0.8	1	1	1.5
$\wedge\mathbf{GF}p_1{\neq}0 \wedge \mathbf{GF}p_1{=}1]$	4	2.1	1	1	11.8	8.7	8.8	1	1	1.3
	5	2.1	1	1	186.3	147.5	146.2	1	1	1.3
P_{max} =?[($\mathbf{G}p_1{\neq}10 \vee \mathbf{G}p_2{\neq}10 \vee \mathbf{G}p_3{\neq}10$)	3	−	32	5.9	−	405	80.1	5.1	8	−
$\wedge\ (\mathbf{FG}p_1{\neq}1 \vee \mathbf{GF}p_2 = 1 \vee \mathbf{GF}p_3 = 1)$	4	−	−	6.4	−	−	703.5	−	8	−
$\wedge\ (\mathbf{FG}p_2{\neq}1 \vee \mathbf{GF}p_1 = 1 \vee \mathbf{GF}p_3 = 1)$	5	−	−	−	−	−	−	−	8	−
P_{min} =?[($\mathbf{FG}p_1{\neq}0 \vee \mathbf{FG}p_2{\neq}0 \vee \mathbf{GF}p_3{=}0$)	3	55.9	4.7	1	289.7	12.6	3.4	3.7	12	84.3
$\vee\ (\mathbf{FG}p_1{\neq}10 \wedge \mathbf{GF}p_2 = 10 \wedge \mathbf{GF}p_3 = 10)$	4	−	4.6	1	−	194.5	33.2	5.9	12	−
	5	−	−	1	−	−	543	−	12	−

Formula 8 is a Streett condition showing the approach still performs competitively. Formulae 9 and 10 combine Rabin and Streett condition requiring both big Rabin automata and big Streett automata. Even in this case, the method scales well. Further, Formula 9 contains non-infinitary behaviour, e.g. $\mathbf{G}p_1{\neq}10$. Therefore, the DGRW is of size greater than 1, and thus also the product is bigger as can be seen in the $s_{\mathbf{GR}}/s_i$ column.

4 Synthesis

In this section, we show how generalized Rabin pairs can be used to speed up the computation of a winning strategy in an LTL(**F**,**G**) game and thus to speed up LTL(**F**,**G**) synthesis. A game is defined like an MDP, but with the stochastic vertices replaced by vertices of an adversarial player.

Definition 13. *A* game $\mathcal{M} = (V, E, (V_0, V_1))$ *consists of a finite directed* game graph (V, E) *and a partition* (V_0, V_1) *of the* finite *set* V *of vertices into player-0 vertices* (V_0) *and player-1 vertices* (V_1).

An *LTL game* is a game together with an LTL formula with vertices as atomic propositions. Similarly, a *Rabin game* and a *game with GRP condition (GRP game)* is a game with a set of Rabin pairs, or a set of generalized Rabin pairs, respectively.

A *strategy* is a function $V^* \to E$ assigning to each *history* an outgoing edge of its last vertex. A play conforming to the strategy f of Player 0 is any infinite sequence $v_0 v_1 \cdots$ satisfying $v_{i+1} = f(v_0 \cdots v_i)$ whenever $v_i \in V_0$, and just $(v_i, v_{i+1}) \in E$ otherwise. Player 0 has a *winning strategy*, if there is a strategy f such that all plays conforming to f of Player 0 satisfy the LTL formula, Rabin condition or GRP condition, depending on the type of the game. For further details, we refer to e.g. [PP06].

One way to solve an LTL game is to make a product of the game arena with the DRW corresponding to the LTL formula, yielding a Rabin game. The current fastest solution of Rabin games works in time $\mathcal{O}(mn^{k+1}kk!)$ [PP06], where $n = |V|, m = |E|$ and k is the number of pairs. Since n is doubly exponential and k singly exponential in the size of the formula, this leads to a doubly exponential algorithm. And indeed, the problem of LTL synthesis is 2-EXPTIME-complete [PR89].

Similarly as for model checking of probabilistic systems, we investigate what happens (1) if we replace the translation to Rabin automata by our new translation and (2) if we employ DGRW instead. The latter leads to the problem of GRP games. In order to solve them, we extend the methods to solve Rabin and Streett games of [PP06].

We show that solving a GRP game is faster than first degeneralizing them and then solving the resulting Rabin game. The induced speed-up factor is $|B|^k$. In the following two subsections we show how to solve GRP games and analyze the complexity. The subsequent section reports on experimental results.

4.1 Generalized Rabin ranking

We shall compute a ranking of each vertex, which intuitively states how far from winning we are. The existence of winning strategy is then equivalent to the existence of a ranking where Player 0 can always choose a successor of the current vertex with smaller ranking, i.e. closer to fulfilling the goal.

Let $(V, E, (V_0, V_1))$ be a game, $\{(F_1, \mathcal{I}_1), \ldots, (F_k, \mathcal{I}_k)\}$ a GRP condition with the corresponding degeneralization domain B. Further, let $n := |V|$ and denote the set of permutations over a set S by $S!$.

Definition 14. *A* ranking *is a function $r : V \times B \to R$ where R is the* ranking domain $\{1, \ldots, k\}! \times \{0, \ldots, n\}^{k+1} \cup \{\infty\}$.

The ranking $r(v, wf)$ gives information important in the situation when we are in vertex v and are waiting for a visit of $\mathcal{I}_i^{wf(i)}$ for each i given by $wf \in B$. As time passes the ranking should decrease. To capture this, we define the following functions.

Definition 15. *For a ranking r and given $v \in V$ and $wf \in B$, we define* $\mathrm{next}_v : B \to B$

$$\mathrm{next}_v(wf)(i) = \begin{cases} wf(i) & \text{if } v \notin \mathcal{I}_i^{wf(i)} \\ wf(i) \oplus 1 & \text{if } v \in \mathcal{I}_i^{wf(i)} \end{cases}$$

and $\mathrm{next} : V \times B \to R$

$$\mathrm{next}(v, wf) = \begin{cases} \min_{(v,w)\in E} r(w, \mathrm{next}_v(wf)) & \text{if } v \in V_0 \\ \max_{(v,w)\in E} r(w, \mathrm{next}_v(wf)) & \text{if } v \in V_1 \end{cases}$$

where the order on $(\pi_1 \cdots \pi_k, w_0 w_1 \cdots w_k) \in R$ is given by the lexicographic order $>_{lg}$ on $w_0\pi_1 w_1 \pi_2 w_2 \cdots \pi_k w_k$ and ∞ being the greatest element.

Intuitively, the ranking $r(v, wf) = (\pi_1 \cdots \pi_k, w_0 w_1 \cdots w_k)$ is intended to bear the following information. The permutation π states the importance of the pairs. The pair $(F_{\pi_1}, \mathcal{I}_{\pi_1})$ is the most important, hence we are not allowed to visit F_{π_1} and we desire to either visit $\mathcal{I}_{\pi_1}$, or not visit F_{π_2} and visit $\mathcal{I}_{\pi_2}$ and so on. If some important F_i is visited it becomes less important. The importance can be freely changed only finitely many (i_0) times. Otherwise, only less important pairs can be permuted if a more important pair makes good progress. Further, w_i measures the worst possible number of steps until visiting $\mathcal{I}_{\pi_i}$. This intended meaning is formalized in the following notion of good rankings.

Definition 16. *A ranking r is* good *if for every $v \in V, wf \in B$ with $r(v, wf) \neq \infty$ we have $r(v, wf) >_{v,wf} \mathrm{next}(v, wf)$.*

We define $(\pi_1 \cdots \pi_k, w_0 w_1 \cdots w_k) >_{v,wf} (\pi'_1 \cdots \pi'_k, w'_0 w'_1 \cdots w'_k)$ if either $w_0 > w'_0$, or $w_0 = w'_0$ with $>^1_{v,wf}$ hold. Recursively, $>^\ell_{v,wf}$ holds if one of the following holds:

- $\pi_\ell > \pi'_\ell$
- $\pi_\ell = \pi'_\ell$, $v \not\models F_{\pi_\ell}$ *and* $w_\ell > w'_\ell$
- $\pi_\ell = \pi'_\ell$, $v \not\models F_{\pi_\ell}$ *and* $v \models \mathcal{I}_{\pi_\ell}^{wf(\pi_\ell)}$
- $\pi_\ell = \pi'_\ell$, $v \not\models F_{\pi_\ell}$ *and* $w_\ell = w'_\ell$ *and* $>^{\ell+1}_{v,wf}$ *holds (where* $>^{k+1}_{v,wf}$ *never holds)*

Moreover, if one of the first three cases holds, we say that $\succ^\ell_{v,wf}$ holds.

Intuitively, $>$ means the second element is closer to the next milestone and $\succ^\ell$, moreover, that it is so because of the first ℓ pairs in the permutation.

Similarly to [PP06], we obtain the following correctness of the construction. Note that for $|B| = 1$, the definitions of the ranking here and the *Rabin ranking* of [PP06] coincide. Further, the extension with $|B| > 1$ bears some similarities with the Streett ranking of [PP06].

Theorem 17. *For every vertex v, Player 0 has a winning strategy from v if and only if there is a good ranking r and $wf \in B$ with $r(v, wf) \neq \infty$.*

4.2 A fixpoint algorithm

In this section, we show how to compute the smallest good ranking and thus solve the GRP game. Consider a lattice of rankings ordered component-wise, i.e. $r_1 >_c r_2$ if for every $v \in V$ and $wf \in B$, we have $r_1(v, wf) >_{lg} r_2(v, wf)$. This induces a complete lattice. The minimal good ranking is then a least fixpoint of the operator Lift on rankings given by:

$$\mathrm{Lift}(r)(v, wf) = \max\left\{r(v, wf), \min\{x \mid x >_{v,wf} \mathrm{next}(v, wf)\}\right\}$$

where the optima are considered w.r.t. $>_{lg}$. Intuitively, if Player 0 cannot choose a successor smaller than the current vertex (or all successors of a Player 1 vertex are greater), the ranking of the current vertex must rise so that it is greater.

Theorem 18. *The smallest good ranking can be computed in time* $\mathcal{O}(mn^{k+1}kk! \cdot |B|)$ *and space* $(nk \cdot |B|)$.

Proof. The lifting operator can be implemented similarly as in [PP06]. With every change, the affected predecessors to be updated are put in a worklist, thus working in time $\mathcal{O}(k \cdot \text{out-deg}(v))$. Since every element can be lifted at most $|R|$-times, the total time is $\mathcal{O}(\sum_{v \in V} \sum_{wf \in B} k \cdot \text{out-deg}(v) \cdot |R|) = |B|km \cdot n^{k+1}k!$. The space required to store the current ranking is $\mathcal{O}(\sum_{v \in V} \sum_{wf \in B} k) = n \cdot |B| \cdot k$. □

We now compare our solution to the one that would solve the degeneralized Rabin game. The number of vertices of the degeneralized Rabin game is $|B|$ times greater. Hence the time needed is multiplied by a factor $|B|^{k+2}$, instead of $|B|$ in the case of a GRP game. Therefore, our approach speeds up by a factor of $|B|^{k+1}$, while the space requirements are the same in both cases, namely $\mathcal{O}(nk \cdot |B|)$.

Example 19. A conjunction of two fairness constraints of example 8 corresponds to $|B| = 2$ and $k = 4$, hence we save by a factor of $2^4 = 16$. A conjunction of three fairness constraints corresponds to $|B| = 24$ and $k = 8$, hence we accelerate $24^8 \approx 10^{11}$ times.

Further, let us note that the computation can be implemented recursively as in [PP06]. The winning set is $\mu Z.\ \mathfrak{GR}(\mathcal{GR}, \mathbf{tt}, \heartsuit Z)$ where $\mathfrak{GR}(\emptyset, \varphi, W) = W$,

$$\mathfrak{GR}(\mathcal{GR}, \varphi, W) = \bigvee_{i \in [1..k]} \nu Y. \bigwedge_{j \in [1..|\mathcal{I}_i|]} \mu X.\ \mathfrak{GR}\Big(\mathcal{GR} \setminus \{(F_i, \mathcal{I}_i)\}, \varphi \wedge \neg F_i, W \vee (\varphi \wedge \neg F_i \wedge I_i^j \wedge \heartsuit Y) \vee (\varphi \wedge \neg F \wedge \heartsuit X)\Big)$$

$\heartsuit\varphi = \{u \in V_0 \mid \exists (u, v) \in E : v \models \varphi\} \cup \{u \in V_1 \mid \forall (u, v) \in E : v \models \varphi\}$ and μ and ν denote the least and greatest fixpoints, respectively. The formula then provides a succinct description of a symbolic algorithm.

4.3 Experimental Evaluation

Reusing the notation of Section 3.2, we compare the performance of the methods for solving LTL games. We build and solve a Rabin game using

L ltl2dstar producing DRW (from LTL formulae),
R Rabinizer producing DRW, and
GR Rabinizer producing DGRW.

We illustrate the methods on three different games and three LTL formulae; see Table 2. The games contain 3 resp. 6 resp. 9 vertices. Similarly to Section 3.2, s_i denotes the number of vertices in the ith arena, $s_{\mathbf{L}}, s_{\mathbf{R}}, s_{\mathbf{GR}}$ the number of vertices in the resulting games for the three methods, and $t_{\mathbf{L}}, t_{\mathbf{R}}, t_{\mathbf{GR}}$ the respective running times.

Formula 1 allows for a winning strategy and the smallest ranking is relatively small, hence computed quite fast. Formula 2, on the other hand, only allows for larger rankings. Hence the computation takes longer, but also because in **L** and **R** cases the automata are larger than for formula 1. While for **L** and **R**, the product is usually too big, there is a chance to find small rankings in **GR** fast. While for e.g. $\mathbf{FG}(a \vee \neg b \vee c)$, the automata and games would be the same for all three methods and the solution would only take less than a second, the more complex formulae 1 and 2 show clearly the speed up.

Table 2. Experimental comparison of **L**, **R**, and **GR** methods for solving LTL games. Again the sign "−" denotes either crash, out-of-memory, time-out after 30 minutes, or a ratio where one operand is −.

Formula	s_i	$\frac{s_{\mathbf{L}}}{s_i}$	$\frac{s_{\mathbf{R}}}{s_i}$	$\frac{s_{\mathbf{GR}}}{s_i}$	$t_{\mathbf{L}}$	$t_{\mathbf{R}}$	$t_{\mathbf{GR}}$	$\frac{t_{\mathbf{R}}}{t_{\mathbf{GR}}}$	$\lvert B\rvert$	$\frac{t_{\mathbf{L}}}{t_{\mathbf{GR}}}$
$(\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c)$	3	22	7.3	4	63.2	1.6	1.1	1.4	9	48.2
$\vee(\mathbf{GF}\neg a \wedge \mathbf{GF}\neg b \wedge \mathbf{GF}\neg c)$	6	21.3	7.3	3.7	878.6	14.1	7.3	2	9	130.3
	9	20.6	7	3.6	−	54.8	31.3	1.8	9	−
$(\mathbf{GF}a \vee \mathbf{FG}b) \wedge (\mathbf{GF}c \vee \mathbf{GF}\neg a)$	3	21	10	4	−	117.5	12	9.8	6	−
$\wedge(\mathbf{GF}c \vee \mathbf{GF}\neg b)$	6	16.2	9.2	3.7	−	−	196.7	−	6	−
	9	17.6	9.2	3.6	−	−	1017.8	−	6	−

5 Conclusions

In this work we considered the translation of the LTL(**F**,**G**) fragment to deterministic ω-automata that is necessary for probabilistic model checking as well as synthesis. The direct translation to deterministic Muller automata gives a compact automata but the explicit representation of the Muller condition is huge and not algorithmically amenable. In contrast to the traditional approach of translation to deterministic Rabin automata that admits efficient algorithms but incurs a blow-up in translation, we consider deterministic automata with generalized Rabin pairs (DGRW). The translation to DGRW produces the same compact

automata as for Muller conditions. We presented efficient algorithms for probabilistic model checking and game solving with DGRW conditions which shows that the blow-up of translation to Rabin automata is unnecessary. Our results establish that DGRW conditions provide the convenient formalism that allows both for compact automata as well as efficient algorithms. We have implemented our approach in PRISM, and experimental results show a huge improvement over the existing methods. Two interesting directions of future works are (1) extend our approach to LTL with the $\mathbf{U}$(until) and the $\mathbf{X}$(next) operators; and (2) consider symbolic computation and Long's acceleration of fixpoint computation (on the recursive algorithm), instead of the ranking function based algorithm for games, and compare the efficiency of both the approaches.

References

[AT04] R. Alur and S. La Torre. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Log.*, 5(1):1–25, 2004.

[BK08] C. Baier and J.-P. Katoen. *Principles of model checking.* MIT Press, 2008.

[CH11] K. Chatterjee and M. Henzinger. Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification. In *SODA*, pages 1318–1336, 2011.

[Chu62] A. Church. Logic, arithmetic, and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35. Institut Mittag-Leffler, 1962.

[CY95] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.

[dA97] L. de Alfaro. *Formal Verification of Probabilistic Systems.* PhD thesis, Stanford University, 1997.

[Ehl11] R. Ehlers. Generalized rabin(1) synthesis with applications to robust system synthesis. In *NASA Formal Methods*, pages 101–115, 2011.

[GKE12] A. Gaiser, J. Křetínský, and J. Esparza. Rabinizer: Small deterministic automata for ltl(f, g). In *ATVA*, pages 72–76, 2012.

[JB06] B. Jobstmann and R. Bloem. Optimizations for LTL synthesis. In *FMCAD*, pages 117–124. IEEE Computer Society, 2006.

[JGWB07] B. Jobstmann, S. J. Galler, Martin Weiglhofer, and Roderick Bloem. Anzu: A tool for property synthesis. In *CAV*, volume 4590 of *LNCS*, pages 258–262. Springer, 2007.

[KB06] J. Klein and C. Baier. Experiments with deterministic ω-automata for formulas of linear temporal logic. *Theor. Comput. Sci.*, 363(2):182–195, 2006.

[KB07] J. Klein and C. Baier. On-the-fly stuttering in the construction of deterministic ω-automata. In *CIAA*, volume 4783 of *LNCS*, pages 51–61. Springer, 2007.

[KE12] J. Křetínský and J. Esparza. Deterministic automata for the (f, g)-fragment of ltl. In P. Madhusudan and Sanjit A. Seshia, editors, *CAV*, volume 7358 of *Lecture Notes in Computer Science*, pages 7–22. Springer, 2012.

[Kle] J. Klein. ltl2dstar - LTL to deterministic Streett and Rabin automata. http://www.ltl2dstar.de/.

[KNP11] M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[Kup12] O. Kupferman. Recent challenges and ideas in temporal synthesis. In *SOFSEM*, volume 7147 of *LNCS*, pages 88–98. Springer, 2012.

[KV05] O. Kupferman and M. Y. Vardi. Safraless decision procedures. In *FOCS*, pages 531–542. IEEE Computer Society, 2005.

[MS08] A. Morgenstern and K. Schneider. From LTL to symbolically represented deterministic automata. In *VMCAI*, volume 4905 of *LNCS*, pages 279–293. Springer, 2008.

[Pit06] N. Piterman. From nondeterministic Buchi and Streett automata to deterministic parity automata. In *LICS*, pages 255–264, 2006.

[Pnu77] A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.

[PP06] N. Piterman and A. Pnueli. Faster solutions of rabin and streett games. In *LICS*, pages 275–284, 2006.

[PPS06] N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. In *VMCAI*, volume 3855 of *LNCS*, pages 364–380. Springer, 2006.

[PR89] A. Pnueli and R. Rosner. On the synthesis of an asynchronous reactive module. In *ICALP*, volume 372 of *LNCS*, pages 652–671. Springer, 1989.

[Put94] M.L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[PZ86] A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1(1):53–72, 1986.

[Saf88] S. Safra. On the complexity of ω-automata. In *FOCS*, pages 319–327. IEEE Computer Society, 1988.

[Var85] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS*, pages 327–338, 1985.

[VW86] M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *J. Comput. Syst. Sci.*, 32(2):183–221, 1986.

Paper D:

Rabinizer 2: Small Deterministic Automata for LTL$_{\setminus GU}$

Jan Křetínský and Ruslán Ledesma Garza

This tool paper is accepted for publication in Hung Van Dang and Mizuhito Ogawa (eds.): Proceedings of Automated Technology for Verification and Analysis - 11th International Symposium, ATVA 2013, Hanoi, Vietnam, October 15-18, 2012. Lecture Notes in Computer Science. Springer, 2013. [KG13]

Summary

Methods for probabilistic LTL model checking and LTL synthesis mostly require to construct a deterministic ω-automaton from a given LTL formula. We extend the approach of [KE12] (Paper A) for the LTL fragment with only **F** and **G** operators to the fragment with **X**, **F**, **G** and **U** operators where **U** does not appear in the scope of any **G** in the positive normal form. Further, instead of Rabin automata we produce the respective generalised Rabin automata since these are directly applicable in probabilistic LTL model checking and LTL synthesis as we show in [CGK13] (Paper C). The idea of the extension is the following. In the previous approach, each state of the automaton had two components: (1) the formula to be currently satisfied, and (2) the current letter read. With the introduction of **X** operator, the second component now carries last n steps of the history where n is the nesting depth of the **X** operator. This is further vastly optimised by keeping only an equivalence class of the history with respect to subformulae occurring in the scope of **G** operators. Experimental results show huge improvement for formulae involving combinations of properties dependent only on suffixes.

Author's contribution: 65 %

- design of the extension and the optimisation,
- the proofs of correctness,
- design of pseudo-codes,
- experimental evaluation,
- writing the paper.

Rabinizer 2: Small Deterministic Automata for $\mathrm{LTL}_{\setminus \mathbf{GU}}$

Jan Křetínský[1,2*] and Ruslán Ledesma Garza[1†]

[1] Institut für Informatik, Technische Universität München, Germany
[2] Faculty of Informatics, Masaryk University, Brno, Czech Republic

Abstract. We present a tool that generates automata for LTL(**X**,**F**,**G**,**U**) where **U** does not occur in any **G**-formula (but **F** still can). The tool generates deterministic generalized Rabin automata (DGRA) significantly smaller than deterministic Rabin automata (DRA) generated by state-of-the-art tools. For complex properties such as fairness constraints, the difference is in orders of magnitude. DGRA have been recently shown to be as useful in probabilistic model checking as DRA, hence the difference in size directly translates to a speed up of the model checking procedures.

1 Introduction

Linear temporal logic (LTL) is a very useful and appropriate language for specifying properties of systems. In the verification process that follows the automata-theoretic approach, an LTL formula is first translated to an ω-automaton and then a product of the automaton and the system is constructed and analyzed. The automata used here are typically non-deterministic Büchi automata (NBA) as they recognize all ω-regular languages and thus also LTL languages. However, for two important applications, *deterministic* ω-automata are important: probabilistic model checking and synthesis of reactive modules for LTL specifications. Here deterministic Rabin automata (DRA) are typically used as deterministic Büchi automata are not as expressive as LTL. In order to transform an NBA to a DRA, one needs to employ either Safra's construction (or some other exponential construction). This approach is taken in `PRISM` [7] a leading probabilistic model checker, which reimplements the optimized Safra's construction of `ltl2dstar` [4]. However, a straight application of this very general construction often yields unnecessarily large automata and thus also large products, often too large to be analyzed.

In order to circumvent this difficulty, one can focus on fragments of LTL. The most prominent ones are GR(1)—a restricted, but useful fragment of LTL(**X**,**F**,**G**) allowing for fast synthesis—and fragments of LTL(**F**,**G**) as investigated in e.g. [1]. Recently [6], we showed how to construct DRA from LTL(**F**,**G**) directly without NBA. As we argued there, this is an interesting fragment also because it can express all complex fairness constraints, which are widely used in verification. We implemented our approach in a tool `Rabinizer` [3] and observed significant improvements, especially for complex formulae: for example, for a conjunction of three fairness constraints `ltl2dstar` produces a DRA with more than a million states, while `Rabinizer` produces 469 states. Moreover, we introduced a new type of automaton a *deterministic generalized Rabin automaton* (DGRA), which

* The author is supported by the Czech Science Foundation, grant No. P202/12/G061.
† The author is supported by the DFG Graduiertenkolleg 1480 (PUMA).

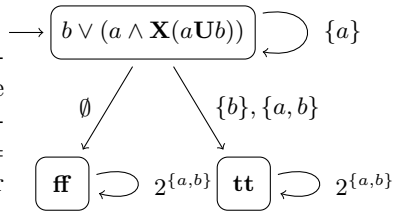
is an intermediate step in our construction, and only has 64 states in the fairness example and only 1 state if transition acceptance is used. In [2], we then show that for probabilistic model checking DGRA are not more difficult to handle than DRA. Hence, without tradeoff, we can use often much smaller DGRA, which are only produced by our construction.

Here, we present a tool `Rabinizer 2` that extends our method and implements it for $\mathrm{LTL}_{\setminus\mathbf{GU}}$ a fragment of LTL($\mathbf{X}$,$\mathbf{F}$,$\mathbf{G}$,$\mathbf{U}$) where $\mathbf{U}$ are not inside $\mathbf{G}$-formulae (but $\mathbf{F}$ still can) in negation normal form. This fragment is not only substantially more complex, but also practically more useful. Indeed, with the unrestricted $\mathbf{X}$-operator, it covers GR(1) and can capture properties describing local structure of systems and is necessary for description of precise sequences of steps. Further, $\mathbf{U}$-operator allows to distinguish paths depending on their initial parts and then we can require different fairness constraints on different paths such as in $\mathit{wait}\mathbf{U}(\mathit{answer}_1 \wedge \phi_1) \vee \mathit{wait}\mathbf{U}(\mathit{answer}_2 \wedge \phi_2)$ where ϕ_1, ϕ_2 are two fairness constraints. As another example, consider patterns for "before": for "absence" we have $\mathbf{F}r \rightarrow (\neg p \mathbf{U} r)$, for "constrained chains" $\mathbf{F}r \rightarrow (p \rightarrow (\neg r \mathbf{U}(s \wedge \neg r \wedge \neg z \wedge \mathbf{X}((\neg r \wedge \neg z)\mathbf{U}t))))\mathbf{U}r$.

Furthermore, as opposed to other tools (including `Rabinizer`), `Rabinizer 2` can also produce DGRA, which are smaller by orders of magnitude for complex formulae. For instance, for a conjunction of four fairness constraints the constructed DGRA has 256 states, while the directly degeneralized DRA is 20736-*times* bigger [2]. As a result, we not only obtain smaller DRA now for much larger fragment (by degeneralizing the DGRA into DRA), but also the power of DGRA is made available for this fragment allowing for the respective speed up of probabilistic model checking.

The tool can be downloaded and additional materials and proofs found at `http://www.model.in.tum.de/~kretinsk/rabinizer2.html`

2 Algorithm



Let us fix a formula φ of $\mathrm{LTL}_{\setminus\mathbf{GU}}$. We construct an automaton $\mathcal{A}(\varphi)$ recognizing models of φ. Details can be found on the tool's webpage. In every step, $\mathcal{A}(\varphi)$ unfolds φ as in [6], now we also define $\mathfrak{Unf}(\psi_1 \mathbf{U} \psi_2) = \mathfrak{Unf}(\psi_2) \vee (\mathfrak{Unf}(\psi_1) \wedge \mathbf{X}(\psi_1 \mathbf{U} \psi_2))$. Then it checks whether the letter currently read complies with thus generated requirements, see the example on the right for $\varphi = a\mathbf{U}b$. E.g. reading $\{a\}$ yields requirement $\mathbf{X}(a\mathbf{U}b)$ for the next step, thus in the next step we have $\mathfrak{Unf}(a\mathbf{U}b)$ which is the same as in the initial state, hence we loop.

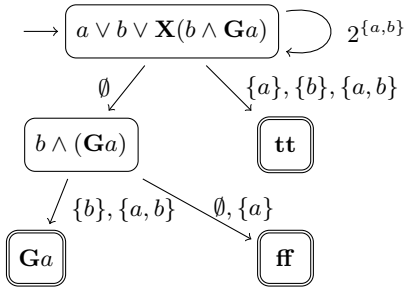
Some requirements can be checked at a finite time by this unfolding, such as $b\mathbf{U}(a \wedge \mathbf{X}b)$, some cannot, such as $\mathbf{GF}(a \wedge \mathbf{X}b)$. The state space has to monitor the latter requirements (such as the repetitive satisfaction of $a \wedge \mathbf{X}b$) separately. To this end, let $\mathbf{G}_\varphi := \{\mathbf{G}\psi \in \mathrm{sf}(\varphi)\}$ and $\mathbf{F}_\varphi := \{\mathbf{F}\psi \in \mathrm{sf}(\omega) \mid \text{for some } \omega \in \mathbf{G}_\varphi\}$ where $\mathrm{sf}(\varphi)$ denotes the set of all subformulae of φ. Then $\mathcal{R}\mathrm{ec} := \{\psi \mid \mathbf{G}\psi \in \mathbf{G}_\varphi \text{ or } \mathbf{F}\psi \in \mathbf{F}_\varphi\}$ is the set of *recurrent* subformulae of φ, whose repeated satisfaction we must check. (Note that no $\mathbf{U}$ occurs in formulae of $\mathcal{R}$ec.) In the case without the $\mathbf{X}$ operator [6, 3], such as with $\mathbf{GF}a$, it was sufficient to record

the currently read letter in the states of $\mathcal{A}(\varphi)$. Then the acceptance condition checks whether e.g. a is visited infinitely often. Now we could extend this to keep history of the last n letters read where n is the nesting depth of the $\mathbf{X}$ operator in φ. In order to reduce the size of the state space, we rather store equivalence classes thereof. This is realized by automata. For every $\xi \in \mathcal{R}\mathrm{ec}$, we have a finite automaton $\mathcal{B}(\xi)$, and $\mathcal{A}(\varphi)$ will keep track of its current states.

Construction of $\mathcal{B}(\xi)$: We define a finite automaton $\mathcal{B}(\xi) = (Q_\xi, i_\xi, \delta_\xi, F_\xi)$ over 2^{Ap} by

- the set of states $Q_\xi = \mathsf{B}^+(\mathrm{sf}(\xi))$, where $\mathsf{B}^+(S)$ is the set of positive Boolean functions over S and $\mathbf{tt}$ and $\mathbf{ff}$,
- the initial state $i_\xi = \xi$,
- the final states F_ξ where each atomic proposition has $\mathbf{F}$ or $\mathbf{G}$ as an ancestor in the syntactic tree (i.e. no atomic propositions are guarded by only $\mathbf{X}$'s and Boolean connectives),
- transition relation δ_ξ is defined by transitions

$$\chi \xrightarrow{\nu} \mathbf{X}^{-1}(\chi[\nu]) \quad \text{for every } \nu \subseteq Ap \text{ and } \chi \notin F$$
$$i \xrightarrow{\nu} i \quad \text{for every } \nu \subseteq Ap$$



where $\chi[\nu]$ is the function χ with $\mathbf{tt}$ and $\mathbf{ff}$ plugged in for atomic propositions according to ν and $\mathbf{X}^{-1}\chi$ strips away the initial $\mathbf{X}$ (whenever there is one) from each formula in the Boolean combination χ. Note that we do not unfold inner $\mathbf{F}$- and $\mathbf{G}$-formulae. See an example for $\xi = a \vee b \vee \mathbf{X}(b \wedge \mathbf{G}a)$ on the right.

Construction of $\mathcal{A}(\varphi)$: The state space has two components. Beside the component keeping track of the input formula, we also keep track of the history for every recurrent formula of $\mathcal{R}\mathrm{ec}$. The second component is then a vector of length $|\mathcal{R}\mathrm{ec}|$ keeping the current set of states of each $\mathcal{B}(\xi)$. Formally, we define $\mathcal{A}(\varphi) = (Q, i, \delta)$ to be a deterministic finite automaton over $\Sigma = 2^{Ap}$ given by

- set of states $Q = \mathsf{B}^+(\mathrm{sf}(\varphi) \cup \mathbf{X}\mathrm{sf}(\varphi)) \times \prod_{\xi \in \mathcal{R}\mathrm{ec}} 2^{Q_\xi}$ where $\mathbf{X}S = \{\mathbf{X}s \mid s \in S\}$,
- the initial state $i = \langle \mathfrak{Unf}(\varphi), (\xi \mapsto \{i_\xi\})_{\xi \in \mathcal{R}\mathrm{ec}} \rangle$;
- the transition function δ is defined by transitions

$$\langle \psi, (R_\xi)_{\xi \in \mathcal{R}\mathrm{ec}} \rangle \xrightarrow{\nu} \langle \mathfrak{Unf}(\mathbf{X}^{-1}(\psi[\nu])), \big(\delta_\xi(R_\xi, \nu)\big)_{\xi \in \mathcal{R}\mathrm{ec}} \rangle$$

On $\mathcal{A}(\varphi)$ it is possible to define an acceptance condition such that $\mathcal{A}(\varphi)$ recognizes models of φ. The approach is similar to [6], but now we have to take the information of each $\mathcal{B}(\xi)$ into account. We use this information to get look-ahead necessary for evaluating $\mathbf{X}$-requirements in the first component of $\mathcal{A}(\varphi)$. However, since storing complete future look-ahead would be costly, $\mathcal{B}(\xi)$ actually stores the compressed information of past. The acceptance condition allows then for deducing enough information about the future.

Further optimizations include not storing states of each $\mathcal{B}(\xi)$, but only the currently relevant ones. E.g. after reading $\emptyset$ in $\mathbf{GF}a \vee (b \wedge \mathbf{GF}c)$, it is no more interesting to track if c occurs infinitely often. Further, since only the infinite behaviour of $\mathcal{B}(\xi)$ is important and it has acyclic structure (except for the initial states), instead of the initial state we can start in any subset of states. Therefore, we start in a subset that will occur repetitively and we thus omit unnecessary initial transient parts of $\mathcal{A}(\varphi)$.

3 Experimental results

We compare our tool to `ltl2dstar`, which yields the same automata as its Java reimplementation in PRISM. We consider some formulae on which `ltl2dstar` was originally tested [5], some formulae used in a network monitoring project Liberouter (https://www.liberouter.org/) showing the $\mathrm{LTL}_{\setminus \mathbf{GU}}$ fragment is practically very relevant, and several other formulae with more involved structure such as ones containing fairness constraints. For results on the LTL($\mathbf{F}$,$\mathbf{G}$) subfragment, we refer to [3]. Due to [2], it only makes sense to use DGRA and we thus display the sizes of DGRA for `Rabinizer 2` (except for the more complex cases this, however, coincides with the degeneralized DRA). Here "?" denotes time-out after 30 minutes. For more experiments, see the webpage.

Formula	`ltl2d*`	`R.2`
$(\mathbf{F}p)\mathbf{U}(\mathbf{G}q)$	4	3
$(\mathbf{G}p)\mathbf{U}q$	5	5
$\neg(p\mathbf{U}q)$	4	3
$\mathbf{G}(p \rightarrow \mathbf{F}q) \wedge ((\mathbf{X}p)\mathbf{U}q) \vee \neg\mathbf{X}(p\mathbf{U}(p \wedge q))$	19	8
$\mathbf{G}(q \vee \mathbf{X}\mathbf{G}p) \wedge \mathbf{G}(r \vee \mathbf{X}\mathbf{G}\neg p)$	5	14
$((\mathbf{G}(\mathbf{F}(p_1) \wedge \mathbf{F}(\neg p_1)))) \rightarrow (\mathbf{G}((p_2 \wedge \mathbf{X}p_2 \wedge \neg p_1 \wedge \mathbf{X}p_1 \rightarrow ((p_3) \rightarrow \mathbf{X}p_4))))$	11	8
$((p_1 \wedge \mathbf{X}\mathbf{G}(\neg p_1)) \wedge (\mathbf{G}((\mathbf{F}p_2) \wedge (\mathbf{F}\neg p_2))) \wedge ((\neg p_2))) \rightarrow (((\neg p_2)\mathbf{U}$ $\mathbf{G}(\neg((p_3 \wedge p_4) \vee (p_3 \wedge p_5) \vee (p_3 \wedge p_6) \vee (p_4 \wedge p_5) \vee (p_4 \wedge p_6) \vee (p_5 \wedge p_6)))))$	17	8
$(\mathbf{X}p_1 \wedge \mathbf{G}((\neg p_1 \wedge \mathbf{X}p_1) \rightarrow XXp_1) \wedge \mathbf{GF}\neg p_1 \wedge \mathbf{GF}p_2 \wedge \mathbf{GF}\neg p_2) \rightarrow$ $(\mathbf{G}(p_3 \wedge p_4 \wedge !p_2 \wedge \mathbf{X}p_2 \rightarrow \mathbf{X}(p_1 \vee \mathbf{X}(\neg p_4 \vee p_1))))$	9	7
$\mathbf{F}r \rightarrow (p \rightarrow (\neg r\mathbf{U}(s \wedge \neg r \wedge \neg z \wedge \mathbf{X}((\neg r \wedge \neg z)\mathbf{U}t))))\mathbf{U}r$	6	5
$((\mathbf{GF}(a \wedge \mathbf{XX}b) \vee \mathbf{FG}b) \wedge \mathbf{FG}(c \vee (\mathbf{X}a \wedge \mathbf{XX}b)))$	353	73
$\mathbf{GF}(\mathbf{XXX}a \wedge \mathbf{XXXX}b) \wedge \mathbf{GF}(b \vee \mathbf{X}c) \wedge \mathbf{GF}(c \wedge \mathbf{XX}a)$	2127	85
$(\mathbf{GF}a \vee \mathbf{FG}b) \wedge (\mathbf{GF}c \vee \mathbf{FG}(d \vee \mathbf{X}e))$	18176	40
$(\mathbf{GF}(a \wedge \mathbf{XX}c) \vee \mathbf{FG}b) \wedge (\mathbf{GF}c \vee \mathbf{FG}(d \vee \mathbf{X}a \wedge \mathbf{XX}b))$	?	142
$a\mathbf{U}b \wedge (\mathbf{GF}a \vee \mathbf{FG}b) \wedge (\mathbf{GF}c \vee \mathbf{FG}d) \vee a\mathbf{U}c \wedge (\mathbf{GF}a \vee \mathbf{FG}d) \wedge (\mathbf{GF}c \vee \mathbf{FG}b)$	?	60

References

1. Rajeev Alur and Salvatore La Torre. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Log.*, 5(1):1–25, 2004.
2. Krishnendu Chatterjee, Andreas Gaiser, and Jan Křetínský. Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis. In *CAV*, pages 559–575, 2013.
3. Andreas Gaiser, Jan Křetínský, and Javier Esparza. Rabinizer: Small deterministic automata for LTL(F,G). In *ATVA*, pages 72–76, 2012.
4. Joachim Klein. ltl2dstar - LTL to deterministic Streett and Rabin automata. http://www.ltl2dstar.de/.
5. Joachim Klein and Christel Baier. Experiments with deterministic *omega*-automata for formulas of linear temporal logic. *Theor. Comput. Sci.*, 363(2):182–195, 2006.
6. Jan Křetínský and Javier Esparza. Deterministic automata for the (F,G)-fragment of LTL. In *CAV*, pages 7–22, 2012.
7. Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, pages 585–591, 2011.

