

Imperial College London
Department of Computing

Finite-State Abstractions for Probabilistic Computation Tree Logic

Daniel Wagner

2011

Supervised by Dr. Michael Huth

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Imperial College London
and the Diploma of Imperial College London

Declaration

I herewith certify that all material in this dissertation which is not my own work has been duly acknowledged. Selected results from this dissertation have been disseminated in scientific publications as detailed in Chapter 1.10.

Daniel Wagner

Abstract

Probabilistic Computation Tree Logic (PCTL) is the established temporal logic for probabilistic verification of discrete-time Markov chains. Probabilistic model checking is a technique that verifies or refutes whether a property specified in this logic holds in a Markov chain. But Markov chains are often infinite or too large for this technique to apply. A standard solution to this problem is to convert the Markov chain to an abstract model and to model check that abstract model. The problem this thesis therefore studies is whether or when such *finite* abstractions of Markov chains for model checking PCTL exist.

This thesis makes the following contributions. We identify a sizeable fragment of PCTL for which 3-valued Markov chains can serve as finite abstractions; this fragment is maximal for those abstractions and subsumes many practically relevant specifications including, e.g., reachability. We also develop game-theoretic foundations for the semantics of PCTL over Markov chains by capturing the standard PCTL semantics via a two-player games. These games, finally, inspire a notion of p-automata, which accept entire Markov chains. We show that p-automata subsume PCTL and Markov chains; that their languages of Markov chains have pleasant closure properties; and that the complexity of deciding acceptance matches that of probabilistic model checking for p-automata representing PCTL formulae. In addition, we offer a simulation between p-automata that under-approximates language containment. These results then allow us to show that p-automata comprise a solution to the problem studied in this thesis.

Contents

1	Introduction	8
1.1	Model checking	10
1.2	Abstraction and refinement	12
1.3	(Finitary) completeness	14
1.4	Finite model property	16
1.5	Abstraction and undecidability	18
1.6	Witnesses for truth and falsity	19
1.7	Research agenda of this thesis	22
1.8	Achievements in this thesis	24
1.9	Outline of this thesis	25
1.10	Publications	27
1.11	Summary of chapter	28
2	Background on Models and Logics	29
2.1	Kripke structures	30
2.2	Linear-time and branching-time temporal logic	36
2.3	Simulation and bisimulation	51
2.4	Markov chains and Markov decision processes	60
2.5	Probabilistic branching-time logic: PCTL	65
2.6	Probabilistic simulation and bisimulation	76
2.7	Unfoldings	79
2.8	Summary of chapter	83
3	Background on Automata and Games	85
3.1	Word automata	86
3.2	Probabilistic automata	95
3.3	Alternating tree automata	99
3.4	Hintikka games	107
3.5	Stochastic parity games	109

3.6	Summary of chapter	112
4	Completeness for a Fragment of PCTL	114
4.1	Existing results	114
4.2	Our setting: Markov chains and PCTL	117
4.3	Completeness for a fragment of PCTL via 3-valued unfoldings	118
4.4	Summary of chapter	131
5	A Game Semantics for PCTL	132
5.1	Game semantics	132
5.2	Winning strategies	149
5.3	Summary of chapter	158
6	Completeness for full PCTL via p-Automata	159
6.1	p-Automata	160
6.2	Acceptance games	165
6.3	Expressiveness of p-automata	173
6.4	Simulation of p-automata	186
6.5	Wrong embedding of Markov chains	203
6.6	Summary of chapter	205
7	Evaluation	206
7.1	Related work on completeness of abstractions	206
7.2	Evaluation of achievements	209
7.3	Future work	210
	Bibliography	213

List of Figures

2.1	A Kripke structure K .	33
2.2	A labelled transition system (T, s_0) .	34
2.3	A partial Kripke structure (K, s_0) .	35
2.4	Hasse diagram of the information order on $\{\text{tt}, \text{?}, \text{ff}\}$.	36
2.5	A Kripke structure C .	53
2.6	A Kripke structure A .	53
2.7	A Kripke structure M .	54
2.8	A Kripke structure N .	55
2.9	A Kripke structure L .	56
2.10	A Kripke structure A .	58
2.11	A Kripke structure C .	58
2.12	A labelled discrete-time Markov chain M .	63
2.13	A 3-valued discrete-time Markov chain M .	64
2.14	A Markov decision process M .	65
2.15	A Markov chain M .	68
2.16	A Markov chain M .	69
2.17	A Markov chain M .	78
2.18	A Markov chain N .	78
2.19	A Markov chain B .	80
2.20	A Markov chain M .	82
2.21	The finite unfolding $M_{3,3}^{s_0}$.	82
3.1	A deterministic word automaton A .	88
3.2	A deterministic Büchi automaton A .	90
3.3	A deterministic Büchi automaton B .	91
3.4	A non-deterministic Büchi automaton C .	94
3.5	A non-deterministic Büchi automaton A .	95
3.6	A probabilistic automaton A .	97
3.7	A probabilistic automaton C .	98

3.8	A probabilistic automaton D .	98
3.9	A Kripke structure M .	104
3.10	An accepting run of B on M .	104
4.1	A labelled Markov chain M .	124
4.2	The finite unfolding $M_2^{s_0}$.	124
4.3	Maximal height of an unfolding.	127
4.4	A concrete Markov chain M .	129
6.1	Graph G_A of automaton A .	163
6.2	A Markov chain M .	164
6.3	Case 3 of acceptance game.	171
6.4	Case 1 of acceptance game.	172
6.5	A Markov chain M .	204

1 Introduction

Computing is becoming an ubiquitous part of our lives and an irreplaceable factor in our modern society. Our daily lives and our society as a whole are becoming more and more reliant and dependent on computer systems and their well-functioning. At the same time the various computer systems and especially interaction between them become more and more complex. The systems grow in size and in complexity as well as in sheer number and interconnectivity. Few systems are truly self-contained and many of them interact with each other and with changing and partially unknown environments. The need to understand such systems and their interaction, and the need to be sure about their functionality and their behaviour is becoming more and more pressing. The size and complexity of such systems makes manual inspections obviously infeasible. Furthermore one would like to be able to audit a system on more than one level between the high-level specification and the source code. Actually, one would like to audit systems at all levels between high-level specification and source code, or at least to understand the relationship between the properties of the system in its various descriptions. The result of such an audit should be a rigorous verification and should be fully integrated in the engineering process.

The importance of (automatic) formal verification of system behaviour has been recognised early on in the history of computing. According to Cliff Jones [107] already some of the work by Herman Goldstine and John von Neumann [83], by Alan Turing [171] and somehow even the “pre-electronic” work by Charles Babbage [15] was concerned with (the formal verification of) the correctness of programs. It was in the 1960s that the research in formal verification and program analysis truly took off the ground [95, 137, 119]. Since then – with the growing importance of computer systems for industry and society – the need for efficient automatic verification techniques is becoming increasingly important, both economically and socially.

Although posed early on, the problem is far from solved. Essentially,

the underlying, fundamental problems are undecidable and so incomplete methods are required. Many verification techniques have been suggested, analysed and deployed in applications, but all of them – even for decidable settings – struggle to keep up with the growth in size and complexity of current systems. Formal verification of system behaviour therefore remains an important and active research area.

The importance of formal verification was recently acknowledged by Tony Hoare’s declaration of *The verifying compiler* as one of the Grand Challenges in Computing [96], and by the UK Computing Research Committee (UKCRC) whose Grand Challenge 6 is *Dependable Systems Evolution* [183, 182]. These Verification Grand Challenges found strong support in the scientific community. Motivating examples, cases of support, different approaches and perspectives were presented, for example, in the position papers presented at the conference “Verified Software: Theories, Tools, Experiments”, which was held at the ETH Zürich in 2005 [2]. As the motivation for their Grand Challenge 6 the UKCRC formulates the general need for a notion of justification and dependability:

The vision of GC6 is of a future in which all computer systems justify the trust that society increasingly places in them. Dependability is a multi-faceted notion which includes fault tolerance, requirements engineering and verification amongst other topics.

This broad definition is not restricted to formal verification but also includes other evaluations of evolving and interacting systems. The available means for evaluating systems are broad, ranging from testing and numerical simulation to the formally rigorous approaches such as theorem proving and model checking. In model checking [69, 42, 155, 154], which this thesis is concerned with, one tries to formally answer a query about a system, such as “After each computation, will the program go back to its idle mode?” In model checking the returned answer is usually rigorously proved as opposed to the approximative nature of, say, testing and Monte Carlo simulation.

1.1 Model checking

Formally speaking, model checking means to compute (effectively and possibly efficiently) the answer to a query $M \models \phi$, where M is a *model* which formalises the system of interest, ϕ is a *query* specified in a formal language, and \models is the *satisfaction relation* which relates such models with the answers of the queries from this formal language. Of course one is interested in choosing models and formal languages which are expressive and intuitive enough to be used for real applications, while making the model-checking problem algorithmically decidable, and efficiently so. There are situations in which one chooses to sacrifice decidability for the sake of usability. Even for decidable instances the model-checking problem can – and in general will – be very complex, i.e. computationally expensive, to solve. One of the main problems for efficient model checking is not so much the complexity of the algorithms but the size of the models. In general models of realistic systems tend to be large; i.e. their size could be infinite and even when finite it usually grows exponentially in the number of interesting components, such as variables or concurrent processes. Thus even efficient algorithms, e.g. even with linear runtime in the size of the model, suffer from the so called *state explosion problem*.

Research in model checking began in the 1980s when Edmund Clarke and Allen Emerson [42]¹, Jean-Pierre Queille and Joseph Sifakis [155], respectively Orna Lichtenstein and Amir Pnueli [134] published the first model-checking algorithms. In [42] Clarke and Emerson describe model checking for branching-time logics as well as the synthesis of programs from branching-time specifications via tableaux. Queille and Sifakis [155] demonstrate how to convert branching-time specifications into Petri nets, and give an algorithm based on a fixpoint iteration to verify branching-time properties on such models. Lichtenstein and Pnueli give a model-checking algorithm for linear-time temporal logic [134]. A lot of progress has since been made, and model checking is already successfully being applied in industry and academia. Data structures and algorithms for various models and logics have been implemented in numerous model-checking tool kits. But although many of the known algorithms for model checking are very efficient, all of them still suffer from the state explosion problem: The model descriptions

¹According to [68] the term *model checking* was “coined” by [42].

themselves tend to be large, typically exponential in the number of critical components. Therefore even efficient algorithms and efficient data structures or symbolic techniques [143] are not always sufficient to handle large systems.

One can distinguish model-checking algorithms by the way they handle the state space of a given model. One group of algorithms, we call them *explicit search*, work on the explicitly constructed state space. The other group of algorithms, we call them *symbolic search*, work on a symbolic representation of the state space [143, 27]. Such a symbolic representation, for example by Boolean formulae represented as BDDs [143], is potentially much smaller than the explicit representation. Therefore symbolic algorithms can handle larger models and avoid the state explosion problem to some extent.

The basic idea of most model-checking algorithms for explicit state representations – as opposed to purely symbolic techniques – is an (exhaustive) labelling of the complete state space of the model. Each state gets tagged with a (partial) answer to the model-checking query until the answer for the full model and query can be read off as sub-queries are being evaluated [105]. Unfortunately such an enumeration of the state space requires a small and in particular finite state space.