Part II

Papers on continuous-time Markovian systems

Paper E:

Continuous-Time Stochastic Games with Time-Bounded Reachability

Tomáš Brázdil, Vojtěch Forejt, Jan Krčál, Jan Křetínský, and Antonín Kučera

This paper has been published in Information and Computation, vol. 224, pages 46-70. Elsevier, 2013. [BFK+13]

Summary

Continuous-time Markov decision processes are an established model in operations research, biology, performance evaluation etc. for modelling either open systems or closed controllable systems. Modelling open controllable systems and optimisation of their behaviour requires a game extension. We give such an extension called continuous-time stochastic games and study its properties. Firstly, we establish determinacy and non/existence of optimal time-abstract strategies in these games. Secondly, we extend and optimise the algorithm for time-bounded reachability with ε-optimal time-abstract strategies in the continuous-time Markov decision processes of [BHHK04] to the setting of continuous-time stochastic games and analyse its complexity. Thirdly, we give an algorithm to compute optimal time-abstract strategies.

Author's contribution: 45 %

- the results and proofs concerning the structure of optimal and ε-optimal strategies,
- the design of parts of the algorithms and the respective proofs of correctness,

- the analysis of the complexity,
- the extension of the conference paper [BFK+09] (12 pages) to the present journal paper (25 pages, 39 pages in the preprint version presented here).

Continuous-Time Stochastic Games with Time-Bounded Reachability[☆]

Tomáš Brázdil, Vojtěch Forejt[1], Jan Krčál, Jan Křetínský[2,*], Antonín Kučera

Faculty of Informatics, Masaryk University, Botanická 68a, 60200 Brno, Czech Republic

Abstract

We study continuous-time stochastic games with time-bounded reachability objectives and time-abstract strategies. We show that each vertex in such a game has a *value* (i.e., an equilibrium probability), and we classify the conditions under which optimal strategies exist. Further, we show how to compute ε-optimal strategies in finite games and provide detailed complexity estimations. Moreover, we show how to compute ε-optimal strategies in infinite games with finite branching and bounded rates where the bound as well as the successors of a given state are effectively computable. Finally, we show how to compute optimal strategies in finite uniform games.

Keywords: continuous time stochastic systems, time-bounded reachability, stochastic games

1. Introduction

Markov models are widely used in many diverse areas such as economics, biology, or physics. More recently, they have also been used for performance and dependability analysis of computer systems. Since faithful modeling of computer systems often requires both *randomized* and *non-deterministic* choice, a lot of attention has been devoted to Markov models where these two phenomena co-exist, such as *Markov decision processes* and *stochastic games*. The latter

[☆]This is an extended version of FSTTCS'09 paper with full proofs and improved complexity bounds on the ε-optimal strategies algorithm. The work has been supported by Czech Science Foundation, grant No. P202/10/1469. Jan Křetínský is a holder of Brno PhD Talent Financial Aid. Vojtěch Forejt was supported in part by ERC Advanced Grant VERIWARE and a Royal Society Newton Fellowship.

[*]Corresponding author, phone no.: +49 89 289 17236 , fax no.: +49 89 289 17207

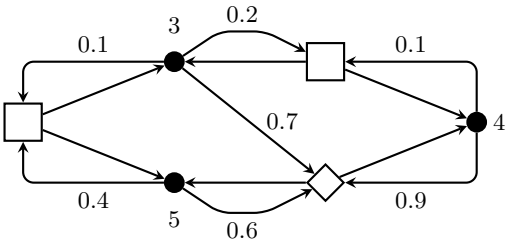
Email addresses: brazdil@fi.muni.cz (Tomáš Brázdil), vojfor@cs.ox.ac.uk (Vojtěch Forejt), krcal@fi.muni.cz (Jan Krčál), jan.kretinsky@fi.muni.cz (Jan Křetínský), kucera@fi.muni.cz (Antonín Kučera)

[1]Present address: Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK

[2]Present address: Institut für Informatik, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany

model of stochastic games is particularly apt for analyzing the interaction between a system and its environment, which are formalized as two *players* with antagonistic objectives (we refer to, e.g., [1, 2, 3] for more comprehensive expositions of results related to games in formal analysis and verification of computer systems). So far, most of the existing results concern *discrete-time* Markov decision processes and stochastic games, and the accompanying theory is relatively well-developed (see, e.g., [4, 5]).

In this paper, we study *continuous-time stochastic games (CTGs)* and hence also *continuous-time Markov decision processes (CTMDPs)* with time-bounded reachability objectives. Roughly speaking, a CTG is a finite or countably infinite graph with three types of vertices—controllable vertices (boxes), adversarial vertices (diamonds), and actions (circles). The outgoing edges of controllable and adversarial vertices lead to the actions that are *enabled* at a given vertex. The outgoing edges of actions lead to controllable or adversarial vertices, and every edge is assigned a positive probability so that the total sum of these probabilities in each vertex is equal to 1. Further, each action is assigned a positive real *rate*. A simple finite CTG is shown below.



A game is played by two players, $\Box$ and $\Diamond$, who are responsible for selecting the actions (i.e., resolving the non-deterministic choice) in the controllable and adversarial vertices, respectively. The selection is timeless, but performing a selected action takes time which is exponentially distributed (the parameter is the rate of a given action). When a given action is finished, the next vertex is chosen randomly according to the fixed probability distribution over the outgoing edges of the action. A *time-bounded reachability objective* is specified by a set T of target vertices and a time bound $t > 0$. The goal of player $\Box$ is to maximize the probability of reaching a target vertex before time t, while player $\Diamond$ aims at minimizing this probability.

Note that events such as component failures, user requests, message receipts, exceptions, etc., are essentially history-independent, which means that the time between two successive occurrences of such events is exponentially distributed. CTGs provide a natural and convenient formal model for systems exhibiting these features, and time-bounded reachability objectives allow to formalize basic liveness and safety properties of these systems.

Previous work. Although the practical relevance of CTGs with time-bounded reachability objectives to verification problems is obvious, to the best of our knowledge there are no previous results concerning even very basic properties of such games. A more restricted model of uniform CTMDPs is studied

in [6, 7]. Intuitively, a uniform CTMDP is a CTG where all non-deterministic vertices are controlled just by one player, and all actions are assigned the same rate. In [6], it is shown that the maximal and minimal probability of reaching a target vertex before time t is efficiently computable up to an arbitrarily small given error, and that the associated strategy is also effectively computable. An open question explicitly raised in [6] is whether this result can be extended to all (not necessarily uniform) CTMDP. In [6], it is also shown that time-dependent strategies are more powerful than time-abstract ones, and this issue is addressed in greater detail in [7] where the mutual relationship between various classes of time-dependent strategies in CTMDPs is studied. Furthermore, in [8] reward-bounded objectives in CTMDPs are studied.

Our contribution is twofold. Firstly, we examine the *fundamental properties* of CTGs, where we aim at obtaining as general (and tight) results as possible. Secondly, we consider the associated *algorithmic issues*. Concrete results are discussed in the following paragraphs.

Fundamental properties of CTGs. We start by showing that each vertex v in a CTG with time-bounded reachability objectives has a *value*, i.e., an *equilibrium probability* of reaching a target vertex before time t. The value is equal to $\sup_\sigma \inf_\pi \mathcal{P}_v^{\sigma,\pi}(\mathit{Reach}^{\leq t}(T))$ and $\inf_\pi \sup_\sigma \mathcal{P}_v^{\sigma,\pi}(\mathit{Reach}^{\leq t}(T))$, where σ and π range over all time-abstract strategies of player $\Box$ and player $\Diamond$, and $\mathcal{P}_v^{\sigma,\pi}(\mathit{Reach}^{\leq t}(T))$ is the probability of reaching T before time t when starting in v in a play obtained by applying the strategies σ and π. This result holds for *arbitrary* CTGs which may have countably many vertices and actions. This immediately raises the question whether each player has an *optimal* strategy which achieves the outcome equal to or better than the value against every strategy of the opponent. We show that the answer is negative in general, but an optimal strategy for player $\Diamond$ is guaranteed to exist in *finitely-branching* CTGs, and an optimal strategy for player $\Box$ is guaranteed to exist in *finitely-branching* CTGs with *bounded rates* (see Definition 2.2). These results are tight, which is documented by appropriate counterexamples. Moreover, we show that in the subclasses of CTGs just mentioned, the players have also optimal CD strategies (a strategy is CD if it is deterministic and "counting", i.e., it only depends on the number of actions in the history of a play, where actions with the same rate are identified). Note that CD strategies still use infinite memory and in general they do not admit a finite description. A special attention is devoted to finite uniform CTGs, where we show a somewhat surprising result—both players have *finite memory optimal strategies* (these finite memory strategies are deterministic and their decision is based on "bounded counting" of actions; hence, we call them "BCD"). Using the technique of uniformization, one can generalize this result to all finite (not necessarily uniform) games, see [9].

Algorithms. We show that for finite CTGs, ε-optimal strategies for both players are computable in $|V|^2 \cdot |A| \cdot bp^2 \cdot \left((\max \mathcal{R}) \cdot t + \ln \frac{1}{\varepsilon}\right)^{2|\mathcal{R}|+\mathcal{O}(1)}$ time, where $|V|$ and $|A|$ is the number of vertices and actions, resp., bp is the maximum bit-length of transition probabilities and rates (we assume that rates and the probabilities in distributions assigned to the actions are represented as fractions

of integers encoded in binary), $|\mathcal{R}|$ is the number of rates, $\max \mathcal{R}$ is the maximal rate, and t is the time bound. This solves the open problem of [6] (in fact, our result is more general as it applies to finite CTGs, not just to finite CTMDPs). Actually, the algorithm works also for *infinite-state* CTGs as long as they are finitely-branching, have bounded rates, and satisfy some natural "effectivity assumptions" (see Corollary 5.26). For example, this is applicable to the class of infinite-state CTGs definable by pushdown automata (where the rate of a given configuration depends just on the current control state), and also to other automata-theoretic models. Finally, we show how to compute the optimal BCD strategies for both players in finite uniform CTGs.

Some proofs that are rather technical have been shifted into Appendix C.

2. Definitions

In this paper, the sets of all positive integers, non-negative integers, rational numbers, real numbers, non-negative real numbers, and positive real numbers are denoted by $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{R}^{\geq 0}$, and $\mathbb{R}^{>0}$, respectively. Let A be a finite or countably infinite set. A *probability distribution* on A is a function $f : A \to \mathbb{R}^{\geq 0}$ such that $\sum_{a \in A} f(a) = 1$. The *support* of f is the set of all $a \in A$ where $f(a) > 0$. A distribution f is *Dirac* if $f(a) = 1$ for some $a \in A$. The set of all distributions on A is denoted by $\mathcal{D}(A)$. A *σ-field* over a set Ω is a set $\mathcal{F} \subseteq 2^\Omega$ that contains Ω and is closed under complement and countable union. A *measurable space* is a pair $(\Omega, \mathcal{F})$ where Ω is a set called *sample space* and $\mathcal{F}$ is a σ-field over Ω whose elements are called *measurable sets*. A *probability measure* over a measurable space $(\Omega, \mathcal{F})$ is a function $\mathcal{P} : \mathcal{F} \to \mathbb{R}^{\geq 0}$ such that, for each countable collection $\{X_i\}_{i \in I}$ of pairwise disjoint elements of $\mathcal{F}$, $\mathcal{P}(\bigcup_{i \in I} X_i) = \sum_{i \in I} \mathcal{P}(X_i)$, and moreover $\mathcal{P}(\Omega) = 1$. A *probability space* is a triple $(\Omega, \mathcal{F}, \mathcal{P})$, where $(\Omega, \mathcal{F})$ is a measurable space and $\mathcal{P}$ is a probability measure over $(\Omega, \mathcal{F})$. Given two measurable sets $X, Y \in \mathcal{F}$ such that $\mathcal{P}(Y) > 0$, the *conditional probability* of X under the condition Y is defined as $\mathcal{P}(X \mid Y) = \mathcal{P}(X \cap Y)/\mathcal{P}(Y)$. We say that a property $A \subseteq \Omega$ holds *for almost all* elements of a measurable set Y if $\mathcal{P}(Y) > 0$, $A \cap Y \in \mathcal{F}$, and $\mathcal{P}(A \cap Y \mid Y) = 1$.

In our next definition we introduce continuous-time Markov chains (CTMCs). The literature offers several equivalent definitions of CTMCs (see, e.g., [10]). For purposes of this paper, we adopt the variant where transitions have discrete probabilities and the rates are assigned to states.

Definition 2.1. A *continuous-time Markov chain (CTMC)* is a tuple $\mathcal{M} = (S, \mathbf{P}, \mathbf{R}, \mu)$, where S is a finite or countably infinite set of *states*, $\mathbf{P}$ is a *transition probability function* assigning to each $s \in S$ a probability distribution over S, $\mathbf{R}$ is a function assigning to each $s \in S$ a positive real *rate*, and μ is the *initial probability distribution* on S.

If $\mathbf{P}(s)(s') = x > 0$, we write $s \xrightarrow{x} s'$ or shortly $s \to s'$. A *time-abstract path* is a finite or infinite sequence $u = u_0, u_1, \ldots$ of states such that $u_{i-1} \to u_i$ for every $1 \leq i < \mathit{length}(u)$, where $\mathit{length}(u)$ is the length of u (the length of

an infinite sequence is ∞). A *timed path* (or just *path*) is a pair $w = (u, t)$, where u is a time-abstract path and $t = t_1, t_2, \ldots$ is a sequence of positive reals such that $\mathit{length}(t) = \mathit{length}(u)$. We put $\mathit{length}(w) = \mathit{length}(u)$, and for every $0 \leq i < \mathit{length}(w)$, we usually write $w(i)$ and $w[i]$ instead of u_i and t_i, respectively.

Infinite paths are also called *runs*. The set of all runs in $\mathcal{M}$ is denoted $\mathit{Run}_{\mathcal{M}}$, or just Run when $\mathcal{M}$ is clear from the context. A *template* is a pair (u, I), where $u = u_0, u_1, \ldots$ is a finite time-abstract path and $I = I_0, I_1, \ldots$ a finite sequence of non-empty intervals in $\mathbb{R}^{\geq 0}$ such that $\mathit{length}(u) = \mathit{length}(I)+1$. Every template (u, I) determines a *basic cylinder* $\mathit{Run}(u, I)$ consisting of all runs w such that $w(i) = u_i$ for all $0 \leq i < \mathit{length}(u)$, and $w[j] \in I_j$ for all $0 \leq i < \mathit{length}(u) - 1$. To $\mathcal{M}$ we associate the probability space $(\mathit{Run}, \mathcal{F}, \mathcal{P})$ where $\mathcal{F}$ is the σ-field generated by all basic cylinders $\mathit{Run}(u, I)$ and $\mathcal{P} : \mathcal{F} \to \mathbb{R}^{\geq 0}$ is the unique probability measure on $\mathcal{F}$ such that

$$\mathcal{P}(\mathit{Run}(u, I)) = \mu(u_0) \cdot \prod_{i=0}^{\mathit{length}(u)-2} \mathbf{P}(u_i)(u_{i+1}) \cdot \left(e^{-\mathbf{R}(u_i) \cdot \inf(I_i)} - e^{-\mathbf{R}(u_i) \cdot \sup(I_i)} \right)$$

Note that if $\mathit{length}(u) = 1$, the "big product" above is empty and hence equal to 1.

Now we formally define continuous-time games, which generalize continuous-time Markov chains by allowing not only probabilistic but also *non-deterministic* choice. Continuous-time games also generalize the model of continuous-time Markov decision processes studied in [6, 7] by splitting the non-deterministic vertices into two disjoint subsets of *controllable* and *adversarial* vertices, which are controlled by two players with antagonistic objectives. Thus, one can model the interaction between a system and its environment.

Definition 2.2. A *continuous-time game (CTG)* is a tuple $G = (V, A, \mathbf{E}, (V_\Box, V_\Diamond), \mathbf{P}, \mathbf{R})$ where V is a finite or countably infinite set of *vertices*, A is a finite or countably infinite set of *actions*, $\mathbf{E}$ is a function which to every $v \in V$ assigns a non-empty set of actions *enabled* in v, $(V_\Box, V_\Diamond)$ is a partition of V, $\mathbf{P}$ is a function which assigns to every $a \in A$ a probability distribution on V, and $\mathbf{R}$ is a function which assigns a positive real *rate* to every $a \in A$.

We require that $V \cap A = \emptyset$ and use N to denote the set $V \cup A$. We say that G is *finitely-branching* if for each $v \in V$ the set $\mathbf{E}(v)$ is finite (note that $\mathbf{P}(a)$ for a given $a \in A$ can still have an infinite support even if G is finitely branching). We say that G has *bounded rates* if $\sup_{a \in A} \mathbf{R}(a) < \infty$, and that G is *uniform* if $\mathbf{R}$ is a constant function. Finally, we say that G is *finite* if N is finite.

If $V_\Box$ or $V_\Diamond$ is empty (i.e., there is just one type of vertices), then G is a *continuous-time Markov decision process (CTMDP)*. Technically, our definition of CTMDP is slightly different from the one used in [6, 7], but the difference is only cosmetic. The two models are equivalent in a well-defined sense (a detailed explanation is included in Appendix B). Also note that $\mathbf{P}$ and $\mathbf{R}$ associate the probability distributions and rates directly to actions, not to pairs of $V \times A$. This

is perhaps somewhat non-standard, but leads to simpler notation (since each vertex can have its "private" set of enabled actions, this is no real restriction).

A *play* of G is initiated in some vertex. The non-deterministic choice is resolved by two players, $\Box$ and $\Diamond$, who select the actions in the vertices of $V_\Box$ and $V_\Diamond$, respectively. The selection itself is timeless, but some time is spent by performing the selected action (the time is exponentially distributed with the rate $\mathbf{R}(a)$), and then a transition to the next vertex is chosen randomly according to the distribution $\mathbf{P}(a)$. The players can also select the actions *randomly*, i.e., they select not just a single action but a *probability distribution* on the enabled actions. Moreover, the players are allowed to play differently when the same vertex is revisited. We assume that both players can see the history of a play, but cannot measure the elapsed time.

Let $\odot \in \{\Box, \Diamond\}$. A *strategy* for player $\odot$ is a function which to each $wv \in N^*V_\odot$ assigns a probability distribution on $\mathbf{E}(v)$. The sets of all strategies for player $\Box$ and player $\Diamond$ are denoted by Σ and Π, respectively. Each pair of strategies $(\sigma, \pi) \in \Sigma \times \Pi$ together with an initial vertex $\hat{v} \in V$ determine a unique *play* of the game G, which is a CTMC $G(\hat{v}, \sigma, \pi)$ where N^*A is the set of states, the rate of a given $wa \in N^*A$ is $\mathbf{R}(a)$ (the rate function of $G(\hat{v}, \sigma, \pi)$ is also denoted by $\mathbf{R}$), and the only non-zero transition probabilities are between states of the form wa and $wava'$ with $wa \xrightarrow{x} wava'$ iff one of the following conditions is satisfied:

- $v \in V_\Box$, $a' \in \mathbf{E}(v)$, and $x = \mathbf{P}(a)(v) \cdot \sigma(wav)(a') > 0$;
- $v \in V_\Diamond$, $a' \in \mathbf{E}(v)$, and $x = \mathbf{P}(a)(v) \cdot \pi(wav)(a') > 0$.

The initial distribution is determined as follows:

- $\mu(\hat{v}a) = \sigma(\hat{v})(a)$ if $\hat{v} \in V_\Box$ and $a \in \mathbf{E}(\hat{v})$;
- $\mu(\hat{v}a) = \pi(\hat{v})(a)$ if $\hat{v} \in V_\Diamond$ and $a \in \mathbf{E}(\hat{v})$;
- in the other cases, μ returns zero.

Note that the set of states of $G(\hat{v}, \sigma, \pi)$ is infinite. Also note that all states reachable from a state $\hat{v}a$, where $\mu(\hat{v}a) > 0$, are alternating sequences of vertices and actions. We say that a state w of $G(\hat{v}, \sigma, \pi)$ *hits* a vertex $v \in V$ if v is the last vertex which appears in w (for example, $v_1a_1v_2a_2$ hits v_2). Further, we say that w hits $T \subseteq V$ if w hits some vertex of T. From now on, the paths (both finite and infinite) in $G(\hat{v}, \sigma, \pi)$ are denoted by Greek letters $\alpha, \beta, \ldots$. Note that for every $\alpha \in Run_{G(\hat{v},\sigma,\pi)}$ and every $i \in \mathbb{N}_0$ we have that $\alpha(i) = wa$ where $wa \in N^*A$.

We denote by $\mathcal{R}(G)$ the set of all rates used in G (i.e., $\mathcal{R}(G) = \{\mathbf{R}(a) \mid a \in A\}$), and by $\mathcal{H}(G)$ the set of all vectors of the form $\mathbf{i} : \mathcal{R}(G) \to \mathbb{N}_0$ satisfying $\sum_{r \in \mathcal{R}(G)} \mathbf{i}(r) < \infty$. When G is clear from the context, we write just $\mathcal{R}$ and $\mathcal{H}$ instead of $\mathcal{R}(G)$ and $\mathcal{H}(G)$, respectively. For every $\mathbf{i} \in \mathcal{H}$, we put $|\mathbf{i}| = \sum_{r \in \mathcal{R}} \mathbf{i}(r)$. For every $r \in \mathcal{R}$, we denote by $\mathbf{1}_r$ the vector of $\mathcal{H}$ such that $\mathbf{1}_r(r) = 1$ and $\mathbf{1}_r(r') = 0$ if $r' \neq r$. Further, for every $wx \in N^*N$ we

define the vector $\mathbf{i}_{wx} \in \mathcal{H}$ such that $\mathbf{i}_{wx}(r)$ returns the cardinality of the set $\{j \in \mathbb{N}_0 \mid 0 \leq j < \mathit{length}(w), w(j) \in A, \mathbf{R}(w(j)) = r\}$. (Note that the last element x of wx is disregarded.) Given $\mathbf{i} \in \mathcal{H}$ and $wx \in N^*N$, we say that wx *matches* $\mathbf{i}$ if $\mathbf{i} = \mathbf{i}_{wx}$.

We say that a strategy τ is *counting (C)* if $\tau(wv) = \tau(w'v)$ for all $v \in V$ and $w, w' \in N^*$ such that $\mathbf{i}_{wv} = \mathbf{i}_{w'v}$. In other words, a strategy τ is counting if the only information about the history of a play w which influences the decision of τ is the vector $\mathbf{i}_{wv}$. Hence, every counting strategy τ can be considered as a function from $\mathcal{H} \times V$ to $\mathcal{D}(A)$, where $\tau(\mathbf{i}, v)$ corresponds to the value of $\tau(wv)$ for every wv matching $\mathbf{i}$. A counting strategy τ is *bounded counting (BC)* if there is $k \in \mathbb{N}$ such that $\tau(wv) = \tau(w'v)$ whenever $\mathit{length}(w) \geq k$ and $\mathit{length}(w') \geq k$. A strategy τ is *deterministic (D)* if $\tau(wv)$ is a Dirac distribution for all wv. Strategies that are not necessarily counting are called *history-dependent (H)*, and strategies that are not necessarily deterministic are called *randomized (R)*. Thus, we obtain the following six types of strategies: BCD, BCR, CD, CR, HD, and HR. The most general (unrestricted) type is HR, and the importance of the other types of strategies becomes clear in subsequent sections.

In this paper, we are interested in continuous-time games with *time-bounded reachability objectives*, which are specified by a set $T \subseteq V$ of *target* vertices and a *time bound* $t \in \mathbb{R}^{>0}$. Let v be an initial vertex. Then each pair of strategies $(\sigma, \pi) \in \Sigma \times \Pi$ determines a unique *outcome* $\mathcal{P}_v^{\sigma,\pi}(\mathit{Reach}^{\leq t}(T))$, which is the probability of all $\alpha \in \mathit{Run}_{G(v,\sigma,\pi)}$ that visit T before time t (i.e., there is $i \in \mathbb{N}_0$ such that $\alpha(i)$ hits T and $\sum_{j=0}^{i-1} \alpha[j] \leq t$). The goal of player $\Box$ is to maximize the outcome, while player $\Diamond$ aims at the opposite. In our next definition we recall the standard concept of an equilibrium outcome called the *value*.

Definition 2.3. We say that a vertex $v \in V$ *has a value* if

$$\sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \mathcal{P}_v^{\sigma,\pi}(\mathit{Reach}^{\leq t}(T)) \quad = \quad \inf_{\pi \in \Pi} \sup_{\sigma \in \Sigma} \mathcal{P}_v^{\sigma,\pi}(\mathit{Reach}^{\leq t}(T))$$

If v has a value, then $\mathit{val}(v)$ denotes the *value of* v defined by the above equality.

The existence of $\mathit{val}(v)$ follows easily by applying the powerful result of Martin about weak determinacy of Blackwell games [11] (more precisely, one can use the determinacy result for stochastic games presented in [12] which builds on [11]). In Section 3, we give a self-contained proof of the existence of $\mathit{val}(v)$, which also brings further insights used later in our algorithms. Still, we think it is worth noting how the existence of $\mathit{val}(v)$ follows from the results of [11, 12] because the argument is generic and can be used also for more complicated timed objectives and a more general class of games over *semi-Markov processes* [4] where the distribution of time spent by performing a given action is not necessarily exponential.

Theorem 2.4. *Every vertex $v \in V$ has a value.*

Proof. Let us consider an infinite path of G initiated in v, i.e., an infinite sequence $v_0, a_0, v_1, a_1, \ldots$ where $v_0 = v$ and $a_i \in \mathbf{E}(v_i)$, $\mathbf{P}(a_i)(v_{i+1}) > 0$ for all

$i \in \mathbb{N}_0$. Let f be a real-valued function over infinite paths of G defined as follows:

- If a given path does not visit a target vertex (i.e., $v_i \notin T$ for all $i \in \mathbb{N}_0$), then f returns 0;
- otherwise, let $i \in \mathbb{N}_0$ be the least index such that $v_i \in T$. The function f returns the probability $\mathcal{P}(X_0 + \cdots + X_{i-1} \leq t)$ where every X_j, $0 \leq j < i$, is an exponentially distributed random variable with the rate $\mathbf{R}(a_j)$ (we assume that all X_j are mutually independent). Intuitively, f returns the probability that the considered path reaches v_i before time t.

Note that f is Borel measurable and bounded. Also note that every run in a play $G(v, \sigma, \pi)$ initiated in v determines exactly one infinite path in G (the time stamps are ignored). Hence, f determines a unique random variable over the runs in $G(v, \sigma, \pi)$ which is denoted by $f_v^{\sigma,\pi}$. Observe that $f_v^{\sigma,\pi}$ does not depend on the time stamps which appear in the runs of $G(v, \sigma, \pi)$, and hence we can apply the results of [12] and conclude that

$$\sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \mathbb{E}[f_v^{\sigma,\pi}] \quad = \quad \inf_{\pi \in \Pi} \sup_{\sigma \in \Sigma} \mathbb{E}[f_v^{\sigma,\pi}]$$

where $\mathbb{E}[f_v^{\sigma,\pi}]$ is the expected value of $f_v^{\sigma,\pi}$. To conclude the proof, it suffices to realize that $\mathcal{P}_v^{\sigma,\pi}(\mathit{Reach}^{\leq t}(T)) = \mathbb{E}[f_v^{\sigma,\pi}]$. □

Since values exist, it makes sense to define *ε-optimal* and *optimal* strategies.

Definition 2.5. Let $\varepsilon \geq 0$. We say that a strategy $\sigma \in \Sigma$ is an *ε-optimal maximizing strategy in v* (or just *ε-optimal in v*) if

$$\inf_{\pi \in \Pi} \mathcal{P}_v^{\sigma,\pi}(\mathit{Reach}^{\leq t}(T)) \;\geq\; \mathit{val}(v) - \varepsilon$$

and that a strategy $\pi \in \Pi$ is an *ε-optimal minimizing strategy in v* (or just *ε-optimal in v*) if

$$\sup_{\sigma \in \Sigma} \mathcal{P}_v^{\sigma,\pi}(\mathit{Reach}^{\leq t}(T)) \;\leq\; \mathit{val}(v) + \varepsilon$$

A strategy is *ε-optimal* if it is ε-optimal in every v. A strategy is *optimal in v* if it is 0-optimal in v, and just *optimal* if it is optimal in every v.

3. The Existence of Values and Optimal Strategies

In this section we first give a self-contained proof that every vertex in a CTG with time-bounded reachability objectives has a value (Theorem 3.6). The argument does not require any additional restrictions, i.e., it works also for CTGs with infinite state-space and infinite branching degree. As we shall see, the ideas presented in the proof of Theorem 3.6 are useful also for designing an algorithm which for a given $\varepsilon > 0$ computes ε-optimal strategies for both

players. Then, we study the existence of optimal strategies. We show that even though optimal minimizing strategies may not exist in infinitely-branching CTGs, they always exist in finitely-branching ones. As for optimal maximizing strategies, we show that they do not necessarily exist even in finitely-branching CTGs, but they are guaranteed to exist if a game is both finitely-branching and has bounded rates (see Definition 2.2).

For the rest of this section, we fix a CTG $G = (V, A, \mathbf{E}, (V_\Box, V_\Diamond), \mathbf{P}, \mathbf{R})$, a set $T \subseteq V$ of target vertices, and a time bound $t > 0$. Given $\mathbf{i} \in \mathcal{H}$ where $|\mathbf{i}| > 0$, we denote by $F_{\mathbf{i}}$ the probability distribution function of the random variable $X_{\mathbf{i}} = \sum_{r \in \mathcal{R}} \sum_{i=1}^{\mathbf{i}(r)} X_i^{(r)}$ where all $X_i^{(r)}$ are mutually independent and each $X_i^{(r)}$ is an exponentially distributed random variable with the rate r (for reader's convenience, basic properties of exponentially distributed random variables are recalled in Appendix A). We also define $F_{\mathbf{0}}$ as a constant function returning 1 for every argument (here $\mathbf{0} \in \mathcal{H}$ is the empty history, i.e., $|\mathbf{0}| = 0$). In the special case when $\mathcal{R}$ is a singleton, we use F_ℓ to denote $F_{\mathbf{i}}$ such that $\mathbf{i}(r) = \ell$, where r is the only element of $\mathcal{R}$. Further, given $\sim \in \{<, \leq, =\}$ and $k \in \mathbb{N}$, we denote by $\mathcal{P}_v^{\sigma,\pi}(\mathit{Reach}_{\sim k}^{\leq t}(T))$ the probability of all $\alpha \in \mathit{Run}_{G(v,\sigma,\pi)}$ that visit T for the first time in the number of steps satisfying the constraint $\sim k$ and before time t (i.e., there is $i \in \mathbb{N}_0$ such that $i = \min\{j \mid \alpha(j) \text{ hits } T\} \sim k$ and $\sum_{j=0}^{i-1} \alpha[j] \leq t$).

We first restate Theorem 2.4 and give its constructive proof.

Theorem 3.6. *Every vertex $v \in V$ has a value.*

Proof. Given $\sigma \in \Sigma$, $\pi \in \Pi$, $\mathbf{j} \in \mathcal{H}$, and $u \in V$, we denote by $P^{\sigma,\pi}(u, \mathbf{j})$ the probability of all runs $\alpha \in \mathit{Run}_{G(u,\sigma,\pi)}$ such that for some $n \in \mathbb{N}_0$ the state $\alpha(n)$ hits T and matches $\mathbf{j}$, and for all $0 \leq j < n$ we have that $\alpha(j)$ does not hit T. Then we introduce two functions $\mathcal{A}, \mathcal{B} : \mathcal{H} \times V \to [0, 1]$ where

$$\mathcal{A}(\mathbf{i}, v) = \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \sum_{\mathbf{j} \in \mathcal{H}} F_{\mathbf{i}+\mathbf{j}}(t) \cdot P^{\sigma,\pi}(v, \mathbf{j})$$

$$\mathcal{B}(\mathbf{i}, v) = \inf_{\pi \in \Pi} \sup_{\sigma \in \Sigma} \sum_{\mathbf{j} \in \mathcal{H}} F_{\mathbf{i}+\mathbf{j}}(t) \cdot P^{\sigma,\pi}(v, \mathbf{j})$$

Clearly, it suffices to prove that $\mathcal{A} = \mathcal{B}$, because then for every vertex $v \in V$ we also have that $\mathcal{A}(\mathbf{0}, v) = \mathcal{B}(\mathbf{0}, v) = \mathit{val}(v)$. The equality $\mathcal{A} = \mathcal{B}$ is obtained by demonstrating that both $\mathcal{A}$ and $\mathcal{B}$ are equal to the (unique) least fixed point of a monotonic function $\mathcal{V} : (\mathcal{H} \times V \to [0, 1]) \to (\mathcal{H} \times V \to [0, 1])$ defined as follows: for every $H : \mathcal{H} \times V \to [0, 1]$, $\mathbf{i} \in \mathcal{H}$, and $v \in V$ we have that

$$\mathcal{V}(H)(\mathbf{i}, v) = \begin{cases} F_{\mathbf{i}}(t) & v \in T \\ \sup_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot H(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) & v \in V_\Box \setminus T \\ \inf_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot H(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) & v \in V_\Diamond \setminus T \end{cases}$$

Let us denote by $\mu\mathcal{V}$ the least fixed point of $\mathcal{V}$. We show that $\mu\mathcal{V} = \mathcal{A} = \mathcal{B}$. The inequality $\mathcal{A} \preceq \mathcal{B}$ (where $\preceq$ is the standard pointwise order) is obvious and follows directly from the definition of $\mathcal{A}$ and $\mathcal{B}$. Hence, it suffices to prove the following two assertions:

1. By the following claim we obtain $\mu\mathcal{V} \preceq \mathcal{A}$.

   Claim 3.7. $\mathcal{A}$ is a fixed point of $\mathcal{V}$.

2. For every $\varepsilon > 0$ there is a CD strategy $\pi_\varepsilon \in \Pi$ such that for every $\mathbf{i} \in \mathcal{H}$ and every $v \in V$ we have that

$$\sup_{\sigma \in \Sigma} \sum_{\mathbf{j} \in \mathcal{H}} F_{\mathbf{i}+\mathbf{j}}(t) \cdot P^{\sigma,\pi_\varepsilon}(v, \mathbf{j}) \quad \leq \quad \mu\mathcal{V}(\mathbf{i}, v) + \varepsilon$$

   from which we get $\mathcal{B} \preceq \mu\mathcal{V}$.

   The strategy π_ε can be defined as follows. Given $\mathbf{i} \in \mathcal{H}$ and $v \in V_\Diamond$, we put $\pi_\varepsilon(\mathbf{i}, v)(a) = 1$ for some $a \in A$ satisfying $\sum_{u \in V} \mathbf{P}(a)(u) \cdot \mu\mathcal{V}(\mathbf{i}+\mathbf{1}_{\mathbf{R}(a)}, u) \leq \mu\mathcal{V}(\mathbf{i}, v) + \frac{\varepsilon}{2^{|\mathbf{i}|+1}}$. We prove that π_ε indeed satisfies the above equality. For every $\sigma \in \Sigma$, every $\mathbf{i} \in \mathcal{H}$, every $v \in V$ and every $k \geq 0$, we denote

$$\mathcal{R}^\sigma_k(\mathbf{i}, v) := \sum_{\substack{\mathbf{j} \in \mathcal{H} \\ |\mathbf{j}| \leq k}} F_{\mathbf{i}+\mathbf{j}}(t) \cdot P^{\sigma,\pi_\varepsilon[\mathbf{i}]}(v, \mathbf{j})$$

   Here $\pi_\varepsilon[\mathbf{i}]$ is the strategy obtained from π_ε by $\pi_\varepsilon[\mathbf{i}](\mathbf{j}, u) := \pi_\varepsilon(\mathbf{i}+\mathbf{j}, u)$.

   The following claim then implies that $\mathcal{R}^\sigma(\mathbf{i}, v) := \lim_{k\to\infty} \mathcal{R}^\sigma_k(\mathbf{i}, v) \leq \mu\mathcal{V}(\mathbf{i}, v) + \varepsilon$.

   Claim 3.8. For every $\sigma \in \Sigma$, $k \geq 0$, $\mathbf{i} \in \mathcal{H}$, $v \in V$, $\varepsilon \geq 0$, we have

$$\mathcal{R}^\sigma_k(\mathbf{i}, v) \leq \mu\mathcal{V}(\mathbf{i}, v) + \sum_{j=1}^{k} \frac{\varepsilon}{2^{|\mathbf{i}|+j}}$$

Both Claim 3.7 and 3.8 are purely technical, for proofs see Appendix C.1. □

It follows directly from Definition 2.3 and Theorem 3.6 that both players have ε-optimal strategies in every vertex v (for every $\varepsilon > 0$). Now we examine the existence of *optimal* strategies. We start by observing that optimal strategies do not necessarily exist in general.

Observation 3.9. *Optimal minimizing and optimal maximizing strategies in continuous-time games with time-bounded reachability objectives do not necessarily exist, even if we restrict ourselves to games with finitely many rates (i.e., $\mathcal{R}(G)$ is finite) and finitely many distinct transition probabilities.*

Proof. Consider a game $G = (V, A, \mathbf{E}, (V_\Box, V_\Diamond), \mathbf{P}, \mathbf{R})$, where $V = \{v_i \mid i \in \mathbb{N}_0\} \cup \{start, down\}$, $A = \{a_i, b_i \mid i \in \mathbb{N}\} \cup \{c, d\}$, $\mathbf{E}(start) = \{a_i \mid i \in \mathbb{N}\}$, $\mathbf{E}(v_i) = \{b_i\}$ for all $i \in \mathbb{N}$, $\mathbf{E}(v_0) = \{c\}$, $\mathbf{E}(down) = \{d\}$, $\mathbf{P}(a_i)(v_i) = 1$, $\mathbf{P}(c)(v_0) = 1$, $\mathbf{P}(d)(down) = 1$, and $\mathbf{P}(b_i)$ is the uniform distribution that chooses *down* and v_{i-1} for all $i \in \mathbb{N}$, and $\mathbf{R}$ assigns 1 to every action. The structure of G is shown in Figure 1 (the partition of V into $(V_\Box, V_\Diamond)$ is not fixed yet, and the vertices are therefore drawn as ovals). If we put $V_\Box = V$, we obtain that $\sup_{\sigma \in \Sigma} \mathcal{P}^{\sigma,\pi}_{start}(Reach^{\leq 1}(\{down\})) = \sum_{\ell=1}^{\infty} \left(\frac{1}{2^\ell} F_{\ell+1}(1)\right)$ where π

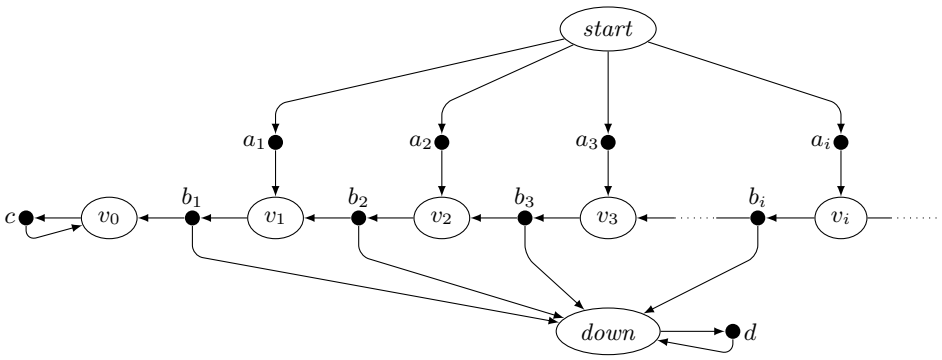


Figure 1: Optimal strategies may not exist.

is the trivial strategy for player $\Diamond$. However, there is obviously no optimal maximizing strategy. On the other hand, if we put $V_\Diamond = V$, we have that $\inf_{\pi\in\Pi} \mathcal{P}^{\sigma,\pi}_{start}(Reach^{\leq 1}(\{v_0\})) = 0$ where σ is the trivial strategy for player $\Box$, but there is no optimal minimizing strategy. $\Box$

However, if G is finitely-branching, then the existence of an optimal minimizing CD strategy can be established by adapting the construction used in the proof of Theorem 3.6.

Theorem 3.10. *If G is finitely-branching, then there is an optimal minimizing CD strategy.*

Proof. It suffices to reconsider the second assertion of the proof of Theorem 3.6. Since G is finitely-branching, the infima over enabled actions in the definition of $\mathcal{V}$ are actually minima. Hence, in the definition of π_ε, we can set $\varepsilon = 0$ and pick actions yielding minimal values. Thus the strategy π_ε becomes an optimal minimizing CD strategy. $\Box$

Observe that for optimal minimizing strategies we did not require that G has bounded rates. The issue with optimal maximizing strategies is slightly more complicated. First, we observe that optimal maximizing strategies do not necessarily exist even in finitely-branching games.

Observation 3.11. *Optimal maximizing strategies in continuous-time games with time-bounded reachability objectives may not exist, even if we restrict ourselves to finitely-branching games.*

Proof. Consider a game $G = (V, A, \mathbf{E}, (V_\Box, V_\Diamond), \mathbf{P}, \mathbf{R})$, where $V = V_\Box = \{v_i, u_i \mid i \in \mathbb{N}_0\} \cup \{win, lose\}$; $A = \{a_i, b_i, end_i \mid i \in \mathbb{N}_0\} \cup \{w, \ell\}$, $\mathbf{E}(win) = \{w\}$, $\mathbf{E}(lose) = \{\ell\}$, and $\mathbf{E}(v_i) = \{a_i, b_i\}$, $\mathbf{E}(u_i) = \{end_i\}$ for all $i \in \mathbb{N}_0$; $\mathbf{R}(w) = \mathbf{R}(\ell) = 1$, and $\mathbf{R}(a_i) = \mathbf{R}(b_i) = 2^i$, $\mathbf{R}(end_i) = 2^{i+1}$ for all $i \in \mathbb{N}_0$; $\mathbf{P}(w)(win) = \mathbf{P}(\ell)(lose) = 1$, and for all $i \in \mathbb{N}_0$ we have that $\mathbf{P}(a_i)(v_{i+1}) = 1$, $\mathbf{P}(b_i)(u_i) = 1$,

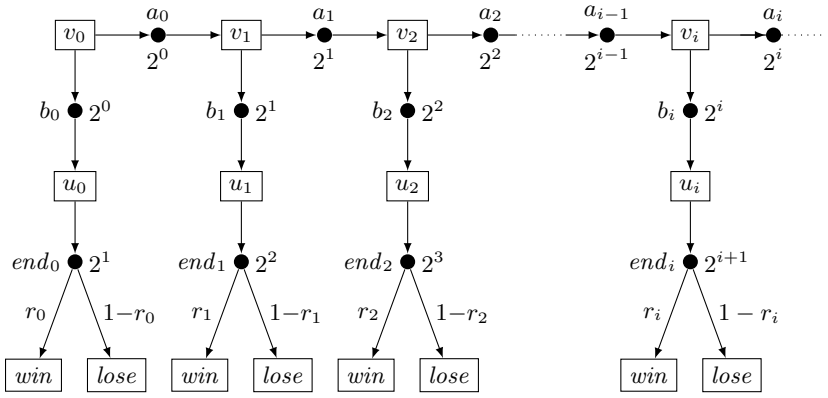


Figure 2: Optimal maximizing strategies may not exist in finitely-branching games.

and $\mathbf{P}(end_i)$ is the distribution that assigns r_i to *win* and $1-r_i$ to *lose*, where r_i is the number discussed below. The structure of G is shown in Figure 2 (note that for clarity, the vertices *win* and *lose* are drawn multiple times, and their only enabled actions w and ℓ are not shown).

For every $k \in \mathbb{N}$, let $\mathbf{i}_k \in \mathcal{H}$ be the vector that assigns 1 to all $r \in \mathcal{R}$ such that $r \leq 2^k$, and 0 to all other rates. Let us fix $t \in \mathbb{Q}$ and $q > \frac{1}{2}$ such that $F_{\mathbf{i}_k}(t) \geq q$ for every $k \in \mathbb{N}$. Note that such t and q exist because the mean value associated to $F_{\mathbf{i}_k}$ is $\sum_{i=0}^{k} 1/2^i < 2$ and hence it suffices to apply Markov inequality. For every $j \geq 0$, we fix some $r_j \in \mathbb{Q}$ such that $q - \frac{1}{2^j} \leq F_{\mathbf{i}_{j+1}}(t) \cdot r_j \leq q - \frac{1}{2^{j+1}}$. It is easy to check that $r_j \in [0,1]$, which means that the function $\mathbf{P}$ is well-defined.

We claim that $\sup_{\sigma \in \Sigma} \mathcal{P}^{\sigma,\pi}_{v_0}(\mathit{Reach}^{\leq t}(\{\mathit{win}\})) = q$ (where π is the trivial strategy for player $\Diamond$), but there is no strategy σ such that $\mathcal{P}^{\sigma,\pi}_{v_0}(\mathit{Reach}^{\leq t}(\{\mathit{win}\})) = q$. The first part follows by observing that player $\Box$ can reach *win* within time t with probability at least $q - \frac{1}{2^j}$ for an arbitrarily large j by selecting the actions $a_0, \ldots, a_{j-1}$ and then b_j. The second part follows from the fact that by using b_j, the probability of reaching *win* from v_0 becomes strictly less than q, and by not selecting b_j at all, this probability becomes equal to 0. $\square$

Observe that again the counterexample is a CTMDP. Now we show that if G is finitely-branching *and* has bounded rates, then there is an optimal maximizing CD strategy. First, observe that for each $k \in \mathbb{N}_0$

$$\sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \mathcal{P}^{\sigma,\pi}_{v}(\mathit{Reach}^{\leq t}_{\leq k}(T)) \;=\; \inf_{\pi \in \Pi} \sup_{\sigma \in \Sigma} \mathcal{P}^{\sigma,\pi}_{v}(\mathit{Reach}^{\leq t}_{\leq k}(T)) \;=\; \mathcal{V}^{k+1}(\mathit{zero})(\mathbf{0}, v) \tag{1}$$

where $\mathcal{V}$ is the function defined in the proof of Theorem 3.6, $\mathit{zero} : \mathcal{H} \times V \to [0,1]$ is a constant function returning zero for every argument, and $\mathbf{0}$ is the empty history. A proof of Equality 1 is obtained by a straightforward induction on k. We use $\mathit{val}^k(v)$ to denote the

k-step value defined by Equality 1, and we say that strategies $\sigma_k \in \Sigma$ and $\pi_k \in \Pi$ are *k-step optimal* if for all $v \in V$, $\pi \in \Pi$, and $\sigma \in \Sigma$ we have $\inf_{\pi \in \Pi} \mathcal{P}_v^{\sigma_k,\pi}(Reach_{\leq k}^{\leq t}(T)) = \sup_{\sigma \in \Sigma} \mathcal{P}_v^{\sigma,\pi_k}(Reach_{\leq k}^{\leq t}(T)) = val^k(v)$. The existence and basic properties of k-step optimal strategies are stated in our next lemma.

Lemma 3.12. *If G is finitely-branching and has bounded rates, then we have the following:*

1. *For all $\varepsilon > 0$, $k \geq (\sup \mathcal{R})te^2 - \ln \varepsilon$, $\sigma \in \Sigma$, $\pi \in \Pi$, and $v \in V$ we have that*

$$\mathcal{P}_v^{\sigma,\pi}(Reach^{\leq t}(T)) - \varepsilon \;\leq\; \mathcal{P}_v^{\sigma,\pi}(Reach_{\leq k}^{\leq t}(T)) \;\leq\; \mathcal{P}_v^{\sigma,\pi}(Reach^{\leq t}(T))$$

2. *For every $k \in \mathbb{N}$, there are k-step optimal BCD strategies $\sigma_k \in \Sigma$ and $\pi_k \in \Pi$. Further, for all $\varepsilon > 0$ and $k \geq (\sup \mathcal{R})te^2 - \ln \varepsilon$ we have that every k-step optimal strategy is also an ε-optimal strategy.*

Proof. See Appendix C.2. □

If G is finitely-branching and has bounded rates, one may be tempted to construct an optimal maximizing strategy σ by selecting those actions that are selected by infinitely many k-step optimal BCD strategies for all $k \in \mathbb{N}$ (these strategies are guaranteed to exist by Lemma 3.12 (2)). However, this is not so straightforward, because the distributions assigned to actions in finitely-branching games can still have an infinite support. Intuitively, this issue is overcome by considering larger and larger finite subsets of the support so that the total probability of all of the infinitely many omitted elements approaches zero. Hence, a proof of the following theorem is somewhat technical.

Theorem 3.13. *If G is finitely-branching and has bounded rates, then there is an optimal maximizing CD strategy.*

Proof. For the sake of this proof, given a set of runs $R \subseteq Run_{G(\hat{v},\sigma,\pi)}$, we denote $\mathcal{P}_{\hat{v}}^{\sigma,\pi}(R)$ the probability of R in $G(\hat{v}, \sigma, \pi)$. For every $k \in \mathbb{N}$ we fix a k-step optimal BCD strategy σ_k of player $\square$ (see Lemma 3.12). Let us order the set $\mathcal{R}$ of rates into an enumerable sequence $r_1, r_2 \ldots$ and the set V into an enumerable sequence $v_1, v_2 \ldots$. We define a sequence of sets of strategies $\Sigma \supseteq \Gamma_0 \supseteq \Gamma_1 \supseteq \cdots$ as follows. We put $\Gamma_0 = \{\sigma_\ell \mid \ell \in \mathbb{N}\}$ and we construct Γ_ℓ to be an infinite subset of $\Gamma_{\ell-1}$ such that we have $\sigma(\mathbf{i}, v_n) = \sigma'(\mathbf{i}, v_n)$ for all $\sigma, \sigma' \in \Gamma_\ell$, all $n \leq \ell$ and all $\mathbf{i} \in \mathcal{H}$ such that $|\mathbf{i}| \leq \ell$ and $\mathbf{i}(r_j) = 0$ whenever $j > \ell$. Note that such a set exists since $\Gamma_{\ell-1}$ is infinite and the conditions above partition it into finitely many classes, one of which must be infinite.

Now we define the optimal strategy σ. Let $\mathbf{i} \in \mathcal{H}$ and $v_n \in V$, we choose a number ℓ such that $\ell > |\mathbf{i}|$, $\ell > n$ and $\mathbf{i}(j) = 0$ for all $j > \ell$ (note that such ℓ exists for any $\mathbf{i} \in \mathcal{H}$ and $v_n \in V$). We put $\sigma(\mathbf{i}, v_n) = \sigma'(\mathbf{i}, v_n)$ where $\sigma' \in \Gamma_\ell$. It is easy to see that σ is a CD strategy, it remains to argue that it is optimal. Suppose the converse, i.e. that it is not ε-optimal in some v_{in} for some $\varepsilon > 0$.

Let us fix k satisfying conditions of part 1 of Lemma 3.12 for $\frac{\varepsilon}{4}$. For each $a \in A$ there is a set $B_a \subseteq V$ such that $V \setminus B_a$ is finite and $\mathbf{P}(a)(B_a) \leq \frac{\varepsilon}{4k}$. For all strategies σ' and π' and all k we have that $\mathcal{P}_v^{\sigma',\pi'}(U_i^{v,\sigma',\pi'}) \leq \frac{\varepsilon}{2k}$ where $U_i^{v,\sigma',\pi'}$ is the set of all runs of $G(v,\sigma',\pi')$ that *do not* contain any state of the form $v_0 a_0 \dots a_{i-1} v_i a_i$ where $v_i \in B_{a_{i-1}}$. As a consequence we have $\mathcal{P}_v^{\sigma',\pi'}(\bigcap_{i=0}^{k} U_i^{v,\sigma',\pi'}) \leq \frac{\varepsilon}{4}$. In the sequel, we denote $U^{v,\sigma',\pi'} = \bigcap_{i=0}^{k} U_i^{v,\sigma',\pi'}$ and we write just U instead of $U^{v,\sigma',\pi'}$ if v, σ and π are clear from the context.

Let W be the set of histories of the form $v_0 a_0 \dots v_{i-1} a_{i-1} v_i$ where $i \leq k$, $v_0 = v_{in}$, and for all $0 \leq j < i$ we have $a_j \in \mathbf{E}(v_j)$, $\mathbf{P}(a_j)(v_{i+j}) > 0$ and $v_{j+1} \notin B_{a_j}$. We claim that there is $m \geq n$ s.t. σ_m is $\frac{\varepsilon}{4}$-optimal and satisfies $\sigma(w) = \sigma_m(w)$ for all $w \in W$. To see that such a strategy exists, observe that W is finite, which means that there is a number ℓ such that $k \leq \ell$ and for all $w \in W$, there is no v_i in w such that $i > \ell$ and whenever a is in w, then $\mathbf{R}(a) = r_i$ for $i < \ell$. Now it suffices to choose arbitrary $\frac{\varepsilon}{4}$-optimal strategy $\sigma_m \in \Gamma_\ell$.

One can prove by induction on the length of path from v_{in} to T that the following equality holds true for all π.

$$\mathcal{P}_{v_{in}}^{\sigma_m,\pi}(Reach_{\leq k}^{\leq t}(T) \setminus U) = \mathcal{P}_{v_{in}}^{\sigma,\pi}(Reach_{\leq k}^{\leq t}(T) \setminus U)$$

Finally, we obtain

$$\begin{aligned}
\min_{\pi\in\Pi} \mathcal{P}_{v_{in}}^{\sigma,\pi}(Reach_{\leq k}^{\leq t}(T) \setminus U) &= \min_{\pi\in\Pi} \mathcal{P}_{v_{in}}^{\sigma_m,\pi}(Reach_{\leq k}^{\leq t}(T) \setminus U) \\
&\geq \min_{\pi\in\Pi} \mathcal{P}_{v_{in}}^{\sigma_m,\pi}(Reach_{\leq m}^{\leq t}(T) \setminus U) - \frac{\varepsilon}{4} \\
&\geq \min_{\pi\in\Pi} \mathcal{P}_{v_{in}}^{\sigma_m,\pi}(Reach_{\leq m}^{\leq t}(T)) - \frac{\varepsilon}{2} \\
&\geq val(v_{in}) - \frac{\varepsilon}{4} - \frac{\varepsilon}{2} \geq val(v_{in}) - \varepsilon
\end{aligned}$$

which means that σ is ε-optimal in v_{in}. □

4. Optimal Strategies in Finite Uniform CTGs

In this section, we restrict ourselves to finite uniform CTGs, i.e. $\mathbf{R}(a) = r > 0$ for all $a \in A$. The histories from $\mathcal{H}$ are thus vectors of length 1, hence we write them as integers. We prove that both players have *optimal BCD strategies* in such games. More precisely, we prove a stronger statement that there are optimal strategies that after some number of steps eventually behave in a stationary way. A CD strategy τ is *stationary* if $\tau(h,v)$ depends just on v for every vertex v. Besides, for a CD strategy τ, a strategy $\tau[h]$ is defined by $\tau[h](h',v) = \tau(h+h',v)$. Further, recall that *bp* is the maximum bit-length of the fractional representation of transition probabilities.

Theorem 4.14. *In a finite uniform CTG, there exist optimal CD strategies $\sigma \in \Sigma$, $\pi \in \Pi$ and $k \in \mathbb{N}$ such that $\sigma[k]$ and $\pi[k]$ are stationary; in particular, σ*

and π are optimal BCD strategies. Moreover, if all transition probabilities are rational then one can choose $k = rt(1 + 2^{bp\cdot|A|^2\cdot|V|^3})$.

We then also show that this result is tight in the sense that optimal BCD strategies do not necessarily exist in uniform CTGs with infinitely many states even if the branching is finite (see Observation 4.23). In Section 5, we use these results to design an algorithm which *computes* the optimal BCD strategies in finite uniform games. Further, using the method of uniformization where a general game is reduced to a uniform one, the results can be extended to general (not necessarilly uniform) finite games, see [9].

Before proving the theorem we note that the crucial point is to understand the behaviour of optimal strategies after many (i.e. k) steps have already been taken. In such a situation, not much time is left and it turns out that in such a situation optimal strategies optimize the probability of reaching T in as few steps as possible. This motivates the central definition of *greedy* strategies. Intuitively, a greedy strategy optimizes the outcome of the first step. If there are more options to do so, it chooses among these options so that it optimizes the second step, etc.

Definition 4.15. For strategies $\sigma \in \Sigma$ and $\pi \in \Pi$ and a vertex v, we define a *step reachability vector* $\overrightarrow{\mathcal{P}}_v^{\sigma,\pi} = \left(\mathcal{P}_v^{\sigma,\pi}(Reach_{=i}^{\leq\infty}(T))\right)_{i\in\mathbb{N}_0}$. A strategy $\sigma \in \Sigma$ is *greedy* if for every v, $\min_{\pi\in\Pi} \overrightarrow{\mathcal{P}}_v^{\sigma,\pi} = \max_{\sigma'\in\Sigma} \min_{\pi\in\Pi} \overrightarrow{\mathcal{P}}_v^{\sigma',\pi}$ where the optima[3] are considered in the lexicographical order. Similarly, a strategy $\pi \in \Pi$ is *greedy* if $\max_{\sigma\in\Sigma} \overrightarrow{\mathcal{P}}_v^{\sigma,\pi} = \min_{\pi'\in\Pi} \max_{\sigma\in\Sigma} \overrightarrow{\mathcal{P}}_v^{\sigma,\pi'}$ for every v.

We prove the theorem as follows:

1. Optimal CD strategies are guaranteed to exist by Theorem 3.10 and Theorem 3.13.

2. For every optimal CD strategy τ, the strategy $\tau[k]$ is *greedy* (see Proposition 4.16).

3. There exist *stationary* greedy strategies (see Proposition 4.21). Let τ_g be such a strategy. Then for an optimal strategy τ, the strategy $\bar{\tau}$ defined by

$$\bar{\tau}(h, v) = \begin{cases} \tau(h, v) & \text{if } h < k; \\ \tau_g(h, v) & \text{otherwise} \end{cases}$$

   is clearly BCD and also optimal. Indeed, all greedy strategies guarantee the same probabilities to reach the target. (This is clear by definition, since their step reachability vectors are the same.) Therefore, we can freely interchange them without affecting the guaranteed outcome.

[3] We can use optima instead of extrema as the optimal strategies obviously exist in finite discrete-time (with the time bound being infinite) games even when the number of steps is fixed.

Proposition 4.16. *Let τ be an optimal strategy. Then there is $k \in \mathbb{N}$ such that $\tau[k]$ is greedy. Moreover, if all transition probabilities are rational then one can choose $k = rt(1 + 2^{bp\cdot|A|^2\cdot|V|^3})$.*

In order to prove the proposition, we relax our definition of greedy strategies. A strategy is *greedy on s steps* if the greedy strategy condition holds for the step reachability vector where only first s elements are considered. A strategy τ is *always greedy on s steps* if for all $i \in \mathbb{N}_0$ the strategy $\tau[i]$ is greedy on s steps. We use this relaxation of greediness to prove the proposition as follows. We firstly prove that every optimal strategy is always greedy on $|\mathbf{E}| := \sum_{v\in V} |\mathbf{E}(v)|$ steps (by instantiating Lemma 4.17 for $s = |\mathbf{E}| \leq |A| \cdot |V|$) and then Lemma 4.18 concludes by proving that being always greedy on $|\mathbf{E}|$ steps guarantees greediness.

Lemma 4.17. *For every $s \in \mathbb{N}$ there is $\delta > 0$ such that for every optimal CD strategy τ the strategy $\tau[rt(1 + 1/\delta)]$ is always greedy on s steps. Moreover, if all transition probabilities are rational, then one can choose $\delta = 1/2^{bp\cdot|V|\cdot|\mathbf{E}|\cdot s}$.*

Proof. We look for a δ such that for every optimal strategy $\sigma \in \Sigma$ if $\sigma[h]$ is not greedy on s steps then $h < k$, where $k = rt(1 + 1/\delta)$. Let thus σ be an optimal CD strategy and $s \in \mathbb{N}$. For $\sigma[h]$ that is not greedy on s steps there is $i \leq s$, a vertex v and a strategy σ^* such that

$$\big(\inf_{\pi\in\Pi} \overrightarrow{\mathcal{P}}_v^{\sigma[h],\pi}\big)_i < \big(\inf_{\pi\in\Pi} \overrightarrow{\mathcal{P}}_v^{\sigma^*,\pi}\big)_i$$

and for all $j < i$

$$\big(\inf_{\pi\in\Pi} \overrightarrow{\mathcal{P}}_v^{\sigma[h],\pi}\big)_j = \big(\inf_{\pi\in\Pi} \overrightarrow{\mathcal{P}}_v^{\sigma^*,\pi}\big)_j$$

This implies that there is $i \leq s$ such that $\inf_{\pi\in\Pi} \mathcal{P}_v^{\sigma^*,\pi}(Reach_{\leq i}^{\leq\infty}(T)) - \inf_{\pi\in\Pi} \mathcal{P}_v^{\sigma[h],\pi}(Reach_{\leq i}^{\leq\infty}(T))$ is positive. Since the game is finite there is a fixed $\delta > 0$ such that difference of this form is (whenever it is non-zero) greater than δ for all deterministic strategies σ and σ^*, $v \in V$ and $i \leq s$. Moreover, if all transition probabilities are rational, then δ can be chosen to be $1/M^s$, where M is the least common multiple of all probabilities denominators. Indeed, $\mathcal{P}_v^{\sigma,\tau}(Reach_{\leq i}^{\leq\infty}(T))$ is clearly expressible as ℓ/M^i for some $\ell \in \mathbb{N}_0$. Since there are at most $|V| \cdot |\mathbf{E}|$ probabilities, we have $\delta \geq 1/2^{bp\cdot|V|\cdot|\mathbf{E}|\cdot s}$.

We define a (not necessarily counting) strategy $\bar{\sigma}$ that behaves like σ, but when h steps have been taken and v is reached, it behaves as σ^*. We show that for $h \geq k$ this strategy $\bar{\sigma}$ would be an improvement against the optimal strategy σ. There is clearly an improvement at the $h + i$th step provided one gets there on time, and this improvement is at least δ. Nonetheless, in the next steps there may be an arbitrary decline. Altogether due to optimality of σ

$$\begin{aligned}
&0 \geq \inf_{\pi\in\Pi} \mathcal{P}_{\hat{v}}^{\bar{\sigma},\pi}(Reach^{\leq t}(T)) - \inf_{\pi\in\Pi} \mathcal{P}_{\hat{v}}^{\sigma,\pi}(Reach^{\leq t}(T)) \geq \\
&\geq \inf_{\pi\in\Pi} \mathcal{P}_{\hat{v}}^{\sigma,\pi}(\xrightarrow{h} v) \cdot \Big[F_{h+i}(t) \cdot \delta - F_{h+i+1}(t) \cdot 1\Big] = (*)
\end{aligned}$$

where $\mathcal{P}^{\sigma,\pi}_{\hat{v}}(\xrightarrow{h} v)$ is the probability that after h steps we will be in v. We need to show that the inequality $0 \geq (*)$ implies $h < k$. We use the following key argument that after taking sufficiently many steps, the probability of taking strictly *more* than one step before the time limit is negligible compared to the probability of taking *precisely* one more step, i.e. that for all $n \geq k = rt(1+1/\delta)$ we have

$$\frac{F_{n+1}(t)}{F_n(t)} < \frac{F_{n+1}(t)}{F_n(t) - F_{n+1}(t)} < \delta$$

As $F_{n+1}(t) = \sum_{i=1}^{\infty} e^{-r\cdot t}(rt)^{n+i}/(n+i)!$, this is proved by the following:

$$\frac{F_{n+1}(t)}{F_n(t) - F_{n+1}(t)} = \sum_{i=1}^{\infty} \frac{n!(rt)^i}{(n+i)!} < \sum_{i=1}^{\infty} \frac{(rt)^i}{(n+1)^i} = \frac{rt}{n+1-rt} < \delta$$

This argument thus implies $h + i < k$, hence we conclude that indeed $h < k$. The minimizing part is dual. □

The following lemma concludes the proof of the proposition.

Lemma 4.18. *A strategy is greedy iff it is always greedy on $|\mathbf{E}|$ steps.*

Proof. We need to focus on the structure of greedy strategies. Therefore, we provide their inductive characterization. Moreover, this characterization can be easily turned into an algorithm computing all greedy strategies.

W.l.o.g. let us assume that all states in T are absorbing, i.e. the only transitions leading from them are self-loops.

Algorithm 1 computes which actions can be chosen in greedy strategies. We begin with the original game and keep on pruning inoptimal transitions until we reach a fix-point. In the first iteration, we compute the value $R_1(v)$ for each vertex v, which is the optimal probability of reaching T in one step. We remove all transitions that are not optimal in this sense. In the next iteration, we consider reachability in precisely two steps. Note that we chose among the one-step optimal possibilities only. Transitions not optimal for two-steps reachability are removed and so forth. After stabilization, using only the remaining "greedy" transitions thus results in greedy behavior.

Claim 4.19. A strategy is always greedy on s steps iff it uses transitions from $\mathbf{E}_s$ only (as defined by Algorithm 1).

In particular, a strategy τ is always greedy on $|\mathbf{E}|$ steps iff it uses transitions from $\mathbf{E}_{|\mathbf{E}|}$ only. For the proof of Claim 4.19 see Appendix C.3.

Claim 4.20. A strategy is greedy iff it uses transitions from $\mathbf{E}_{|\mathbf{E}|}$ only.

The proof of Claim 4.20 now follows easily. Since the number of edges is finite, there is a fix-point $\mathbf{E}_n = \mathbf{E}_{n+1}$, moreover, $n \leq |\mathbf{E}|$. Therefore, any strategy using $\mathbf{E}_{|\mathbf{E}|}$ only is by Claim 4.19 always greedy on s steps for all $s \in \mathbb{N}_0$, hence clearly greedy. On the other hand, every greedy strategy is in particular always greedy on $|\mathbf{E}|$ steps and thus uses transitions from $\mathbf{E}_{|\mathbf{E}|}$ only again by Claim 4.19. This concludes the proof of the Lemma and thus also of Proposition 4.16. □

Algorithm 1 computing all greedy edges

$$R_0(v) = \begin{cases} 1 & \text{if } v \in T, \\ 0 & \text{otherwise.} \end{cases}$$
$$\mathbf{E}_0(v) = \mathbf{E}(v)$$

$$R_{i+1}(a) = \sum_{u \in V} \mathbf{P}(a)(u) \cdot R_i(u)$$
$$R_{i+1}(v) = \begin{cases} \max_{a \in \mathbf{E}_i(v)} R_{i+1}(a) & \text{if } v \in V_\Box, \\ \min_{a \in \mathbf{E}_i(v)} R_{i+1}(a) & \text{otherwise.} \end{cases}$$
$$\mathbf{E}_{i+1}(v) = \mathbf{E}_i(v) \cap \{a \mid R_{i+1}(a) = R_{i+1}(v)\}$$

We now move on to Proposition 4.21 that concludes the proof of the theorem.

Proposition 4.21. *There are greedy stationary strategies $\sigma_g \in \Sigma$ and $\pi_g \in \Pi$. Moreover, the strategies σ_g and π_g are computable in polynomial time.*

Proof. The complexity of Algorithm 1 is polynomial in the size of the game graph as the fix-point is reached within $|\mathbf{E}|$ steps. And as there is always a transition enabled in each vertex (the last one is trivially optimal), we can choose one transition in each vertex arbitrarily and thus get a greedy strategy (by Claim 4.20) that is stationary. □

Corollary 4.22. *In a finite uniform game with rational transition probabilities, there are optimal strategies τ such that $\tau[rt(1 + 2^{bp \cdot |A|^2 \cdot |V|^3})]$ is a greedy stationary strategy.*

A natural question is whether Theorem 4.14 and Corollary 4.22 can be extended to infinite-state uniform CTGs. The question is answered in our next observation.

Observation 4.23. *Optimal BCD strategies do not necessarily exist in uniform infinite-state CTGs, even if they are finitely-branching and use only finitely many distinct transition probabilities.*

Proof. Consider a game $G = (V, A, \mathbf{E}, (V_\Box, V_\Diamond), \mathbf{P}, \mathbf{R})$ where $V = V_\Box = \{v_i, u_i, \bar{u}_i, \hat{u}_i \mid i \in \mathbb{N}_0\} \cup \{down\}$, $A = \{a_i, hat_i, bar_i, \hat{b}_i, \bar{b}_i \mid i \in \mathbb{N}_0\}$, $\mathbf{E}(v_i) = \{a_i\}$, $\mathbf{E}(u_i) = \{bar_i, hat_i\}$, $\mathbf{E}(\hat{u}_i) = \{\hat{b}_i\}$, and $\mathbf{E}(\bar{u}_i) = \{\bar{b}_i\}$ for all $i \in \mathbb{N}_0$. $\mathbf{P}$ is defined as follows:

- $\mathbf{P}(a_0)$ is the uniform distribution on $\{v_0, v_1, u_0\}$, $\mathbf{P}(a_i)$ is the uniform distribution on $\{u_i, v_{i+1}\}$ for $i > 0$,
- $\mathbf{P}(hat_i)(\hat{u}_i) = 1$ and $\mathbf{P}(bar_i)(\bar{u}_i) = 1$ for $i \geq 0$,

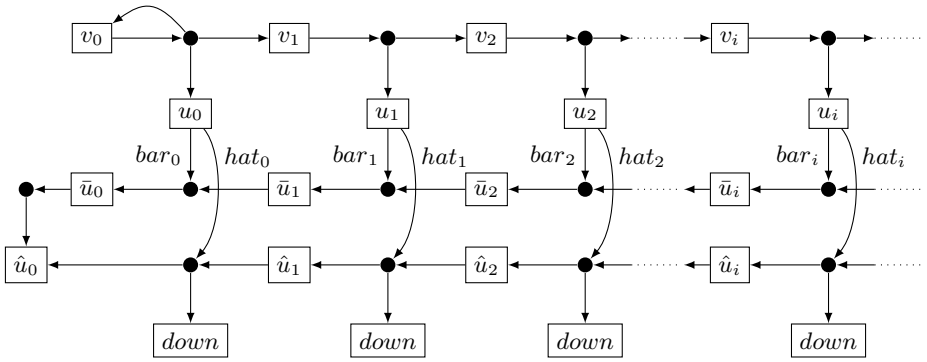


Figure 3: Optimal BCD strategies may not exist in infinite uniform games.

- $\mathbf{P}(\bar{b}_0)(\hat{u}_0) = 1$, and $\mathbf{P}(\bar{b}_i)(\bar{u}_{i-1}) = 1$ for $i > 0$,
- $\mathbf{P}(\hat{b}_i)$ is the uniform distribution on $\{\hat{u}_{i-1}, down\}$ for $i \geq 1$.

We set $\mathbf{R}(a) = 1$ for all $a \in A$. The structure of G is shown in Figure 3. Observe that player $\Box$ has a real choice only in states u_i.

We show that if the history is of the form $v_0a_0v_1a_1 \dots v_ia_iu_i$ (where $i \in \mathbb{N}_0$), the optimal strategy w.r.t. reaching $\hat{u}_0$ within time $t = 1$ must choose the action bar_i. We need to show that $F_{2i+3}(1) > \frac{1}{2^{i+2}} \cdot F_{2i+2}(1)$, i.e. that $\frac{F_{2i+2}(1)}{F_{2i+3}(1)} < 2^{i+2}$, for infinitely many is. This follows by observing that for $i > 0$

$$\frac{F_{2i+2}(1)}{F_{2i+3}(1)} = \frac{\sum_{j=2i+2}^{\infty} \frac{1}{j!}}{\sum_{j=2i+3}^{\infty} \frac{1}{j!}} < \frac{\sum_{j=2i+2}^{\infty} \frac{1}{j!}}{\frac{1}{(2i+3)!}} < (2i+3) + \sum_{k=0}^{\infty} \frac{1}{(2i+3)^k} < 2i + 5 < 2^{i+2}$$

On the other hand, from Lemma 4.17 one can deduce that for all i there is $j \geq i$ such that any optimal strategy must choose hat_i if the history is of the form $(v_0a_0)^j v_1a_1 \dots v_ia_iu_i$. Thus no strategy with counting bounded by $k \in \mathbb{N}$ can be optimal as one can choose $j \geq i > k$.

□

5. Algorithms

Now we present algorithms which compute ε-optimal BCD strategies in finitely-branching CTGs with bounded rates and optimal BCD strategies in finite uniform CTGs. In this section, we assume that all rates and distributions used in the considered CTGs are *rational*.

5.1. Computing ε-optimal BCD strategies

For this subsection, let us fix a CTG $G = (V, A, \mathbf{E}, (V_\Box, V_\Diamond), \mathbf{P}, \mathbf{R})$, a set $T \subseteq V$ of target vertices, a time bound $t > 0$, and some $\varepsilon > 0$. For simplicity, let us

Algorithm 2 Compute the function C

{1st phase: compute the approximations of $F_{\mathbf{i}}(t)$ and $\mathbf{P}$}
for all vectors $\mathbf{i} \in \mathcal{H}$, where $|\mathbf{i}| \leq k$ **do**
  compute a number $\ell_{\mathbf{i}}(t) > 0$ such that $\frac{|F_{\mathbf{i}}(t)-\ell_{\mathbf{i}}(t)|}{F_{\mathbf{i}}(t)} \leq \left(\frac{\varepsilon}{2}\right)^{2|\mathbf{i}|+1}$.
for all actions $a \in A$ and vertices $u \in V$ **do**
  compute a floating point representation $\mathbf{p}(a)(u)$ of $\mathbf{P}(a)(u)$ satisfying $\frac{|\mathbf{P}(a)(u)-\mathbf{p}(a)(u)|}{\mathbf{P}(a)(u)} \leq \left(\frac{\varepsilon}{2}\right)^{2k+1}$.
{2nd phase: compute the functions R and C in a bottom up manner}
for all vector lenghts j from k down to 0 **do**
  for all vectors $\mathbf{i} \in \mathcal{H}$ of length $|\mathbf{i}| = j$ **do**
    for all vertices $v \in V$ **do**
      if $v \in T$ **then**
        $R(\mathbf{i}, v) \leftarrow \ell_{\mathbf{i}}(t)$
      else if $|\mathbf{i}| = k$ **then**
        $R(\mathbf{i}, v) \leftarrow 0$
      else if $v \in V_{\Box}$ **then**
        $R(\mathbf{i}, v) \leftarrow \max_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{p}(a)(u) \cdot R(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u)$
        $C(\mathbf{i}, v) \leftarrow a$ where a is the action that realizes the maximum above
      else if $v \in V_{\Diamond}$ **then**
        $R(\mathbf{i}, v) \leftarrow \min_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{p}(a)(u) \cdot R(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u)$
        $C(\mathbf{i}, v) \leftarrow a$ where a is the action that realizes the minimum above

first assume that G is finite; as we shall see, our algorithm does not really depend on this assumption, as long as the game is finitely-branching, has bounded rates, and its structure can be effectively generated (see Corollary 5.26). Let $k = (\max \mathcal{R})te^2 - \ln(\frac{\varepsilon}{2})$. Then, due to Lemma 3.12, all k-step optimal strategies are $\frac{\varepsilon}{2}$-optimal. We use the remaining $\frac{\varepsilon}{2}$ for numerical imprecisions.