Complementary to symbolic model-checking techniques there are various approaches which address the problem of state explosion and try to reduce the size of the state space. Important examples are

- *compositional model checking* [10, 132, 86, 123], which tries to answer queries on the whole system by combining the answers for independent components;
- *symmetry reduction* [45, 106, 34], which tries to avoid duplication of work by pruning redundancy from the state space;
- *partial order reduction* [172], which tries to avoid unnecessary combinatorial complexity introduced by the permutation of executions in concurrent processes;
- *on-the-fly model checking* [51, 80], and *bounded model checking* [41], which try to perform model checking on large or even infinite-state systems by deferring the evaluation of all but a finite subset of states;

- *abstraction* [44], which tries to shrink the model in a way which maintains enough information so that a query computed on the abstraction implies a meaningful answer for the original model.

The main research question of this thesis is connected to this last technique, i.e. *abstraction*, as we will explain in Section 1.2 below.

Of course every approach to model checking depends heavily on the chosen models and the query language. Many such combinations have been proposed and investigated, and it is rather difficult to organise the field of model checking in a coherent and canonical form. One way to organise the field, which suits our work, is to distinguish between probabilistic [125] and non-probabilistic [47] model checking first. Within both fields one can further distinguish between qualitative [175] and quantitative [40, 35] model checking. Other organisations of the field of model checking are possible, and some model-checking scenarios might not fit very well into our simple classification. Nevertheless, the given classification is sufficient in the context of this thesis.

One major objective of this thesis is to find a solution to a problem – let’s call it the *completeness problem* for now – for a particular instance of probabilistic qualitative model checking, namely for Markov chains with the probabilistic branching-time logic PCTL [14, 88] as query language. Previous work on the completeness problem was focused on non-probabilistic settings [31, 117, 61, 58, 59, 73, 74], and did not address completeness of probabilistic logics over Markov chains.

For the rest of this chapter we assume the query $M \models \phi$ to yield a Boolean answer. This allows us to informally talk about some important concepts (e.g. “ $A \models \phi$ implies $M \models \phi$ ”) whose meanings are rather intuitive in a Boolean setting but less clear for other answer domains.

1.2 Abstraction and refinement

The two complementary notions of *abstraction* and *refinement* are central concepts in model checking. Abstraction is one possible means to counter the state explosion problem: Given a *sound* abstraction notion one can try to solve the model-checking problem on the potentially smaller abstract model and deduce the answer to the concrete problem from there. Thus, if ϕ is

a query, and A is an abstraction of M , then we want that $A \models \phi$ implies $M \models \phi$ for a class of queries of interest.

Abstraction is not necessarily just a relation between concrete and abstract models. In general, an abstraction of the concrete model-checking problem $M \models \phi$ is an abstract model-checking problem $M^\alpha \models^\alpha \phi^\alpha$, where M^α is an abstract model, ϕ^α an abstract query and \models^α an abstract satisfaction relation [99]. Indeed, there are examples of useful abstractions where the abstraction is the identity on the model, i.e. $M^\alpha = M$, and only the query or the satisfaction relation are abstracted. (For example: sample testing, numerical simulation and finite-horizon model checking can be seen as abstracted satisfaction relations \models^α .) Such a general notion of abstraction as triple $(M^\alpha, \models^\alpha, \phi^\alpha)$ is very powerful, and notions such as soundness, precision and completeness can be defined in terms of this general framework [99].² Nevertheless, in this thesis we focus on the abstraction of models, while leaving the query language as it is, and only adapting the satisfaction relation as far as necessary to match the abstract models. We thus omit a more detailed discussion of the general notion of abstraction as triple, and refer to [99] for further details.

Refinement is the inverse concept to abstraction. But furthermore refinement can be seen as formalising what it means to implement a specification. If a model M refines a more abstract model A , it *implements* the formal specification given by the abstract model A . Depending on the properties of the refinement notion one can then investigate which properties ϕ the more concrete implementation M inherits from the more abstract specification A . An outstanding example for this idea of preserving properties all the way from formal specifications to the actual source code is the so called B-method developed by Jean-Raymond Abrial [5, 4]. The B-method uses so called Abstract Machine Notation [3] for specifications, which are then refined, step by step, all the way to an actual implementation in a programming language. The Abstract Machines and their refinement techniques in the B-method enable proving that

1. the specifications are consistent; and

²An interesting undertaking beyond the scope of this thesis would be to look at abstraction between triples $(M^\alpha, \models^\alpha, \phi^\alpha)$ and (M, \models, ϕ) in a category theoretic framework [150], and to find out whether desirable properties of abstractions have a natural correspondence in terms of category theory.

2. each refinement is a correct implementation of the more abstract specifications.

In particular one can prove that the resulting source code is a correct implementation of the initial specification.

In this thesis we are concerned with abstraction rather than refinement. In particular, we are interested in the abstraction notion called *simulation* [145], which is introduced in Chapter 2.3. Essentially, a model N *simulates* another model M , if N can mimic the transition behaviour of M .

1.3 (Finitary) completeness

Informally, let an *abstraction framework* consist of a class of abstract models, their abstract semantics, and an abstraction notion defined via some kind of simulation relation. Anticipating a formal definition of completeness in Section 4, let's call such an abstraction framework *complete* for a formula ϕ iff for all concrete models M that satisfy ϕ there is a finite-state model A such that A abstracts M and A satisfies ϕ in the abstract semantics, denoted as $A \preceq M$ and $A \models \phi$ respectively. The abstraction framework is complete for a set of formulae Γ if it is complete for each $\phi \in \Gamma$.

This is motivated by the following practical question: Given a model-checking problem, say $M \models \phi$, is there a model A such that

1. A abstracts M , e.g. $A \preceq M$;
2. the answer to $A \models \phi$ is easier to compute than the answer to $M \models \phi$;
and
3. the answer to $A \models \phi$ implies that for $M \models \phi$.

The ability to infer an answer for the original problem from the solution of the abstract model-checking query is ensured by the *soundness* of the abstraction notion. Formally soundness of abstraction \preceq for formula ϕ means, whenever $A \preceq M$ then $A \models \phi$ implies $M \models \phi$. We say abstraction is sound for a set of formulae Γ if it is sound for every $\phi \in \Gamma$. This implication is necessary in our context because our motivation for abstraction is to perform the model check on A as replacement for the, potentially infeasible, model check on M . If we could not infer an answer for the query on M

the whole abstraction would become useless for our application context. We are therefore interested in abstractions which are sound for all queries of interest, e.g. for the full query language.

With regard to the existence of an “easier to solve” abstract model, we are in particular interested in models A which have a smaller state space than the concrete model M . For finite model M one would like to find an abstraction A with *fewer* states; for infinite model M one would like an abstract model A with *finitely* many states. This latter reduction, from infinite-state to finite-state models, is particularly important for model checking as explicit search model-checking algorithms require a finite state space as input. The abstraction of infinite-state models M by finite-state abstractions A is thus also the primary concern of this thesis. In this sense we are concerned with *finitary completeness* throughout this thesis: For every concrete (potentially infinite-state) model M and query ϕ with $M \models \phi$ does there exist a *finite-state* abstraction A such that $A \preceq M$ and $A \models \phi$?

The main research challenge in this thesis is the completeness of one particular abstraction framework, namely discrete-time Markov chains as concrete models, PCTL as query language, and a suitable class of abstract models with a matching variant of probabilistic simulation as abstraction.

A lot of work on completeness of abstraction has been done for non-probabilistic model checking. Yonit Kesten and Amir Pnueli showed how to achieve completeness for linear-time temporal logics, e.g. LTL, on labelled transition systems via over-approximations and an explicit encoding of fairness constraints in so called *progress monitors* [117]. There were several approaches aiming at completeness for branching-time temporal logic such as CTL and the modal μ -calculus. The main breakthrough in this context was the work by Dennis Dams and Kedar Namjoshi [58, 59]. In their first article [58] they prove that even 3-valued labelling and fairness on labelled transition systems is insufficient to achieve completeness for CTL. They then present *focused transition systems* as a complete abstraction framework. The second article [59] re-formulates and summarises these and other existing results in terms of automata. The abstractions presented in these two articles are complete for the full modal μ -calculus, and by the use of automata for encoding the models as well as the formulae the proofs in [59] become stunningly elegant. There were other abstractions suggested which achieve completeness for (fragments) of branching-time logics, e.g. disjunc-

tive modal transition systems [73] and generalised Kripke modal transition systems [61, 74]. These existing frameworks were summarised within the terminology of tree-automata in Dams and Namjoshi’s article *Automata as abstractions* [59] and will be re-visited in Chapter 7 of this thesis.

1.4 Finite model property

It is necessary and worthwhile to differentiate our research question, whether an abstraction framework is finitary complete, from the so called *finite model property*³. A logic \mathcal{L} has the *finite model property*, with regard to a class of models \mathcal{M} , if and only if whenever a formula $\phi \in \mathcal{L}$ is satisfiable then there also exists a finite-state model for that formula. Formally,

whenever there exists a model $M \in \mathcal{M}$ with $M \models \phi$
then there exists a finite model $M' \in \mathcal{M}$ with $M' \models \phi$.

At first finitary completeness of an abstraction frameworks for a logic \mathcal{L} on a class of models \mathcal{M} looks very similar to the finite model property for this combination of logic and models. Under certain circumstances both properties actually coincide, but in general they are not the same. In fact this thesis is concerned with such a case: PCTL does not have the finite model property with regard to Markov chains [89, 30] but our p-automata of Chapter 6 are nevertheless a complete abstraction framework for PCTL over Markov chains.

While a complete abstraction framework (under some reasonable assumptions about the abstract class of models) always has the finite model property⁴, the finite model property of a logic does not in general yield a complete abstraction framework. For each formula $\phi \in \mathcal{L}$ the finite model property guarantees the existence of a finite-state model M' with $M' \models \phi$ but this model does not need to be in any particular relationship with a given model M with $M \models \phi$. Thus, in general M' does not abstract the concrete model

³Sometimes also called *small model property*.

⁴In the settings of this thesis, this means that PCTL has the finite model property with regard to p-automata but not with regard to Markov chains. Although Markov chains can be expressed as p-automata, this is no contradiction but illustrates that p-automata are significantly more expressive than Markov chains, which they indeed need to be for achieving completeness.

M .

The situation looks different if every model M has a so called *characteristic formula* with the following properties. Let M be a model from a class of models \mathcal{M} . A formula $\Phi_M \in \mathcal{L}$ is a *characteristic formula* of M iff for all $M' \in \mathcal{M}$

$$M' \models \Phi_M \text{ iff } M \text{ refines } M', \text{ i.e. } M' \preceq M .$$

So $M' \models \Phi_M$ and $M' \preceq M$ are then equivalent.

With such characteristic formulae and a suitable logic \mathcal{L} which is closed under conjunction, the finite model property and completeness as considered in this thesis are equivalent. Let every model $M \in \mathcal{M}$ have a characteristic formula Φ_M in \mathcal{L} and let the satisfaction relation satisfy the axiom of conjunction elimination, then the following statements are equivalent:

1. \mathcal{L} has the finite model property with regards to \mathcal{M} .
2. Abstraction within \mathcal{M} is finitary complete for all formulae of \mathcal{L} .

This equivalence is easy to proof.⁵The key point in this argument is the expressiveness of the logic \mathcal{L} : The characteristic formulae need to be expressible within the query logic \mathcal{L} , and the logic must support conjunction. The latter constraint is hardly a restriction since most logics used in applications contain conjunction. The other condition is much more restrictive. In fact, for many classes of models it is not even clear if they have characteristic formulae or not, let alone in which logic these formulae could be expressed if they existed. But then again, proving that a logic \mathcal{L} has the finite model property is, in general, not trivial either. Indeed, the finite model property is strongly related to the decidability of a logic. If a logic \mathcal{L} is finitely axiomatisable and has the finite model property then it is decidable [28].