We need to specify the ε-optimal BCD strategies $\sigma_\varepsilon \in \Sigma$ and $\pi_\varepsilon \in \Pi$ on the first k steps. For every $\mathbf{i} \in \mathcal{H}$, where $|\mathbf{i}| < k$, and for every $v \in V$, our algorithm computes an action $C(\mathbf{i}, v) \in \mathbf{E}(v)$ which represents the choice of the constructed strategies. That is, for every $\mathbf{i} \in \mathcal{H}$, where $|\mathbf{i}| < k$, and for every $v \in V_{\Box}$, we put $\sigma_\varepsilon(\mathbf{i}, v)(C(\mathbf{i}, v)) = 1$, and for the other arguments we define σ_ε arbitrarily so that σ_ε remains a BCD strategy. The strategy π_ε is induced by the function C in the same way.

The procedure to compute the function C is described in Algorithm 2. For computing $C(\mathbf{i}, v)$ it uses a family of probabilities $R(\mathbf{i}, u)$ of reaching T from u before time t in at most $k - |\mathbf{i}|$ steps using the strategies σ_ε and π_ε (precisely, using the parts of strategies σ_ε and π_ε computed so far) and assuming that the history matches $\mathbf{i}$. Actually, our algorithm computes the probabilities $R(\mathbf{i}, u)$ only up to a sufficiently small error so that the actions chosen by C are "sufficiently optimal" (i.e., the strategies σ_ε and π_ε are ε-optimal, but they are not necessarily k-step optimal for the k chosen above).

Lemma 5.24. *The strategies σ_ε and π_ε are ε-optimal.*

Proof. See Appendix C.4. □

Assuming that the probabilities $\mathbf{P}(a)(u)$ and rates are given as fractions with both numerator and denominator represented in binary with length bounded by bp, a complexity analysis of the algorithm reveals the following.

Theorem 5.25. *Assume that G is finite. Then for every $\varepsilon > 0$ there are ε-optimal BCD strategies $\sigma_\varepsilon \in \Sigma$ and $\pi_\varepsilon \in \Pi$ computable in time $|V|^2 \cdot |A| \cdot bp^2 \cdot \left((\max \mathcal{R}) \cdot t + \ln \frac{1}{\varepsilon}\right)^{2|\mathcal{R}|+\mathcal{O}(1)}$.*

Proof. We analyze the complexity of Algorithm 2. We start with 1st phase. Recall that $k = (\max \mathcal{R})te^2 + \ln \frac{1}{\varepsilon}$. (Here we use ε instead of $\varepsilon/2$ as this difference is clearly absorbed in the $\mathcal{O}$-notation.)

We approximate the value of $F_{\mathbf{i}}(t)$ to the relative precision $(\varepsilon/2)^{2k+1}$ as follows. According to [13], the value of $F_{\mathbf{i}}(t)$ is expressible as $\sum_{r \in \mathcal{R}} q_r e^{-rt}$. First, q_r here is a polynomial in t and can be precisely computed using polynomially many (in $|\mathbf{i}| \leq k$ and $|\mathcal{R}|$) arithmetical operations on integers with length bounded by $bp + \ln k + \ln t$. Hence the computation of q_r as a fraction can be done in time $bp^2 \cdot k^{\mathcal{O}(1)} \cdot |\mathcal{R}|^{\mathcal{O}(1)}$ and both the numerator and the denominator are of length $bp \cdot k^{\mathcal{O}(1)} \cdot |\mathcal{R}|^{\mathcal{O}(1)}$. We approximate this fraction with a floating point representation with relative error $(\varepsilon/4)^{2k+1}$. This can be done in linear time w.r.t. the length of the fraction and $k \ln \frac{1}{\varepsilon}$, hence again in the time $bp^2 \cdot k^{\mathcal{O}(1)} \cdot |\mathcal{R}|^{\mathcal{O}(1)}$. Secondly, according to [14], the floating point approximation of e^{-rt} with the relative error $(\varepsilon/4)^{2k+1}$ can be computed in time less than quadratic in $k \ln \frac{1}{\varepsilon}$. Altogether, we can compute an $(\varepsilon/2)^{2k+1}$-approximation of each $F_{\mathbf{i}}(t)$ in time $|\mathcal{R}| \cdot bp^2 \cdot k^{\mathcal{O}(1)} \cdot |\mathcal{R}|^{\mathcal{O}(1)} = bp^2 \cdot k^{\mathcal{O}(1)} \cdot |\mathcal{R}|^{\mathcal{O}(1)}$. This procedure has to be repeated for every $\mathbf{i} \in \mathcal{H}$, where $|\mathbf{i}| \leq k$. The number of such $\mathbf{i}$'s is bounded by $\binom{|\mathcal{R}|+k}{k} \leq \mathcal{O}(k^{|\mathcal{R}|})$. So computing all $(\varepsilon/2)^{2k+1}$-approximations $\ell_{\mathbf{i}}(t)$ of values $F_{\mathbf{i}}(t)$ can be done in time $\mathcal{O}(k^{|\mathcal{R}|}) \cdot bp^2 \cdot k^{\mathcal{O}(1)} \cdot |\mathcal{R}|^{\mathcal{O}(1)} \subseteq bp^2 \cdot k^{|\mathcal{R}|+\mathcal{O}(1)}$.

Using a similar procedure as above, for every $a \in A$ and $u \in V$, we compute the floating point approximation $\mathbf{p}(a)(u)$ of $\mathbf{P}(a)(u)$ to the relative precision $(\varepsilon/2)^{2k+1}$ in time linear in $bp \cdot k \ln \frac{1}{\varepsilon}$. So the first phase takes time $bp^2 \cdot k^{|\mathcal{R}|+\mathcal{O}(1)} + |A| \cdot |V| \cdot \mathcal{O}(bp \cdot k \ln \frac{1}{\varepsilon}) \subseteq |V| \cdot |A| \cdot bp^2 \cdot k^{|\mathcal{R}|+\mathcal{O}(1)}$.

In 2nd phase, the algorithm computes the table R and outputs the results into the table C. The complexity is thus determined by the product of the table size and the time to compute one item in the table. The size of the tables is $\binom{|\mathcal{R}|+k}{k} \cdot |V| \leq \mathcal{O}(k^{|\mathcal{R}|} \cdot |V|)$.

The value of $R(\mathbf{i}, u)$ according to the first case has already been computed in 1st phase. To compute the value according to the third or fourth case we have to compare numbers whose representation has at most $bp^2 \cdot k^{|\mathcal{R}|+\mathcal{O}(1)} + k \cdot bp \cdot k \ln(\frac{1}{\varepsilon})$ bits. To compute $R(\mathbf{i}, v)$, we need to compare $|A|$ such sums of $|V|$ numbers. So the 2nd phase takes at most time $\mathcal{O}(k^{|\mathcal{R}|} \cdot |V|) \cdot |V| \cdot |A| \cdot bp^2 \cdot k^{|\mathcal{R}|+\mathcal{O}(1)} \subseteq |V|^2 \cdot |A| \cdot bp^2 \cdot k^{2|\mathcal{R}|+\mathcal{O}(1)}$.

Altogether, the overall time complexity of Algorithm 2 is bounded by

$$|V|^2 \cdot |A| \cdot bp^2 \cdot k^{2|\mathcal{R}|+\mathcal{O}(1)} = |V|^2 \cdot |A| \cdot bp^2 \cdot \left((\max \mathcal{R})t + \ln \frac{1}{\varepsilon}\right)^{2|\mathcal{R}|+\mathcal{O}(1)} \qquad \square$$

Note that our algorithm needs to analyze only a finite part of G. Hence, it also works for infinite games which satisfy the conditions formulated in the next corollary.

Corollary 5.26. *Let G be a finitely-branching game with bounded rates and let $v \in V$. Assume that the vertices and actions of G reachable from v in a given finite number of steps are effectively computable, and that an upper bound on rates is also effectively computable. Then for every $\varepsilon > 0$ there are effectively computable BCD strategies $\sigma_\varepsilon \in \Sigma$ and $\pi_\varepsilon \in \Pi$ that are ε-optimal in v.*

Proof. By Lemma 3.12, there is $k \in \mathbb{N}$ such that all k-step optimal strategies are $\frac{\varepsilon}{4}$-optimal. Thus we may safely restrict the set of vertices of the game G to the set V_{reach} of vertices reachable from v in at most k steps (i.e. for all $v' \in V_{reach}$ there is a sequence $v_0 \dots v_k \in V^*$ and $a_0 \dots a_k \in A^*$ such that, $v_0 = v$, $v_k = v'$, $a_i \in \mathbf{E}(v_i)$ for all $0 \leq i \leq k$ and $\mathbf{P}(a_i)(v_{i+1}) > 0$ for all $0 \leq i < k$). Moreover, for every action $a \in A$ which is enabled in a vertex of V_{reach} there is a finite set B_a of vertices such that $1 - \sum_{u \in B_a} \mathbf{P}(a)(u) < \frac{\varepsilon}{4k}$. We restrict the domain of $\mathbf{P}(a)$ to B_a by assigning the probability 0 to all vertices of $V \setminus B_a$ and adding the probability $1 - \sum_{u \in B_a} \mathbf{P}(a)(u)$ to an arbitrary vertex of B_a. Finally, we restrict the set of vertices once more to the vertices reachable in k steps from v using the restricted $\mathbf{P}$. Then the resulting game is finite and by Theorem 5.25 there is an $\frac{\varepsilon}{4}$-optimal BCD strategy σ' in this game. Now it suffices to extend σ' to a BCD strategy σ in the original game by defining, arbitrarily, its values for vertices and actions removed by the above procedure. It is easy to see that σ is an ε-optimal BCD strategy in G. □

5.2. Computing optimal BCD strategies in uniform finite games

For the rest of this subsection, we fix a finite uniform CTG $G = (V, A, \mathbf{E}, (V_\Box, V_\Diamond), \mathbf{P}, \mathbf{R})$ where $\mathbf{R}(a) = r > 0$ for all $a \in A$. Let $k = rt(1 + 2^{bp \cdot |A|^2 \cdot |V|^3})$ (see Corollary 4.22).

The algorithm works similarly as the one of Section 5.1, but there are also some differences. Since we have just one rate, the vector $\mathbf{i}$ becomes just a number i. Similarly as in Section 5.1, our algorithm computes an action $C(i, v) \in \mathbf{E}(v)$ representing the choice of the constructed optimal BCD strategies $\sigma_{max} \in \Sigma$ and $\pi_{min} \in \Pi$. By Corollary 4.22, every optimal strategy can, from the k-th step on, start to behave as a fixed greedy stationary strategy, and we can compute such a greedy stationary strategy in polynomial time. Hence, the optimal BCD strategies σ_{max} and π_{min} are defined as follows:

$$\sigma_{max}(i, v) = \begin{cases} C(i, v) & \text{if } i < k; \\ \sigma_g(v) & \text{otherwise.} \end{cases} \qquad \pi_{min}(i, v) = \begin{cases} C(i, v) & \text{if } i < k; \\ \pi_g(v) & \text{otherwise.} \end{cases}$$

To compute the function C, our algorithm uses a table of symbolic representations of the (precise) probabilities $R(i, v)$ (here $i \leq k$ and $v \in V$) of reaching T from v before time t in at most $k - i$ steps using the strategies σ_{max} and π_{min} and assuming that the history matches i.

The function C and the family of all $R(i,v)$ are computed (in a bottom up fashion) as follows: For all $0 \leq i \leq k$ and $v \in V$ we have that

$$R(i,v) = \begin{cases} F_i(t) & \text{if } v \in T \\ \sum_{j=0}^{\infty} F_{i+j}(t) \cdot \mathcal{P}_v^{\sigma_g,\pi_g}(Reach_{=j}^{<\infty}(T)) & \text{if } v \notin T \text{ and } i = k \\ \max_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot R(i+1,u) & \text{if } v \in V_\Box \setminus T \text{ and } i < k \\ \min_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot R(i+1,u) & \text{if } v \in V_\Diamond \setminus T \text{ and } i < k \end{cases}$$

For all $i < k$ and $v \in V$, we put $C(i,v) = a$ where a is an action maximizing or minimizing $\sum_{u \in V} \mathbf{P}(a)(u) \cdot R(i+1,u)$, depending on whether $v \in V_\Box$ or $v \in V_\Diamond$, respectively. The effectivity of computing such an action (this issue is not trivial) is discussed in the proof of the following theorem.

Theorem 5.27. *The BCD strategies σ_{max} and π_{min} are optimal and effectively computable.*

Proof. We start by showing that σ_{max} and π_{min} are optimal. Let us denote by Σ_g (resp. Π_g) the set of all CD strategies $\sigma \in \Sigma$ (resp. $\pi \in \Pi$) such that for all $u \in V_\Box$ ($u \in V_\Diamond$) and $i \geq k$ we have $\sigma(i,u) = \sigma_g(u)$, which is a stationary greedy strategy. By Corollary 4.22, for every $v \in V$ we have

$$val(v) = \max_{\sigma \in \Sigma_g} \min_{\pi \in \Pi_g} \mathcal{P}_v^{\sigma,\pi}(Reach^{\leq t}(T)) = \min_{\pi \in \Pi_g} \max_{\sigma \in \Sigma_g} \mathcal{P}_v^{\sigma,\pi}(Reach^{\leq t}(T))$$

Recall that given a CD strategy τ and $i \geq 0$, we denote by $\tau[i]$ a strategy obtained from τ by $\tau[i](j,u) = \tau(i+j,u)$. Let us denote

$$\bar{P}^{\sigma,\pi}(i,v) = \sum_{j=0}^{\infty} F_{i+j}(t) \cdot \mathcal{P}_v^{\sigma[i],\pi[i]}(Reach_{=j}^{<\infty}(T))$$

For every $i \geq 0$ we put

$$val(i,v) = \max_{\sigma \in \Sigma_g} \min_{\pi \in \Pi_g} \bar{P}^{\sigma,\pi}(i,v) = \min_{\pi \in \Pi_g} \max_{\sigma \in \Sigma_g} \bar{P}^{\sigma,\pi}(i,v)$$

Given $i \geq 0$ and $\pi \in \Pi$, we define

$$\bar{K}^{\pi}(i,v) := \bar{P}^{\sigma_{max},\pi}(i,v)$$

Similarly, given $i \in \mathcal{H}$ and $\sigma \in \Sigma$, we define

$$\bar{K}^{\sigma}(i,v) := \bar{P}^{\sigma,\pi_{min}}(i,v)$$

Using this fomulation, the optimality of σ_{max} and π_{min} is proven in the following claim.

Claim 5.28. Let $i \leq k$ and $v \in V$. We have

$$\min_{\pi \in \Pi_g} \bar{K}^{\pi}(i,v) = R(i,v) = \max_{\sigma \in \Sigma_g} \bar{K}^{\sigma}(i,v) \tag{2}$$

$$R(i,v) = val(i,v) \tag{3}$$

Proof. We start by proving the equation (2). If $v \in T$, then $\bar{K}^{\pi}(i,v) = \bar{K}^{\sigma}(i,v) = F_i(t) = R(i,v)$. Assume that $v \notin T$. We proceed by induction on $n = k - i$. For $n = 0$ we have

$$\bar{K}^{\pi}(i,v) = \bar{K}^{\sigma}(i,v) = \bar{P}^{\sigma_g,\pi_g}(i,v) = R(i,v)$$

Assume the claim holds true for n and consider $n+1$. If $v \in V_{\Box}$ and $\sigma_{max}(\mathbf{i},v)(b) = 1$,

$$\begin{aligned}
\min_{\pi\in\Pi_g} \bar{K}^{\pi}(i,v) &= \min_{\pi\in\Pi_g} \sum_{u\in V} \mathbf{P}(b)(u)\cdot \bar{K}^{\pi}(i+1,u) \\
&= \sum_{u\in V} \mathbf{P}(b)(u)\cdot \min_{\pi\in\Pi_g} \bar{K}^{\pi}(i+1,u) \\
&= \sum_{u\in V} \mathbf{P}(b)(u)\cdot R(i+1,u) \\
&= \max_{a\in\mathbf{E}(v)} \sum_{u\in V} \mathbf{P}(a)(u)\cdot R(i+1,u) \\
&= R(i,v)
\end{aligned}$$

and

$$\begin{aligned}
\max_{\sigma\in\Sigma_g} \bar{K}^{\sigma}(i,v) &= \max_{\sigma\in\Sigma_g} \sum_{a\in\mathbf{E}(v)} \sigma(i,v)(a) \sum_{u\in V} \mathbf{P}(a)(u)\cdot \bar{K}^{\sigma}(i+1,u) \\
&= \max_{a\in\mathbf{E}(v)} \sum_{u\in V} \mathbf{P}(a)(u)\cdot \max_{\sigma\in\Sigma_g} \bar{K}^{\sigma}(i+1,u) \\
&= \max_{a\in\mathbf{E}(v)} \sum_{u\in V} \mathbf{P}(a)(u)\cdot R(i+1,u) \\
&= R(i,v)
\end{aligned}$$

For $u \in V_{\Diamond}$ the proof is similar.

Now the equation (3) follows easily:

$$\begin{aligned}
R(i,v) = \min_{\pi\in\Pi_g} \bar{K}^{\pi}(i,v) \leq \max_{\sigma\in\Sigma_g} \min_{\pi\in\Pi_g} \bar{P}^{\sigma,\pi}(i,v) = \\
\min_{\pi\in\Pi_g} \max_{\sigma\in\Sigma_g} \bar{P}^{\sigma,\pi}(i,v) \leq \max_{\sigma\in\Sigma_g} \bar{K}^{\sigma}(i,v) = R(i,v)
\end{aligned}$$

□

This proves that σ_{max} and π_{min} are optimal.

Effective computability of σ_{max} and π_{min}. We show how to compute the table $C(i,v)$. Assume that we have already computed the symbolic representations of the values $R(i+1,u)$ for all $u \in V$. Later we show that

$\sum_{j=0}^{\infty} F_{i+j}(t) \cdot \mathcal{P}_v^{\sigma_g,\pi_g}(Reach_{=j}^{<\infty}(T))$ can effectively be expressed as a linear combination of transcendental numbers of the form e^{ct} where c is algebraic. Therefore, each difference of the compared numbers can effectively be expressed as a finite sum $\sum_j \eta_j e^{\delta_j}$ where the η_j and δ_j are algebraic numbers and the δ_j's are pairwise distinct. Now it suffices to apply Lemma 2 of [15] to decide whether the difference is greater than 0, or not.

It remains to show that $\sum_{j=0}^{\infty} F_{i+j}(t) \cdot \mathcal{P}_v^{\sigma_g,\pi_g}(Reach_{=j}^{<\infty}(T))$ is effectively expressible in the form $\sum_j \eta_j e^{\delta_j}$. Consider a game G' obtained from G by adding new vertices $v_1, \ldots, v_i$ and new actions $a_1, \ldots, a_i$, setting $\mathbf{E}(v_j) = \{a_j\}$ for $1 \leq j \leq i$, and setting $\mathbf{P}(a_i)(v) = 1$, and $\mathbf{P}(a_j)(v_{j+1}) = 1$ for $1 \leq j < i$ (intuitively, we have just added a simple path of length i from a new vertex v_1 to v). We put $\mathbf{R}(a_j) = r$ for $1 \leq j \leq i$. As the strategies σ_g and π_g are stationary, they can be used in G' (we just make them select a_j in v_j).

Since $v_j \notin T$ for all $1 \leq j \leq i$ we obtain

$$\mathcal{P}_{v_1}^{\sigma_g,\pi_g}(Reach^{\leq t}(T)) = \sum_{j=0}^{\infty} F_j(t) \cdot \mathcal{P}_{v_1}^{\sigma_g,\pi_g}(Reach_{=j}^{<\infty}(T)) =$$

$$\sum_{j=0}^{\infty} F_{i+j}(t) \cdot \mathcal{P}_{v_1}^{\sigma_g,\pi_g}(Reach_{=i+j}^{<\infty}(T)) = \sum_{j=0}^{\infty} F_{i+j}(t) \cdot \mathcal{P}_v^{\sigma_g,\pi_g}(Reach_{=j}^{<\infty}(T))$$

As σ_g and π_g are stationary, the chain $G'(v_1, \sigma_g, \pi_g)$ can be treated as a finite continuous time Markov chain. Therefore we may apply results of [13] and obtain the desired form of $\mathcal{P}_{v_1}^{\sigma_g,\pi_g}(Reach^{\leq t}(T))$, and hence also of $\sum_{j=0}^{\infty} F_{i+j}(t) \cdot \mathcal{P}_v^{\sigma_g,\pi_g}(Reach_{=j}^{<\infty}(T))$. □

6. Conclusions, Future Work

We have shown that vertices in CTGs with time bounded reachability objectives have a value, and we classified the subclasses of CTGs where a given player has an optimal strategy. We also proved that in finite uniform CTGs, both players have optimal BCD strategies. Finally, we designed algorithms which compute ε-optimal BCD strategies in finitely-branching CTGs with bounded rates, and optimal BCD strategies in finite uniform CTGs.

There are at least two interesting directions for future research. First, we can consider more general classes of strategies that depend on the elapsed time (in our setting, strategies are time-abstract). In [6], it is demonstrated that time-dependent strategies are more powerful (i.e., can achieve better results) than the time-abstract ones. However, this issue is somewhat subtle—in [7], it is shown that the power of time-dependent strategies is different when the player knows only the total elapsed time, the time consumed by the last action, or the complete timed history of a play. The analog of Theorem 3.6 in this setting is examined in [16]. In [17] ε-optimal time-dependent strategies are computed for CTMDPs. Second, a generalization to semi-Markov processes and games, where arbitrary (not only exponential) distributions are considered, would be desirable.

[1] W. Thomas, Infinite games and verification, in: Proceedings of CAV 2003, Vol. 2725 of LNCS, Springer, 2003, pp. 58–64.

[2] E. Grädel, W. Thomas, T. Wilke, Automata, Logics, and Infinite Games, no. 2500 in LNCS, Springer, 2002.

[3] I. Walukiewicz, A landscape with games in the background, in: Proceedings of LICS 2004, IEEE, 2004, pp. 356–366.

[4] M. Puterman, Markov Decision Processes, Wiley, 1994.

[5] J. Filar, K. Vrieze, Competitive Markov Decision Processes, Springer, 1996.

[6] C. Baier, H. Hermanns, J.-P. Katoen, B. Haverkort, Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes, TCS 345 (2005) 2–26.

[7] M. Neuhäußer, M. Stoelinga, J.-P. Katoen, Delayed nondeterminism in continuous-time Markov decision processes, in: Proceedings of FoSSaCS 2009, Vol. 5504 of LNCS, Springer, 2009, pp. 364–379.

[8] C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen, Reachability in continuous-time Markov reward decision processes, in: E. Graedel, J. Flum, T. Wilke (Eds.), Logic and Automata: History and Perspectives, Vol. 2 of Texts in Logics and Games, Amsterdam University Press, 2008, pp. 53–72.

[9] M. Rabe, S. Schewe, Optimal time-abstract schedulers for CTMDPs and Markov games, in: A. D. Pierro, G. Norman (Eds.), QAPL, Vol. 28 of EPTCS, 2010, pp. 144–158.

[10] J. Norris, Markov Chains, Cambridge University Press, 1998.

[11] D. Martin, The determinacy of Blackwell games, JSL 63 (4) (1998) 1565–1581.

[12] A. Maitra, W. Sudderth, Finitely additive stochastic games with Borel measurable payoffs 27 (1998) 257–267.

[13] S. Amari, R. Misra, Closed-form expressions for distribution of sum of exponential random variables, IEEE transactions on reliability 46 (1997) 519–522.

[14] R. P. Brent, Fast multiple-precision evaluation of elementary functions, Journal of the ACM 23 (1976) 242–251.

[15] A. Aziz, K. Sanwal, V. Singhal, R. Brayton, Model-checking continuous-time Markov chains, ACM Trans. on Comp. Logic 1 (1) (2000) 162–170.

[16] M. Rabe, S. Schewe, Finite optimal control for time-bounded reachability in CTMDPs and continuous-time Markov games, CoRR abs/1004.4005.

[17] M. R. Neuhäußer, L. Zhang, Time-bounded reachability probabilities in continuous-time Markov decision processes, in: QEST, IEEE Computer Society, 2010, pp. 209–218.

Appendix A. Exponentially Distributed Random Variables

For reader's convenience, in this section we recall basic properties of exponentially distributed random variables.

A *random variable* over a probability space $(\Omega, \mathcal{F}, \mathcal{P})$ is a function $X : \Omega \to \mathbb{R}$ such that the set $\{\omega \in \Omega \mid X(\omega) \leq c\}$ is measurable for every $c \in \mathbb{R}$. We usually write just $X{\sim}c$ to denote the set $\{\omega \in \Omega \mid X(\omega) \sim c\}$, where $\sim$ is a comparison and $c \in \mathbb{R}$. The *expected value* of X is defined by the Lebesgue integral $\int_{\omega\in\Omega} X(\omega)\, d\mathcal{P}$. A function $f : \mathbb{R} \to \mathbb{R}^{\geq 0}$ is a *density* of a random variable X if for every $c \in \mathbb{R}$ we have that $\mathcal{P}(X{\leq}c) = \int_{-\infty}^{c} f(x)\, dx$. If a random variable X has a density function f, then the expected value of X can also be computed by a (Riemann) integral $\int_{-\infty}^{\infty} x{\cdot}f(x)\, dx$. Random variables X, Y are *independent* if for all $c, d \in \mathbb{R}$ we have that $\mathcal{P}(X{\leq}c \cap Y{\leq}d) = \mathcal{P}(X{\leq}c){\cdot}\mathcal{P}(Y{\leq}d)$. If X and Y are independent random variables with density functions f_X and f_Y, then the random variable $X + Y$ (defined by $X + Y\,(\omega) = X(\omega) + Y(\omega)$) has a density function f which is the *convolution* of f_X and f_Y, i.e., $f(z) = \int_{-\infty}^{\infty} f_X(x) \cdot f_Y(z - x)\, dx$.

A random variable X has an *exponential distribution with rate* λ if $\mathcal{P}(X \leq c) = 1 - e^{-\lambda c}$ for every $c \in \mathbb{R}^{\geq 0}$. The density function f_X of X is then defined as $f_X(c) = \lambda e^{-\lambda c}$ for all $c \in \mathbb{R}^{\geq 0}$, and $f_X(c) = 0$ for all $c < 0$. The expected value of X is equal to $\int_{-\infty}^{\infty} x \cdot \lambda e^{-\lambda x} dx = 1/\lambda$.

Lemma A.29. *Let $\mathcal{M} = (S, \mathbf{P}, \mathbf{R}, \mu)$ be a CTMC, $j \in \mathbb{N}_0$, $t \in \mathbb{R}^{\geq 0}$, and $u_0, \ldots, u_j \in S$. Let U be the set of all runs (u, s) where u starts with $u_0, \ldots, u_j$ and $\sum_{i=0}^{j} s_j \leq t$. We have that*

$$\mathcal{P}(U) = F_{\mathbf{i}}(t) \cdot \mu(u_0) \cdot \prod_{\ell=0}^{j-1} \mathbf{P}(u_\ell)(u_{\ell+1})$$

where $\mathbf{i}$ assigns to every rate r the cardinality of the set $\{k \mid \mathbf{R}(u_k) = r, 0 \leq k \leq j\}$

Proof. By induction on j. For $j = 0$ the lemma holds, because we $\mathcal{P}(U) = \mu(u_0)$ by definition.

Now suppose that $j > 0$ and the lemma holds for all $k < j$. We denote by $U_k^{t'}$ the set of all runs (u, s) where u starts with $u_0, \ldots, u_j$ and $\sum_{i=0}^{k} s_i = t'$. We have that

$$\begin{aligned}
\mathcal{P}(U) &= \int_0^t \mathcal{P}(U_{j-1}^x) \cdot \mathbf{P}(u_{j-1})(u_j) \cdot e^{-\mathbf{R}(u_{j-1})\cdot(t-x)}\, dx \\
&= \int_0^t F_{\mathbf{i}-\mathbf{1}_{\mathbf{R}(u_{j-1})}}(x) \cdot \left(\prod_{\ell=0}^{j-2} \mathbf{P}(u_\ell)(u_{\ell+1})\right) \mathbf{P}(u_{j-1})(u_j) \cdot e^{-\mathbf{R}(u_{j-1})\cdot(t-x)}\, dx \\
&= \prod_{\ell=0}^{j-1} \mathbf{P}(u_\ell)(u_{\ell+1}) \cdot \int_0^t F_{\mathbf{i}-\mathbf{1}_{\mathbf{R}(u_{j-1})}}(x) \cdot e^{-\mathbf{R}(u_{j-1})\cdot(t-x)}\, dx \\
&= F_{\mathbf{i}}(t) \cdot \prod_{\ell=0}^{j-1} \mathbf{P}(u_\ell)(u_{\ell+1})
\end{aligned}$$

□

Appendix B. A Comparison of the Existing Definitions of CTMDPs

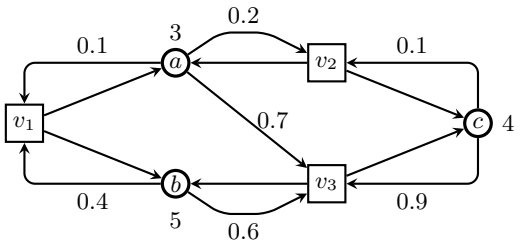
As we already mentioned in Section 2, our definition of CTG (and hence also CTMDP) is somewhat different from the definition of CTMDP used in [6, 7]. To prevent misunderstandings, we discuss the issue in greater detail in here and show that the two formalisms are in fact equivalent. First, let us recall the alternative definition CTMDP used in [6, 7].

Definition B.30. A CTMDP is a triple $\mathcal{M} = (S, A, \mathbf{R})$, where S is a finite or countably infinite set of *states*, A is a finite or countably infinite set of *actions*, and $\mathbf{R} : (S \times A \times S) \to \mathbb{R}^{\geq 0}$ is a *rate matrix*.

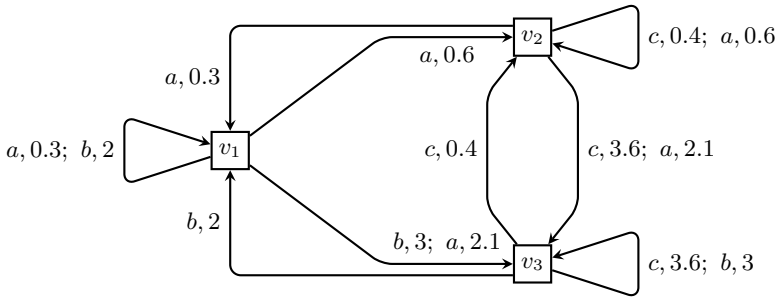
A CTMDP $\mathcal{M} = (S, A, \mathbf{R})$ can be depicted as a graph where S is the set of vertices and $s \to s'$ is an edge labeled by (a, r) iff $\mathbf{R}(s, a, s') = r > 0$. The conditional probability of selecting the edge $s \xrightarrow{(a,r)} s'$, under the condition that the action a is used, is defined as $r/\mathbf{R}(s, a)$, where $\mathbf{R}(s, a) = \sum_{s \xrightarrow{(a,\hat{r})} \hat{s}} \hat{r}$. The time needed to perform the action a in s is exponentially distributed with the rate $\mathbf{R}(s, a)$. This means that $\mathcal{M}$ can be translated into an equivalent CTG where the set of vertices is S, the set of actions is

$$\{(s, a) \mid s \in S, a \in A, \mathbf{R}(s, a, s') > 0 \text{ for some } s' \in S\}$$

where the rate of a given action (s, a) is $\mathbf{R}(s, a)$, and $\mathbf{P}((s, a))(s') = \mathbf{R}(s, a, s')/\mathbf{R}(s, a)$. This translation also works in the opposite direction (assuming that $V = V_\Box$ or $V = V_\Diamond$). To illustrate this, consider the following CTG:



An equivalent CTMDP (in the sense of Definition B.30) looks as follows:



However, there is one subtle issue regarding strategies. In [6, 7], a strategy (controller) selects an action in every vertex. The selection may depend on the history of a play. In [6, 7], it is noted that if a controller is deterministic, then the resulting play is a CTMC. If a controller is *randomized*, one has to add "intermediate" discrete-time states which implement the timeless randomized choice, and hence the resulting play is *not* a CTMC, but a mixture of discrete-time and continuous-time Markov chains. In our setting, this problem disappears, because the probability distribution chosen by a player is simply "multiplied" with the probabilities of outgoing edges of actions. For deterministic strategies, the two approaches are of course completely equivalent.

Appendix C. Technical Proofs

Appendix C.1. Proofs of Claim 3.7 and Claim 3.8

Claim 3.7. *$\mathcal{A}$ is a fixed point of $\mathcal{V}$.*

Proof. If $v \in T$, we have

$$\mathcal{A}(\mathbf{i}, v) = \sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} F_{\mathbf{i}}(t) = \mathcal{V}(\mathcal{A})(\mathbf{i}, v)$$

Assume that $v \notin T$. Given a strategy $\tau \in \Sigma \cup \Pi$ and $a \in A$, we denote by τ^a a strategy defined by $\tau^a(wu) := \tau(vawu)$. Note that $\sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} P^{\sigma,\pi}(\cdot,\cdot) = \sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} P^{\sigma^a,\pi^a}(\cdot,\cdot)$ for any $a \in A$.

If $v \in V_{\Box}$,

$$\begin{aligned}
\mathcal{V}(A)(\mathbf{i}, v) &= \sup_{a\in\mathbf{E}(v)} \sum_{u\in V} \mathbf{P}(a)(u) \cdot \sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} \sum_{\mathbf{j}\in\mathcal{H}} F_{\mathbf{i}+\mathbf{1}_{\mathbf{R}(a)}+\mathbf{j}}(t) \cdot P^{\sigma,\pi}(u, \mathbf{j}) \\
&= \sup_{d\in\mathcal{D}(\mathbf{E}(v))} \sum_{a\in A} d(a) \sum_{u\in V} \mathbf{P}(a)(u) \cdot \sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} \sum_{\mathbf{j}\in\mathcal{H}} F_{\mathbf{i}+\mathbf{1}_{\mathbf{R}(a)}+\mathbf{j}}(t) \cdot P^{\sigma,\pi}(u, \mathbf{j}) \\
&= \sup_{d\in\mathcal{D}(\mathbf{E}(v))} \sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} \sum_{a\in A} d(a) \sum_{u\in V} \mathbf{P}(a)(u) \sum_{\mathbf{j}\in\mathcal{H}} F_{\mathbf{i}+\mathbf{1}_{\mathbf{R}(a)}+\mathbf{j}}(t) \cdot P^{\sigma,\pi}(u, \mathbf{j}) \\
&= \sup_{d\in\mathcal{D}(\mathbf{E}(v))} \sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} \sum_{a\in A} d(a) \sum_{u\in V} \mathbf{P}(a)(u) \sum_{\substack{\mathbf{j}\in\mathcal{H} \\ \mathbf{j}(\mathbf{R}(a))>0}} F_{\mathbf{i}+\mathbf{j}}(t) \cdot P^{\sigma^a,\pi^a}(u, \mathbf{j} - \mathbf{1}_{\mathbf{R}(a)}) \\
&= \sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} \sum_{a\in A} \sigma(v)(a) \sum_{u\in V} \mathbf{P}(a)(u) \sum_{\substack{\mathbf{j}\in\mathcal{H} \\ \mathbf{j}(\mathbf{R}(a))>0}} F_{\mathbf{i}+\mathbf{j}}(t) \cdot P^{\sigma^a,\pi^a}(u, \mathbf{j} - \mathbf{1}_{\mathbf{R}(a)}) \\
&= \sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} \sum_{a\in A} \sum_{\substack{\mathbf{j}\in\mathcal{H} \\ \mathbf{j}(\mathbf{R}(a))>0}} F_{\mathbf{i}+\mathbf{j}}(t) \cdot \sigma(v)(a) \sum_{u\in V} \mathbf{P}(a)(u) \cdot P^{\sigma^a,\pi^a}(u, \mathbf{j} - \mathbf{1}_{\mathbf{R}(a)}) \\
&= \sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} \sum_{\mathbf{j}\in\mathcal{H}} F_{\mathbf{i}+\mathbf{j}}(t) \sum_{\substack{a\in A \\ \mathbf{j}(\mathbf{R}(a))>0}} \sigma(v)(a) \sum_{u\in V} \mathbf{P}(a)(u) \cdot P^{\sigma^a,\pi^a}(u, \mathbf{j} - \mathbf{1}_{\mathbf{R}(a)}) \\
&= \sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} \sum_{\mathbf{j}\in\mathcal{H}} F_{\mathbf{i}+\mathbf{j}}(t) P^{\sigma,\pi}(v, \mathbf{j})
\end{aligned}$$

If $v \in V_\diamondsuit$,

$$\begin{aligned}
\mathcal{V}(A)(\mathbf{i}, v) &= \inf_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \sum_{\mathbf{j} \in \mathcal{H}} F_{\mathbf{i}+\mathbf{1}_{\mathbf{R}(a)}+\mathbf{j}}(t) \cdot P^{\sigma,\pi}(u, \mathbf{j}) \\
&= \inf_{d \in \mathcal{D}(\mathbf{E}(v))} \sum_{a \in A} d(a) \sum_{u \in V} \mathbf{P}(a)(u) \cdot \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \sum_{\mathbf{j} \in \mathcal{H}} F_{\mathbf{i}+\mathbf{1}_{\mathbf{R}(a)}+\mathbf{j}}(t) \cdot P^{\sigma,\pi}(u, \mathbf{j}) \\
&= \inf_{d \in \mathcal{D}(\mathbf{E}(v))} \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \sum_{a \in A} d(a) \sum_{u \in V} \mathbf{P}(a)(u) \sum_{\mathbf{j} \in \mathcal{H}} F_{\mathbf{i}+\mathbf{1}_{\mathbf{R}(a)}+\mathbf{j}}(t) \cdot P^{\sigma,\pi}(u, \mathbf{j}) \\
&= \sup_{\sigma \in \Sigma} \inf_{d \in \mathcal{D}(\mathbf{E}(v))} \inf_{\pi \in \Pi} \sum_{a \in A} d(a) \sum_{u \in V} \mathbf{P}(a)(u) \sum_{\mathbf{j} \in \mathcal{H}} F_{\mathbf{i}+\mathbf{1}_{\mathbf{R}(a)}+\mathbf{j}}(t) \cdot P^{\sigma,\pi}(u, \mathbf{j}) \\
&= \sup_{\sigma \in \Sigma} \inf_{d \in \mathcal{D}(\mathbf{E}(v))} \inf_{\pi \in \Pi} \sum_{a \in A} d(a) \sum_{u \in V} \mathbf{P}(a)(u) \sum_{\substack{\mathbf{j} \in \mathcal{H} \\ \mathbf{j}(\mathbf{R}(a))>0}} F_{\mathbf{i}+\mathbf{j}}(t) \cdot P^{\sigma^a,\pi^a}(u, \mathbf{j} - \mathbf{1}_{\mathbf{R}(a)}) \\
&= \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \sum_{a \in A} \pi(v)(a) \sum_{u \in V} \mathbf{P}(a)(u) \sum_{\substack{\mathbf{j} \in \mathcal{H} \\ \mathbf{j}(\mathbf{R}(a))>0}} F_{\mathbf{i}+\mathbf{j}}(t) \cdot P^{\sigma^a,\pi^a}(u, \mathbf{j} - \mathbf{1}_{\mathbf{R}(a)}) \\
&= \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \sum_{a \in A} \sum_{\substack{\mathbf{j} \in \mathcal{H} \\ \mathbf{j}(\mathbf{R}(a))>0}} F_{\mathbf{i}+\mathbf{j}}(t) \cdot \pi(v)(a) \sum_{u \in V} \mathbf{P}(a)(u) \cdot P^{\sigma^a,\pi^a}(u, \mathbf{j} - \mathbf{1}_{\mathbf{R}(a)}) \\
&= \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \sum_{\mathbf{j} \in \mathcal{H}} F_{\mathbf{i}+\mathbf{j}}(t) \sum_{\substack{a \in A \\ \mathbf{j}(\mathbf{R}(a))>0}} \pi(v)(a) \sum_{u \in V} \mathbf{P}(a)(u) \cdot P^{\sigma^a,\pi^a}(u, \mathbf{j} - \mathbf{1}_{\mathbf{R}(a)}) \\
&= \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \sum_{\mathbf{j} \in \mathcal{H}} F_{\mathbf{i}+\mathbf{j}}(t) P^{\sigma,\pi}(v, \mathbf{j})
\end{aligned}$$

□

Claim 3.8. *For every $\sigma \in \Sigma$, $k \geq 0$, $\mathbf{i} \in \mathcal{H}$, $v \in V$, $\varepsilon \geq 0$, we have*

$$\mathcal{R}_k^\sigma(\mathbf{i}, v) \leq \mu\mathcal{V}(\mathbf{i}, v) + \sum_{j=1}^{k} \frac{\varepsilon}{2^{|\mathbf{i}|+j}}$$

Proof. For $v \in T$ we have

$$\mathcal{R}_k^\sigma(\mathbf{i}, v) = F_{\mathbf{i}}(t) = \mu\mathcal{V}(\mathbf{i}, v)$$

Assume that $v \notin T$. We proceed by induction on k. For $k = 0$ we have

$$\mathcal{R}_k^\sigma(\mathbf{i}, v) = 0 \leq \mu\mathcal{V}(\mathbf{i}, v)$$

For the induction step, first assume that $v \in V_\Box \setminus T$

$$\begin{aligned}
\mathcal{R}^\sigma_k(\mathbf{i}, v) &= \sum_{a \in \mathbf{E}(v)} \sigma(v)(a) \sum_{u \in V} \mathbf{P}(a)(u) \cdot \mathcal{R}^{\sigma^a}_{k-1}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \\
&\leq \sum_{a \in \mathbf{E}(v)} \sigma(v)(a) \sum_{u \in V} \mathbf{P}(a)(u) \cdot \left(\mu\mathcal{V}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) + \sum_{j=2}^{k} \frac{\varepsilon}{2^{|\mathbf{i}|+j}} \right) \\
&= \left(\sum_{a \in \mathbf{E}(v)} \sigma(v)(a) \cdot \sum_{u \in V} \mathbf{P}(a)(u) \cdot \mu\mathcal{V}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \right) + \sum_{j=2}^{k} \frac{\varepsilon}{2^{|\mathbf{i}|+j}} \\
&\leq \mu\mathcal{V}(\mathbf{i}, v) + \sum_{j=2}^{k} \frac{\varepsilon}{2^{|\mathbf{i}|+j}}
\end{aligned}$$

Finally, assume that $v \in V_\Diamond \setminus T$, and let $a \in A$ be the action such that $\pi_\varepsilon(\mathbf{i}, v)(a) = 1$

$$\begin{aligned}
\mathcal{R}^\sigma_k(\mathbf{i}, v) &= \sum_{u \in V} \mathbf{P}(a)(u) \cdot \mathcal{R}^{\sigma^a}_{k-1}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \\
&\leq \sum_{u \in V} \mathbf{P}(a)(u) \cdot \left(\mu\mathcal{V}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) + \sum_{j=2}^{k} \frac{\varepsilon}{2^{|\mathbf{i}|+j}} \right) \\
&= \left(\sum_{u \in V} \mathbf{P}(a)(u) \cdot \mu\mathcal{V}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \right) + \sum_{j=2}^{k} \frac{\varepsilon}{2^{|\mathbf{i}|+j}} \\
&\leq \mu\mathcal{V}(\mathbf{i}, v) + \frac{\varepsilon}{2^{|\mathbf{i}|}} + \sum_{j=2}^{k} \frac{\varepsilon}{2^{|\mathbf{i}|+j}} \\
&\leq \mu\mathcal{V}(\mathbf{i}, v) + \sum_{j=1}^{k} \frac{\varepsilon}{2^{|\mathbf{i}|+j}} \qquad \Box
\end{aligned}$$

Appendix C.2. Proof of Lemma 3.12

Lemma 3.12. *If G is finitely-branching and has bounded rates, then we have the following:*

1. *For all $\varepsilon > 0$, $k \geq (\sup \mathcal{R})te^2 - \ln \varepsilon$, $\sigma \in \Sigma$, $\pi \in \Pi$, and $v \in V$ we have that*

$$\mathcal{P}^{\sigma,\pi}_v(\mathit{Reach}^{\leq t}(T)) - \varepsilon \;\leq\; \mathcal{P}^{\sigma,\pi}_v(\mathit{Reach}^{\leq t}_{\leq k}(T)) \;\leq\; \mathcal{P}^{\sigma,\pi}_v(\mathit{Reach}^{\leq t}(T))$$

2. *For every $k \in \mathbb{N}$, there are k-step optimal BCD strategies $\sigma^k \in \Sigma$ and $\pi^k \in \Pi$. Further, for all $\varepsilon > 0$ and $k \geq (\sup \mathcal{R})te^2 - \ln \varepsilon$ we have that every k-step optimal strategy is also an ε-optimal strategy.*

Proof. **ad 1.** Let us fix a rate $r = \sup \mathcal{R}$. It suffices to see that (here, the random variables used to define F_i have rate r)

$$\begin{aligned}\sum_{n=k+1}^{\infty} \mathcal{P}_v^{\sigma,\pi}(Reach_{=n}^{\leq t}(T)) &\leq \sum_{n=k+1}^{\infty} F_n(t) \cdot \mathcal{P}_v^{\sigma,\pi}(Reach_{=n}^{\leq \infty}(T)) \\ &\leq \sum_{n=k+1}^{\infty} F_{k+1}(t) \cdot \mathcal{P}_v^{\sigma,\pi}(Reach_{=n}^{\leq \infty}(T)) \\ &= F_{k+1}(t) \cdot \sum_{n=k+1}^{\infty} \mathcal{P}_v^{\sigma,\pi}(Reach_{=n}^{\leq \infty}(T)) \\ &\leq F_{k+1}(t)\end{aligned}$$

which is less than ε for $k \geq rte^2 - \ln \varepsilon$ by the following claim.

Claim C.34. For every $\varepsilon \in (0,1)$ and $n \geq rte^2 - \ln \varepsilon$ we have $F_n(t) < \varepsilon$.

Proof.

$$F_n(t) = 1 - e^{-rt} \sum_{i=0}^{n-1} \frac{(rt)^i}{i!} = e^{-rt} \sum_{i=n}^{\infty} \frac{(rt)^i}{i!} = (*)$$

By Taylor's theorem for $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$ and Lagrange form of the remainder we get

$$(*) \leq e^{-rt} \frac{(rt)^n}{n!} e^{rt} = \frac{(rt)^n}{n!} = (**)$$

By Stirling's formula $n! \approx \sqrt{n}(n/e)^n$ we get

$$(**) < \left(\frac{rte}{n}\right)^n < \left(\frac{1}{e}\right)^n < \left(\frac{1}{e}\right)^{-\ln \varepsilon} = \varepsilon$$

by assumptions. □

ad 2. We proceed similarly as in the proof of Theorem 3.6 (we also use some notation of the proof of Theorem 3.6). Recall that given $\sigma \in \Sigma$, $\pi \in \Pi$, $\mathbf{j} \in \mathcal{H}$, and $u \in V$, we denote by $P^{\sigma,\pi}(u, \mathbf{j})$ the probability of all runs $\alpha \in Run_{G(u,\sigma,\pi)}$ such that for some $n \in \mathbb{N}_0$ the state $\alpha(n)$ hits T and matches $\mathbf{j}$, and for all $0 \leq j < n$ we have that $\alpha(j)$ does not hit T.

Given $(\sigma, \pi) \in \Sigma \times \Pi$, $\mathbf{i} \in \mathcal{H}$ such that $|\mathbf{i}| \leq k$, and $v \in V$, we define

$$\bar{P}^{\sigma,\pi}(\mathbf{i}, v) := \sum_{\substack{\mathbf{j} \in \mathcal{H} \\ |\mathbf{j}| \leq k - |\mathbf{i}|}} F_{\mathbf{i}+\mathbf{j}}(t) \cdot P^{\sigma,\pi}(v, \mathbf{j})$$

the probability of reaching T from v before time t in at most $k - |\mathbf{i}|$ steps using the strategies σ and π and assuming that the history matches $\mathbf{i}$.

To define the CD strategies σ^k and π^k we express the value $\sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \bar{P}^{\sigma,\pi}(\mathbf{i}, v)$ $(= \inf_{\pi \in \Pi} \sup_{\sigma \in \Sigma} \bar{P}^{\sigma,\pi}(\mathbf{i}, v)$, see below) using the following recurrence.

Given $\mathbf{i} \in \mathcal{H}$, where $|\mathbf{i}| \leq k$, and $v \in V$, we define

$$\bar{R}(\mathbf{i}, v) := \begin{cases} F_{\mathbf{i}}(t) & \text{if } v \in T \\ 0 & \text{if } v \notin T \text{ and } |\mathbf{i}| = k \\ max_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot \bar{R}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) & \text{if } v \in V_{\Box} \setminus T \text{ and } |\mathbf{i}| < k \\ min_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot \bar{R}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) & \text{if } v \in V_{\Diamond} \setminus T \text{ and } |\mathbf{i}| < k \end{cases}$$

For $v \notin T$ and $|\mathbf{i}| < k$ we define $\sigma^k(\mathbf{i}, v)$ and $\pi^k(\mathbf{i}, v)$ in the following way. If $v \in V_{\Box}$, we put $\sigma^k(\mathbf{i}, v)(a) = 1$ for some action a which realizes the maximum in the definition of $\bar{R}(\mathbf{i}, v)$. Similarly, if $v \in V_{\Diamond}$, we put $\pi^k(\mathbf{i}, v)(a) = 1$ for some action a which realizes the minimum in the definition of $\bar{R}(\mathbf{i}, v)$. For $|\mathbf{i}| \geq k$ and $v \in V$ we define $\sigma^k(\mathbf{i}, v)$ and $\pi^k(\mathbf{i}, v)$ arbitrarily so that σ^k and π^k remain BCD.

For every CD strategy $\tau \in \Sigma \cup \Pi$ and $\mathbf{i} \in \mathcal{H}$, we denote by $\tau[\mathbf{i}]$ the strategy obtained from τ by $\tau[\mathbf{i}](\mathbf{j}, u) := \tau(\mathbf{i} + \mathbf{j}, u)$.

Given $\pi \in \Pi$, $\mathbf{i} \in \mathcal{H}$ where $|\mathbf{i}| \leq k$, and $v \in V$, we define

$$Z^{\pi}(\mathbf{i}, v) := \bar{P}^{\sigma^k[\mathbf{i}], \pi}(\mathbf{i}, v)$$

Similarly, given $\sigma \in \Sigma$, $\mathbf{i} \in \mathcal{H}$ where $|\mathbf{i}| \leq k$, and $v \in V$, we define

$$Z^{\sigma}(\mathbf{i}, v) := \bar{P}^{\sigma, \pi^k[\mathbf{i}]}(\mathbf{i}, v)$$

We prove the following claim.

Claim C.35. Let $\mathbf{i} \in \mathcal{H}$, where $|\mathbf{i}| \leq k$, and $v \in V$. Then

$$\begin{aligned} \bar{R}(\mathbf{i}, v) &= \inf_{\pi \in \Pi} Z^{\pi}(\mathbf{i}, v) && \text{(C.1)} \\ &= \sup_{\sigma \in \Sigma} Z^{\sigma}(\mathbf{i}, v) && \text{(C.2)} \\ &= \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \bar{P}^{\sigma, \pi}(\mathbf{i}, v) && \text{(C.3)} \\ &= \inf_{\pi \in \Pi} \sup_{\sigma \in \Sigma} \bar{P}^{\sigma, \pi}(\mathbf{i}, v) && \text{(C.4)} \end{aligned}$$

In particular, the strategies σ^k and π^k are k-step optimal because $\bar{P}^{\sigma,\pi}(\mathbf{0}, v) = \mathcal{P}_v^{\sigma,\pi}(Reach_{\leq k}^{\leq t}(T))$.

Proof. First, if $v \in T$, then for all $(\sigma, \pi) \in \Sigma \times \Pi$ we have $\bar{P}^{\sigma,\pi}(\mathbf{i}, v) = F_{\mathbf{i}}(t) = \bar{R}(\mathbf{i}, v)$. Assume that $v \notin T$. We proceed by induction on $n = k - |\mathbf{i}|$. For $n = 0$ we have $\bar{P}^{\sigma,\pi}(\mathbf{i}, v) = 0 = \bar{R}(\mathbf{i}, v)$. Assume the lemma holds for n, we show that it holds also for $n + 1$.

We start by proving the equation (C.1). Using the notation of the proof of Theorem 3.6, given a strategy $\tau \in \Sigma \cup \Pi$ and $a \in A$, we denote by τ^a a strategy defined by $\tau^a(wu) := \tau(vawu)$.

If $v \in V_\Box$ and $\sigma^k(\mathbf{i}, v)(b) = 1$,

$$\begin{aligned}
\inf_{\pi\in\Pi} Z^\pi(\mathbf{i}, v) &= \inf_{\pi\in\Pi} \sum_{u\in V} \mathbf{P}(b)(u) \cdot Z^{\pi^b}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(b)}, u) \\
&= \sum_{u\in V} \mathbf{P}(b)(u) \cdot \inf_{\pi\in\Pi} Z^{\pi^b}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(b)}, u) \\
&= \sum_{u\in V} \mathbf{P}(b)(u) \cdot \inf_{\pi\in\Pi} Z^{\pi}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(b)}, u) \\
&= \sum_{u\in V} \mathbf{P}(b)(u) \cdot \bar{R}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(b)}, u) \\
&= \max_{a\in\mathbf{E}(v)} \sum_{u\in V} \mathbf{P}(a)(u) \cdot \bar{R}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \\
&= \bar{R}(\mathbf{i}, v)
\end{aligned}$$

If $u \in V_\Diamond$,

$$\begin{aligned}
\inf_{\pi\in\Pi} Z^\pi(\mathbf{i}, v) &= \inf_{\pi\in\Pi} \sum_{a\in\mathbf{E}(v)} \pi(v)(a) \sum_{u\in V} \mathbf{P}(a)(u) \cdot Z^{\pi^a}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \\
&= \inf_{d\in\mathcal{D}(\mathbf{E}(v))} \sum_{a\in\mathbf{E}(v)} d(a) \sum_{u\in V} \mathbf{P}(a)(u) \cdot \inf_{\pi\in\Pi} Z^{\pi^a}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \\
&= \min_{a\in\mathbf{E}(v)} \sum_{u\in V} \mathbf{P}(a)(u) \cdot \inf_{\pi\in\Pi} Z^{\pi}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \\
&= \min_{a\in\mathbf{E}(v)} \sum_{u\in V} \mathbf{P}(a)(u) \cdot \bar{R}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \\
&= \bar{R}(\mathbf{i}, v)
\end{aligned}$$

The equation (C.2) can be proved in a similar manner.

The claim follows from the following

$$\begin{aligned}
\bar{R}(\mathbf{i}, v) = \inf_{\pi\in\Pi} Z^\pi(\mathbf{i}, v) \le \sup_{\sigma\in\Sigma} \inf_{\pi\in\Pi} \bar{P}^{\sigma,\pi}(\mathbf{i}, v) \le \\
\le \inf_{\pi\in\Pi} \sup_{\sigma\in\Sigma} \bar{P}^{\sigma,\pi}(\mathbf{i}, v) \le \sup_{\sigma\in\Sigma} Z^\sigma(\mathbf{i}, v) = \bar{R}(\mathbf{i}, v)
\end{aligned}$$

□

The rest of the lemma is easily obtained from 1. as follows. Let $\varepsilon > 0$ and consider $k \ge (\sup \mathcal{R})te^2 - \ln\varepsilon$. Then 1. implies that the value $\bar{R}(\mathbf{0}, v)$ of the k-step game initiated in v satisfies $val(v) - \varepsilon \le \bar{R}(\mathbf{0}, v) \le val(v)$. Therefore all k-step optimal strategies are ε-optimal. □

Appendix C.3. Proof of Claim 4.19

Claim 4.19. *A strategy is always greedy on s steps iff it uses transitions from $\mathbf{E}_s$ only (as defined by Algorithm 1).*

Proof. For the 'if' direction, we prove by induction on n that

$$\max_{\sigma\in\Sigma}\min_{\pi\in\Pi}\big(\overrightarrow{\mathcal{P}}_v^{\sigma,\pi}\big)_{(1,\dots,n)} = \min_{\pi\in\Pi}\big(\overrightarrow{\mathcal{P}}_v^{\bar\sigma,\pi}\big)_{(1,\dots,n)} = \max_{\sigma\in\Sigma}\big(\overrightarrow{\mathcal{P}}_v^{\sigma,\bar\pi}\big)_{(1,\dots,n)}$$

for all strategies $\bar\sigma\in\Sigma$ and $\bar\pi\in\Pi$ that use edges from $\mathbf{E}_n$ only. It is sufficient to prove that

$$\big(\max_{\sigma\in\Sigma}\min_{\pi\in\Pi}\overrightarrow{\mathcal{P}}_v^{\sigma,\pi}\big)_n \overset{(1)}{=} R_n(v) \overset{(2)}{=} \min_{\pi\in\Pi}\big(\overrightarrow{\mathcal{P}}_v^{\bar\sigma,\pi}\big)_n$$

for every strategy $\bar\sigma\in\Sigma$ that uses edges from $\mathbf{E}_n$ only. (The minimizing part is dual.) The case $n=0$ is trivial. Now consider $n+1$. For $v\in V_\Box\setminus T$,

$$\begin{aligned}
\big(\min_{\pi\in\Pi}\overrightarrow{\mathcal{P}}_v^{\bar\sigma,\pi}\big)_{n+1} &= \sum_{u\in V}\bar\sigma(0,v)(u)\big(\min_{\pi\in\Pi}\overrightarrow{\mathcal{P}}_u^{\bar\sigma[1],\pi}\big)_n\\
\text{by IH (2) and } \mathbf{E}_{n+1}\subseteq\mathbf{E}_n \quad &= \sum_{u\in V}\bar\sigma(0,v)(u)\cdot R_n(u)\\
\bar\sigma \text{ uses } \mathbf{E}_{n+1} \text{ only} \quad &= \max_{a\in\mathbf{E}_n(v)}\sum_{u\in V}\mathbf{P}(a)(u)\cdot R_n(u) \quad = \quad (*)\\
\text{by IH (1)} \quad &= \max_{a\in\mathbf{E}_n(v)}\sum_{u\in V}\mathbf{P}(a)(u)\cdot\big(\max_{\sigma\in\Sigma}\min_{\pi\in\Pi}\overrightarrow{\mathcal{P}}_u^{\sigma,\pi}\big)_n\\
\text{by IH} \quad &= \max_{a\in A_n(v)}\sum_{u\in V}\mathbf{P}(a)(u)\cdot\big(\max_{\sigma\in\Sigma}\min_{\pi\in\Pi}\overrightarrow{\mathcal{P}}_u^{\sigma,\pi}\big)_n\\
&= \big(\max_{\sigma\in\Sigma}\min_{\pi\in\Pi}\overrightarrow{\mathcal{P}}_v^{\sigma,\pi}\big)_{n+1}
\end{aligned}$$

where $A_n(v)$ is the set of all edges going from v that any strategy always greedy on n steps can choose. I.e. it is the desired abstractly defined set of greedy edges, which is equal to the computed set $\mathbf{E}_n(v)$ by the induction hypothesis. Since $(*)=R_{n+1}(v)$, the equality with the first and the last expression proves the claim. Similarly for $v\in V_\Diamond\setminus T$,

$$\begin{aligned}
\big(\min_{\pi\in\Pi}\overrightarrow{\mathcal{P}}_v^{\bar\sigma,\pi}\big)_{n+1} &= \min_{a\in A_n(v)}\sum_{u\in V}\mathbf{P}(a)(u)\cdot\big(\min_{\pi\in\Pi}\overrightarrow{\mathcal{P}}_u^{\bar\sigma[1],\pi}\big)_n\\
\text{by IH (2) and } \mathbf{E}_{n+1}\subseteq\mathbf{E}_n \quad &= \min_{a\in A_n(v)}\sum_{u\in V}\mathbf{P}(a)(u)\cdot R_n(u)\\
\text{by IH for the minimizing part} \quad &= \min_{a\in\mathbf{E}_n(v)}\sum_{u\in V}\mathbf{P}(a)(u)\cdot R_n(u) \quad = \quad (**)\\
\text{by IH (1)} \quad &= \min_{a\in\mathbf{E}(v)}\sum_{u\in V}\mathbf{P}(a)(u)\cdot\big(\max_{\sigma\in\Sigma}\min_{\pi\in\Pi}\overrightarrow{\mathcal{P}}_u^{\sigma,\pi}\big)_n\\
\text{by IH for the minimizing part} \quad &= \max_{a\in A_n(v)}\sum_{u\in V}\mathbf{P}(a)(u)\cdot\big(\max_{\sigma\in\Sigma}\min_{\pi\in\Pi}\overrightarrow{\mathcal{P}}_u^{\sigma,\pi}\big)_n\\
&= \big(\max_{\sigma\in\Sigma}\min_{\pi\in\Pi}\overrightarrow{\mathcal{P}}_v^{\sigma,\pi}\big)_{n+1}
\end{aligned}$$

where $(**) = R_{n+1}(v)$. The case with $v \in T$ is trivial as states in T are absorbing.

We prove the "only if" direction by contraposition. If a strategy τ uses a transition $a \in \mathbf{E} \setminus \mathbf{E}_s$ in v then there is $i \leq s$ such that a has been cut off in the ith step. Therefore a did not realize the i steps optimum (equal to $R_i(v)$). Hence τ is not greedy on n steps. □

Appendix C.4. Proof of Lemma 5.24

Lemma 5.24. *The strategies σ_ε and π_ε are ε-optimal.*

Proof. We use some notation of the proof of Theorem 3.6. Recall that given $\sigma \in \Sigma$, $\pi \in \Pi$, $\mathbf{j} \in \mathcal{H}$, and $u \in V$, we denote by $P^{\sigma,\pi}(u, \mathbf{j})$ the probability of all runs $\alpha \in Run_{G(u,\sigma,\pi)}$ such that for some $n \in \mathbb{N}_0$ the state $\alpha(n)$ hits T and matches $\mathbf{j}$, and for all $0 \leq j < n$ we have that $\alpha(j)$ does not hit T.

Given $(\sigma, \pi) \in \Sigma \times \Pi$, $\mathbf{i} \in \mathcal{H}$, where $|\mathbf{i}| \leq k$, and $v \in V$, we define

$$\bar{P}^{\sigma,\pi}(\mathbf{i}, v) := \sum_{\substack{\mathbf{j} \in \mathcal{H} \\ |\mathbf{j}| \leq k - |\mathbf{i}|}} F_{\mathbf{i}+\mathbf{j}}(t) \cdot P^{\sigma,\pi}(v, \mathbf{j})$$

the probability of reaching T from v before time t in at most $k - |\mathbf{i}|$ steps using the strategies σ and π and assuming that the history already matches $\mathbf{i}$. We have shown in the proof of Claim C.35 that for every $\mathbf{i} \in \mathcal{H}$, where $|\mathbf{i}| \leq k$, and $v \in V$, the value

$$\max_{\sigma \in \Sigma} \min_{\pi \in \Pi} \bar{P}^{\sigma,\pi}(\mathbf{i}, v) = \min_{\pi \in \Pi} \max_{\sigma \in \Sigma} \bar{P}^{\sigma,\pi}(\mathbf{i}, v)$$

is equal to $\bar{R}(\mathbf{i}, v)$ defined by the following equations:

$$\bar{R}(\mathbf{i}, v) := \begin{cases} F_{\mathbf{i}}(t) & \text{if } v \in T \\ 0 & \text{if } v \notin T \text{ and } |\mathbf{i}| = k \\ \max_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot \bar{R}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) & \text{if } v \in V_\Box \setminus T \text{ and } |\mathbf{i}| < k \\ \min_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot \bar{R}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) & \text{if } v \in V_\Diamond \setminus T \text{ and } |\mathbf{i}| < k \end{cases}$$

Note that $\bar{P}^{\sigma,\pi}(\mathbf{0}, v) = \mathcal{P}_v^{\sigma,\pi}(Reach^{\leq t}_{\leq k}(T))$ and thus $\bar{R}(\mathbf{0}, v) = val^k(v)$, the k-step value in v.

Note that assuming $l_{\mathbf{i}}(t) = F_{\mathbf{i}}(t)$ for all $\mathbf{i} \in \mathcal{H}$ satisfying $|\mathbf{i}| \leq k$, we would obtain that each $R(\mathbf{i}, v)$ is precisely $\bar{R}(\mathbf{i}, v)$ and hence that σ_ε and π_ε are k-step optimal strategies.

Let us allow imprecisions in the computation of $l_{\mathbf{i}}(t)$. We proceed as follows: First we show, by induction, that each value $R(\mathbf{i}, v)$ approximates the value $\bar{R}(\mathbf{i}, v)$ with relative error $\left(\frac{\varepsilon}{2}\right)^{2|\mathbf{i}|+1}$ (Claim C.38 below). From this we get, also by induction, that both $\min_{\pi \in \Pi} \bar{P}^{\sigma_\varepsilon,\pi}(\mathbf{i}, v)$ and $\max_{\sigma \in \Sigma} \bar{P}^{\sigma,\pi_\varepsilon}(\mathbf{i}, v)$ approximate $\bar{R}(\mathbf{i}, v)$ with relative error $\left(\frac{\varepsilon}{2}\right)^{2|\mathbf{i}|+1}$ as well (Claim C.39 below). In other words, σ_ε and π_ε are $\frac{\varepsilon}{2}$-optimal strategies in the k-step game. Together with the assumptions imposed on k we obtain that σ_ε and π_ε are ε-optimal strategies.

For $n \geq 0$, we denote by err_n the number $\left(\frac{\varepsilon}{2}\right)^{2n+1}$.

Claim C.38. For all $\mathbf{i} \in \mathcal{H}$ and $v \in V$ we have

$$(1 - err_{|\mathbf{i}|}) \cdot \bar{R}(\mathbf{i}, v) \quad \leq \quad R(\mathbf{i}, v) \quad \leq \quad (1 + err_{|\mathbf{i}|}) \cdot \bar{R}(\mathbf{i}, v)$$

Proof. If $v \in T$, then $\bar{R}(\mathbf{i}, v) = F_{\mathbf{i}}(t)$ and $R(\mathbf{i}, v) = l_{\mathbf{i}}(t)$, and the inequality follows from the definition of $l_{\mathbf{i}}(t)$. Assume that $v \notin T$. We proceed by induction on $n = k - |\mathbf{i}|$. For $n = 0$ we have $\bar{R}(\mathbf{i}, v) = 0 = R(\mathbf{i}, v)$. Assume the inequality holds for any v and $\mathbf{i} \in \mathcal{H}$ such that $|\mathbf{i}| = k - n$. Let us consider $\mathbf{i} \in \mathcal{H}$ such that $|\mathbf{i}| = k - n - 1$ and $v \in V$. If $v \in V_{\Box}$ we have

$$\begin{aligned}
R(\mathbf{i}, v) &= \max_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{p}(a)(u) \cdot R(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \\
&\leq \max_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot (1 + err_{|\mathbf{i}|+1}) \cdot \bar{R}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \cdot (1 + err_{|\mathbf{i}|+1}) \\
&= (1 + err_{|\mathbf{i}|+1})^2 \cdot \max_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot \bar{R}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \\
&\leq (1 + err_{|\mathbf{i}|}) \cdot \bar{R}(\mathbf{i}, v)
\end{aligned}$$

and, similarly,

$$R(\mathbf{i}, v) \geq (1 - err_{|\mathbf{i}|}) \cdot \bar{R}(\mathbf{i}, v)$$

For $v \in V_{\Diamond}$ the proof is similar.

□

We denote by Σ_{CD} and Π_{CD} the sets of all CD strategies of Σ and Π, respectively. Recall that given a strategy $\tau \in \Sigma_{CD} \cup \Pi_{CD}$ and $\mathbf{i} \in \mathcal{H}$, we denote by $\tau[\mathbf{i}]$ the strategy obtained from τ by $\tau[\mathbf{i}](\mathbf{j}, u) := \tau(\mathbf{i} + \mathbf{j}, u)$.

Given $\mathbf{i} \in \mathcal{H}$ and $\pi \in \Pi$, we define

$$K^{\pi}(\mathbf{i}, v) := \bar{P}^{\sigma_{\varepsilon}[\mathbf{i}], \pi[\mathbf{i}]}(\mathbf{i}, v)$$

Similarly, given $\mathbf{i} \in \mathcal{H}$ and $\sigma \in \Sigma$, we define

$$K^{\sigma}(\mathbf{i}, v) := \bar{P}^{\sigma[\mathbf{i}], \pi_{\varepsilon}[\mathbf{i}]}(\mathbf{i}, v)$$

Claim C.39. Let $\mathbf{i} \in \mathcal{H}$, where $|\mathbf{i}| \leq k$, and $v \in V$. We have

$$\begin{aligned}
\min_{\pi \in \Pi_{CD}} K^{\pi}(\mathbf{i}, v) &\geq \bar{R}(\mathbf{i}, v) \cdot (1 - err_{|\mathbf{i}|}) \\
\max_{\sigma \in \Sigma_{CD}} K^{\sigma}(\mathbf{i}, v) &\leq \bar{R}(\mathbf{i}, v) \cdot (1 + err_{|\mathbf{i}|})
\end{aligned}$$

Proof. If $v \in T$, then $K^{\pi}(\mathbf{i}, v) = K^{\sigma}(\mathbf{i}, v) = F_{\mathbf{i}}(t)$ and $\bar{R}(\mathbf{i}, v) = l_{\mathbf{i}}(t)$, and similarly as above, the result follows from the definition of $l_{\mathbf{i}}(t)$. Assume that $v \notin T$. We proceed by induction on $n := k - |\mathbf{i}|$. For $n = 0$ we have $0 = K^{\pi}(\mathbf{i}, v) = K^{\sigma}(\mathbf{i}, v) = \bar{R}(\mathbf{i}, u)$. Assume the lemma holds true for n and consider

$n+1$. If $v \in V_\Box$ and $\sigma_\varepsilon(\mathbf{i}, v)(b) = 1$,

$$\begin{aligned}
\min_{\pi \in \Pi_{CD}} K^\pi(\mathbf{i}, v) &= \min_{\pi \in \Pi_{CD}} \sum_{u \in V} \mathbf{P}(b)(u) \cdot K^\pi(\mathbf{i} + \mathbf{1}_{\mathbf{R}(b)}, u) \\
&= \sum_{u \in V} \mathbf{P}(b)(u) \cdot \min_{\pi \in \Pi_{CD}} K^\pi(\mathbf{i} + \mathbf{1}_{\mathbf{R}(b)}, u) \\
&\geq \sum_{u \in V} \mathbf{P}(b)(u) \cdot \bar{R}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(b)}, u) \cdot (1 - err_{|\mathbf{i}|+1}) \\
&\geq \sum_{u \in V} \mathbf{p}(b)(u) \cdot \frac{1}{1 + err_{|\mathbf{i}|+1}} \cdot R(\mathbf{i} + \mathbf{1}_{\mathbf{R}(b)}, u) \cdot \frac{1 - err_{|\mathbf{i}|+1}}{1 + err_{|\mathbf{i}|+1}} \\
&= R(\mathbf{i}, v) \cdot \frac{1 - err_{|\mathbf{i}|+1}}{(1 + err_{|\mathbf{i}|+1})^2} \\
&\geq \bar{R}(\mathbf{i}, v) \cdot (1 - err_{|\mathbf{i}|+1}) \cdot \frac{1 - err_{|\mathbf{i}|+1}}{(1 + err_{|\mathbf{i}|+1})^2} \\
&\geq \bar{R}(\mathbf{i}, v) \cdot (1 - err_{|\mathbf{i}|})
\end{aligned}$$

and

$$\begin{aligned}
\max_{\sigma \in \Sigma_{CD}} K^\sigma(\mathbf{i}, v) &= \max_{\sigma \in \Sigma} \sum_{a \in \mathbf{E}(v)} \sigma(\mathbf{i}, v)(a) \sum_{u \in V} \mathbf{P}(a)(u) \cdot K^\sigma(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \\
&= \max_{a \in \mathbf{E}(v)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot \max_{\sigma \in \Sigma} K^\sigma(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \\
&\leq \max_{a \in \mathbf{E}(u)} \sum_{u \in V} \mathbf{P}(a)(u) \cdot \bar{R}(\mathbf{i} + \mathbf{1}_{\mathbf{R}(a)}, u) \cdot (1 + err_{|\mathbf{i}|+1}) \\
&\leq \bar{R}(\mathbf{i}, u) \cdot (1 + err_{|\mathbf{i}|}) \,.
\end{aligned}$$

For $u \in V_\Diamond$ the proof is similar. □

This proves that σ_ε and π_ε are ε-optimal, since the absolute error is smaller than the relative error as the probabilities are at most 1. □

Paper F:

Verification of Open Interactive Markov Chains

Tomáš Brázdil, Holger Hermanns, Jan Krčál, Jan Křetínský, and Vojtěch Řehák

This paper has been published in Deepak D'Souza, Telikepalli Kavitha and Jaikumar Radhakrishnan (eds.): Proceedings of IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India. LIPIcs, vol. 18, pages 474-485. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012. [BFK+09]

Summary

Interactive Markov chains are a slight extension of the widespread continuous-time Markov decision processes. Their main advantage is compositionality, which facilitates hierarchical design and analysis of systems. However, the analysis of interactive Markov chains has so far been limited to closed controllable systems or open, but uncontrollable systems. We provide a framework for analysis and optimisation of controllable systems operating in an unknown environment. This in turn enables compositional verification of separate components of the system. We give an algorithm computing the optimal guarantee to reach a given state within a given time bound when the system is composed with an unknown IMC environment, provided the system has at each moment only internal or only synchronising actions available. In such a case the time complexity of the analysis is the same as for the earlier analysis of closed systems.

Author's contribution: 35 %

- co-design of the framework in the group discussions,
- writing most of the paper (excluding Section 5),
- discussing the proofs,
- writing the proofs concerning the reduction of the problem to the game setting.