There are some probabilistic logics [65, 176] which have the finite model property, but in this thesis we focus on the (more expressive) probabilistic

⁵Proof: “1 implies 2:” Let $M \in \mathcal{M}$, $\phi \in \mathcal{L}$ and $M \models \phi$. By assumption M has a characteristic formula $\Phi_M \in \mathcal{L}$ and \mathcal{L} is closed under conjunction. Therefore $\Phi_M \wedge \phi$ exists and is in \mathcal{L} . Now, if \mathcal{L} has the finite model property, then there exists a finite-state model M' for $\Phi_M \wedge \phi$. By the definition of conjunction $M' \models \phi$; and by the definition of characteristic formulae $M' \preceq M$. Hence \mathcal{M} is a complete abstraction framework for \mathcal{L} . “2 implies 1:” Conversely, if \mathcal{M} is a finitary complete abstraction for \mathcal{L} , then for each satisfiable formula ϕ , i.e. for each formula ϕ which has a model $M \in \mathcal{M}$, there also exists a finite-state model M' which abstracts M and for which $M' \models \phi$. Therefore every satisfiable formula ϕ has a finite model in \mathcal{M} , and \mathcal{L} has the finite model property.

branching-time logic PCTL over Markov chains as models. PCTL does not have the finite model property [30], and we do not know yet if Markov chains have characteristic formulae in PCTL or any other logic.

1.5 Abstraction and undecidability

In Section 1.4 above we related our completeness question to the finite model property and therefore to other problems, such as the decidability of a logic. This, of course, raises the question of decidability for abstraction in general, and for our completeness question in particular. As one would expect, already the abstraction problem is undecidable.

For example, Dams writes in [56]:

Appropriate abstractions of infinite state programs are not computable in general: it would imply that the program verification problem is decidable

Similarly, in [99] Michael Huth writes about probabilistic model checking:

The specification of \models may not be computable, e.g. [...] we could encode the Halting problem as a model check. Since Markov decision processes faithfully subsume such qualitative systems, the [model-checking problem] of infinite-state Markov decision processes and CTL is undecidable as well. [...]

The answers to individual model-checking instances $M \models \phi$ may be very hard to determine and such a determination may be closely tied to finding a ‘magic’ abstraction.

Therefore even a complete abstraction framework, as the one we develop in this thesis, can not provide the machinery to find these “magic” abstractions. Nevertheless, the completeness question is an important question: if an abstraction framework is known to be incomplete, the search for abstractions or abstraction refinements will not necessarily terminate and thus will need to be aborted at some point or replaced by a totally different approach, e.g. adjusting the specification to yield a complete formula. Furthermore, to address the completeness problem can be a way to attack the often closely related problem of decidability of satisfiability of the underlying logic.

1.6 Witnesses for truth and falsity

Another perspective on the work in this thesis is to look at abstractions as witnesses for truth or falsity.

One of the big advantages of model checking over other techniques for formal verification, e.g. theorem proving, is the availability of counter-examples. If a model checker fails to establish $M \models \phi$ then it can often point out where in its search space the verification process failed. Since explicit search model-checking algorithms work, more or less, directly on the actual state space of the model, such a negative answer can provide debugging information about the model, e.g. a counter-example which explains *why* $M \not\models \phi$. Such a counter-example is very valuable as it provides an explanation for why the model does not satisfy the property. This can help to uncover problems in the model and to drive the engineering process.

There are several issues connected to the (re-)presentation of counter-examples. For linear-time temporal logics the situation is quite convenient: formulae are interpreted over all traces of the model and therefore a counter-example is just a single trace. Reasonably short, single traces can be understood by human users fairly easily and usually the shortest counter-example is also the most interesting and meaningful one. For branching-time logics the situation is more complicated and the counter-examples to a formula are potentially more complex, e.g. tree-like counter-examples [46]. Let's take for example the CTL formula $\text{AFAX}q$, which informally says that

one can always reach a state where q is true at all successor states.

This formula “does in general not have linear counter-examples” [46]. Such complex counter-examples are potentially much more difficult to apprehend than trace-like, linear counter-examples. So far little work has been done on more satisfying representations of counter-examples for CTL model checking, but we mention work on tree-shaped counter-examples [46].

In probabilistic model checking the construction of meaningful counter-examples is even more difficult since counter-examples are sets of certain paths with a sufficiently large or small probability measure. While each of the paths on its own might not violate the (probabilistic) formula and may even have probability 0, their entirety can constitute a counter-example.

One is then concerned to find suitable representations, e.g. instructive subsets which are maximal with regard to their probability but minimal with regard to the number of (substantially different) traces and the length of these traces [87, 11, 6]. Again, a sensible presentation of useful information to the human user is crucial for debugging. Suggested approaches include projecting counter-examples, i.e. sets of traces, on the graph structure of the model [7], or encoding sets of counter-example as the automata or regular expressions which accept them [55].

Another way to represent the answer to a query $M \models \phi$ and its justification, are winning strategies in suitable game-based semantics [167, 94]. In this approach the truth or falsehood of a formula corresponds to a winning strategy for one of two adversarial players. The respective winning strategy, interpreted as an operational description, can be seen as a witness itself. For example Colin Stirling writes in [167]:

in the model checking case not only do [games] allow a user to know that a process has a property, but also *why* it has it. Games also allow a user to know *why* a process fails to have a property. In both these cases the justification can be given as a winning strategy.

Our game-based semantics for PCTL in Chapter 5 can be seen from this perspective of winning strategies as witnesses.

Counter-examples are also connected to abstractions and to the completeness of abstractions – the main research focus of this thesis. Let A be an abstraction of the concrete model M . Now let this abstraction be *sound*, i.e.

$$A \models \phi \text{ implies } M \models \phi \text{ for all formulae } \phi \in \mathcal{L} .$$

Even for sound abstraction one can in general not deduce an answer to the model-checking query on M from a negative answer on A , i.e. $A \not\models \phi$ does not imply $M \not\models \phi$. In this sense one could refer to soundness of abstraction as *soundness with regard to verification*. In the context of counter-examples one could be interested in *soundness with regard to refutation*:

$$A \not\models \phi \text{ implies } M \not\models \phi \text{ for all formulae } \phi \in \mathcal{L} .$$

Both types of soundness are related if the logic allows for a syntactic negation whose semantics is equivalent to non-satisfaction: Whenever satisfaction of $\neg\phi$ is equivalent to not satisfying ϕ on abstract and concrete models, then soundness for verification is equivalent to soundness for refutation.⁶

For sound abstraction and logics with negation this suggests to check $A \models \neg\phi$ whenever the query $A \models \phi$ is inconclusive. Unfortunately, in many abstraction frameworks, an abstraction A of M might be fine-tuned especially for checking a particular formula ϕ . This abstraction does not need to be equal or even related to an abstraction A' which would be suitable to check $\neg\phi$. To achieve small abstract models it can even be reasonable to drop the duality between negation and non-satisfaction on the abstract models. For example 3-valued frameworks [102, 31], e.g. with *may* and *must* conditions, allow for models which *must*-satisfy neither ϕ nor $\neg\phi$ but *may*-satisfy both of them.

Let the abstraction A of M be sound (for verification only). If model checking returns a negative answer $A \not\models \phi$ together with an abstract counter-example c , then this answer is in-conclusive for M and we are left with two possibilities:

1. The counter-example c is a *real* counter-example, i.e. one can derive a concrete counter-example on M from c and hence show that $M \not\models \phi$.
2. The counter-example c is *spurious*, i.e. it is an artifact of the abstraction. It does not decide whether $M \models \phi$ or $M \not\models \phi$. A spurious counter-example only reveals that the abstraction A is too coarse to yield a conclusive answer to the query on M .

In an approach called *counter-example guided abstraction refinement* (CEGAR) [43] such spurious counter-examples are used to refine the abstraction A incrementally. If model checking the abstract model A_i yields a spurious counter-example, then A_i is refined to a model A_{i+1} which rules out this particular spurious counter-example. The refined abstraction A_{i+1} is then model-checked, and the next iteration of the so called CEGAR loop begins. The refinement of abstract models in the CEGAR loop is monotone, in the

⁶Proof: Let $X \models \neg\phi$ be equivalent to $X \not\models \phi$, let A abstract M , and let $A \not\models \phi$. Then $A \not\models \phi$ iff $A \models \neg\phi$. Soundness for verification of $\neg\phi$ yields $A \models \neg\phi$ implies $M \models \neg\phi$. Again we invoke the semantic equivalence of negation and non-satisfaction, now on M : $M \models \neg\phi$ iff $M \not\models \phi$. Thus $A \not\models \phi$ implies $M \not\models \phi$.

sense that its iterations remove more and more spurious counter-examples, but in general it is not guaranteed that the CEGAR loop will ever terminate. The CEGAR loop might not terminate for two quite different reasons.

1. The class of abstract models, e.g. all finite-state ones, from which the CEGAR loop chooses the next refinement in each iteration, does simply not contain a witnessing abstraction at all. (This connects to our question of completeness.)
2. Even if the class of abstract models contains suitable witnesses it is not guaranteed that the CEGAR method will terminate and find a suitable abstraction. The algorithm might choose the wrong “branch” of the search space for the refinement and follow it to infinity.⁷

Recently CEGAR algorithms for probabilistic model checking were proposed in [93, 114]. Of course the same theoretical questions as for the non-probabilistic CEGAR algorithms apply. Furthermore, the problem of (finding and representing) probabilistic counter-examples has to be taken into account [87, 180].

1.7 Research agenda of this thesis

The aim of this thesis is to find a complete abstraction framework for PCTL on discrete-time Markov chains. Furthermore this thesis aims to provide a deeper insight into the particulars of PCTL and probabilistic verification in general. To this end we define an operational semantics for PCTL in Chapter 5, in addition to the work on the PCTL completeness problem in Chapter 4 and 6.

We chose the combination of PCTL and discrete-time Markov chains because they have been well established and are widely used as a means of modeling and specification. PCTL model checking on Markov chains is decidable in polynomial time [14] while the logic is rich enough to express specification patterns which are of interest for the specification of real applications.⁸ There also is well established tool support for PCTL model checking, e.g. PRISM [1, 62]. Our work thus contributes to the theoretical

⁷In general, depth-first search over infinite domains is not guaranteed to terminate even if the search space contains a correct solution.

⁸The extension PCTL* is already in PSPACE [14].

background for state space reduction methods which are used in existing model-checking toolkits to counter the state space explosion problem, e.g. symmetry reduction [45] and predicate abstraction [23, 113]. We will comment on possible extensions beyond PCTL and Markov chains in this thesis, but our focus is firmly set on PCTL as the probabilistic logic of choice in most probabilistic model checkers for discrete-time Markov chains.

Despite the theoretical nature and the non-constructiveness of the completeness question, it has some very important implications for practitioners. A non-existence result would be stunning because it would expose an inherent problem of the logic and models. If a complete abstraction framework could not exist, it would imply that there are (infinite) systems which can not be model-checked for certain properties ϕ by any method which depends on explicit but finite state representations. On the other hand the existence of a complete abstraction framework may allow us to use predicate abstraction, CEGAR, and other approximation techniques, with the knowledge that a finite abstraction does at least exist.

The class of models which constitutes a complete abstraction framework also gives a deeper insight into the expressiveness and the pitfalls of the logic PCTL, and into discrete-time Markov chains as a means of specification and modeling. For example, our result on completeness for a fragment of PCTL via *3-valued unfoldings* in Chapter 4 identifies the benign part of PCTL for which finitary abstractions can be “traditional” 3-valued models, and reveals the part of PCTL which needs new technical machinery for complete abstractions.