Verification of Open Interactive Markov Chains

Tomáš Brázdil[1], Holger Hermanns[2], Jan Krčál[1], Jan Křetínský[1,3], and Vojtěch Řehák[1]

1 **Faculty of Informatics, Masaryk University, Czech Republic**
{brazdil,krcal,jan.kretinsky,rehak}@fi.muni.cz
2 **Saarland University – Computer Science, Saarbrücken, Germany**
hermanns@cs.uni-saarland.de
3 **Institut für Informatik, Technical University Munich, Germany**

Abstract

Interactive Markov chains (IMC) are compositional behavioral models extending both labeled transition systems and continuous-time Markov chains. IMC pair modeling convenience - owed to compositionality properties - with effective verification algorithms and tools - owed to Markov properties. Thus far however, IMC verification did not consider compositionality properties, but considered closed systems. This paper discusses the evaluation of IMC in an open and thus compositional interpretation. For this we embed the IMC into a game that is played with the environment. We devise algorithms that enable us to derive bounds on reachability probabilities that are assured to hold in any composition context.

1998 ACM Subject Classification D.4.8 Performance

Keywords and phrases IMC, compositional verification, synthesis, time bounded reachability, discretization

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

With the increasing complexity of systems and software reuse, component based development concepts gain more and more attention. In this setting developers are often facing the need to develop a component with only partial information about the surrounding components at hand, especially when relying on third-party components to be inter-operated with. This motivates verification approaches that ensure the functionality of a component in an environment whose behavior is unknown or only partially known. *Compositional verification* approaches aim at methods to prove guarantees on isolated components in such a way that when put together, the entire system's behavior has the desired properties based on the individual guarantees.

The assurance of reliable functioning of a system relates not only to its correctness, but also to its performance and dependability. This is a major concern especially in embedded system design. A natural instantiation of the general component-based approach in the continuous-time setting are *interactive Markov chains* [24]. Interactive Markov chains (IMC) are equipped with a sound compositional theory. IMC arise from classical labeled transition systems by incorporating the possibility to change state according to a random delay governed by some negative exponential distribution. This twists the model to one that is running in continuous real time. State transitions may be triggered by delay expirations, or may be triggered by the execution of actions. By dropping the new type of transitions, labeled transition systems are regained in their entirety. By dropping action-labeled transitions instead, one arrives at one of the simplest but also most widespread class of performance and de-

pendability models, *continuous-time Markov chains* (CTMC). IMC have a well-understood compositional theory, rooted in process algebra [3], and are in use as semantic backbones for dynamic fault trees [6], architectural description languages [5, 8], generalized stochastic Petri nets [25] and Statemate [4] extensions, and are applied in a large spectrum of practical applications, ranging from networked hardware on chips [15] to water treatment facilities [21] and ultra-modern satellite designs [16].

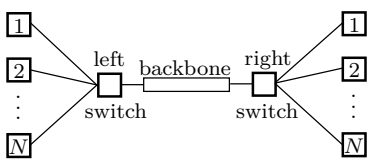
In recent years, various analysis techniques have been proposed [18, 27, 23, 26, 34, 19] for IMC. The pivotal verification problem considered is that of *time-bounded reachability*. It is the problem to calculate or approximate the probability that a given state (set) is reached within a given deadline. However, despite the fact that IMC support compositional model generation and minimization very well, the analysis techniques considered thus far are not compositional. They are all bound to the assumption that the analyzed IMC is closed, i.e. does not depend on interaction with the environment. Technically, this is related to the *maximal-progress assumption* governing the interplay of delay and action execution of an IMC component: Internal actions are assumed to happen instantaneously and therefore take precedence over delay transitions while external actions do not. External actions are the process algebraic means for interaction with other components. Remarkably, in all the published IMC verification approaches, all occurring actions are assumed to be internal (respectively internalized by means of a hiding operator prior to analysis).

In this paper, we instead consider *open* IMC, where the control over external actions is in the hands of and possibly delayed by an environment. The environment can be thought of as summarizing the behavior of one or several interacting components. As a consequence, we find ourselves in the setting of a timed game, where the environment has the (timed) control over external actions, while the IMC itself controls choices over internal actions. The resulting game turns out to be remarkably difficult, owed to the interplay of timed moves with external and internal moves of both players.

Concretely, assume we are given an IMC $\mathcal{C}$ which contains some internal non-deterministic transitions and also offers some external actions for synchronization to an unknown environment. Our goal is to synthesize a scheduler controlling the internal transitions which maximizes the probability of reaching a set G of goal states, in time T no matter what and when the environment E decides to synchronize with the external actions. The environment E ranges over all possible IMC able to synchronize with the external actions of $\mathcal{C}$.

To get a principal understanding of the complications faced, we need to consider a restricted setting, where $\mathcal{C}$ does not enable internal and external transitions at the same state. We provide an algorithm which approximates the probability in question up to a given precision $\varepsilon > 0$ and also computes an ε-optimal scheduler. The algorithm consists of two steps. First, we reduce the problem to a game where the environment is not an IMC but can decide to execute external actions at *non-deterministically* chosen time instances. In a second step, we solve the resulting game on $\mathcal{C}$ using discretization. Our discretization is based on the same approach as the algorithm of [34]. However, the algorithm as well as its proof of correctness is considerably more complicated due to presence of non-deterministic choices of the player controlling the environment. We finally discuss what happens if we allow internal and external transitions to be enabled at the same time.

Example. To illustrate the concepts by an example application, we can consider a variant of the *fault-tolerant workstation cluster* [22] depicted on the right. The overall system consists of two sub-clusters connected via a backbone; each of them contains N workstations.



Any component can fail and then needs to be repaired to become operational again. There is a single repair unit (not depicted) which must take decisions what to repair next when multiple components are failed. The entire system can be modelled using the IMC composition operators [22], but we are now also in the position to study a partial model, where some components, such as one of the switches, are left unspecified. We seek for the optimal repair schedule regardless of how the unknown components are implemented. We can answer questions such as: *"What is the worst case probability to hit a state in which premium service is not guaranteed within T time units?"* with premium service only being guaranteed if there are at least N operational workstations connected to each other via operational switches.

Our contribution. We investigate the problem of compositionally verifying open IMC. In particular, we introduce the problem of synthesizing optimal control for time-bounded reachability in an IMC interacting in an unknown environment, provided no state enables internal and external transition. Thereafter, we solve the problem of finding ε-optimal schedulers using the established method of discretization, give bounds on the size of the game to be solved for a given ε and thus establish upper complexity bound for the problem. Complete proofs and further relevant details can be found in the full version [10].

Related work. Model checking of *open* systems has been proposed in [28]. The synthesis problem is often stated as a *game* where the first player controls a component and the second player simulates an environment [31]. There is a large body of literature on games in verification, including recent surveys [1, 13]. *Stochastic* games have been applied to e.g. concurrent program synthesis [33] and for collaboration strategies among compositional stochastic systems [14]. Although most papers deal with discrete time games, lately games with stochastic *continuous-time* have gained attention [7, 30, 9, 11]. Some of the games we consider in the present paper exploit special cases of the games considered in [7, 11]. However, both papers prove decidability only for qualitative reachability problems and do not discuss compositionality issues. Further, while systems of [30, 9] are very similar to ours, the structure of the environment is fixed there and the verification is thus not compositional. The same holds for [32, 20], where time is under the control of the components.

The *time-bounded reachability* problem for closed IMC has been studied in [23, 34] and compositional abstraction techniques to compute it are developed in [26]. In the closed interpretation, IMC have some similarities with continuous-time Markov decision processes, CTMDP. Algorithms for time-bounded reachability in CTMDP and corresponding games are developed in [2, 9, 30]. A numerically stable algorithm for time-bounded properties for CTMDP is developed in [12].

2 Interactive Markov Chains

In this section, we introduce the formalism of interactive Markov chains together with the standard way to compose them. After giving the operational interpretation for closed systems, we define the fundamental problem of our interest, namely we define the value of time-bounded reachability and introduce the studied problems.

We denote by $\mathbb{N}$, $\mathbb{N}_0$, $\mathbb{R}_{>0}$, and $\mathbb{R}_{\geq 0}$ the sets of natural numbers, natural numbers with zero, positive real numbers and non-negative real numbers, respectively.

▶ **Definition 1** (IMC). An interactive Markov chain (IMC) is a tuple $\mathcal{C} = (S, \mathbb{A}\mathrm{ct}^\tau, \hookrightarrow, \rightsquigarrow, s_0)$ where S is a finite set of *states*, $\mathbb{A}\mathrm{ct}^\tau$ is a finite set of *actions* containing a designated *internal action* τ, $s_0 \in S$ is an *initial* state,

- $\hookrightarrow \subseteq S \times \mathbb{A}\mathrm{ct}^\tau \times S$ is an *interactive transition* relation, and
- $\rightsquigarrow \subseteq S \times \mathbb{R}_{>0} \times S$ is a *Markovian transition* relation.

Elements of $\mathbb{A}\mathrm{ct} := \mathbb{A}\mathrm{ct}^\tau \setminus \{\tau\}$ are called *external actions*. We write $s \overset{a}{\hookrightarrow} t$ whenever $(s,a,t) \in \hookrightarrow$, and further $\mathrm{succ}_e(s) = \{t \in S \mid \exists a \in \mathbb{A}\mathrm{ct} : s \overset{a}{\hookrightarrow} t\}$ and $\mathrm{succ}_\tau(s) = \{t \in S \mid s \overset{\tau}{\hookrightarrow} t\}$. Similarly, we write $s \overset{\lambda}{\rightsquigarrow} t$ whenever $(s,\lambda,t) \in \rightsquigarrow$ where λ is called a *rate* of the transition, and $\mathrm{succ}_M(s) = \{t \in S \mid \exists \lambda : s \overset{\lambda}{\rightsquigarrow} t\}$. We assume w.l.o.g. that for each pair of states s and t, there is at most one Markovian transition from s to t. We say that an external, or internal, or Markovian transition is available in s if $\mathrm{succ}_e(s) \neq \emptyset$, or $\mathrm{succ}_\tau(s) \neq \emptyset$, or $\mathrm{succ}_M(s) \neq \emptyset$, respectively.

We also define a *total exit rate* function $\mathbf{E} : S \to \mathbb{R}_{\geq 0}$ which assigns to each state the sum of rates of all outgoing Markovian transitions, i.e. $\mathbf{E}(s) = \sum_{s \overset{\lambda}{\rightsquigarrow} t} \lambda$ where the sum is zero if $\mathrm{succ}_M(s)$ is empty. Furthermore, we define a probability matrix $\mathbf{P}(s,t) = \lambda/\mathbf{E}(s)$ if $s \overset{\lambda}{\rightsquigarrow} t$; and $\mathbf{P}(s,t) = 0$, otherwise.

IMC are well suited for compositional modeling, where systems are built out of smaller ones using composition operators. Parallel composition and hiding operators are central to the modeling style, where parallel components synchronize using shared action, and further synchronization can be prohibited by hiding (i.e. internalizing) some actions. IMC employ the *maximal progress assumption*: Internal actions take precedence over the advance of time [24].

▶ **Definition 2** (Parallel composition). For IMC $\mathcal{C}_1 = (S_1, \mathbb{A}\mathrm{ct}_1^\tau, \hookrightarrow_1, \rightsquigarrow_1, s_{01})$ and $\mathcal{C}_2 = (S_2, \mathbb{A}\mathrm{ct}_2^\tau, \hookrightarrow_2, \rightsquigarrow_2, s_{02})$ and a *synchronization alphabet* $A \subseteq \mathbb{A}\mathrm{ct}_1 \cap \mathbb{A}\mathrm{ct}_2$, the parallel composition $\mathcal{C}_1 \parallel_A \mathcal{C}_2$ is the IMC $\mathcal{C}_1 = (S_1 \times S_2, \mathbb{A}\mathrm{ct}_1^\tau \cup \mathbb{A}\mathrm{ct}_2^\tau, \hookrightarrow, \rightsquigarrow, (s_{01}, s_{02}))$ where $\hookrightarrow$ and $\rightsquigarrow$ are defined as the smallest relations satisfying

- $s_1 \overset{a}{\hookrightarrow} s_1'$ and $s_2 \overset{a}{\hookrightarrow} s_2'$ and $a \in A$ implies $(s_1,s_2) \overset{a}{\hookrightarrow} (s_1',s_2')$,
- $s_1 \overset{a}{\hookrightarrow} s_1'$ and $a \notin A$ implies $(s_1,s_2) \overset{a}{\hookrightarrow} (s_1',s_2)$ for each $s_2 \in S_2$,
- $s_2 \overset{a}{\hookrightarrow} s_2'$ and $a \notin A$ implies $(s_1,s_2) \overset{a}{\hookrightarrow} (s_1,s_2')$ for each $s_1 \in S_1$,
- $s_1 \overset{\lambda}{\rightsquigarrow} s_1'$ implies $(s_1,s_2) \overset{\lambda}{\rightsquigarrow} (s_1',s_2)$ for each $s_2 \in S_2$, and
- $s_2 \overset{\lambda}{\rightsquigarrow} s_2'$ implies $(s_1,s_2) \overset{\lambda}{\rightsquigarrow} (s_1,s_2')$ for each $s_1 \in S_1$.

▶ **Definition 3** (Hiding). For an IMC $\mathcal{C} = (S, \mathbb{A}\mathrm{ct}^\tau, \hookrightarrow, \rightsquigarrow, s_0)$ and a *hidden alphabet* $A \subseteq \mathbb{A}\mathrm{ct}$, the hiding $\mathcal{C} \setminus A$ is the IMC $(S, \mathbb{A}\mathrm{ct}^\tau \setminus A, \hookrightarrow', \rightsquigarrow, s_0)$ where $\hookrightarrow'$ is the smallest relation satisfying for each $s \overset{a}{\hookrightarrow} s'$ that $a \in A$ implies $s \overset{\tau}{\hookrightarrow}{}' s'$, and $a \notin A$ implies $s \overset{a}{\hookrightarrow}{}' s'$.

The analysis of IMC has thus far been restricted to *closed* IMC [18, 27, 23, 26, 34, 19]. In a closed IMC, external actions do not appear as transition labels (i.e. $\hookrightarrow \subseteq S \times \{\tau\} \times S$). In practice, this is achieved by an outermost hiding operator $\setminus \mathbb{A}\mathrm{ct}$ closing the composed system. Non-determinism among internal τ transitions is resolved using a (history-dependent) scheduler σ [34].

Let us fix a *closed* IMC $\mathcal{C} = (S, \mathbb{A}\mathrm{ct}^\tau, \hookrightarrow, \rightsquigarrow, s_0)$. The IMC $\mathcal{C}$ under a scheduler σ moves from state to state, and in every state may wait for a random time. This produces a *run* which is an infinite sequence of the form $s_0\, t_0\, s_1\, t_1 \cdots$ where s_n is the n-th visited state and t_n is the time spent there. After n steps, the scheduler resolves the non-determinism based on the *history* $\mathfrak{h} = s_0\, t_0 \cdots s_{n-1}\, t_{n-1}\, s_n$ as follows.

▶ **Definition 4** (Scheduler). A scheduler[1] for an IMC $\mathcal{C} = (S, \mathbb{A}\mathrm{ct}^\tau, \hookrightarrow, \rightsquigarrow, s_0)$ is a measurable[2] function $\sigma : (S \times \mathbb{R}_{\geq 0})^* \times S \to S$ such that for each history $\mathfrak{h} = s_0\, t_0\, s_1 \cdots s_n$ with $\mathrm{succ}_\tau(s_n) \neq \emptyset$ we have $\sigma(\mathfrak{h}) \in \mathrm{succ}_\tau(s_n)$. The set of all schedulers for $\mathcal{C}$ is denoted by $\mathfrak{S}(\mathcal{C})$.

---

[1] For the sake of simplicity, we only consider deterministic schedulers in this paper.

[2] More precisely, $\sigma^{-1}(s)$ is measurable in the product topology of the discrete topology on S and the Borel topology on $\mathbb{R}_{\geq 0}$.

The decision of the scheduler $\sigma(\mathfrak{h})$ determines t_n and s_{n+1} as follows. If $\mathrm{succ}_\tau(s_n) \neq \emptyset$, then the run proceeds immediately, i.e. in time $t_n := 0$, to the state $s_{n+1} := \sigma(\mathfrak{h})$. Otherwise, if $\mathrm{succ}_\tau(s_n) = \emptyset$, then only Markovian transitions are available in s_n. In such a case, the run moves to a randomly chosen next state s_{n+1} with probability $\mathbf{P}(s_n, s_{n+1})$ after waiting for a random time t_n chosen according to the exponential distribution with the rate $\mathbf{E}(s_n)$.

One of the fundamental problems in verification and performance analysis of continuous-time stochastic systems is the time-bounded reachability. Given a set of goal states $G \subseteq S$ and a time bound $T \in \mathbb{R}_{\geq 0}$, the *value of time-bounded reachability* is defined as $\sup_{\sigma \in \mathfrak{S}(\mathcal{C})} \mathcal{P}^\sigma_{\mathcal{C}}\left[\Diamond^{\leq T} G\right]$ where $\mathcal{P}^\sigma_{\mathcal{C}}\left[\Diamond^{\leq T} G\right]$ denotes the probability that a run of $\mathcal{C}$ under the scheduler σ visits a state of G before time T. The pivotal problem in the algorithmic analysis of IMC is to compute this value together with a scheduler that achieves the supremum. As the value is not rational in most cases, the aim is to provide an efficient approximation algorithm and compute an ε-optimal scheduler. The value of time-bounded reachability can be approximated up to a given error tolerance $\varepsilon > 0$ in time $\mathcal{O}(|S|^2 \cdot (\lambda T)^2/\varepsilon)$ [29], where λ is the maximal rate of $\mathcal{C}$, and the procedure also yields an ε-optimal scheduler. We generalize both the notion of the value as well as approximation algorithms to the setting of *open* IMC, i.e. those that are not closed, and motivate this extension in the next section.

3 Compositional Verification

In this section we turn our attention to the central questions studied in this paper. How can we decide how well an IMC component $\mathcal{C}$ performs (w.r.t. time-bounded reachability) when acting in parallel with an unknown environment? And how to control the component to establish a guarantee as high as possible?

Speaking thus far in vague terms, this amounts to finding a scheduler σ for $\mathcal{C}$ which maximizes the probability of reaching a target set G before T no matter what environment E is composed with $\mathcal{C}$. As we are interested in compositional modeling using IMC, the environments are supposed to be IMC with the same external actions as $\mathcal{C}$ (thus resolving the external non-determinism of $\mathcal{C}$). We also need to consider all resolutions of the internal non-determinism of E as well as the non-determinism arising from synchronization of $\mathcal{C}$ and E using another scheduler π. So we are interested in the following value:

$$\sup_{\sigma} \inf_{E,\pi} \mathcal{P}[G \text{ is reached in composition of } \mathcal{C} \text{ and } E \text{ before } T \text{ using } \sigma \text{ and } \pi].$$

Now, let us be more formal and fix an IMC $\mathcal{C} = (S, \mathbb{A}\mathrm{ct}^\tau, \hookrightarrow, \rightsquigarrow, s_0)$. For a given environment IMC E with the same action alphabet $\mathbb{A}\mathrm{ct}^\tau$, we introduce a composition

$$\mathcal{C}(E) = (\mathcal{C} \parallel_{\mathbb{A}\mathrm{ct}} E) \setminus \mathbb{A}\mathrm{ct}$$

where all open actions are hidden, yielding a closed system. Note that the states of $\mathcal{C}(E)$ are pairs (c, e) where c is a state of $\mathcal{C}$ and e is a state of E. We consider a scheduler σ of $\mathcal{C}$ and a scheduler π of $\mathcal{C}(E)$ respecting σ on internal actions of $\mathcal{C}$. We say that π *respects* σ, denoted by $\pi \in \mathfrak{S}(\mathcal{C}(E), \sigma)$, if for every history $\mathfrak{h} = (c_0, e_0)\, t_0 \cdots t_{n-1} (c_n, e_n)$ of $\mathcal{C}(E)$ the scheduler π satisfies one of the following conditions:

- $\pi(\mathfrak{h}) = (c, e)$ where $c_n \xhookrightarrow{a} c$ and $e_n \xhookrightarrow{a} e$ (π resolves synchronization)
- $\pi(\mathfrak{h}) = (c_n, e)$ where $e_n \xhookrightarrow{\tau} e$ (π chooses a move in the environment)
- $\pi(\mathfrak{h}) = (\sigma(\mathfrak{h}_{\mathcal{C}}), e_n)$ where $\mathfrak{h}_{\mathcal{C}} = c_0 t_0 \cdots t_{n-1} c_n$ (π chooses a move in $\mathcal{C}$ according to σ).

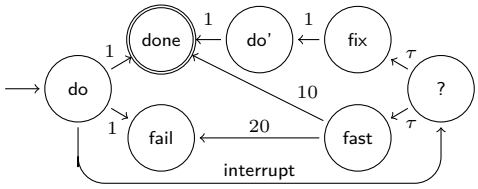
Given a set of goal states $G \subseteq S$ and a time bound $T \in \mathbb{R}_{\geq 0}$, the *value of compositional*

time-bounded reachability is defined as

$$\sup_{\sigma\in\mathfrak{S}(\mathcal{C})} \inf_{\substack{E\in\mathrm{ENV}\\ \pi\in\mathfrak{S}(\mathcal{C}(E),\sigma)}} \mathcal{P}^{\pi}_{\mathcal{C}(E)}\left[\Diamond^{\leq T} G_E\right] \qquad (*)$$

where ENV denotes the set of all IMC with the action alphabet $\mathbb{A}\mathrm{ct}^{\tau}$ and $G_E = G \times S_E$ where S_E is the set of states of E. As for the closed IMC, our goal is to efficiently approximate this value together with a maximizing scheduler. Before we present an approximation algorithm based on discretization, we illustrate some of the effects of the open system perspective.

Example. The figure on the right depicts an IMC on which we approximate the value $(*)$ for $T = 2$ and $G = \{\mathsf{done}\}$. From the initial state do, the system may go randomly either to the target done or to fail. Concurrently, the external action interrupt may switch the run to the state ?, where the scheduler σ chooses between two successors (1) the state fast allowing fast but risky run to the target and (2) the state fix that guarantees reaching the target but takes longer time. The value $(*)$ is approximately 0.47 and an optimal scheduler goes to fix only if there are more than 1.2 minutes left. Note that the probability of reaching the target in time depends on when the external action interrupt is taken. The most "adversarial" environment executes interrupt after 0.8 minutes from the start.



Results. We now formulate our main result concerning efficient approximation of the value of compositional time-bounded reachability. In fact, we provide an approximation algorithm for a restricted subclass of IMC defined by the following two assumptions:

▶ **Assumption 1.** *Each cycle contains a Markovian transition.*

This assumption is standard over all analysis techniques published for IMC [18, 27, 23, 26, 34, 19]. It implies that the probability of taking infinitely many transitions in finite time, i.e. of Zeno behavior, is zero. This is a rather natural assumption and does not restrict the modeling power much, since no real system will be able to take infinitely many transitions in finite time anyway. Furthermore, the assumed property is a compositional one, i.e. it is preserved by parallel composition and hiding.

▶ **Assumption 2.** *Internal and external actions are not enabled at the same time, i.e. for each state s, either* $\mathrm{succ}_e(s) = \emptyset$ *or* $\mathrm{succ}_\tau(s) = \emptyset$.

Note that both assumptions are met by the above mentioned example. However, Assumption 2 is not compositional; specifically, it is not preserved by applications of the hiding operator. A stronger assumption would require the environment not to trigger external actions in zero time after a state change. This is indeed implied by Assumption 2 which basically asks *internal* transitions of the component to be executed before any *external* actions are taken into account.[3] In fact, the reverse precedence cannot be implemented in real systems, if internal actions are assumed to be executed without delay. Any procedure implemented in $\mathcal{C}$ for checking the availability of external actions will involve some non-zero delay (unless one resorts to quantum effects). From a technical point of view, lifting Assumption 2 makes the studied problems considerably more involved; see Section 6 for further discussion.

[3] To see this one can construct a weak simulation relation between a system violating Assumption 2 and one satisfying it, where any state with both internal and external transitions is split into two: the first one enabling the internal transitions and a new τ to the second one only enabling the external ones.

▶ **Theorem 5.** *Let $\varepsilon > 0$ be an approximation bound and $\mathcal{C} = (S, \mathbb{A}\mathrm{ct}^\tau, \hookrightarrow, \rightsquigarrow, s_0)$ be an IMC satisfying Assumptions 1 and 2. Then one can approximate the value of compositional time-bounded reachability of $\mathcal{C}$ up to ε and compute an ε-optimal scheduler in time $\mathcal{O}(|S|^2 \cdot (\lambda T)^2/\varepsilon)$, where λ is the maximal rate of $\mathcal{C}$ and T is the reachability time-bound.*

In the remainder of the paper, we prove this theorem and discuss its restrictions. First, we introduce a new kind of real-time games, called CE games, that are played on open IMC. Then we reduce the compositional time-bounded reachability of $\mathcal{C}$ to time-bounded reachability objective in the CE game played just on the component $\mathcal{C}$ (see Proposition 6). In Section 5, we show how to reduce, using discretization, the time-bounded reachability in CE games to step-bounded reachability in discrete-time stochastic games (see Proposition 8), that in turn can be solved using simple backward propagation. Finally, we show, in Proposition 9, how to transform optimal strategies in the discretized stochastic games to ε-optimal schedulers for $\mathcal{C}$.

4 Game of Controller and Environment

In order to approximate $(*)$, the value of compositional time-bounded reachability, we turn the IMC $\mathcal{C}$ into a two-player *controller–environment game* (CE game) $\mathcal{G}$. The CE game naturally combines two approaches to real-time systems, namely the *stochastic* flow of time as present in CTMC with the *non-deterministic* flow of time as present in timed automata. The game $\mathcal{G}$ is played on the graph of an IMC $\mathcal{C}$ played by two players: **con** (controlling the component $\mathcal{C}$) and **env** (controlling/simulating the environment). In essence, **con** chooses in each state with internal transitions one of them, and **env** chooses in each state with external (and hence synchronizing) transitions either which of them should be taken, or a delay $t_e \in \mathbb{R}_{>0}$. Note that, due to Assumption 2, the players control the game in disjoint sets of states, hence $\mathcal{G}$ is a turn-based game. The internal and external transitions take zero time to be executed once chosen. If no zero time transition is chosen, the delay t_e determined by **env** competes with the Markovian transitions, i.e. with a random time sampled from the exponential distribution with the rate $\mathbf{E}(s)$. We consider time-bounded reachability objective, so the goal of **con** is to reach a given subset of states G before a given time T, and **env** opposes it.

Formally, let us fix an IMC $\mathcal{C} = (S, \mathbb{A}\mathrm{ct}^\tau, \hookrightarrow, \rightsquigarrow, s_0)$ and thus a CE game $\mathcal{G}$. A *run* of $\mathcal{G}$ is again an infinite sequence $s_0\, t_0\, s_1\, t_1 \cdots$ where $s_n \in S$ is the n-th visited state and $t_n \in \mathbb{R}_{\geq 0}$ is the time spent there. Based on the *history* $s_0\, t_0 \cdots t_{n-1}\, s_n$ went through so far, the players choose their moves as follows.

- If $\mathrm{succ}_\tau(s_n) \neq \emptyset$, the player **con** chooses a state $s_\tau \in \mathrm{succ}_\tau(s_n)$.
- Otherwise, the player **env** chooses either a state $s_e \in \mathrm{succ}_e(s_n)$, or a delay $t_e \in \mathbb{R}_{>0}$. (Note that if $\mathrm{succ}_e(s_n) = \emptyset$ only a delay can be chosen.)

Subsequently, Markovian transitions (if available) are resolved by randomly choosing a target state s_M according to the distribution $\mathbf{P}(s_n, \cdot)$ and randomly sampling a time t_M according to the exponential distribution with rate $\mathbf{E}(s_n)$. The next waiting time t_n and state s_{n+1} are given by the following rules in the order displayed.

- If $\mathrm{succ}_\tau(s_n) \neq \emptyset$ and s_τ was chosen, then $t_n = 0$ and $s_{n+1} = s_\tau$.
- If s_e was chosen, then $t_n = 0$ and $s_{n+1} = s_e$.
- If t_e was chosen then:
  - if $\mathrm{succ}_M(s_n) = \emptyset$, then $t_n = t_e$ and $s_{n+1} = s_n$;
  - if $t_e \leq t_M$, then $t_n = t_e$ and $s_{n+1} = s_n$;
  - if $t_M < t_e$, then $t_n = t_M$ and $s_{n+1} = s_M$.

According to the definition of schedulers in IMC, we formalize the choice of **con** as a *strategy* $\sigma : (S \times \mathbb{R}_{\geq 0})^* \times S \to S$ and the choice of **env** as a strategy $\pi : (S \times \mathbb{R}_{\geq 0})^* \times S \to S \cup \mathbb{R}_{>0}$. We denote by Σ and Π the sets of all strategies of the players **con** and **env**, respectively. In order to keep CE games out of Zeno behavior, we consider in Π only those strategies of the player **env** for which the induced Zeno runs have zero measure, i.e. the sum of the chosen delays diverges almost surely no matter what **con** is doing.

Given goal states $G \subseteq S$ and a time bound $T \in \mathbb{R}_{\geq 0}$, the *value of* $\mathcal{G}$ is defined as

$$\sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \mathcal{P}_{\mathcal{G}}^{\sigma,\pi}\left[\Diamond^{\leq T} G\right] \qquad (**)$$

where $\mathcal{P}_{\mathcal{G}}^{\sigma,\pi}\left[\Diamond^{\leq T} G\right]$ is the probability of all runs of $\mathcal{G}$ induced by σ and π and reaching a state of G before time T. We now show that the value of the CE game coincides with the value of compositional time-bounded reachability. This result is interesting and important as it allows us to replace unknown probabilistic behaviour with non-deterministic choices.

▶ **Proposition 6.** $(*) = (**)$, *i.e.*

$$\sup_{\sigma \in \mathfrak{S}(\mathcal{C})} \inf_{\substack{E \in \mathrm{ENV} \\ \pi \in \mathfrak{S}(\mathcal{C}(E),\sigma)}} \mathcal{P}_{\mathcal{C}(E)}^{\pi}\left[\Diamond^{\leq T} G_E\right] = \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \mathcal{P}_{\mathcal{G}}^{\sigma,\pi}\left[\Diamond^{\leq T} G\right]$$

Proof Idea. We start with the inequality $(*) \geq (**)$. Let $\sigma \in \Sigma$ $(= \mathfrak{S}(\mathcal{C}))$ and let us fix an environment E together with a scheduler $\pi \in \mathfrak{S}(\mathcal{C}(E), \sigma)$. The crucial observation is that the purpose of the environment E (controlled by π) is to choose delays of external actions (the delay is determined by a sequence of internal and Markovian actions of E executed before the external action), which is in fact similar to the role of the player **env** in the CE game. The only difference is that the environment E "chooses" the delays randomly as opposed to deterministic strategies of **env**. However, using a technically involved argument, we show how to get rid of this randomization and obtain a strategy π' in the CE game satisfying $\mathcal{P}_{\mathcal{G}}^{\sigma,\pi'}\left[\Diamond^{\leq T} G\right] \leq \mathcal{P}_{\mathcal{C}(E)}^{\pi}\left[\Diamond^{\leq T} G_E\right]$.

Concerning the second inequality $(*) \leq (**)$, we show that every strategy of **env** can be (approximately) implemented using a suitable environment together with a scheduler π. The idea is to simulate every deterministic delay, say t, chosen by **env** using a random delay tightly concentrated around t (roughly corresponding to an Erlang distribution) that is implemented as an IMC. We show that the imprecision of delays introduced by this randomization induces only negligible alteration to the value. ◀

5 Discretization

In this section we show how to approximate the value $(**)$ of the CE game up to an arbitrarily small error $\varepsilon > 0$ by reduction to a discrete-time (turn-based) stochastic game Δ.

A stochastic game Δ is played on a graph $(V, \mapsto)$ partitioned into $V_\square \uplus V_\Diamond \uplus V_\bigcirc$. A play starts in the initial vertex v_0 and forms a run $v_0 v_1 \cdots$ as follows. For a history $v_0 \cdots v_i$, the next vertex v_{i+1} satisfying $v_i \mapsto v_{i+1}$ is determined by a strategy $\sigma \in \Sigma_\Delta$ of player $\square$ if $v_i \in V_\square$ and by a strategy $\pi \in \Pi_\Delta$ of player $\Diamond$ if $v_i \in V_\Diamond$. Moreover, v_{i+1} is chosen randomly according to a fixed distribution $Prob(v_i)$ if $v_i \in V_\bigcirc$. For a formal definition, see, e.g., [17].

Let us fix a CE game $\mathcal{G}$ and a discretization step $\delta > 0$ that divides the time bound T into $N \in \mathbb{N}$ intervals of equal length (here $\delta = T/N$). We construct a discrete-time stochastic game Δ by substituting each state of $\mathcal{G}$ by a gadget of one or two vertices (as illustrated in Figure 1).[4] Intuitively, the game Δ models passing of time as follows. Each discrete step

[4] We assume w.l.o.g. that (1) states with internal transitions have no Markovian transitions available and

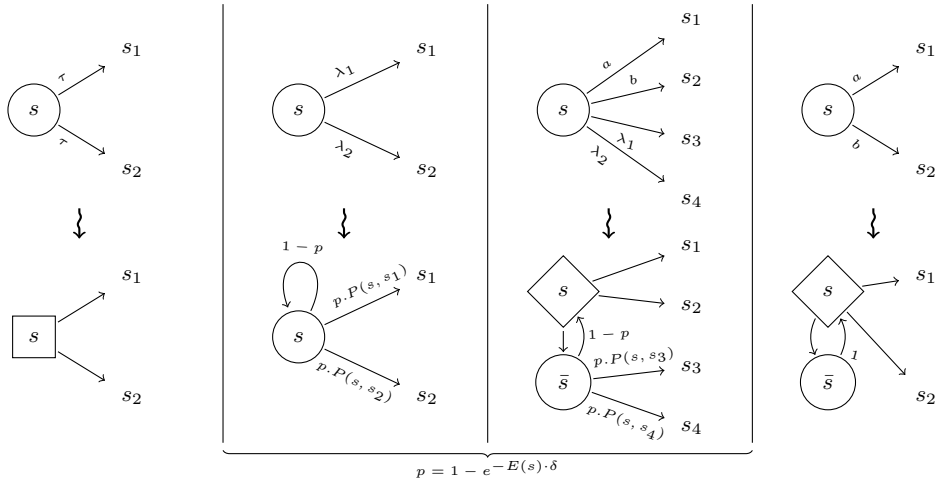


Figure 1 Four gadgets for transforming a CE game into a discrete game. The upper part shows types of states in the original CE game, the lower part shows corresponding gadgets in the transformed discrete game. In the lower part, the square-shaped, diamond-shaped and circle-shaped vertices belong to $V_\Box$, $V_\Diamond$ and $V_\bigcirc$, respectively. Binary branching is displayed only in order to simplify the figure.

"takes" either time δ or time 0. Each step from a vertex of $V_\bigcirc$ takes time δ whereas each step from vertex of $V_\Box \cup V_\Diamond$ takes zero time. The first gadget transforms internal transitions into edges of player $\Box$ taking zero time. The second gadget transforms Markovian transitions into edges of player $\bigcirc$ taking time δ where the probability p is the probability that any Markovian transition is taken in $\mathcal{G}$ before time δ. The third gadget deals with states with both external and Markovian transitions available where the player $\Diamond$ decides in vertex s in zero time whether an external transition is taken or whether the Markovian transitions are awaited in $\overline{s}$ for time δ. The fourth gadget is similar, but no Markovian transition can occur and from $\overline{s}$ the play returns into s with probability 1.

Similarly to $(*)$ and $(**)$, we define the *value of the discrete-time game* Δ as

$$\sup_{\sigma\in\Sigma_\Delta} \inf_{\pi\in\Pi_\Delta} \mathcal{P}_\Delta^{\sigma,\pi}\left[\Diamond^{\#_\bigcirc \leq N} G\right] \qquad (***)$$

where $\mathcal{P}_\Delta^{\sigma,\pi}\left[\Diamond^{\#_\bigcirc \leq N} G\right]$ is the probability of all runs of Δ induced by σ and π that reach G before taking more than N steps from vertices in $V_\bigcirc$. According to the intuition above, such a step bound corresponds to a time bound $N \cdot \delta = T$.

We say that a strategy *is counting* if it only considers the last vertex and the current count $\#_\bigcirc$ of steps taken from vertices in $V_\bigcirc$. We may represent it as a function $V \times \{0, \ldots, N\} \to V$ since it is irrelevant what it does after more than N steps.

▶ **Lemma 7.** *There are counting strategies optimal in* $(***)$. *Moreover, they can be computed together with* $(***)$ *in time* $\mathcal{O}(N|V|^2)$.

We now show that the value $(***)$ of the discretized game Δ approximates the value $(**)$ of the CE game $\mathcal{G}$ and give the corresponding error bound.

(2) every state has at least one outgoing transition.This is no restriction since (1) Markovian transitions are never taken in such states and (2) any state without transitions can be endowed with a Markovian self-loop transition without changing the time-bounded reachability.

▶ **Proposition 8** (Error bound). *For every approximation bound $\varepsilon > 0$ and discretization step $\delta \leq \varepsilon/(\lambda^2 T)$ where $\lambda = \max_{s \in S} \mathbf{E}(s)$, the value $(***)$ induced by δ satisfies*

$$(***) \;\leq\; (**) \;\leq\; (***) + \varepsilon.$$

Proof Idea. The proof is inspired by the techniques for closed IMC [29]. Yet, there are several new issues to overcome, caused mainly by the fact that the player **env** in the CE game may choose an arbitrary real delay $t_e > 0$ (so **env** has uncountably many choices). The discretized game Δ is supposed to simulate the original CE game but restricts possible behaviors as follows: (1) Only one Markovian transition is allowed in any interval of length δ. (2) The delay t_e chosen by player $\Diamond$ (which simulates the player **env** from the CE game) must be divisible by δ. We show that none of these restrictions affects the value.

ad (1) As pointed out in [29], the probability of two or more Markovian transitions occurring in an interval $[0, \delta]$ is bounded by $(\lambda\delta)^2/2$ where $\lambda = \max_{s \in S} \mathbf{E}(s)$. Hence, the probability of multiple Markovian transitions occurring in any of the discrete steps of Δ is $\leq \varepsilon$.

ad (2) Assuming that at most one Markovian transition is taken in $[0, \delta]$ in the CE game, we reduce the decision when to take external transitions to minimization of a linear function on $[0, \delta]$, which in turn is minimized either in 0, or δ. Hence, the optimal choice for the player **env** in the CE game is either to take the transitions immediately at the beginning of the interval (before the potential Markovian transition) or to wait for time δ (after the potential Markovian transition). ◀

Finally, we show how to transform an optimal counting strategy $\sigma : V \times \{0, \ldots, N\} \to V$ in the discretized game Δ into an ε-optimal scheduler $\overline{\sigma}$ in the IMC $\mathcal{C}$. For every $\mathfrak{p} = s_0\, t_0 \cdots s_{n-1}\, t_{n-1}\, s_n$ we put $\overline{\sigma}(\mathfrak{p}) = \sigma(s_n, \lceil (t_0 + \ldots + t_{n-1})/\delta \rceil)$.