We break down our research agenda into a list of concrete objectives:

1. Summarise and reformulate the existing relevant work in a coherent fashion.
2. Define an operational semantics for PCTL over discrete-time Markov chains via an infinite two-player game.
3. Derive a first partial completeness result for a fragment of PCTL based on existing work and on insights from our operational semantics.
4. Define a new abstraction framework based on automata which achieves completeness for full PCTL.

5. Explore the complete abstraction framework with regard to its application above and beyond answering our completeness question.

1.8 Achievements in this thesis

This thesis presents three main contributions:

- We identify a “benign” fragment of PCTL for which completeness can be achieved with rather simple methods. These methods, namely, 3-valued abstractions (in fact, a 3-valued form of finite unfoldings) with approximative but compositional semantics, suggest themselves from a model-checking perspective.

A finite unfolding A is essentially an unwinding of the concrete model M up to a fixed depth k . Abstracting the concrete model M by a 3-valued finite unfolding A intuitively means that the concrete model is explored up to a certain depth k (as in bounded model checking) and that all the information beyond this finite horizon is ignored (or rather aggregated and approximated) in some meaningful way. This meaningful aggregation is achieved by 3-valued propositional labels under a conservative (“pessimistic”) semantic interpretation. A 3-valued label can assert that a property may be true or may be false in the abstract model A . This *uncertainty* in the abstract model is then resolved by further information in the refinement of A , e.g. the concrete model M . Such 3-valued labels are a widely used means of abstraction for (branching-time) model checking [102].

Our benign PCTL fragment is maximal for our choice of abstractions – and for most sensible choices of abstraction relation – as we provide counter-examples which show that any syntactic enrichment of the fragment is in general not complete.⁹

- We present a Hintikka game (i.e. an operational semantics) for PCTL which provides a deeper insight into the logic and the corresponding completeness problem.

⁹In a different context, “non-extendable” subsets – which we call *maximal* here – are sometimes called *complete*, too. To avoid confusion we use the term “complete” exclusively for the finitary completeness as described in Section 1.3 and as formally defined in Chapter 4.

A Hintikka game [94] is a way to capture the semantics of a logic as a 2-player game. This allows for an operational understanding of the truth and falsity of formulae. The fact that model M satisfies formula ϕ (in the standard semantics) corresponds to the existence of a winning strategy for one particular player in the Hintikka game for the logic. Such winning strategies are useful witnesses for $M \models \phi$ [167], and they have applications, e.g., in programme synthesis and counter-example generation.

- We develop a new class of automata, called *p-automata*, to achieve completeness for full PCTL. The fundamental difference of these p-automata to other probabilistic automata, such as the classical variant for finite words [157] (called *probabilistic finite automata* in [16]) or *probabilistic Büchi automata* and other probabilistic automata for infinite words [16], is that our automata accept entire Markov chains rather than words. Our p-automata combine the combinatorial structure of alternating automata [179] with the ability to quantify the probabilities of regular sets of paths. They constitute a new framework for Markov chains and PCTL, which opens a range of future work, e.g. on automata based probabilistic verification.

1.9 Outline of this thesis

Chapter 2 provides the necessary background on models, logics and abstraction relations, while Chapter 3 provides more specific background on automata and games. Chapter 4 formally defines the notion of finitary completeness, i.e. our research question. In the same chapter we present incompleteness results for full PCTL and prove completeness of abstraction by 3-valued Markov chains for a sizeable fragment of PCTL. Chapter 5 defines a game-based semantics for PCTL. In Chapter 6 we develop p-automata as complete abstraction framework for full PCTL. Chapter 7 evaluates our results, discusses relevant related work and gives an outlook on open questions and directions for future work.

The background material of Chapter 2 is needed throughout this thesis while the material in Chapter 3 is specific to Chapter 5 and Chapter 6. The three core Chapters 4, 5 and 6 present our technical contributions. These

three chapters are independent of each other and can be read in any order. Thus, there are several alternative approaches to read (the main part of) this thesis. Besides the linear read from cover to cover, there are the following two noteworthy variants.

- For a model checking approach, starting with classical models and achieving partial completeness via 3-valued abstractions: read Chapters 2 and 4.
- Focusing on automata and games one can move straight to the most general completeness result – namely completeness for full PCTL – by skipping Chapter 4 and 5: read Chapters 2, 3, and 6.

The order in which we present our contributions in this thesis differs from the chronological order in which we achieved the results. Chronologically, we started with counter-examples as in [58] to show that we will need a more sophisticated machinery to achieve completeness for full PCTL. Next we developed the Hintikka game for PCTL, now presented in Chapter 5. This was also motivated by the work of Dams and Namjoshi [58, 59], which suggests that some sort of game or automata machinery will be necessary to achieve completeness. The game-based approach also aimed at gaining a better understanding of PCTL and the PCTL completeness problem. The Hintikka game indeed provided deep insights into the completeness question, and we were able to derive the complete fragment presented in Chapter 4 more or less directly from certain winning strategies in the Hintikka game. After establishing this complete fragment, which requires rather simple abstract models, we addressed the full completeness problem. We approached the full problem by capturing PCTL formulae as automata. This required the development of a new class of automata which are substantially different from existing (probabilistic) automata: our so called *p-automata* accept and reject whole Markov chains as inputs; their languages are unions of bisimulation equivalence classes of Markov chains; and Markov chains and PCTL formulae can both be expressed as *p-automata*. These *p-automata* are presented in Chapter 6. They achieve completeness for full PCTL and thus answer the main research questions of this thesis.

In this thesis we present the results in a slightly different order which moves from the simple concepts to the more complex ideas. First we discuss how far one gets with standard methods of 3-valued abstractions, which

is an obvious idea from the model-checking perspective and which already yields completeness for a sizeable fragment of PCTL. We then construct certain counter-examples which show that the 3-valued approach alone can not be sufficient for full completeness and that we need to move on to more complex methods. Next we introduce our Hintikka game, to pinpoint the remaining problems with solving the completeness problem, and to provide technical machinery for the following introduction of our p-automata. We then introduce p-automata and related stochastic games, and show how they capture and exceed the previous results and finally achieve full completeness.

1.10 Publications

Selected results from this thesis have been disseminated in scientific publications. Details of the four publications which are based on the work of this thesis are given below.

The game-based semantics for PCTL (Chapter 5 in this thesis) was developed in joint work with Harald Fecher, Michael Huth and Nir Piterman as a first step toward finitary abstractions. It was published in the article *Hintikka Games for PCTL on Labelled Markov Chains* in the Proceedings of the International Conference on Quantitative Evaluation of Systems (QEST) 2008 [75]. Based on these results we consequently were invited to submit an extended paper – titled *PCTL Model Checking of Markov Chains: Truth and Falsity as Winning Strategies in Games* – for a special issue of the International Journal on Performance Evaluation, which was published in 2010 [76].

The completeness results which were achieved for a fragment of PCTL via 3-valued unfoldings of Markov chains (Chapter 4) was published jointly with Michael Huth and Nir Piterman in 2009. The article appeared under the title *Three-Valued Abstractions of Markov Chains: Completeness for a Sizeable Fragment of PCTL* in the proceedings of the International Symposium on Fundamentals of Computation Theory (FCT) [103].

Our p-automata framework (Chapter 6) was accepted for publication as *p-Automata: New Foundations for Discrete-Time Probabilistic Verification*, to appear in the Proceedings of the International Conference on Quantitative Evaluation of Systems (QEST) 2010 [104]. The article is co-authored with Michael Huth and Nir Piterman.

1.11 Summary of chapter

This chapter gave an informal introduction and motivation to our research. It explained our main research questions and the contributions made in this thesis.

The context of this thesis was outlined as model checking. One important motivation for our research are finite-state abstractions as a means to counter the state explosion problem for model checking.

This chapter then introduced the notion of soundness and completeness of an abstraction. Soundness guarantees the usefulness of an abstraction for model checking as it allows to derive an answer for the concrete model checking problem from the answer for a corresponding abstract model checking problem. Completeness guarantees the existence of a finite state abstraction for every concrete model checking problem. This notion of completeness is in general distinct from the finite model property as it requires a prescribed abstraction relation, e.g. simulation, between the concrete model checking problem and its abstraction.

As research topic for this thesis we identified probabilistic verification, i.e. for PCTL over Markov chains, as one particular model checking framework with previously unsolved completeness question. This chapter then stated our main research aims, i.e. to develop a complete abstraction framework for PCTL over Markov and to advance the understanding of probabilistic verification through a new automata based framework for PCTL and Markov chains.

Other themes touched in this thesis were identified as, e.g., game based semantics for probabilistic logics; winning strategies as witnesses of truth and falsity; and probabilistic automata.

2 Background on Models and Logics

In this chapter we provide relevant technical background on the models and logics used in this thesis. Before we introduce the probabilistic models and logics in Section 2.4 which are used in the main part of this thesis – i.e. in Chapters 4, 5 and 6 – we introduce some non-probabilistic models and logics first. We discuss general concepts and intuitions on these non-probabilistic models and refer back to them in the probabilistic context. Completeness results from the literature for these non-probabilistic models are discussed in Section 4.1 and Section 7.1.

The non-probabilistic part (Section 2.1–2.3) and the probabilistic part (Section 2.4–2.6) of this chapter each follow the same structural outline:

- (i) presentation of models;
- (ii) logics and accompanying satisfaction relation;
- (iii) refinement and equivalence of models.

In the following we assume familiarity with basic set theory and graph theory. We try to use well established notations wherever possible but modify some standard notations to facilitate the presentation of our main results later in the thesis. While we aim for clarity and consistency of notations, we sometimes use “sloppy” notations if they enhance readability without jeopardising clarity.

In this thesis most sets come with relations on them, for example sets of states come with a transition relation on them. Therefore we use *direct product* not only for sets, but also for sets with relations. The direct product then lifts the relations to the (set-theoretic) direct product of the respective sets.

Definition 1 (Direct product)

Let $\mathcal{S}_i = (S_i, R_i)$ for $i \in \{1, 2\}$ be a set S_i with a relation $R_i \subseteq S_i \times S_i$. The *direct product*¹ of \mathcal{S}_1 and \mathcal{S}_2 is

$$\mathcal{T} = \mathcal{S}_1 \times \mathcal{S}_2$$

where $\mathcal{T} = (T, Q)$ with set $T = \{(a, b) \mid a \in S_1, b \in S_2\}$ and with relation $Q \subseteq T \times T$ such that

$$((a, b), (c, d)) \in Q \text{ iff } (a, c) \in R_1 \text{ and } (b, d) \in R_2 . \quad \blacklozenge$$

2.1 Kripke structures

For non-probabilistic model checking two important classes of models are *Kripke structures* and *labelled transition systems*. They are largely equivalent but offer different perspectives on the modeled systems [82]. We take a (state-based) model-checking approach and therefore focus on Kripke structures. Kripke structures also resemble discrete-time Markov chains, which are the models used in the main part of this thesis, more closely than labelled transition systems. We do normally think of models as graph structures decorated with labels of some sort. More formally we use the following definitions:

Definition 2 (Kripke structure)

A *Kripke structure* K is a tuple (S, R, L) where

- (i) S is a non-empty set of *states*,
- (ii) $R \subseteq S \times S$ is a (total) *transition relation*,
- (iii) L is a total labelling function $L: S \times \mathbf{AP} \longrightarrow \{\mathbf{tt}, \mathbf{ff}\}$ for a finite set \mathbf{AP} of *atomic propositions*. \blacklozenge

The labelling function L determines which atomic propositions $\mathbf{q} \in \mathbf{AP}$ are true (\mathbf{tt}) respectively false (\mathbf{ff}) in a state s . Thus the labelling function

¹Sometimes also called *Cartesian product*.

L defines the set of all propositions \mathbf{q} which hold at state s . By abuse of notation, we write:

$$L(s) := \{\mathbf{q} \in \text{AP} \mid L(s, \mathbf{q}) = \text{tt}\}$$

For the transition relation R we often use the infix notation $s \longrightarrow_R s'$ instead of $(s, s') \in R$, and omit the subscript R whenever R is determined by the context. For Kripke structures $K = (S, R, L)$ we sometimes abuse notation and write $s \in K$, meaning $s \in S$.

We now introduce further concepts and notations for Kripke structures.

Definition 3

Let K be a Kripke structure.

1. K is called *finite branching* iff for all states $s \in S$ the set of successors of s , $\text{Succ}(s) := \{s' \in S \mid s \longrightarrow s'\}$, is finite.
2. K is called *finite state* iff the set S is finite. A finite state Kripke structure is always finite-branching.
3. K is called *pointed* iff it has a designated *initial state* $s^{\text{in}} \in S$. We write (K, s^{in}) to indicate the initial state s^{in} of K .
4. K is called *serial* iff each state $s \in S$ has at least one *successor*, i.e. some s' with $s \longrightarrow s'$. (Thus K is serial iff the relation R is total.) In a serial structure every sequence of states can be extended to an infinite sequence, i.e. a path as defined below.
5. We call a state s *absorbing* iff it has a *self-loop* $s \longrightarrow s$ and no other outgoing transitions. ◆

A Kripke structure $K = (S, R, L)$ contains a directed graph $G_K := (S, R)$ where the vertices are the states S and the edges are defined by the transition relation R . In a graphical representation of this graph propositions \mathbf{q} annotate states s to indicate $L(s, \mathbf{q}) = \text{tt}$, as seen in Figure 2.1. We define *paths* in K as paths in the directed graph G_K .

Definition 4 (Path)

A *path* π in $K = (S, R, L)$ is an infinite sequence of states $s_0 s_1 \dots$ where $s_i \in S$ and $(s_i, s_{i+1}) \in R$ for all $i \in \mathbb{N}$. Furthermore, let

- $\pi[i] := s_i$ be the element at position i in π ;
- $\pi[i, \dots, j]$ be the subsequence $s_i \dots s_j$ of π ; and
- π^i be the suffix $s_i s_{i+1} \dots$ of π starting at the i -th state for $i \geq 0$.

We denote $\text{Path}(s)$ the set of all paths starting at state s . Whenever we write about *paths of Kripke structure* K without stating another start state, we mean paths which start at the initial state s^{in} of K . \blacklozenge

Example 1

Figure 2.1 shows a Kripke structure K over $\text{AP} = \{\mathbf{q}, \mathbf{r}, \mathbf{s}\}$ represented as directed graph G_K . State s_0 is marked as initial state by an incoming arrow. State names s_i are written in the vertices; atomic propositions \mathbf{x} for which $L(s_i, \mathbf{x}) = \text{tt}$ are written next to the state s_i . Throughout this thesis we use a typewriter font for propositions, e.g. \mathbf{s} , and a serif font for states, e.g. s_0 .

All paths in K are infinite sequences of one of the following four forms:

$$\begin{aligned} & s_0^\omega \\ & s_0^* s_1^\omega \\ & s_0^* s_1^* s_2^\omega \\ & s_0^* s_2^\omega \end{aligned}$$

where we use the standard notation of regular expressions. In particular x^* stands for zero or finitely many repetitions of x , and x^ω stands for infinitely many repetition of x . \blacklozenge

Example 2

The Kripke structure K in Figure 2.1 is finite-branching, finite state, pointed with initial state s_0 , and serial. State s_2 is absorbing and the only absorbing state of K . \blacklozenge

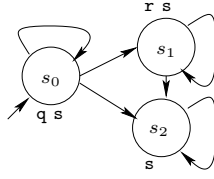


Figure 2.1: A Kripke structure K .

Seriality of Kripke structures can be enforced by adding a *self-loop* $s \rightarrow s$ to each state $s \in S$ which has no other outgoing transitions. Alternatively a *sink state* d , which has only one outgoing transition to itself, can be introduced and connected to each state s without outgoing transitions via $s \rightarrow d$. In particular, seriality is convenient and a standard assumption for the definition of semantics for linear time temporal logics. If s_n is an absorbing state, we sometimes abbreviate the (infinite) path $\pi = s_0 \dots s_n s_n s_n \dots$ by writing $\pi = s_0 \dots s_n$.

Assumption 1 (Pointed serial Kripke structures)

In this thesis all Kripke structures K are pointed and serial. ◆

As mentioned above, another important class of models are *labelled transition systems*. The notion of a labelled transition system $T = (S, R, \text{Act})$ is somehow dual to that of a Kripke structure $K = (S, R, L)$. While the labelling function of a Kripke structure decorates the states, the *action labels* Act of a labelled transition system decorate the transitions. Formally we have the following definition:

Definition 5 (Labelled transition system)

A *labelled transition system* T is a tuple (S, R, Act) , where

- (i) S is a non-empty set of *states*; and
- (ii) $R \subseteq S \times \text{Act} \times S$ is a *transition relation*;

where Act is a finite set of *action labels*. ◆

A labelled transition system $T = (S, R, \text{Act})$ contains a directed graph $G_T := (S, R')$ where vertices are the states S and edges are defined according

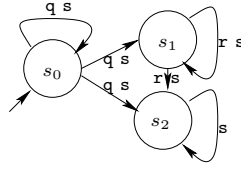


Figure 2.2: A labelled transition system (T, s_0) .

to R . An edge from s to s' is annotated with action $\alpha \in \text{Act}$ if $(s, \alpha, s') \in R$. An example of such a graph is illustrated in Figure 2.2.

Example 3

Figure 2.2 shows a labelled transition system T represented as directed graph G_T . State names s_i are written in the vertices; the initial state s_0 is designated by an incoming arrow; action labels \mathbf{x} are written next to transitions. Rather than drawing separate edges (s, \mathbf{x}, s') between the same states for multiple actions, the actions are written on a single edge in the illustration. \blacklozenge

The notions of Kripke structures and labelled transition systems are basically equivalent but differ in that they “provide complementary views for reasoning about reactive systems” [82]. Kripke structures “model state changes of reactive systems”, while labelled transition systems “model the external behaviour of a system (i.e. sequences of actions the system can perform)” [82].

As a unifying formalism *doubly labelled transition systems* were introduced in [97] as models which include action labels on transitions as well as propositional labels on states. Doubly labelled transition systems contain Kripke structures and labelled transition systems as special cases, but are not more expressive than either of them [82].

In the following we restrict ourselves to Kripke structures as our choice of non-probabilistic models, because they resemble Markov chains more closely than labelled transition systems. Markov chains – the models which we will use in the main part of this thesis – are labelled with propositions on states and with probabilities (not actions) on transitions.

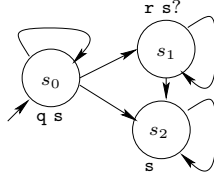


Figure 2.3: A partial Kripke structure (K, s_0) .

Especially in the context of abstractions, 3-valued generalisation of models arise rather naturally [31, 32, 98, 102]. For Kripke structures such generalisations are called *partial Kripke structures* [31]. They are Kripke structures where the propositional labels have three different truth-values (tt , ? , ff), rather than just two (tt , ff). For each proposition $p \in \text{AP}$, at a state $s \in S$ the truth-value of p can be true (tt), false (ff) or “unknown” (?). We introduce this generalisation now. It is then revisited in Section 2.2.4, where we discuss the practical application of 3-valuedness for abstraction, and in Section 2.3 in the context of *sound* abstractions.

Definition 6 (Partial Kripke structure)

A *partial Kripke structure* K is a tuple (S, R, L) , where

- (i) S is a non-empty set of *states*;
- (ii) $R \subseteq S \times S$ is a *transition relation*; and
- (iii) L is a total labelling function $L: S \times \text{AP} \longrightarrow \{\text{tt}, \text{?}, \text{ff}\}$ for a finite set AP of *atomic propositions*. ◆

In the graph representation we use the label $q?$ to mark states s for which $L(s, q) = \text{?}$. The absence of any label q or $q?$ indicates $L(s, q) = \text{ff}$.

Example 4

A partial Kripke structure K is illustrated in Figure 2.3. ◆

The 3-valued generalisations of labelled transition systems are called *modal transition systems* [133]. A unifying framework for partial Kripke structures

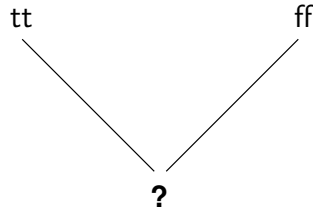


Figure 2.4: Hasse diagram of the information order on $\{\text{tt}, \text{?}, \text{ff}\}$.

and modal transition systems is called *doubly labelled modal transition systems* [102] which is the 3-valued generalisation of the doubly labelled transition systems mentioned above. Again, doubly labelled modal transition systems do not increase the expressiveness of the models; translations between the three types of 3-valued models are possible and require only linear time and logarithmic space [82].

To establish a partial order of refinement respectively abstraction between 3-valued models or between 2-valued and 3-valued models, one needs an underlying partial order on the three possible truth-values. With regard to soundness of abstraction we choose the so-called *information order*.

Definition 7 (Information order)

We define the following partial order \leq on the set of truth-values $\{\text{tt}, \text{?}, \text{ff}\}$:

$$\begin{aligned} \text{?} &\leq \text{tt} \\ \text{?} &\leq \text{ff} \end{aligned}$$

where these are the only non-reflexive instances of \leq . This partial order is called *information order* and goes back at least to Stephen Kleene [118]. It is illustrated as Hasse diagram in Figure 2.4. \blacklozenge

2.2 Linear-time and branching-time temporal logic

There is a trade-off between the expressiveness of a logic and the complexity of the related decision problems such as, for example, the corresponding model-checking problem. In practice model checking is done for modal log-

ics, because they are considered a good balance between expressiveness and complexity.

For temporal modal logics there are two main paradigms: linear time [151, 152] and branching time [42].

In linear time one reasons about time as *linear* sequences of states; each state of such a sequence in time has exactly one successor, the next step in time. Linear time logics then quantify over all such sequences for a system. Such logics express properties of paths, for example “Every path eventually reaches a state that satisfies proposition *good*”.

Branching time on the other hand treats time as a sequence of states each of which is a branching point for multiple possible futures. In contrast to linear time the states in such a sequence may have multiple successors. Branching time logics then reason about sequences of such states in terms of their branching behaviour, for example “Every state has at least one successor which satisfies proposition *good*”. Hence a process in branching time is essentially a tree of all its possible temporal developments, while in linear time it is a set of linear paths capturing all its possible temporal developments.

Linear time and branching time are in general incomparable with regard to their expressiveness. Which one of the two paradigms is more suitable as specification language and for model checking is still an ongoing debate [128, 127, 70, 174]. Allen Emerson and Joseph Halpern [70] recognise linear-time temporal logic for its simplicity but advocate branching-time temporal logic for writing specifications. Indeed branching-time logic seems to have the easier model-checking problem and is closed under negation which linear-time logic is not – in the sense that it “cannot express a fact such as ‘property *P* does not hold along some execution of the program’” [70]. More recently Moshe Vardi [174] argued strongly for linear-time logic for specifications and model checking especially from a practical perspective.