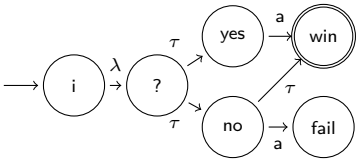
▶ **Proposition 9** (ε-optimal scheduler). *Let $\varepsilon > 0$, Δ be a corresponding discrete game, and $\overline{\sigma}$ be induced by an optimal counting strategy in Δ, then*

$$(*) \;\leq\; \inf_{\substack{E \in \mathrm{ENV} \\ \pi \in \mathfrak{S}(\mathcal{C}(E), \overline{\sigma})}} \mathcal{P}^{\pi}_{\mathcal{C}(E)}\left[\Diamond^{\leq T} G_E\right] + \varepsilon$$

This together with the complexity result of Lemma 7 finishes the proof of Theorem 5.

6 Summary, Discussion and Future Work

We discussed the computation of maximal timed bounded reachability for IMC operating in an unknown IMC environment to synchronize with. All prior analysis approaches considered closed systems, implicitly assuming that external actions do happen in zero time. Our analysis for open IMC works essentially with the opposite assumption, which is arguably more realistic. We have shown that the resulting stochastic two-player game has the same extremal values as a CE-game, where the player controlling the environment can choose exact times. The latter is approximated up to a given precision by discretization and the resulting control strategy translated back to a scheduler of the IMC achieving the bound.



Finally, we argue that lifting Assumption 2 makes analysis considerably more involved as the studied game may contain imperfect information and concurrent decisions. Let us illustrate the problems on an example. Consider an IMC depicted on the right. This IMC violates Assumption 2 in its state no. Let us fix an arbitrary environment E (controlled by π) and a scheduler σ. Since internal transitions of E take zero time, the environment must spend almost all the time in states without internal transitions. Hence, E is almost surely

in such a state when ? is entered. Assume E is in a state with the (external) action a being available. The scheduler σ wins if he chooses the internal transition to yes since the synchronizing transition a is then taken immediately, and fails if he chooses to proceed to no, as a (reasonable) scheduler π will now force synchronization on action a. If, otherwise, on entering state ?, E is in a state without the action a being available, the scheduler σ fails if he chooses yes because a (reasonable) environment never synchronizes, and wins if he chooses no since the environment E cannot immediately synchronize and the τ transition is taken. Note that the scheduler σ cannot observe whether a is available in the current state of E. As this is crucial for the further evolution of the game from state ?, the game is intrinsically of imperfect information.

We conjecture that solving even this special case of imperfect information games is PSPACE-hard. Yet, the complexity might only increase in the number of internal transitions that can be taken in a row. For systems, where a bound on the length of internal transition sequences can be assumed, this problem would then still be feasible.

Acknowledgement

The work has been supported by the Czech Science Foundation, grant No. P202/12/P612 (T. Brázdil, J. Křetínský and V. Řehák) and No. 102/09/H042 (J. Krčál) and by the DFG as part of the SFB/TR 14 AVACS, by the DFG/NWO project ROCKS, and by the EU FP7 project MEALS, grant agreement no. 295261. (H. Hermanns).

References

1 K. Apt and E. Grädel, editors. *Lectures in Game Theory for Computer Scientists*. Cambridge, 2011.

2 C. Baier, H. Hermanns, J.-P. Katoen, and B.R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *Theor. Comp. Sci.*, 345(1):2–26, 2005.

3 J.A. Bergstra, A. Ponse, and S.A. Smolka, editors. *Handbook of Process Algebra*. Elsevier, 2001.

4 E. Böde, M. Herbstritt, H. Hermanns, S. Johr, T. Peikenkamp, R. Pulungan, J. Rakow, R. Wimmer, and B. Becker. Compositional dependability evaluation for STATEMATE. *IEEE Trans. on Soft. Eng.*, 35(2):274–292, 2009.

5 H. Boudali, P. Crouzen, B.R. Haverkort, M. Kuntz, and M. I. A. Stoelinga. Architectural dependability evaluation with Arcade. In *Proc. of DSN*, pages 512–521. IEEE, 2008.

6 H. Boudali, P. Crouzen, and M. Stoelinga. A rigorous, compositional, and extensible framework for dynamic fault tree analysis. *IEEE Trans. on DSC*, 7(2):128–143, 2010.

7 P. Bouyer and V. Forejt. Reachability in stochastic timed games. In *Proc. of ICALP*, volume 5556 of *LNCS*, pages 103–114. Springer, 2009.

8 M. Bozzano, A. Cimatti, J.-P. Katoen, V.Y. Nguyen, T. Noll, and M. Roveri. Safety, dependability and performance analysis of extended AADL models. *The Computer Journal*, 54(5):754–775, 2011.

9 T. Brázdil, V. Forejt, J. Krčál, J. Křetínský, and A. Kučera. Continuous-time stochastic games with time-bounded reachability. In *Proc. of FSTTCS*, volume 4 of *LIPIcs*, pages 61–72. Schloss Dagstuhl, 2009.

10 T. Brázdil, H. Hermanns, J. Krčál, J. Křetínský, and V. Řehák. Verification of open interactive markov chains. Technical Report FIMU-RS-2012-04, Faculty of Informatics MU, 2012.

11 T. Brázdil, J. Krčál, J. Křetínský, A. Kučera, and V. Řehák. Stochastic real-time games with qualitative timed automata objectives. In *Proc. of CONCUR*, volume 6269 of *LNCS*, pages 207–221. Springer, 2010.

12 P. Buchholz and I. Schulz. Numerical Analysis of Continuous Time Markov Decision processes over Finite Horizons. *Computers and Operations Research*, 38:651–659, 2011.

13 K. Chatterjee and T.A. Henzinger. A survey of stochastic ω-regular games. *J. Comput. Syst. Sci.*, 78(2):394–413, 2012.

14 T. Chen, V. Forejt, M.Z. Kwiatkowska, D. Parker, and A. Simaitis. Automatic verification of competitive stochastic systems. In *Proc. of TACAS*, volume 7214 of *LNCS*, pages 315–330. Springer, 2012.

15 N. Coste, H. Hermanns, E. Lantreibecq, and W. Serwe. Towards performance prediction of compositional models in industrial GALS designs. In *Proc. of CAV*, volume 5643, pages 204–218. Springer, 2009.

16 M.-A. Esteve, J.-P. Katoen, V.Y. Nguyen, B. Postma, and Y. Yushtein. Formal correctness, safety, dependability and performance analysis of a satellite. In *Proc. of ICSE*. ACM and IEEE press, 2012.

17 J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1996.

18 H. Garavel, R. Mateescu, F. Lang, and W. Serwe. CADP 2006: A toolbox for the construction and analysis of distributed processes. In *Proc. of CAV*, volume 4590 of *LNCS*, pages 158–163. Springer, 2007.

19 D. Guck, T. Han, J.-P. Katoen, , and M.R. Neuhäußer. Quantitative timed analysis of interactive markov chains. In *NFM*, volume 7226 of *LNCS*, pages 8–23. Springer, 2012.

20 E.M. Hahn, G. Norman, D. Parker, B. Wachter, and L. Zhang. Game-based abstraction and controller synthesis for probabilistic hybrid systems. In *QEST*, pages 69–78, 2011.

21 B.R. Haverkort, M. Kuntz, A. Remke, S. Roolvink, and M.I.A. Stoelinga. Evaluating repair strategies for a water-treatment facility using Arcade. In *Proc. of DSN*, pages 419–424, 2010.

22 H. Hermanns and S. Johr. Uniformity by construction in the analysis of nondeterministic stochastic systems. In *DSN*, pages 718–728. IEEE Computer Society, 2007.

23 H. Hermanns and S. Johr. May we reach it? Or must we? In what time? With what probability? In *Proc. of MMB*, pages 125–140. VDE Verlag, 2008.

24 H. Hermanns and J.-P. Katoen. The how and why of interactive Markov chains. In *Proc. of FMCO*, volume 6286 of *LNCS*, pages 311–337. Springer, 2009.

25 H. Hermanns, J.-P. Katoen, M. R. Neuhäußer, and L. Zhang. GSPN model checking despite confusion. Technical report, RWTH Aachen University, 2010.

26 J.-P. Katoen, D. Klink, and M. R. Neuhäußer. Compositional abstraction for stochastic systems. In *Proc. of FORMATS*, volume 5813 of *LNCS*, pages 195–211. Springer, 2009.

27 J.-P. Katoen, I.S. Zapreev, E.M. Hahn, H. Hermanns, and D.N. Jansen. The ins and outs of the probabilistic model checker MRMC. *Performance Evaluation*, 68(2):90–104, 2011.

28 O. Kupferman and M. Vardi. Module checking. In *CAV*, volume 1102 of *LNCS*, pages 75–86. Springer, 1996.

29 M.R. Neuhäußer. *Model checking nondeterministic and randomly timed systems*. PhD thesis, University of Twente, 2010.

30 M.N. Rabe and S. Schewe. Finite optimal control for time-bounded reachability in CTMDPs and continuous-time Markov games. *Acta Informatica*, 48(5-6):291–315, 2011.

31 P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1), 1989.

32 J. Sproston. Discrete-time verification and control for probabilistic rectangular hybrid automata. In *QEST*, pages 79–88, 2011.

33 P. Černý, K. Chatterjee, T.A. Henzinger, A. Radhakrishna, and R. Singh. Quantitative synthesis for concurrent programs. In *CAV*, pages 243–259, 2011.

34 L. Zhang and M.R. Neuhäußer. Model checking interactive Markov chains. In *Proc. of TACAS*, volume 6015 of *LNCS*, pages 53–68. Springer, 2010.

Paper G:

Compositional Verification and Optimization of Interactive Markov Chains

Holger Hermanns, Jan Krčál, and Jan Křetínský

This paper has been published in Pedro R. D'Argenio and Hernán Melgratti (eds.): Proceedings of Concurrency Theory, 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Lecture Notes in Computer Science. [BHK+12]

Summary

Interactive Markov chains are a slight extension of the widespread continuous-time Markov decision processes. Their main advantage is compositionality, which facilitates hierarchical design and analysis of systems. However, analysis of interactive Markov chains has so far been limited to closed controllable systems or open, but uncontrollable systems. The first step towards the analysis of controllable systems operating in an unknown environment is taken in [BHK+12] (Paper F). However, the environment considered there is completely unknown and the systems under consideration are structurally limited. Here we lift the assumption on the systems, introduce a framework for specifying environments (or generally IMC) and give an algorithm for time-bounded reachability when the system under analysis operates in an unknown environment conforming to a given specification. The specifications are given as modal continuous time automata—a modal extension of time-automata-like framework where we employ continuous time constraints instead of hard constant guards. This enables the first truly compositional verification and assume-guarantee reasoning in the stochastic continuous-time setting.

Author's contribution: 60 %

- design of the specification framework of modal continuous time automata,
- group discussions of the problem,
- writing most of the paper (excluding Section 6),
- discussing the proofs,
- writing most of the proofs concerning the reduction of the problem to the game setting.

Compositional Verification and Optimization of Interactive Markov Chains*

Holger Hermanns[1], Jan Krčál[2], and Jan Křetínský[2,3]

[1] Saarland University – Computer Science, Saarbrücken, Germany
[2] Faculty of Informatics, Masaryk University, Czech Republic
[3] Institut für Informatik, Technical University Munich, Germany

Abstract. Interactive Markov chains (IMC) are compositional behavioural models extending labelled transition systems and continuous-time Markov chains. We provide a framework and algorithms for compositional verification and optimization of IMC with respect to time-bounded properties. Firstly, we give a specification formalism for IMC. Secondly, given a time-bounded property, an IMC component and the assumption that its unknown environment satisfies a given specification, we synthesize a scheduler for the component optimizing the probability that the property is satisfied in any such environment.

1 Introduction

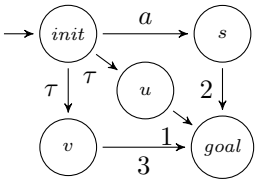
The ever increasing complexity and size of systems together with software reuse strategies naturally enforce the need for component based system development. For the same reasons, checking reliability and optimizing performance of such systems needs to be done in a *compositional* way. The task is to get useful guarantees on the behaviour of a component of a larger system. The key idea is to incorporate assumptions on the rest of the system into the verification process. This *assume-guarantee reasoning* is arguably a successful divide-and-conquer technique in many contexts [MC81,AH96,HMP01].

In this work, we consider a continuous-time stochastic model called *interactive Markov chains* (IMC). First, we give a language for expressing assumptions about IMC. Second, given an IMC, an assumption on its environment and a property of interest, we synthesize a controller of the IMC that optimizes the guarantee, and we compute this optimal guarantee, too.

Interactive Markov chains are behavioural models of probabilistic systems running in continuous real time appropriate for the component-based approach [HK09]. IMC have a well-understood compositional theory rooted in process algebra, and are in use as semantic backbones for dynamic fault trees,

* The work has received support from the Czech Science Foundation, project No. P202/12/G061, from the German Science Foundation DFG as part of SFB/TR 14 AVACS, and by the EU FP7 Programme under grant agreement no. 295261 (MEALS) and 318490 (SENSATION).

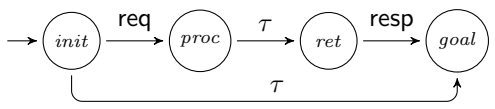
architectural description languages, generalized stochastic Petri nets and Statemate extensions, see [HK09] for a survey. IMC are applied in a large spectrum of practical applications, ranging from water treatment facilities [HKR+10] to ultra-modern satellite designs [EKN+12].



IMC arise from classical labelled transition systems by incorporating the possibility to change state according to a random *delay* governed by a negative exponential distribution with a given rate, see transitions labelled 1, 2 and 3 in the figure. Apart from delay expirations, state transitions may be triggered by the execution of *internal* (τ) actions or *external* (synchronization) actions. Internal actions are assumed to happen instantaneously and therefore take precedence over delay transitions. External actions are the process algebraic means for interaction with other components, see a in the figure. By dropping the delay transitions, labelled transition systems are regained in their entirety. Dropping action-labelled transitions instead yields continuous-time Markov chains – one of the most used performance and reliability models.

The fundamental problem in the analysis of IMC is that of *time-bounded reachability*. It is the problem to approximate the probability that a given set of states is reached within a given deadline. We illustrate the compositional setting of this problem in the following examples.

Examples. In the first example, consider the IMC $\mathcal{C}$ from above and an unknown environment $\mathcal{E}$ with no assumptions. Either $\mathcal{E}$ is initially not ready to synchronize on the external action a and thus one of the internal actions is taken, or $\mathcal{E}$ is willing to synchronize on a at the beginning. In the latter case, whether τ or a happens is resolved non-deterministically. Since this is out of control of $\mathcal{C}$, we must assume the worst case and let the environment decide which of the two options will happen. For more details on this design choice, see [BHK+12]. If there is synchronization on a, the probability to reach *goal* within time $t = 1.5$ is $1 - e^{-2t} \approx 0.95$. Otherwise, $\mathcal{C}$ is given the choice to move to u or v. Naturally, v is the choice maximizing the chance to get to *goal* on time as it has a higher rate associated. In this case the probability amounts to $1 - e^{-3t} \approx 0.99$, while if u were chosen, it would be only 0.78. Altogether, the guaranteed probability is 95% and the strategy of $\mathcal{C}$ is to choose v in *init*.



The example depicted on the right illustrates the necessity of assumptions on the environment: As it is, the environment can drive the component to state *ret* and let it get stuck there by not synchronising on resp ever. Hence no better guarantee than 0 can be derived. However, this changes if we know some specifics about the behaviour of the environment: Let us assume that we know that once synchronization on req occurs, the environment must be ready to synchronise on resp within some random time according to, say, an exponential distribution with rate 2. Under this assumption, we are able to derive a guarantee of 95%, just as in the previous example.

Observe the form of the time constraint we imposed in the last example: "within a random time distributed according to Exp(2)" or symbolically $\Diamond_{\leq Exp(2)}\varphi$. We call this a *continuous time constraint*. If a part of the environment is e.g. a model of a communication network, it is clear we cannot impose hard bounds (discrete time constraints) such as "within 1.5" as in e.g. a formula of MTL $\Diamond_{\leq 1.5}\varphi$. Folklore tells us that messages might get delayed for longer than that. Yet we want to express high assurance that they arrive on time. In this case one might use e.g. a formula of CSL $Pr_{\geq 0.95}(\Diamond_{\leq 1.5}\varphi)$. However, consider now a system with two transitions labelled with resp in a row. Then this CSL formula yields only a zero guarantee. By splitting the time 1.5 in halves, the respective $Pr_{\geq 0.77}(\Diamond_{\leq 0.75}\varphi)$ yields only the guarantee $0.77^2 = 0.60$. The actual guarantee 0.80 is given by the convolution of the two exponential distributions and as such can be exactly obtained from our continuous time constraint $\Diamond_{\leq Exp(2)}\varphi$.

Our contribution is the following:

1. We introduce a specification formalism to express assumptions on continuous-time stochastic systems. The novel feature of the formalism are the continuous time constraints, which are vital for getting guarantees with respect to time-bounded reachability in IMC.
2. We incorporate the assume-guarantee reasoning to the IMC framework. We show how to synthesize ϵ-optimal schedulers for IMC in an *unknown environment satisfying a given specification* and approximate the respective guarantee.

In our recent work [BHK+12] we considered a very restricted setting of the second point. Firstly, we considered no assumptions on the environment as the environment of a component might be entirely unknown in many scenarios. Secondly, we were restricted to IMC that never enable internal and external transitions at the same state. This was also a severe limitation as this property is not preserved during the IMC composition process and restricts the expressivity significantly. Both examples above violate this assumption. In this paper, we lift the assumption.

Each of the two extensions shifts the solution methods from complete information stochastic games to (one-sided) *partial observation* stochastic games, where we need to solve the quantitative reachability problem. While this is undecidable in general, we reduce our problem to a game played on an *acyclic graph* and show how to solve our problem in exponential time. (Note that even the qualitative reachability in the acyclic case is PSPACE-hard [CD10].)

Related work. The *synthesis* problem is often stated as a game where the first player controls a component and the second player simulates an environment [RW89]. Model checking of *open* systems, i.e. operating in an unknown environment, has been proposed in [KV96]. There is a body of work on *assume-guarantee* reasoning for parallel composition of *real-time* systems [TAKB96,HMP01]. Lately, games with *stochastic continuous-time* have gained attention, for a very general class see [BF09]. While the second player models possible schedulers of the environment, the structure of the environment is fixed there and the veri-

fication is thus not compositional. The same holds for [Spr11,HNP+11], where time is under the control of the components.

A compositional framework requires means for specification of systems. A specification can be also viewed as an *abstraction* of a set of systems. Three valued abstractions stemming from [LT88] have also been applied to the timed setting, namely in [KKLW07] to continuous-time Markov chains (IMC with no non-determinism), or in [KKN09] to IMC. Nevertheless, these abstractions do not allow for constraints on time distributions. Instead they would employ abstractions on transition probabilities. Further, a compositional framework with timed specifications is presented in [DLL+12]. This framework explicitly allows for time constraints. However, since the systems under consideration have non-deterministic flow of time (not stochastic), the natural choice was to only allow for discrete (not continuous) time constraints.

Although IMC support compositional design very well, analysis techniques for IMC proposed so far (e.g. [KZH+11,KKN09,ZN10,GHKN12] are not compositional. They are all bound to the assumption that the analysed IMC is a *closed* system, i.e. it does not depend on interaction with the environment (all actions are internal). Some preliminary steps to develop a framework for synthesis of controllers based on models of hardware and control requirements have been taken in [Mar11]. The first attempt at compositionality is our very recent work [BHK+12] discussed above.

Algorithms for the *time-bounded reachability* problem for closed IMC have been given in [ZN10,BS11,HH13] and compositional abstraction techniques to compute it are developed in [KKN09]. In the closed interpretation, IMC have some similarities with continuous-time Markov decision processes. For this formalism, algorithms for time-bounded reachability are developed in [BHKH05,BS11].

2 Interactive Markov Chains

In this section, we introduce the formalism of interactive Markov chains together with the standard way to compose them. We denote by $\mathbb{N}$, $\mathbb{R}_{>0}$, and $\mathbb{R}_{\geq 0}$ the sets of positive integers, positive real numbers and non-negative real numbers, respectively. Further, let $\mathcal{D}(S)$ denote the set of probability distributions over the set S.

Definition 1 (IMC). *An* interactive Markov chain (IMC) *is a quintuple* $\mathcal{C} = (S, \mathbb{A}\mathrm{ct}^\tau, \hookrightarrow, \rightsquigarrow, s_0)$ *where* S *is a finite set of* states, $\mathbb{A}\mathrm{ct}^\tau$ *is a finite set of* actions *containing a designated* internal action τ, $s_0 \in S$ *is an* initial *state,*

- $\hookrightarrow \subseteq S \times \mathbb{A}\mathrm{ct}^\tau \times S$ *is an* interactive transition *relation, and*
- $\rightsquigarrow \subseteq S \times \mathbb{R}_{>0} \times S$ *is a* Markovian transition *relation.*

Elements of $\mathbb{A}\mathrm{ct} := \mathbb{A}\mathrm{ct}^\tau \setminus \{\tau\}$ are called *external actions*. We write $s \overset{a}{\hookrightarrow} t$ whenever $(s, a, t) \in \hookrightarrow$, and $s \overset{\lambda}{\rightsquigarrow} t$ whenever $(s, \lambda, t) \in \rightsquigarrow$ where λ is called a *rate* of the transition. We say that an external action a, or internal τ, or Markovian transition is *available* in s, if $s \overset{a}{\hookrightarrow} t$, $s \overset{\tau}{\hookrightarrow} t$ or $s \overset{\lambda}{\rightsquigarrow} t$ for some t (and λ), respectively.

IMC are well suited for compositional modelling, where systems are built out of smaller ones using standard composition operators. *Parallel composition* $\|_A$ over a *synchronization alphabet* A produces a product of two IMC with transitions given by the rules

(PC1) $(s_1, s_2) \overset{a}{\hookrightarrow} (s_1', s_2')$ for each $s_1 \overset{a}{\hookrightarrow} s_1'$ and $s_2 \overset{a}{\hookrightarrow} s_2'$ and $a \in A$,
(PC2, PC3) $(s_1, s_2) \overset{a}{\hookrightarrow} (s_1', s_2)$ for each $s_1 \overset{a}{\hookrightarrow} s_1'$ and $a \notin A$, and symmetrically,
(PC4, PC5) $(s_1, s_2) \overset{\lambda}{\rightsquigarrow} (s_1', s_2)$ for each $s_1 \overset{\lambda}{\rightsquigarrow} s_1'$, and symmetrically.

Further, *hiding* $\setminus A$ an alphabet A, yields a system, where each $s \overset{a}{\hookrightarrow} s'$ with $a \notin A$ is left as it is, and each $s \overset{a}{\hookrightarrow} s'$ with $a \in A$ is replaced by internal $s \overset{\tau}{\hookrightarrow} s'$.

Hiding $\setminus \mathbb{A}\mathrm{ct}$ thus yields a *closed* IMC, where external actions do not appear as transition labels (i.e. $\hookrightarrow \subseteq S \times \{\tau\} \times S$). A closed IMC (under a scheduler σ, see below) moves from state to state and thus produces a *run* which is an infinite sequence of the form $s_0\, t_1\, s_1\, t_2\, s_2 \cdots$ where s_n is the n-th visited state and t_n is the time of arrival to s_n. After n steps, the scheduler resolves the non-determinism among internal τ transitions based on the *path* $\mathfrak{p} = s_0\, t_1 \cdots t_n\, s_n$.

Definition 2 (Scheduler). *A* scheduler *of an IMC* $\mathcal{C} = (S, \mathbb{A}\mathrm{ct}^\tau, \hookrightarrow, \rightsquigarrow, s_0)$ *is a measurable function* $\sigma : (S \times \mathbb{R}_{\geq 0})^* \times S \to \mathcal{D}(S)$ *such that for each path* $\mathfrak{p} = s_0\, t_1\, s_1 \cdots t_n\, s_n$ *with* s_n *having* τ *available,* $\sigma(\mathfrak{p})(s) > 0$ *implies* $s_n \overset{\tau}{\hookrightarrow} s$. *The set of all schedulers for* $\mathcal{C}$ *is denoted by* $\mathfrak{S}(\mathcal{C})$.

The decision of the scheduler $\sigma(\mathfrak{p})$ determines t_{n+1} and s_{n+1} as follows. If s_n has available τ, then the run proceeds immediately, i.e. at time $t_{n+1} := t_n$, to a state s_{n+1} randomly chosen according to the distribution $\sigma(\mathfrak{p})$. Otherwise, only Markovian transitions are available in s_n. In such a case, after waiting for a random time t chosen according to the exponential distribution with the rate $\mathrm{R}(s_n) = \sum_{s_n \overset{\lambda}{\rightsquigarrow} s'} \lambda$, the run moves at time $t_{n+1} := t_n + t$ to a randomly chosen next state s_{n+1} with probability λ / r where $s_n \overset{\lambda}{\rightsquigarrow} s_{n+1}$. This defines a probability space $(\mathbb{R}\mathrm{uns}, \mathcal{F}, \mathcal{P}_{\mathcal{C}}^{\sigma})$ over the runs in the standard way [ZN10].

3 Time-Bounded Reachability

In this section, we introduce the studied problems. One of the fundamental problems in verification and performance analysis of continuous-time stochastic systems is time-bounded reachability. Given a *closed* IMC $\mathcal{C}$, a set of goal states $G \subseteq S$ and a time bound $T \in \mathbb{R}_{\geq 0}$, the *value of time-bounded reachability* is defined as $\sup_{\sigma \in \mathfrak{S}(\mathcal{C})} \mathcal{P}_{\mathcal{C}}^{\sigma}\left[\Diamond^{\leq T} G\right]$ where $\mathcal{P}_{\mathcal{C}}^{\sigma}\left[\Diamond^{\leq T} G\right]$ denotes the probability that a run of $\mathcal{C}$ under the scheduler σ visits a state of G before time T. We have seen an example in the introduction. A standard assumption over all analysis techniques published for IMC [KZH+11,KKN09,ZN10,GHKN12] is that each cycle contains a Markovian transition. It implies that the probability of taking infinitely many transitions in finite time, i.e. of Zeno behaviour, is zero. One can ε-approximate the value and compute the respective scheduler in time $\mathcal{O}(\lambda^2 T^2 / \varepsilon)$ [ZN10] recently improved to $\mathcal{O}(\sqrt{\lambda^3 T^3 / \varepsilon})$ [HH13].

For an *open* IMC to be put in parallel with an unknown environment, the optimal scheduler is computed so that it optimizes the guarantee against all possible environments. Formally, for an IMC $\mathcal{C} = (C, \mathbb{A}\mathrm{ct}^\tau, \hookrightarrow, \rightsquigarrow, c_0)$ and an environment IMC $\mathcal{E}$ with the same action alphabet $\mathbb{A}\mathrm{ct}^\tau$, we introduce a composition $\mathcal{C}|\mathcal{E} = (\mathcal{C} \parallel_{\mathbb{A}\mathrm{ct}} \mathcal{E}) \setminus \mathbb{A}\mathrm{ct}$ where all open actions are hidden, yielding a closed system. In order to compute guarantees on $\mathcal{C}|\mathcal{E}$ provided we use a scheduler σ in $\mathcal{C}$, we consider schedulers π of $\mathcal{C}|\mathcal{E}$ that *respect* σ on the internal actions of $\mathcal{C}$, written $\pi \in \mathfrak{S}_\sigma(\mathcal{C}|\mathcal{E})$; the formal definition is below. The *value of compositional time-bounded reachability* is then defined in [BHK+12] as

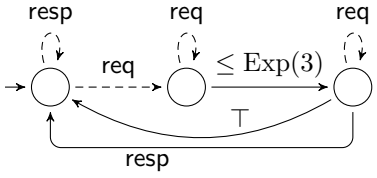
$$\sup_{\sigma \in \mathfrak{S}(\mathcal{C})} \inf_{\substack{\mathcal{E} \in \mathrm{ENV} \\ \pi \in \mathfrak{S}_\sigma(\mathcal{C}|\mathcal{E})}} \mathcal{P}^\pi_{\mathcal{C}|\mathcal{E}}\left[\Diamond^{\leq T} G\right]$$

where ENV denotes the set of all IMC with the action alphabet $\mathbb{A}\mathrm{ct}^\tau$ and $\Diamond^{\leq T} G$ is the set of runs that reach G in the first component before T. Now π *respects* σ on internal actions of $\mathcal{C}$ if for every path $\mathfrak{p} = (c_0, e_0)\, t_1 \cdots t_n (c_n, e_n)$ of $\mathcal{C}|\mathcal{E}$ there is $p \in [0,1]$ such that for each internal transition $c_n \xhookrightarrow{\tau} c$ of $\mathcal{C}$, we have $\pi(\mathfrak{p})(c, e_n) = p \cdot \sigma(\mathfrak{p}_\mathcal{C})(c)$. Here $\mathfrak{p}_\mathcal{C}$ is the projection of $\mathfrak{p}$ where σ can only see the path of moves in $\mathcal{C}$ and not in which states $\mathcal{E}$ is. Formally, we define *observation* of a path $\mathfrak{p} = (c_0, e_0)\, t_1 \cdots t_n (c_n, e_n)$ as $\mathfrak{p}_\mathcal{C} = c_0 t_1 \cdots t_n c_n$ where each maximal consecutive sequence $t_i\, c_i \cdots t_j\, c_j$ with $c_k = c_i$ for all $i \leq k \leq j$ is rewritten to $t_i\, c_i$. This way, σ ignores precisely the internal steps of $\mathcal{E}$.

3.1 Specifications of environments

In the second example in the introduction, without any assumptions on the environment only zero guarantees could be derived. The component was thus indistinguishable from an entirely useless one. In order to get a better guarantee, we introduce a formalism to specify assumptions on the behaviour of environments.

Example 1. In the mentioned example, if we knew that after an occurrence of req the environment is ready to synchronize on resp in time distributed according to Exp(3) or faster, we would be able to derive a guarantee of 0.26. We will depict this assumption as shown below.



The dashed arrows denote *may* transitions, which may or may not be available, whereas the full arrows denote *must* transitions, which the environment is ready to synchronize on. Full arrows are further used for time transitions.

Although such a system resembles a timed automaton, there are several fundamental differences. Firstly, the time constraints are given by probability distributions instead of constants. Secondly, there is only one clock that, moreover, gets reset whenever the state is *changed*. Thirdly, we allow modalities of may and must transitions. Further, as usual with timed or stochastic specifications, we require determinism.

Definition 3 (MCA syntax). *A* continuous time constraint *is either* $\top$ *or of the form* $\bowtie d$ *with* $\bowtie \in \{\leq, \geq\}$ *and* d *a continuous distribution. We denote the set of all continuous time constraints by* $\mathcal{CTC}$*. A* modal continuous-time automaton (MCA) *over* Σ *is a tuple* $\mathcal{S} = (Q, q_0, \dashrightarrow, \longrightarrow, \rightsquigarrow)$*, where*

- Q *is a non-empty finite set of* locations *and* $q_0 \in Q$ *is an* initial location*,*
- $\longrightarrow, \dashrightarrow : Q \times \Sigma \to Q$ *are* must *and* may *transition functions, respectively, satisfying* $\longrightarrow \subseteq \dashrightarrow$*,*
- $\rightsquigarrow : Q \to \mathcal{CTC} \times Q$ *is a* time flow *function.*

We have seen an example of an MCA in the previous example. Note that upon taking req from the first state, the waiting time is chosen and the waiting starts. On the other hand, when req *self-loop* is taken in the middle state, the waiting process is not restarted, but continues on the background independently.[(1)] We introduce this independence as a useful feature to model properties as "response follows within some time after request" in the setting with concurrently running processes. Further, we have transitions under $\top$ corresponding to "> 0", meaning there is no restriction on the time distribution except that the transition takes non-zero time. We formalize this in the following definition. With other respects, the semantics of may and must transitions follows the standards of modal transition systems [LT88].

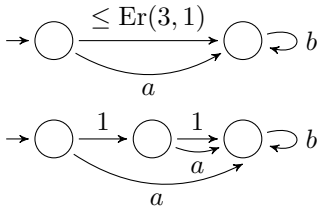
Definition 4 (MCA semantics). *An IMC* $\mathcal{E} = (E, \mathbb{A}\mathrm{ct}^\tau, \hookrightarrow, \rightsquigarrow, e_0)$ conforms *to an MCA specification* $\mathcal{S} = (Q, q_0, \dashrightarrow, \longrightarrow, \rightsquigarrow)$*, written* $\mathcal{E} \models \mathcal{S}$*, if there is a* satisfaction relation $\mathcal{R} \subseteq E \times Q$ *containing* (e_0, q_0) *and satisfying for each* $(e, q) \in \mathcal{R}$ *that whenever*

1. $q \xrightarrow{a} q'$ *then there is some* $e \overset{a}{\hookrightarrow} e'$ *and if, moreover,* $q \neq q'$ *then* $e' \mathcal{R} q'$*,*
2. $e \overset{a}{\hookrightarrow} e'$ *then there is (unique)* $q \overset{a}{\dashrightarrow} q'$ *and if, moreover,* $q \neq q'$ *then* $e' \mathcal{R} q'$*,*
3. $e \overset{\tau}{\hookrightarrow} e'$ *then* $e' \mathcal{R} q$*,*
4. $q \overset{ctc}{\rightsquigarrow} q'$ *then for every IMC* $\mathcal{C}$ *and every scheduler* $\pi \in \mathfrak{S}(\mathcal{C}|e)$*,*[(2)] *there is a random variable* $Stop : \mathbb{R}\mathrm{uns} \to \mathbb{R}_{>0}$ *on the probability space* $(\mathbb{R}\mathrm{uns}, \mathcal{F}, \mathcal{P}^\pi_{\mathcal{C}|e})$ *such that*
   - *if ctc is of the form* $\bowtie d$ *then the cumulative distribution function of Stop is point-wise* $\bowtie$ *cumulative distribution function of* d *(there are no constraints when* $ctc = \top$*), and*
   - *for every run* ρ *of* $\mathcal{C}|e$ *under* π*, either a transition corresponding to synchronization on action* a *with* $q \overset{a}{\dashrightarrow} q' \neq q$ *is taken before time* $Stop(\rho)$*, or*
     - *the state* (c, e') *visited at time* $Stop(\rho)$ *satisfies* $e' \mathcal{R} q'$*, and*
     - *for all states* $(\bar{c}, \bar{e})$ *visited prior to that, whenever*
       (a) $q \xrightarrow{a} q'$ *then there is* $e \overset{a}{\hookrightarrow} e'$*,*
       (b) $e \overset{a}{\hookrightarrow} e'$ *then there is* $q \overset{a}{\dashrightarrow} q'$*.*

The semantics of $\mathcal{S}$ *is the set* $[\![\mathcal{S}]\!] = \{\mathcal{E} \in \mathrm{IMC} \mid \mathcal{E} \models \mathcal{S}\}$ *of all conforming IMC.*

(1) This makes no difference for memoryless exponential distributions, but for all other distributions it does.

(2) Here e stands for the IMC $\mathcal{E}$ with the initial state e.



Example 2. We illustrate this definition. Consider the MCA on the right above specifying that a is ready and b will be ready either immediately after taking a or within the time distributed according to the Erlang distribution $\mathrm{Er}(3,1)$, which is a convolution of three $\mathrm{Exp}(1)$ distributions. The IMC below conforms to this specification (here, $\mathit{Stop} \sim \mathit{Er}(2,1)$ can be chosen). However, observe that it would not conform, if there was no transition under a from the middle to the right state. Satisfying the modalities throughout the waiting is namely required by the last bullet of the previous definition.

3.2 Assume-Guarantee Optimization

We can now formally state what guarantees on time-bounded reachability we can derive provided the unknown environment conforms to a specification $\mathcal{S}$. Given an *open* IMC $\mathcal{C}$, a set of goal states $G \subseteq C$ and a time bound $T \in \mathbb{R}_{\geq 0}$, the *value of compositional time-bounded reachability conditioned by an MCA* $\mathcal{S}$ is defined as

$$v_{\mathcal{S}}(\mathcal{C}) \quad := \quad \sup_{\sigma \in \mathfrak{S}(\mathcal{C})} \inf_{\substack{\mathcal{E} \in \mathrm{ENV}: \mathcal{E} \models \mathcal{S} \\ \pi \in \mathfrak{S}_\sigma(\mathcal{C}|\mathcal{E})}} \mathcal{P}^{\pi}_{\mathcal{C}|\mathcal{E}}\left[\Diamond^{\leq T} G\right]$$

In this paper, we pose a technical assumption on the set of schedulers of $\mathcal{C}$. For some clock resolution $\delta > 0$, we consider only such schedulers σ that take the same decision for any pair of paths $c_0 t_1 \dots t_n c_n$ and $c_0 t'_1 \dots t'_n c_n$ with t_i and t'_i equal when rounded down to a multiple of δ for all $1 \leq i \leq n$. This is no practical restriction as it is not possible to achieve arbitrary resolution of clocks when implementing the scheduler. Observe this is a safe assumption as it is *not* imposed on the unknown environment.