In the main part of this thesis we are concerned with PCTL which is essentially a probabilistic branching-time logic. For probabilistic systems, such as Markov chains, the location of the (probabilistic) branching is crucial, so that a branching-time approach seems appropriate. At the same time certain parts of PCTL – namely the predicates for path formulae – resemble a linear-time logic. Therefore, we now define and discuss both linear-time and branching-time logics.

In the following we call two formulae ϕ and ψ *semantically equivalent* if they have the same set of models for an underlying satisfaction relation between models and formulae.

2.2.1 LTL

In this section we introduce linear-time logic [151, 152].

Definition 8 (LTL syntax)

The formulae of *Linear Time Logic (LTL)* are as follows:

$$\phi, \psi ::= \mathbf{q} \mid \neg\phi \mid \phi \wedge \psi \mid \mathbf{X}\phi \mid \phi \mathbf{U} \psi$$

where \mathbf{q} ranges over a finite set of *atomic proposition* \mathbf{AP} , and \mathbf{X} , \mathbf{U} are the temporal operators called *Next* and (*strong*) *Until* respectively. \blacklozenge

We use the usual abbreviations:

$$\begin{aligned} \mathbf{ff} &\equiv \neg\mathbf{q} \wedge \mathbf{q} \\ \mathbf{tt} &\equiv \neg\mathbf{ff} \\ \phi \vee \psi &\equiv \neg(\neg\phi \wedge \neg\psi) \\ \phi \Rightarrow \psi &\equiv \neg\phi \vee \psi \\ \mathbf{F}\phi &\equiv \mathbf{tt} \mathbf{U} \phi \\ \mathbf{G}\phi &\equiv \neg\mathbf{F}\neg\phi \end{aligned}$$

The temporal modalities \mathbf{F} and \mathbf{G} are called *Eventually* (“*in the Future*”) and *Globally* respectively.

Recall that all our Kripke structures $K = (S, R, L)$ are pointed and serial, and that a path π is an *infinite* sequence of states. (Refer to Definition 4 on page 31 for path notations such as suffixes π^i .)

Definition 9 (LTL semantics)

The semantics of LTL on pointed Kripke structures $K = (S, R, L)$ is defined via paths π as follows:

- $\pi \models \mathbf{q}$ iff $L(\pi[0], \mathbf{q}) = \mathbf{tt}$.

- $\pi \models \neg\phi$ iff $\pi \not\models \phi$.
- $\pi \models \phi \wedge \psi$ iff $\pi \models \phi$ and $\pi \models \psi$.
- $\pi \models X\phi$ iff $\pi^1 \models \phi$.
- $\pi \models \phi U \psi$ iff there exists $k \in \mathbb{N}$ such that $\pi^k \models \psi$ and $\pi^i \models \phi$ for all $i < k$.

The Kripke structure (K, s_0) satisfies an LTL formula ϕ , denoted $K \models \phi$, iff $\pi \models \phi$ for every $\pi \in \text{Path}(s_0)$, i.e. for all paths $\pi = s_0 s_1 \dots$ which begin in the initial state s_0 . \blacklozenge

Definition 10

Let K be a Kripke structure and α an LTL formula. Then $\text{Path}(s, \alpha)$ is the set of paths π which start at s and satisfy α :

$$\text{Path}(s, \alpha) := \{\pi \text{ path in } K \mid \pi[0] = s \text{ and } \pi \models \alpha\} \quad \blacklozenge$$

Clearly we have $\text{Path}(s, \text{tt}) = \text{Path}(s)$ which justifies the similarity of notation.

Example 5

Kripke structure K of Figure 2.1 on page 33 satisfies Xs and Gs but neither Xq nor Xr . We have $\text{Path}(s_0) = \{s_0^\omega\} \cup \{s_0^* s_1^\omega\} \cup \{s_0^* s_2^\omega\} \cup \{s_0^* s_1^* s_2^\omega\}$ and $\text{Path}(s_0, q U r) = \{s_0^* s_1^\omega\} \cup \{s_0^* s_1^* s_2^\omega\}$. As $\text{Path}(s_0, q U r)$ is only a subset of all paths from s_0 , namely the ones which visit s_1 , Kripke structure K does not satisfy $q U r$. \blacklozenge

Interestingly, the negation of an Until formula can be expressed as a disjunction of an Until and a negated Until:

$$\begin{aligned} \neg(\phi U \psi) &= ((\neg\psi) U (\neg\phi \wedge \neg\psi)) \vee G\neg\psi \\ &= ((\neg\psi) U (\neg\phi \wedge \neg\psi)) \vee \neg(\text{tt} U \psi) \end{aligned} \quad (2.1)$$

A widely used syntactic extension of LTL uses the *weak Until* operator W , which encodes this disjunction explicitly. The operator $\phi W \psi$, sometimes

also called *Unless*, behaves like the strong Until $\phi \mathbf{U} \psi$ except that the second condition ψ does not have to be realised. Formally:

- $\pi \models \phi \mathbf{W} \psi$ iff for all $l \in \mathbb{N}$ we have either $\pi^l \models \phi$ or there is $0 \leq j \leq l$ with $\pi^j \models \psi$.

The weak Until \mathbf{W} can be expressed by strong Until \mathbf{U} and vice versa:

$$\begin{aligned} \phi \mathbf{U} \psi &\equiv \neg((\neg\psi) \mathbf{W}(\neg\phi \wedge \neg\psi)) \\ \phi \mathbf{W} \psi &\equiv \neg((\neg\psi) \mathbf{U}(\neg\phi \wedge \neg\psi)) \end{aligned} \tag{2.2}$$

Thus, the weak Until does not increase the expressiveness of LTL, but it allows to write another intuitive equivalence:

$$\begin{aligned} \mathbf{G} \phi &\equiv \neg \mathbf{F} \neg\phi \\ &\equiv \neg \mathbf{tt} \mathbf{U} \neg\phi \\ &\equiv \phi \mathbf{W} \mathbf{ff} . \end{aligned}$$

With the two variants of Until in the syntax one can convert every LTL formula ϕ into an equivalent formula ϕ' in the following *negation normal form*.

Definition 11 (LTL negation normal form)

The formulae of LTL in *negation normal form*² (NNF) are as follows:

$$\phi, \psi ::= \mathbf{q} \mid \neg\mathbf{q} \mid \phi \wedge \psi \mid \phi \vee \psi \mid \mathbf{X}\phi \mid \phi \mathbf{U} \psi \mid \phi \mathbf{W} \psi \quad \blacklozenge$$

Arbitrary LTL formulae ϕ can be converted to an equivalent formula in negation normal form by applying the following rules recursively to the subformulae of ϕ , starting with ϕ itself:

$$\begin{aligned} \mathbf{nnf}(\mathbf{q}) &= \mathbf{q} \\ \mathbf{nnf}(\neg\mathbf{q}) &= \neg\mathbf{q} \\ \mathbf{nnf}(\neg\neg\phi) &= \phi \end{aligned}$$

²Also called *positive normal form* by other authors.

$$\begin{aligned}
\text{nnf}(\phi \wedge \psi) &= \text{nnf}(\phi) \wedge \text{nnf}(\psi) \\
\text{nnf}(\neg(\phi \wedge \psi)) &= \text{nnf}(\neg\phi) \vee \text{nnf}(\neg\psi) \\
\text{nnf}(\phi \vee \psi) &= \text{nnf}(\phi) \vee \text{nnf}(\psi) \\
\text{nnf}(\neg(\phi \vee \psi)) &= \text{nnf}(\neg\phi) \wedge \text{nnf}(\neg\psi) \\
\\
\text{nnf}(X\phi) &= X \text{nnf}(\phi) \\
\text{nnf}(\neg X\phi) &= X \text{nnf}(\neg\phi) \\
\text{nnf}(\phi U \psi) &= \text{nnf}(\phi) U \text{nnf}(\psi) \\
\text{nnf}(\neg(\phi U \psi)) &= \text{nnf}(\neg\psi) W \text{nnf}(\neg\phi \wedge \neg\psi) \\
\text{nnf}(\phi W \psi) &= \text{nnf}(\phi) W \text{nnf}(\psi) \\
\text{nnf}(\neg(\phi W \psi)) &= \text{nnf}(\neg\psi) U \text{nnf}(\neg\phi \wedge \neg\psi)
\end{aligned}$$

Both, weak and strong Until, can be expressed in terms of the *Release* operator R which is formally defined as follows:

$$\begin{aligned}
\pi \models \phi R \psi &\text{ if and only if there is no } l \in \mathbb{N} \text{ such that:} \\
\pi^l \models \neg\psi &\text{ and } \pi^j \models \neg\phi \text{ for all } 0 \leq j < l .
\end{aligned}$$

Weak and strong Until can be written in terms of R:

$$\begin{aligned}
\phi U \psi &\equiv (\neg\phi \vee \psi) R(\phi \vee \psi) \\
\phi W \psi &\equiv \neg(\neg\phi R \neg\psi)
\end{aligned}$$

Since the Release is rather un-intuitive in its meaning and thus not well suited for writing specifications, we do not include R explicitly in the syntax of LTL.

Example 6

Below we state some LTL formulae, each followed by an equivalent formula in negation normal form:

$$\begin{aligned}
G(\neg(\text{started} \wedge \neg\text{ready})) &\equiv (\neg\text{started} \vee \text{ready}) W \text{ff} \\
G(\neg\text{started} \vee F \text{acknowledged}) &\equiv (\neg\text{started} \vee (\text{tt} U \text{acknowledged})) W \text{ff} \\
\neg G F \text{enabled} \vee G F \text{running} &\equiv ((\text{tt} U \neg\text{enabled}) W \text{ff}) \vee \\
&\quad ((\text{tt} U \text{running}) W \text{ff}) \quad \blacklozenge
\end{aligned}$$

Assumption 2

We assume W to be part of the LTL syntax, although we may not explicitly mention it in, for example, structural inductions or definitions. We do not include the Release R explicitly in the LTL syntax. \blacklozenge

2.2.2 CTL

In this section we introduce the branching-time logic CTL [42].

Definition 12 (CTL syntax)

The formulae of *Computation Tree Logic* (CTL) are state formulae ϕ generated as follows:

$$\begin{aligned}\phi, \psi &::= \mathbf{q} \mid \neg\phi \mid \phi \wedge \psi \mid \mathbf{A}[\alpha] \mid \mathbf{E}[\alpha] \\ \alpha &::= \mathbf{X}\phi \mid \phi \mathbf{U}\psi\end{aligned}$$

where \mathbf{q} ranges over a finite set of *atomic proposition* \mathbf{AP} , \mathbf{X} and \mathbf{U} are the temporal modalities *Next* and (*strong*) *Until* respectively, and \mathbf{E} and \mathbf{A} are the existential and universal quantifier respectively. \blacklozenge

The CTL semantics casts *path formulae* α into state formulae ϕ by quantifying over the paths $\text{Path}(s)$ starting from a particular state s .

Definition 13 (CTL semantics)

The semantics of CTL on Kripke structures $K = (S, R, L)$ is defined as follows. Let $s \in S$ be a state.

- $s \models \mathbf{q}$ iff $L(s, \mathbf{q}) = \text{tt}$.
- $s \models \neg\phi$ iff $s \not\models \phi$.
- $s \models \phi \wedge \psi$ iff $s \models \phi$ and $s \models \psi$.
- $s \models \mathbf{E}[\mathbf{X}\phi]$ iff there exists s' such that $(s, s') \in R$ and $s' \models \phi$.
- $s \models \mathbf{A}[\mathbf{X}\phi]$ iff for all s' with $(s, s') \in R$ we have $s' \models \phi$.
- $s \models \mathbf{E}[\phi \mathbf{U}\psi]$ iff there exists a path $\pi \in \text{Path}(s)$ starting at s with $\pi \models \phi \mathbf{U}\psi$, i.e. there exists $i \in \mathbb{N}$ such that $\pi[i] \models \psi$ and $\pi[j] \models \phi$ for all $j < i$.

- $s \models A[\phi U \psi]$ iff for all paths $\pi \in \text{Path}(s)$ starting at s we have $\pi \models \phi U \psi$, i.e. there exists $i \in \mathbb{N}$ such that $\pi[i] \models \psi$ and $\pi[j] \models \phi$ for all $j < i$.

A pointed Kripke structure (K, s_0) satisfies ϕ , denoted $K \models \phi$, iff its initial state s_0 satisfies ϕ . \blacklozenge

Example 7

The Kripke structure K in Figure 2.1 on page 33 satisfies $E[q U r]$ but not $A[q U r]$. It also satisfies $A[X s]$ and $A[s W ff]$. \blacklozenge

We use similar abbreviations as for LTL, but with the difference that CTL temporal modalities always occur in the immediate scope of a quantifier:

$$EF \phi \equiv E[tt U \phi]$$

$$AF \phi \equiv A[tt U \phi]$$

$$EG \phi \equiv \neg AF \neg \phi$$

$$AG \phi \equiv \neg EF \neg \phi$$

Dualities for the Until operators in CTL are not as straightforward as in LTL. In particular EU and AU are not dual to each other, since the disjunction which results from a negated strong Until formula in LTL (see equation (2.2.1) on page 39) does not fulfill the syntactic restrictions on path formulae α . As a result EU cannot be expressed as combination of other CTL operators [139]. The expressiveness of CTL operators is discussed in [130] and [139]. The latter article identifies *adequate*³ fragments of CTL. The main problem is that the quantifiers E and A do not distribute over the logical operators \wedge and \vee respectively. The logic CTL* [70] allows combinations of path-formulae within the scope of a quantifier. Thus equation (2.2.1) can be lifted to CTL* with the usual duality between universal and existential

³A syntactic subset \mathcal{A} of a logic \mathcal{L} (seen as a set of formulae) is *adequate* iff it is as expressive as the full logic \mathcal{L} , i.e. for each $\phi \in \mathcal{L}$ there exists some $\psi \in \mathcal{A}$ such that ϕ and ψ are semantically equivalent.

quantification:

$$\neg A[\phi \text{ U } \psi] = E[((\neg\psi) \text{ U } (\neg\phi \wedge \neg\psi)) \vee G \neg\psi]$$

$$\neg E[\phi \text{ U } \psi] = A[((\neg\psi) \text{ U } (\neg\phi \wedge \neg\psi)) \vee G \neg\psi]$$

As for LTL, CTL syntax can be extended with a *weak Until* operator W . The weak Until in its two quantified variants $E[\phi W \psi]$ and $A[\phi W \psi]$ behaves like the corresponding variants of strong Until, i.e. $E[\phi U \psi]$ and $A[\phi U \psi]$ respectively, where the second condition ψ does not need to be reached. Formally:

- $s \models E[\phi W \psi]$ iff there exists a path $\pi \in \text{Path}(s)$ starting at s with $\pi \models \phi W \psi$, i.e. for all $l \in \mathbb{N}$ we have either $\pi^l \models \phi$ or there is $0 \leq j \leq l$ with $\pi^j \models \psi$.
- $s \models A[\phi W \psi]$ iff for all paths $\pi \in \text{Path}(s)$ starting at s we have $\pi \models \phi W \psi$, i.e. for all $l \in \mathbb{N}$ we have either $\pi^l \models \phi$ or there is $0 \leq j \leq l$ with $\pi^j \models \psi$.

Weak Until W can be expressed by strong Until U and vice versa:

$$E[\phi \text{ U } \psi] \equiv \neg A[(\neg\psi) \text{ W } (\neg\phi \wedge \neg\psi)]$$

$$E[\phi \text{ W } \psi] \equiv \neg A[(\neg\psi) \text{ U } (\neg\phi \wedge \neg\psi)]$$

$$A[\phi \text{ U } \psi] \equiv \neg E[(\neg\psi) \text{ W } (\neg\phi \wedge \neg\psi)]$$

$$A[\phi \text{ W } \psi] \equiv \neg E[(\neg\psi) \text{ U } (\neg\phi \wedge \neg\psi)]$$

Nevertheless, not all dualities for LTL formulae have a direct correspondent in CTL. For example the duality

$$\phi \text{ W } \psi \equiv (\phi \text{ U } \psi) \vee \neg(\text{tt U } \neg\phi)$$

does not in general have a direct correspondent in CTL. The universal quantifier does not distribute over the disjunction. Even if it would, a quantified variant of the right conjunct, e.g. $A[\neg(\text{tt U } \neg\phi)]$ syntactically would not belong to CTL but to CTL* – a more general logic which we discuss in Section 2.2.3.

Nevertheless, with the dualities at hand we can convert CTL formulae into equivalent formulae in the following negation normal form.

Definition 14 (CTL negation normal form)

The formulae of CTL in *negation normal form*⁴ (NNF) are state formula ϕ generated as follows:

$$\begin{aligned}\phi, \psi ::= & \mathbf{q} \mid \neg \mathbf{q} \mid \phi \wedge \psi \mid \phi \vee \psi \mid \mathbf{A}[\alpha] \mid \mathbf{E}[\alpha] \\ \alpha ::= & \mathbf{X}\phi \mid \phi \mathbf{U}\psi \mid \phi \mathbf{W}\psi\end{aligned}$$

◆

CTL formulae in negation normal form are all those CTL formulae in which negation only occurs on propositions. In addition to the syntax introduced in Definition 12 the negation normal form explicitly contains the logical disjunction \vee , which in NNF can not be derived from conjunction \wedge via negation, and the weak Until \mathbf{W} as negated counterpart for the strong Until \mathbf{U} . Every formula ϕ of CTL can be transformed into a semantically equivalent formula ϕ' in NNF by recursively pushing negations inward using DeMorgan's laws and the dualities between \mathbf{U} and \mathbf{W} as described above. This conversion of formula ϕ is formalised by the following recursive rules:

$$\begin{aligned}\mathbf{nnf}(\mathbf{q}) &= \mathbf{q} \\ \mathbf{nnf}(\neg \mathbf{q}) &= \neg \mathbf{q} \\ \mathbf{nnf}(\neg \neg \phi) &= \phi \\ \mathbf{nnf}(\phi \wedge \psi) &= \mathbf{nnf}(\phi) \wedge \mathbf{nnf}(\psi) \\ \mathbf{nnf}(\neg(\phi \wedge \psi)) &= \mathbf{nnf}(\neg \phi) \vee \mathbf{nnf}(\neg \psi) \\ \mathbf{nnf}(\phi \vee \psi) &= \mathbf{nnf}(\phi) \vee \mathbf{nnf}(\psi) \\ \mathbf{nnf}(\neg(\phi \vee \psi)) &= \mathbf{nnf}(\neg \phi) \wedge \mathbf{nnf}(\neg \psi) \\ \mathbf{nnf}(\mathbf{A} \alpha) &= \mathbf{A} \mathbf{nnf}(\alpha) \\ \mathbf{nnf}(\neg \mathbf{A} \alpha) &= \mathbf{E} \mathbf{nnf}(\neg \alpha) \\ \mathbf{nnf}(\mathbf{E} \alpha) &= \mathbf{E} \mathbf{nnf}(\alpha) \\ \mathbf{nnf}(\neg \mathbf{E} \alpha) &= \mathbf{A} \mathbf{nnf}(\neg \alpha)\end{aligned}$$

⁴Also called *positive normal form* by other authors.

$$\begin{aligned}
\text{nnf}(X\phi) &= X\text{nnf}(\phi) \\
\text{nnf}(\neg X\phi) &= X\text{nnf}(\neg\phi) \\
\text{nnf}(\phi U \psi) &= \text{nnf}(\phi) U \text{nnf}(\psi) \\
\text{nnf}(\neg(\phi U \psi)) &= \text{nnf}(\neg\psi) W \text{nnf}(\neg\phi \wedge \neg\psi) \\
\text{nnf}(\phi W \psi) &= \text{nnf}(\phi) W \text{nnf}(\psi) \\
\text{nnf}(\neg(\phi W \psi)) &= \text{nnf}(\neg\psi) U \text{nnf}(\neg\phi \wedge \neg\psi)
\end{aligned}$$

For every formula ϕ in negation normal form, the formulae

$$\begin{aligned}
AF\phi &= A[\text{tt} U \phi] \\
EF &= E[\text{tt} U \phi] \\
AG &= A[\phi W \text{ff}] \\
EG &= E[\phi W \text{ff}]
\end{aligned}$$

are in negation normal form, too. Hence one can use the abbreviations **F** and **G** also for formulae in negation normal form.

Example 8

Some CTL formulae and equivalent formulae in negation normal form are:

$$\begin{aligned}
\neg EF(\text{started} \wedge \neg \text{ready}) &\equiv AG(\neg \text{started} \vee \text{ready}) \\
\neg AF(AG \text{ deadlock}) &\equiv EG(EF \neg \text{deadlock}) \quad \blacklozenge
\end{aligned}$$

In negation normal form one can syntactically define meaningful fragments of CTL, such as ACTL and ECTL.

Definition 15 (ACTL)

The *universal* fragment of CTL is called *ACTL*. It contains the CTL formulae in negation normal form restricted to universal quantification over paths:

$$\begin{aligned}
\phi, \psi &::= \mathbf{q} \mid \neg \mathbf{q} \mid \phi \wedge \psi \mid \phi \vee \psi \mid A[\alpha] \\
\alpha &::= X\phi \mid \phi U \psi \mid \phi W \psi \quad \blacklozenge
\end{aligned}$$

Definition 16 (ECTL)

The *existential* fragment of CTL is called *ECTL*. It contains the CTL formulae in negation normal form restricted to existential quantification over paths:

$$\begin{aligned} \phi, \psi &::= \mathbf{q} \mid \neg \mathbf{q} \mid \phi \wedge \psi \mid \phi \vee \psi \mid \mathbf{E}[\alpha] \\ \alpha &::= \mathbf{X} \phi \mid \phi \mathbf{U} \psi \mid \phi \mathbf{W} \psi \end{aligned} \quad \blacklozenge$$

Some authors define ACTL and ECTL as the sets of CTL formulae ϕ (without explicit syntactical restrictions) whose negation normal form is in the syntactic fragment which we use for the respective definitions.

ACTL contains the *safety-properties* [138, 8] expressible in CTL. Intuitively, safety formulae state properties of the form “Something bad will never happen”:

$$\mathbf{A} \mathbf{G} \neg \text{bad}$$

Another important class of properties, besides safety, are the so called *liveness properties*. Informally they state that the system stays “alive” in the sense that it always remains possible to reach a “good” state.

$$\mathbf{A} \mathbf{G} \mathbf{A} \mathbf{F} \text{good}$$

Safety and liveness are more naturally related to each other than it might seem at first. Safety asserts something about finite prefixes of paths, e.g. no bad state may occur in a finite prefix of any path. Liveness on the other hand states something about behaviours which are independent of finite prefixes of paths, e.g. no matter how much of a path was already examined, it will always eventually visit another good state. This allows for a topological definition of safety and liveness. The topological approach is rather elegant in its independence from concrete syntax, and reveals the natural duality of safety and liveness very clearly. As it is not relevant for the remainder of this thesis, we omit the topological definitions of safety and liveness and refer to [8, 138, 9] and [177].