We consider specifications $\mathcal{S}$ where distributions have differentiable density functions. In the rest of the paper we show how to approximate $v_{\mathcal{S}}(\mathcal{C})$ for such $\mathcal{S}$. Firstly, we make a product of the given IMC and MCA. Secondly, we transform the product to a game. This game is further discretized into a partially observable stochastic game played on a dag where the quantitative reachability is solved. For full proofs, see [HKK13].

4 Product of IMC and Specification

In this section, we first translate MCA $\mathcal{S}$ into a sequence of IMC $(\mathcal{S}_i)_{i \in \mathbb{N}}$. Second, we combine the given IMC $\mathcal{C}$ with the sequence $(\mathcal{S}_i)_{i \in \mathbb{N}}$ into a sequence of product IMC $(\mathcal{C} \times \mathcal{S}_i)_{i \in \mathbb{N}}$ that will be further analysed. The goal is to reduce the case where the unknown environment is bound by the specification to a setting where we solve the problem for the product IMC while quantifying over all possible environments (satisfying only a simple technical assumption discussed at the end of the section), denoted ENV'. The reason why we need a sequence of products

instead of one product is that we need to approximate arbitrary distributions with more and more precise and detailed hyper-Erlang distributions expressible in IMC. Formally, we want to define the sequence of the products $\mathcal{C} \times \mathcal{S}_i$ so that

$$v_{product}(\mathcal{C} \times \mathcal{S}_i) \quad := \quad \sup_{\sigma \in \mathfrak{S}(\mathcal{C})} \inf_{\substack{\mathcal{E} \in \mathrm{ENV}' \\ \pi \in \mathfrak{S}_\sigma((\mathcal{C} \times \mathcal{S}_i)|\mathcal{E})}} \mathcal{P}^\pi_{(\mathcal{C} \times \mathcal{S}_i)|\mathcal{E}} \left[\Diamond^{\leq T} G \right]$$

approximates the compositional value:

Theorem 1. *For every IMC $\mathcal{C}$ and MCA $\mathcal{S}$, $v_{\mathcal{S}}(\mathcal{C}) = \lim_{i \to \infty} v_{product}(\mathcal{C} \times \mathcal{S}_i)$.*

Note that in $v_{product}$, σ is a scheduler over $\mathcal{C}$, not the whole product $\mathcal{C} \times \mathcal{S}_i$.[(3)] Constructing a product with the specification intuitively corresponds to adding a known, but uncontrollable and unobservable part of the environment to $\mathcal{C}$. We proceed as follows: We translate the MCA $\mathcal{S}$ into a sequence of IMC $\mathcal{S}_i$ and then the product will be defined as basically a parallel composition of $\mathcal{C}$ and $\mathcal{S}_i$.

There are two steps in the translation of $\mathcal{S}$ to $\mathcal{S}_i$. Firstly, we deal with the modal transitions. A *may* transition under a is translated to a standard external transition under a that has to synchronize with a in both $\mathcal{C}$ and $\mathcal{E}$ simultaneously, so that the environment may or may not let the synchronization occur. Further, each *must* transition under a is replaced by an external transition, that synchronizes with a in $\mathcal{C}$, but is hidden before making product with the environment. This way, we guarantee that $\mathcal{C}$ can take a and make progress no matter if the general environment $\mathcal{E}$ would like to synchronize on a or not.

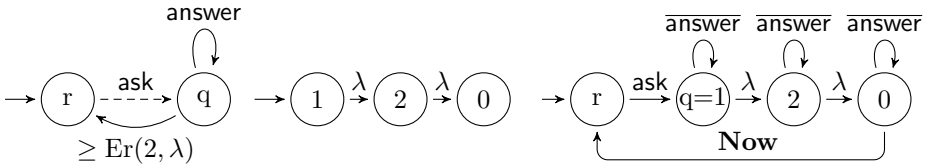
Formally, the must transitions are transformed into special "barred" transitions that will be immediately hidden in the product $\mathcal{C} \times \mathcal{S}_i$ as opposed to transitions arising from may transitions. Let $\overline{\mathbb{A}\mathrm{ct}} = \{\bar{a} \mid a \in \mathbb{A}\mathrm{ct}\}$ denote a fresh copy of the original alphabet. We replace all modal transitions as follows

- whenever $q \overset{a}{\dashrightarrow} r$ set $q \overset{a}{\hookrightarrow} r$,
- whenever $q \overset{a}{\longrightarrow} r$ set $q \overset{\bar{a}}{\hookrightarrow} r$.

The second step is to deal with the *timed* transitions, especially with the constraints of the form $\bowtie d$. Such a transition is, roughly speaking, replaced by a phase-type approximation of d. This is a continuous-time Markov chain (an IMC with only timed transitions) with a sink state such that the time to reach the sink state is distributed with d'. For any continuous distribution d, we can find such d' arbitrarily close to d.

Example 3. Consider the following MCA on the left. It specifies that whenever ask is taken, it cannot be taken again for at least the time distributed by $\mathrm{Er}(2, \lambda)$ and during all that time, it is ready to synchronize on answer. This specifies systems that are allowed to ask, but not too often, and whenever they ask, they must be ready to receive (possibly more) answers for at least the specified time.

(3) Here we overload the notation $\mathfrak{S}_\sigma((\mathcal{C} \times \mathcal{S}_i)|\mathcal{E})$ introduced for pairs in a straightforward way to triples, where σ ignores both the second and the third components.



After performing the first step of replacing the modal transitions as described above, we proceed with the second step as follows. We replace the timed transition with a phase-type, e.g. the one represented by the IMC in the middle. Observe that while the Markovian transitions are taken, answer must still be available. Hence, we duplicate the corresponding self-loops on all the new states. Further, since the time constraint is of the form $\geq$, getting to the state $(q, 0)$ does not guarantee that we already get to the state r. It can possibly take longer. To this end, we connect the states $(q, 0)$ and r by a special external action **Now**. Since this action is synchronized with $\mathcal{E} \in \mathrm{ENV}'$, the environment can block the progress for arbitrarily long time. Altogether, we obtain the IMC on the right.

In the case of "$\leq$" condition, we would instead add the **Now** transition from each auxiliary state to the sink, which could instead shorten the waiting time.

When constructing $\mathcal{S}_i$, we replace each distribution d with its hyper-Erlang phase-type approximation d_i with i branches of lengths 1 to i and rates $\sqrt{i}$ in each branch. For formal description, see [HKK13]. Formally, let $\mathbf{Now} \notin \mathbb{A}\mathrm{ct} \cup \overline{\mathbb{A}\mathrm{ct}}$ be a fresh action. We replace all timed transitions as follows:

- whenever $q \overset{\top}{\rightsquigarrow} r$ such that $q \neq r$ set $q \overset{\mathbf{Now}}{\hookrightarrow} r$,
- whenever $q \overset{\bowtie d}{\rightsquigarrow} r$ where the phase-type d_i corresponds to a continuous-time Markov chain (IMC with only timed transitions) with the set of states D, the initial state *1* and the sink state *0*, then
  1. identify the states q and *1*,
  2. for every $u \in D$ and $q \overset{\alpha}{\hookrightarrow} q$, set $u \overset{\alpha}{\hookrightarrow} u$,
  3. for every $u \in D$ and $q \overset{\alpha}{\hookrightarrow} p$ with $p \neq q$, set $u \overset{\alpha}{\hookrightarrow} p$,
  4. if $\bowtie = \leq$, then identify r and *0*, and set $u \overset{\mathbf{Now}}{\hookrightarrow} r$ for each $u \in D$,
  5. if $\bowtie = \geq$, then set $0 \overset{\mathbf{Now}}{\hookrightarrow} r$.

Intuitively, the new timed transitions model the delays, while in the "$\leq$" case, the action **Now** can be taken to speed up the process of waiting, and in the "$\geq$" case, **Now** can be used to block further progress even after the delay has elapsed.

The product is now the parallel composition of $\mathcal{C}$ and $\mathcal{S}_i$, where each action $\bar{a}$ synchronizes with a and the result is immediately hidden. Formally, the product $\mathcal{C} \times \mathcal{S}$ is defined as $\mathcal{C} \parallel^{\mathbf{PC6}}_{\mathbb{A}\mathrm{ct} \cup \overline{\mathbb{A}\mathrm{ct}}} \mathcal{S}_i$, where $\parallel^{\mathbf{PC6}}_{\mathbb{A}\mathrm{ct} \cup \overline{\mathbb{A}\mathrm{ct}}}$ is the parallel composition with one additional axiom:

(PC6) $s_1 \overset{a}{\hookrightarrow} s_1'$ and $s_2 \overset{\bar{a}}{\hookrightarrow} s_2'$ implies $(s_1, s_2) \overset{\tau}{\hookrightarrow} (s_1', s_2')$,

saying that a synchronizes also with $\bar{a}$ and, in that case, is immediately hidden (and any unused $\bar{a}$ transitions are thrown away).

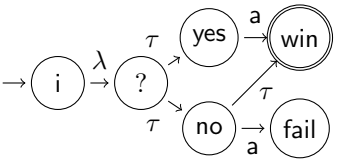
The idea of **Now** is that it can be taken in arbitrarily short, but non-zero time. To this end, we define ENV′ in the definition of $v_{product}(\mathcal{C} \times \mathcal{S}_i)$ to denote all environments where **Now** is only available in states that can be entered by only a Markovian transition. Due to this requirement, each **Now** can only be taken after waiting for some time.

5 Controller-Environment Games

So far, we have reduced our problem to computing $\lim_{i\to\infty} v_{product}(\mathcal{C} \times \mathcal{S}_i)$. Note that we are still quantifying over unknown environments. Further, the behaviour of each environment is limited by the uncontrollable *stochastic* flow of time caused by its Markovian transitions. This setting is still too difficult to be solved directly. Therefore, in this section, we reduce this setting to one, where the stochastic flow of time of the environment (limited in an unknown way) is replaced by a free *non-deterministic* choice of the *second player*.

We want to turn the product IMC $\mathcal{C} \times \mathcal{S}_i$ into a two-player *controller–environment game* (CE game) $\mathcal{G}_i$, where player **con** controls the decisions over internal transitions in $\mathcal{C}$; and player **env** simulates the environment including speeding-up/slowing-down $\mathcal{S}$ using **Now** transitions. In essence, **con** chooses in each state with internal transitions one of them, and **env** chooses in each state with external (and hence synchronizing) transitions either which of them should be taken, or a *delay* $d \in \mathbb{R}_{>0}$ during which no synchronization occurs. The internal and external transitions take zero time to be executed if chosen. Otherwise, the game waits until either the delay d elapses or a Markovian transition occurs.

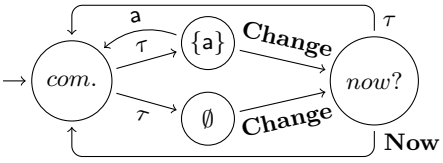
This is the approach taken in [BHK+12] where no specification is considered. However, there is a catch. This construction is only correct under the assumption of [BHK+12] that there are no states of $\mathcal{C}$ with both external and internal transitions available.



Example 4. Consider the IMC $\mathcal{C}$ on the right (for instance with a trivial specification not restricting the environment). Note that there are both internal and external actions available in no.

As τ transitions take zero time, the environment $\mathcal{E}$ must spend almost all the time in states without τ. Hence, when ? is entered, $\mathcal{E}$ is almost surely in such a state e. Now τ form ? is taken and $\mathcal{E}$ cannot move to another state when yes/no is entered. Since action a either *is* or *is not* available in e, the environment cannot choose to synchronize in no and not to synchronize in yes. As a result, the environment *"commits" in advance* to synchronize over a either in both yes and no or in none of them. Therefore, in the game we define, **env** cannot completely freely choose which external transition is/is not taken. Further, note that the scheduler of $\mathcal{C}$ cannot observe whether a is currently available in $\mathcal{E}$, which intrinsically induces imperfect information.

In order to transfer these "commitments" to the game, we again make use of the compositionality of IMC and put the product $\mathcal{C} \times \mathcal{S}_i$ in parallel with an IMC *Commit* and then define the game on the result.



The action alphabet of *Commit* is $\mathbb{A}\mathrm{ct} \cup \{\mathbf{Now}, \mathbf{Change}\}$ and the state space is $2^{\mathbb{A}\mathrm{ct}} \cup \{\mathit{commit}, \mathit{now}?\}$ (in the figure, $\mathbb{A}\mathrm{ct} = \{\mathsf{a}\}$; for formal description, see [HKK13]). State $A \subseteq \mathbb{A}\mathrm{ct}$ corresponds to $\mathcal{E}$ being committed to the set of currently available actions A. Thus $A \xrightarrow{a} \mathit{commit}$ for each $a \in A$. This commitment must be respected until the state of $\mathcal{E}$ is changed: either (1) by an external transition from the commitment set (which in *Commit* leads to the state *commit* where a new commitment is immediately chosen); or (2) by a **Change** transition (indicating the environment changed its state due to its Markovian transition).

The game $\mathcal{G}_i$ is played on the arena $\big(\mathcal{C}\times\mathcal{S}_i \parallel_{\mathbb{A}\mathrm{ct}\cup\{\mathbf{Now}\}} \mathit{Commit}\big) \setminus \big(\mathbb{A}\mathrm{ct} \cup \{\mathbf{Now}\}\big)$ with its set of states denoted by G_i. Observe that external actions have either been hidden (whenever they were available in the commitment), or discarded (whenever not present in the current commitment). The only external action that remains is **Change**. The game $\mathcal{G}_i$ is played as follows. There are two types of states: *immediate* states with some τ transitions available and *timed* states with no τ available. The game starts in $v_0 = (c_0, q_0, \mathit{commit})$.

- In an immediate state $v_n = (c, q, e)$, **con** chooses a probability distribution over transitions corresponding to the internal transitions in $\mathcal{C}$ (if there are any). Then, **env** either approves this choice (chooses ✓) and v_{n+1} is chosen randomly according to this distribution, or rejects this choice and chooses a τ transition to some v_{n+1} such that the transition does *not* correspond to any internal transitions of $\mathcal{C}$. Then the game moves at time $t_{n+1} = t_n$ to v_{n+1}.
- In a timed state $v_n = (c, q, e)$, **env** chooses a delay $d > 0$. Then Markovian transitions (if available) are resolved by randomly sampling a time t according to the exponential distribution with rate $\mathrm{R}(v_n)$ and randomly choosing a target state v_{n+1} where each $v_n \overset{\lambda}{\rightsquigarrow} v$ is chosen with probability $\lambda/\mathrm{R}(v_n)$.
  - If $t < d$, $\mathcal{G}_i$ moves at time $t_{n+1} = t_n + t$ to v_{n+1}, (Markovian transition wins)
  - else $\mathcal{G}_i$ moves at time $t_{n+1} = t_n + d$ to $(c, q, \mathit{now}?)$. ($\mathcal{E}$ takes **Change**)

This generates a run $v_0 t_1 v_1 t_1 \cdots$. The set $(G_i \times \mathbb{R}_{\geq 0})^* \times G_i$ of prefixes of runs is denoted $\mathbb{H}\mathrm{istories}(\mathcal{G})$. We formalize the choice of **con** as a *strategy* $\sigma : \mathbb{H}\mathrm{istories}(\mathcal{G}_i) \to \mathcal{D}(G_i)$. We further allow the **env** to randomize and thus his *strategy* is $\pi : \mathbb{H}\mathrm{istories}(\mathcal{G}_i) \to \mathcal{D}(\{\checkmark\} \cup G_i) \cup \mathcal{D}(\mathbb{R}_{>0})$. We denote by Σ and Π the sets of all strategies of the players **con** and **env**, respectively.

Since **con** is not supposed to observe the state of the specification and the state of *Commit*, we consider in Σ only those strategies that satisfy $\sigma(p) = \sigma(p')$, whenever *observations* of p and p' are the same. Like before, the observation of $(c_0, q_0, e_0)t_1 \cdots t_n(c_n, q_n, e_n) \in \mathbb{H}\mathrm{istories}(\mathcal{G})$ is a sequence obtained from $c_0 t_1 \cdots t_n c_n$ by replacing each maximal consecutive sequence $t_i\, c_i \cdots t_j\, c_j$ with all c_k the same, by $t_i\, c_i$. This replacement takes place so that the player cannot observe transitions that do not affect $\mathcal{C}$. Notice that now $\mathfrak{S}(\mathcal{C})$ is in one-to-one correspondence with Σ. Further, in order to keep CE games out of Zeno behaviour, we consider in Π only those strategies for which the induced Zeno runs

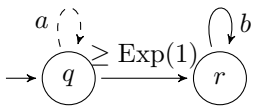
have zero measure, i.e. the sum of the chosen delays diverges almost surely no matter what **con** is doing. The *value of* $\mathcal{G}_i$ is now defined as

$$v_{\mathcal{G}_i} \quad := \quad \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \mathcal{P}_{\mathcal{G}_i}^{\sigma,\pi}\left[\Diamond^{\leq T} G\right]$$

where $\mathcal{P}_{\mathcal{G}_i}^{\sigma,\pi}\left[\Diamond^{\leq T} G\right]$ is the probability of all runs of $\mathcal{G}_i$ induced by σ and π and reaching a state with the first component in G before time T. We now show that it coincides with the value of the ith product:

Theorem 2. *For every IMC $\mathcal{C}$, MCA $\mathcal{S}$, $i \in \mathbb{N}$, we have $v_{\mathcal{G}_i} = v_{product}(\mathcal{C} \times \mathcal{S}_i)$.*

This result allows for approximating $v_{\mathcal{S}}(\mathcal{C})$ through computing $v_{\mathcal{G}_i}$'s. However, from the algorithmic point of view, we would prefer approximating $v_{\mathcal{S}}(\mathcal{C})$ by solving a single game $\mathcal{G}$ whose value $v_{\mathcal{G}}$ we could approximate directly. This is indeed possible. But first, we need to clarify, why the approximation sequence $\mathcal{S}_i$ was crucial even in the case where all distributions of $\mathcal{S}$ are already exponential.



Consider the MCA on the right and a conforming environment $\mathcal{E}$, in which a is available iff b becomes available within 0.3 time units. If Player **env** wants to simulate this behaviour, he needs to know how long the transition to r is going to take so that he can plan his behaviour freely, only sticking to satisfying the specification. If we translate Exp(1) directly to a single Markovian transition (with no error incurred), **env** knows nothing about this time as exponential distributions are memoryless. On the other hand, with finer hyper-Erlang, he knows how long the current branch of hyper-Erlang is roughly going to take. In the limit, he knows the precise waiting time right after coming to q.

To summarize, **env** is too weak in $\mathcal{G}_i$, because it lacks the information about the precise time progress of the specification. The environment needs to know how much time is left before changing the location of $\mathcal{S}$. Therefore, the game $\mathcal{G}$ is constructed from $\mathcal{G}_1$ by multiplying the state space with $\mathbb{R}_{\geq 0}$ where we store the exact time to be waited. After the product changes the state so that the specification component switches to a state with $\bowtie d$ constraint, this last component is overwritten with a number generated according to d. This way, the environment knows precisely how much time is left in the current specification location. This corresponds to the infinitely precise hyper-Erlang, where we at the beginning randomly enter a particular branch, which is left in time with Dirac distribution. For more details, see [HKK13].

Denoting the *value of* $\mathcal{G}$ by $v_{\mathcal{G}} := \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \mathcal{P}_{\mathcal{G}}^{\sigma,\pi}\left[\Diamond^{\leq T} G\right]$, we obtain:

Theorem 3. *For every IMC $\mathcal{C}$ and MCA $\mathcal{S}$, we have $v_{\mathcal{G}} = \lim_{i \to \infty} v_{\mathcal{G}_i}$.*

6 Approximation using discrete-time PO games

In this section, we briefly discuss the approximation of $v_{\mathcal{G}}$ by a discrete time turn-based partial-observation stochastic game Δ. The construction is rather standard; hence, we do not treat the technical difficulties in great detail (see [HKK13]).

We divide the time bound T into N intervals of length $\kappa = T/N$ such that the clock resolution δ (see Section 3.2) satisfies $\delta = n\kappa$ for some $n \in \mathbb{N}$.

1. We enhance the state space with a counter $i \in \{0, \ldots, N\}$ that tracks that $i \cdot \kappa$ time has already elapsed. Similarly, the $\mathbb{R}_{\geq 0}$-component of the state space is discretized to κ-multiples. In timed states, time is assumed to pass exactly by κ. In immediate states, actions are assumed to take zero time.
2. We let at most one Markovian transition occur in one step in a timed state.
3. We unfold the game into a tree until on each branch a timed state with $i = N$ is reached. Thereafter, Δ stops. We obtain a graph of size bounded by $b^{\leq N \cdot |G|}$ where b is the maximal branching and G is the state space of $\mathcal{G}$.

Let Σ_Δ and Π_Δ denote the set of randomized history-dependent strategies of **con** and **env**, respectively, where player **con** observes in the history only the first components of the states, i.e. the states of $\mathcal{C}$, and the elapsed time $\lfloor i/n \rfloor$ up to the precision δ. Then $v_\Delta := \sup_{\sigma \in \Sigma_\Delta} \inf_{\pi \in \Pi_\Delta} \mathcal{P}_\Delta^{\sigma,\pi}(\Diamond G)$ denotes the value of the game Δ where $\mathcal{P}_\Delta^{\sigma,\pi}(\Diamond G)$ is the probability of the runs of Δ induced by σ and π and reaching a state with first component in G. Let b be a constant bounding (a) the sum of outgoing rates for any state of $\mathcal{C}$, and (b) densities and their first derivative for any distribution in $\mathcal{S}$.

Theorem 4. *For every IMC $\mathcal{C}$ and MCA $\mathcal{S}$, $v_\mathcal{G}$ is approximated by v_Δ:*

$$|v_\mathcal{G} - v_\Delta| \quad \leq \quad 10\kappa(bT)^2 \ln \tfrac{1}{\kappa}.$$

A strategy σ^* optimal in Δ defines a strategy $(10\kappa(bT)^2 \ln \frac{1}{\kappa})$-optimal in $\mathcal{G}$. Further, v_Δ and σ^* can be computed in time polynomial in $|\Delta|$, hence in time $2^{\mathcal{O}(|\mathcal{G}|)}$.

The proof of the error bound extends the technique of the previous bounds of [ZN10] and [BHK+12]. Its technical difficulty stems from partial observation and from semi-Markov behaviour caused by the arbitrary distributions in the specification. The game is unfolded into a tree in order to use the result of [KMvS94]. Without the unfolding, the best known (naive) solution would be a reduction to the theory of reals, yielding an EXPSPACE algorithm.

7 Summary

We have introduced an assume-guarantee framework for IMC. We have considered the problem to approximate the guarantee on time-bounded reachability properties in an unknown environment $\mathcal{E}$ that satisfies a given assumption. The assumptions are expressed in a new formalism, which introduces continuous time constraints. The algorithmic solution results from Theorems 1 to 4:

Corollary 1. *For every IMC $\mathcal{C}$ and MCA $\mathcal{S}$ and $\varepsilon > 0$, a value v and a scheduler σ can be computed in exponential time such that $|v_\mathcal{S}(\mathcal{C}) - v| \leq \varepsilon$ and σ is ε-optimal in $v_\mathcal{S}(\mathcal{C})$.*

In future work, we want to focus on identifying structural subclasses of IMC allowing for polynomial analysis.

Acknowledgement We thank Tomáš Brázdil and Vojtěch Řehák for fruitful discussions and for their feedback.

References

[AH96] R. Alur and T.A. Henzinger. Reactive modules. In *LICS*, pages 207–218, 1996.

[BF09] P. Bouyer and V. Forejt. Reachability in stochastic timed games. In *Proc. of ICALP*, volume 5556 of *LNCS*, pages 103–114. Springer, 2009.

[BHK+12] T. Brázdil, H. Hermanns, J. Krčál, J. Křetínský, and V. Řehák. Verification of open interactive markov chains. In *FSTTCS*, pages 474–485, 2012.

[BHKH05] C. Baier, H. Hermanns, J.-P. Katoen, and B.R. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *Theor. Comp. Sci.*, 345(1):2–26, 2005.

[BS11] P. Buchholz and I. Schulz. Numerical Analysis of Continuous Time Markov Decision processes over Finite Horizons. *Computers and Operations Research*, 38:651–659, 2011.

[CD10] K. Chatterjee and L. Doyen. The complexity of partial-observation parity games. In *LPAR (Yogyakarta)*, pages 1–14, 2010.

[DLL+12] A. David, K.G. Larsen, A. Legay, M.H. Møller, U. Nyman, A.P. Ravn, A. Skou, and A. Wasowski. Compositional verification of real-time systems using ECDAR. *STTT*, 14(6):703–720, 2012.

[EKN+12] M.-A. Esteve, J.-P. Katoen, V.Y. Nguyen, B. Postma, and Y. Yushtein. Formal correctness, safety, dependability and performance analysis of a satellite. In *Proc. of ICSE*. ACM and IEEE press, 2012.

[GHKN12] D. Guck, T. Han, J.-P. Katoen, and M.R. Neuhäußer. Quantitative timed analysis of interactive Markov chains. In *NFM*, volume 7226 of *LNCS*, pages 8–23. Springer, 2012.

[HH13] H. Hatefi and H. Hermanns. Improving time bounded computations in interactive Markov chain. In *FSEN*, 2013. to appear.

[HK09] H. Hermanns and J.-P. Katoen. The how and why of interactive Markov chains. In *FMCO*, volume 6286 of *LNCS*, pages 311–337. Springer, 2009.

[HKK13] H. Hermanns, J. Krčál, and J. Křetínský. Compositional verification and optimization of interactive markov chains. *CoRR*, abs/1305.7332, 2013.

[HKR+10] B.R. Haverkort, M. Kuntz, A. Remke, S. Roolvink, and M.I.A. Stoelinga. Evaluating repair strategies for a water-treatment facility using Arcade. In *Proc. of DSN*, pages 419–424, 2010.

[HMP01] T.A. Henzinger, M. Minea, and V.S. Prabhu. Assume-guarantee reasoning for hierarchical hybrid systems. In *HSCC*, pages 275–290, 2001.

[HNP+11] E.M. Hahn, G. Norman, D. Parker, B. Wachter, and L. Zhang. Game-based abstraction and controller synthesis for probabilistic hybrid systems. In *QEST*, pages 69–78, 2011.

[KKLW07] J.-P. Katoen, D. Klink, M. Leucker, and V. Wolf. Three-valued abstraction for continuous-time Markov chains. In *CAV*, pages 311–324, 2007.

[KKN09] J.-P. Katoen, D. Klink, and M.R. Neuhäußer. Compositional abstraction for stochastic systems. In *FORMATS*, pages 195–211, 2009.

[KMvS94] D. Koller, N. Megiddo, and B. von Stengel. Fast algorithms for finding randomized strategies in game trees. In *STOC*, pages 750–759, 1994.

[KV96] O. Kupferman and M. Vardi. Module checking. In *CAV*, volume 1102 of *LNCS*, pages 75–86. Springer, 1996.

[KZH+11] J.-P. Katoen, I.S. Zapreev, E.M. Hahn, H. Hermanns, and D.N. Jansen. The ins and outs of the probabilistic model checker MRMC. *Performance Evaluation*, 68(2):90–104, 2011.

[LT88] K. G. Larsen and B. Thomsen. A modal process logic. In *LICS*, pages 203–210, 1988.

[Mar11] J. Markovski. Towards supervisory control of interactive Markov chains: Controllability. In *ACSD*, pages 108–117, 2011.

[MC81] J. Misra and K. Mani Chandy. Proofs of networks of processes. *IEEE Trans. Software Eng.*, 7(4):417–426, 1981.

[RW89] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1), 1989.

[Spr11] J. Sproston. Discrete-time verification and control for probabilistic rectangular hybrid automata. In *QEST*, pages 79–88, 2011.

[TAKB96] S. Tasiran, R. Alur, R.P. Kurshan, and R.K. Brayton. Verifying abstractions of timed systems. In *CONCUR*, pages 546–562, 1996.

[ZN10] L. Zhang and M.R. Neuhäußer. Model checking interactive Markov chains. In *Proc. of TACAS*, volume 6015 of *LNCS*, pages 53–68. Springer, 2010.

Part III

Auxiliary materials

Appendix H

Beyond Markov chains

The requirements (F), (M) and (H) of Section 2.1 are very realistic since the systems we have discussed are (1) typically controlled by finite programs with finite memory, and (2) often operating in environments where the random behaviour depends only on the current state of the system (or equivalently, finitely many recent states). Nonetheless, the requirements do restrict possible applications. For instance, consider a thermo-regulator in a room. If we want to check properties concerning the real-valued temperature, it needs to be captured in the states. And even if we satisfy (F) by abstracting the temperature values into e.g. intervals, (M) and (H) do not hold. Indeed, the temperature in the next moment depends also on the current derivative of the temperature, not only on its current value. Since we cannot store the real-valued derivation in the state, all requirements would only hold if we stored a precise finite abstraction of the derivation in the state. For non-linear behaviour, every abstraction yielding a Markov chain only approximates the original system. Note that using a more general modelling framework such as (non-linear) hybrid systems often leads to undecidability [ACHH92] of the problems we are interested in.

Further features that Markov chains fail to capture are general distributions on waiting and awaiting more events in parallel. Consider cooking meat and potatoes both happening in 15 to 18 minutes from the start. If the meat is done, the state is changed as only potatoes are being cooked. However, now the waiting time for potatoes is not between 15 and 18 minutes, but definitely less than 3. This behaviour is inherently non-Markovian, but it can again be approximated by a Markov chain using e.g. the phase type methods [Kom12]. More general framework of generalised semi-Markov processes can be employed, but again the analysis becomes significantly harder.

Appendix I

Quantitative analysis overview

We now give a concise overview of the model checking results for systems of Figure 2.2, i.e. for non-stochastic, discrete-time stochastic and continuous-time stochastic systems each of which is considered in its deterministic, non-deterministic and game form, denoted 0, 1, and 2, respectively, according to the number of players. We consider model checking non-stochastic systems with respect to reachability, LTL and CTL, stochastic systems with respect to their probabilistic (both qualitative and quantitative) extensions, and finally we consider quantitative continuous-time specifications. These are time-bounded reachability (TBR), deterministic timed automata (DTA), and continuous stochastic logic (CSL). For better readability, we display the complexity of the respective algorithms in the tables below.

Table I.1: Non-stochastic analysis

	0	1	2
reachability	NL	NL	P
LTL	P	PSPACE	2-EXP
CTL	P	P	P

Non-stochastic analysis

While reachability for LTS can be solved by graph reachability, for games the attractor construction is used. Checking CTL can be done inductively, hence in linear time w.r.t. the length of the formula [CES83]. Further, CTL and LTL

collapse when interpreted over a single run. However, checking LTL becomes more expensive for LTS and games. For LTS, a Büchi automaton for a formula ϕ exponential in $|\phi|$ is constructed, but only on the fly yielding a PSPACE algorithm [VW86]. However, the search through the state space of the product of the LTS and the automaton is only linear in the size of the LTS. In contrast, solving the LTL game [PR89] requires to construct a deterministic ω-automaton for ϕ, e.g. a deterministic Rabin automaton (the reason for this is explained in Chapter 3). This automaton requires up to doubly exponential space (and has exponentially many pairs) and then the respective game must be solved on the product. This solution can be computed in time polynomial in the product and exponential in the number of Rabin pairs, hence altogether doubly exponential in $|\phi|$.

Table I.2: Qualitative stochastic analysis

	0	1	2
reachability	P	P	P
PLTL	PSPACE	2-EXP	3-NEXP∩co-3-NEXP
PCTL	P	EXP	?

Table I.3: Quantitative stochastic analysis

	0	1	2
reachability	P	P	NP∩co-NP
PLTL	2-EXP	2-EXP	3-NEXP∩co-3-NEXP
PCTL	P	undecidable	undecidable

Stochastic analysis

Qualitative reachability in the > 0 case is almost the same as the non-stochastic reachability. However, in the $= 1$ case it requires a decomposition to strongly connected components for MC and maximum end-component (MEC) decomposition for MDP and followed by > 0 reachability analysis of the resulting graph. In the quantitative case, linear equalities must be solved for MC, linear programming for MDP and strategy iteration is used in games to obtain (deterministic and positional) strategies.

For probabilistic (qualitative) interpretation of LTL, [CY88] gives (1) an algorithm for MC, which modifies the chain step by step and works in time exponential in $|\phi|$, and (2) an algorithm for MDP, where a deterministic (or "almost-deterministic") ω-automaton is constructed (doubly exponential in $|\phi|$) and the product is examined by MEC decomposition, for details see Chapter 3 and Paper D. In both cases, the complexity w.r.t. the system size is only polynomial. For the quantitative LTL on MDP, we only need to solve quantitative

reachability on the MEC decomposition. In order to analyse stochastic games, one can transform the product with the Rabin automaton into a stochastic parity game by the exponential last appearance record construction, and then solve the latter in NP∩co-NP [CJH04]. For deterministic strategies (choosing only Dirac distribution) the problem is shown to be in 3-EXP [BGL+04].

For PCTL over MC, only qualitative/quantitative reachability computations are iterated for all subformulae. For qualitative PCTL, synthesis for MDP is shown to be in EXP in [BFK08] even though the strategies may require infinite memory. On the other hand, for quantitative PCTL the problem is undecidable [BBFK06] even for fragments not requiring infinite memory. However, the decidability of the qualitative PCTL over games remains open. It has been proven (ibid.) that infinite memory is needed. The largest qualitative fragment know to be decidable contains only $\mathbf{F}^{=1}$, $\mathbf{F}^{>0}$, and $\mathbf{G}^{=1}$ operators (ibid.).

Finally, note that this analysis also applies to continuous-time systems by simply ignoring the timing. This is done by ignoring the rates R.

Table I.4: Continuous-time analysis

	0	1	2
TBR	decidable	approximable	approximable
DTA	approximable	?	?
CSL	decidable	?	?

Continuous-time analysis

The qualitative analysis of continuous-time systems can safely ignore the continuous timing aspect as the exponential distribution is positively supported on $[0, \infty)$. However, the results in quantitative analysis are much less encouraging. The main difficulty with computing exact probabilities is caused by the irrationality and transcendence of e, which is omnipresent due to the exponential distribution on the waiting times. Time-bounded reachability can be expressed and computed as an expression of the form $\sum q_i \cdot e^{r_i}$ where the sum is finite and q_i, r_i are rational [ASSB96]. The comparison of such a sum to a rational number is decidable using the Lindemann-Weierstrass theorem, hence quantitative reachability and CSL model checking is decidable (ibid.). However, apart from that not much is known to be decidable. In [BHK+12] (Paper E) we show that time bounded reachability is decidable for games, but only when schedulers ignoring the times in the histories are taken into account—so called time-abstract strategies.

As a result, approximations are usually considered instead. Algorithms for the time-bounded reachability problem have first been given in the time-abstract setting where strategies only take into account the sequence of the states visited, but not the times. The first one is for CTMDP [BHHK04] further extended to

games [BHK$^+$12, RS10]. Later, strategies taking time into account have been considered and polynomial algorithms for CTMDP and IMC (easily extensible to games) have been given in [ZN10, NZ10] and improved in [BS11, HH13]. For more details, see Chapter 4. This yields algorithms for model checking CSL of IMC/CTMDP [ZN10] where the formulae are not nested. If nesting is allowed then approximability is not known and depends on whether TBR is decidable. Finally, objectives given by DTA and their approximability have been considered in [CHKM09].

Satisfiability Let us further mention another problem for logics, namely the *satisfiability* problem, i.e. the problem whether there exists a model satisfying a given formula. It is important especially for logics capable of expressing game arenas of non-deterministic programs as strategy synthesis for these programs can then be encoded into satisfiability. The satisfiability problem for LTL and its variants is PSPACE-complete [SC85], for CTL [EH82] and qualitative PCTL [HS84, BFKK08] it is EXP-complete, and for quantitative PCTL decidability of this problem remains open [BFKK08, HPW10].

Appendix J

Note on copyrights

According to the Consent to Publish in Lecture Notes in Computer Science with Springer-Verlag GmbH, the author of the thesis is allowed to include Papers A-D and G in the thesis:

> Author retains the right to use his/her Contribution for his/her further scientific career by including the final published paper in his/her dissertation or doctoral thesis provided acknowledgement is given to the original source of publication.

For more information, please see the copyright form electronically accessible at ftp.springer.de/pub/tex/latex/llncs/LNCS-Springer_Copyright_Form.pdf

According to the Journal Publishing Agreement with Elsevier B.V., the author of the thesis is allowed to include Paper E in the thesis:

> The Retained Rights include the right to use the Preprint or Accepted Author Manuscript for Personal Use, Internal Institutional Use and for Permitted Scholarly Posting, the right to use the Published Journal Article for Personal Use and Internal Institutional Use.

where "Personal Use" is defined as follows:

> Use by an author in the author's classroom teaching (including distribution of copies, paper or electronic), distribution of copies to research colleagues for their personal use, use in a subsequent compilation of the author's works, inclusion in a thesis or dissertation, preparation of other derivative works such as extending the article to book-length form, or otherwise using or re-using portions or excerpts in other works (with full acknowledgement of the original publication of the article).

For more information on this and on the copyright, please visit the webpage http://www.elsevier.com/copyright

According to the rules for publishing in LIPIcs (Leibniz International Proceedings in Informatics) with Schloss Dagstuhl Leibniz-Zentrum für Informatik GmbH, the author of the thesis is allowed to include Paper F in the thesis:

> All publication series follow the concept of OpenAccess, i.e., the articles are freely available online for the reader and the rights are retained by the author.

For more information, please see http://www.dagstuhl.de/publikationen/