Historically safety and liveness are concepts of LTL [8, 153, 137] which is, in general, semantically incomparable to CTL [127, 70]. Our interest lies in branching-time logics such as CTL and PCTL, however. Within the logic

CTL*, which contains both LTL and CTL, the corresponding fragment to ACTL is ACTL*. This fragment contains full LTL, ACTL and all safety-properties expressible in CTL*. See Section 2.2.3 below for more details on the incomparability of LTL and CTL, and for CTL*. For more information on safety and liveness in branching time see [138, 102] and in particular [20, Chapters 3 and 7]. For safety and liveness in the modal μ -calculus see, for example, [161].

2.2.3 Modal μ -calculus

At first, one might think that branching-time logics are strictly more expressive than linear-time logics, and that it should be possible to express every LTL property in CTL. Actually, this is not the case; the expressiveness of the two logics is incomparable. Many properties can be expressed in both logics, but there are properties which can be expressed in CTL but not in LTL and vice versa. Here we say a property P can be expressed by two formulae ϕ and ψ , if the formulae are semantically equivalent, i.e. they have the same set of models, namely the set of all models with property P . In this sense a formula ϕ expresses (the intent of) another formula ψ , if ϕ and ψ are semantically equivalent.

Example 9

The following formulae are in LTL but can not be expressed in CTL:

$$F q \Rightarrow F r$$

$$F G q$$

$$F(q \wedge X q)$$

Conversely, the following formulae are in CTL but can not be expressed in LTL:

$$A F A G q$$

$$A F(q \wedge A[X q])$$

The examples are taken from [98] and [20] respectively. \blacklozenge

For a more detailed comparison of LTL and CTL expressiveness see [20, Chapter 6.3].

The logic CTL^* was developed by Allen Emerson and Joseph Halpern to unify LTL and CTL [70]. Unfortunately the model-checking problem, i.e. whether a formula ϕ holds for Kripke structure K , for CTL^* is PSPACE-complete, while there are efficient algorithms for model checking CTL and important fragments of LTL [70, 165, 98, 164]. In particular LTL model-checking is linear in the size of the Kripke structure K and PSPACE-complete in the length⁵ of the formula ϕ [164]; CTL model-checking is linear (i.e. P-complete) in both the size of K and the length of ϕ [164].

The universal fragment $ACTL^*$ of CTL^* contains LTL as a strict subset. CTL^* itself is subsumed by Dexter Kozen's *modal μ -calculus* [122] which is even more expressive and contains least and greatest fixpoint operators [54]. As we are interested in PCTL, which is more similar to CTL than CTL^* , we do not discuss CTL^* further. Instead we move straight on to the modal μ -calculus. The modal μ -calculus has a strong connection to tree automata which are introduced in Section 3.3. This connection provides some motivation for our probabilistic automata in Chapter 6.

Definition 17 (Modal μ -calculus)

The formulae of the modal μ -calculus are defined as follows:

$$\phi, \psi ::= q \mid Z \mid \neg\phi \mid \phi \wedge \psi \mid EX\phi \mid \mu Z.\phi$$

where q is a proposition from a finite set AP and Z a variable. The *least* fixpoint operator μZ binds recursion variables Z . In every fixpoint formula $\mu Z.\phi$ all free occurrence of Z in ϕ are in the scope of an even number of negations. \blacklozenge

The even scope of negations in $\mu Z.\phi$ is necessary for the formula ϕ to be monotone and thus to guarantee the existence of the fixpoint. Indeed,

⁵The exponential complexity actually stems from the number of nested Until operators.

Kozen calls this requirement *syntactic monotonicity* [122].

We include the standard derived operators **tt**, **ff** and \vee in the above syntax. Further, we write $\phi[Z/Y]$ for the syntactic substitution with Y for all free occurrences of Z in ϕ . We also include the *greatest* fixpoint in the μ -calculus syntax. It is defined as

$$\nu Z.\phi Z \equiv \neg\mu Z.\neg\phi[Z/\neg Z]$$

where $\phi[Z/\neg Z]$ is the formula ϕ except that all occurrences of variable Z are replaced by $\neg Z$.

Including $\nu Z.\phi Z$, $\phi \wedge \psi$ and $\mathbf{AX}\phi$ explicitly in the syntax and enlarging \mathbf{AP} by $\bar{\mathbf{q}} = \neg\mathbf{q}$ for every $\mathbf{q} \in \mathbf{AP}$ allows for a negation normal form of the modal μ -calculus where all negations are pushed inwards and regarded as propositional. Formulae of this form, i.e. without negated subformulae, are called *positive* in the original article by Kozen [122].

Example 10

The Kripke K structure of Figure 2.1 on page 33 satisfies, for example, the μ -calculus formulae

$$\begin{aligned}\phi &= \nu Z.(\mathbf{s} \wedge \mathbf{AX} Z) \\ \psi &= \mu Z.(\mathbf{r} \vee (\mathbf{q} \wedge \mathbf{EX} Z))\end{aligned}$$

which correspond to $\mathbf{AG}\mathbf{s}$ and $\mathbf{E}[\mathbf{q} \mathbf{U} \mathbf{r}]$ respectively in CTL. ◆

This concludes our discussion of the modal μ -calculus in this chapter. The modal μ -calculus is revisited briefly in Chapter 3 when we discuss its connection to (alternating) tree automata.

2.2.4 LTL and CTL semantics for partial Kripke structures

Partial Kripke structures allow for two interpretation of $L(s, \mathbf{q}) = ?$, where $L(s, \mathbf{q}) = ?$ is either under-approximated as **ff** or over-approximated as **tt**. In the terminology of Glenn Bruns and Patrice Godefroid [31] we call these two modes of interpretation *optimistic* and *pessimistic* respectively. For example, an optimistic interpretation of $L(s, \mathbf{q}) = ?$ justifies the CTL assertion $s \models \mathbf{q}$;

while pessimistically it justifies $s \not\models q$. This gives rise to a 3-valued semantics with truth values in $\{\text{tt}, \text{ff}, \text{?}\}$.

We omit a formal definition of 3-valued semantics for LTL and CTL here and refer to Bruns's and Godefroid's work on *generalised model checking* [31, 32] for details.

Partial Kripke structures and 3-valued semantics have a very practical application in abstraction. We omit formal definitions and only briefly sketch the idea. Intuitively, states labelled with $L(s, \mathbf{q}) = \text{?}$ can abstract infinitely many concrete states such that a formula is true respectively false on the concrete model whenever it is so in both the pessimistic and optimistic interpretation on the abstract model. That is, whenever the pessimistic and optimistic interpretation of a formula yields the same truth-value on a 3-valued abstraction, then the formula has this truth-value also on every refinement of the abstraction. The abstraction is inconclusive, i.e. too coarse, only if pessimistic and optimistic pessimistic yields different answers.

This semantics allows to verify a variety of useful formulae on partial Kripke structures with 3-valued semantics. A formal description of this approach and some instructive examples can be found, e.g., in [31, Section 5].

Example 11

Partial Kripke structure K of Figure 2.3 on page 35 satisfies $A[X \mathbf{s}]$ optimistically but not pessimistically as s_1 is labelled with $\mathbf{s}?$. ◆

We define a 3-valued semantics for PCTL over 3-valued Markov chains in Section 2.4. This semantics is then used for our first partial completeness result in Section 4. The partial completeness result is achieved by abstraction via 3-valued unfoldings of Markov chains which is similar to the abstraction via partial Kripke structures mentioned above.

2.3 Simulation and bisimulation

We now define some important relations between states of a Kripke structure respectively between whole Kripke structures, and discuss their connection to the logic CTL. Our presentation follows in part [20, Chapter 7] and we

refer to [20] for a much more detailed discussion, for further examples and for many of the proofs which we omit in this background chapter.

First we introduce *simulation*, a relation between the observable behaviour of systems, whose definition goes back to Robin Milner [145].

Definition 18 (Simulation)

Let $J = (S, R, L)$ and $K = (T, R, L)$ be pointed Kripke structures with initial state s^{in} and t^{in} respectively.⁶ A relation $Q \subseteq S \times T$ is called *simulation* iff for all $s \in S$ and $t \in T$ we have whenever $(s, t) \in Q$:

- $L(s) = L(t)$; and
- for all $s' \in S$ with $s \longrightarrow s'$ there is some $t' \in T$ with $t \longrightarrow t'$ and $(s', t') \in Q$.

We say “ t simulates s ” (or “ s is simulated by t ”), denoted by $t \preceq s$, iff there is a simulation Q with $(s, t) \in Q$.

We say K simulates J , denoted $K \preceq J$, iff the initial state of K simulates the initial state of J . ◆

Intuitively, t simulates s if it matches the “local information” of s , i.e. t has the same labelling as s , and if t “can mimic all stepwise behaviour” of s [20, Chapter 7.4].

Unfortunately there is no universal agreement in the literature on the direction of the relational sign \preceq ; i.e. on whether to write $K \preceq J$ or $J \preceq K$ for “ K simulates J ”. There are arguments for both variants:

- Seen as abstract specification K and more concrete (or refined) specification J , then K (potentially) allows more implementations than J , e.g. describes a larger set of models.
- Seen as abstract model K and concrete model J , then K can be understood as (potentially) smaller than J , e.g. having a smaller state space.

⁶To simplify presentation we use R and L to denote the transition relation and labelling function respectively of both Kripke structures. Their domain, e.g. $R \subseteq S \times S$ or $R \subseteq T \times T$, is clear from the context.

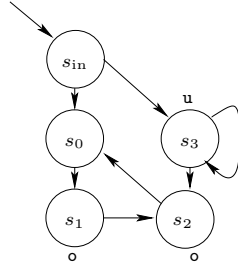


Figure 2.5: A Kripke structure C .

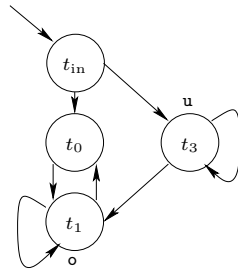


Figure 2.6: A Kripke structure A .

We favour the latter intuition, and therefore write $K \preceq J$ whenever K simulates J . In this sense of “abstract specification K *simulates* concrete implementation J ”, the definition yields that K can mimic every transition of J , i.e. every transition of J is allowed by the specification K , but K can feature additional transitions which are not implemented in J . In the same sense one could read “simulate” as “abstracts”, and as the opposite of “implements”. The \preceq sign then indicates “more abstract” and “potentially smaller”.

The following example is adapted from David Schmidt’s tutorial-style article on abstraction and refinement [161].

Example 12

Let C and A be the Kripke structures illustrated in Figure 2.5 and Figure 2.6 respectively. Then A simulates C .

The simulation relation is $R = \{(s_{\text{in}}, t_{\text{in}}), (s_i, t_i), (s_2, t_1) \mid i = 0, \dots, 3\}$. ♦

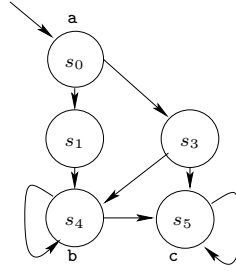


Figure 2.7: A Kripke structure M .

Simulation can be seen as relation between specifications and implementations. We now introduce two notions for models which satisfy the same specification, or which in some sense exhibit the same observable behaviour. The first notion is mutual simulation, called *similarity*. The second notion is *bisimulation*, which is a co-inductive variant of mutual simulation.

Definition 19 (Similarity)

We say Kripke structures K and J are *similar* (to each other) iff $K \preceq J$ and $J \preceq K$. ◆

The following example is adapted from Example 7.63 in the book by Christel Baier and Joost-Pieter Katoen [20].

Example 13

Let M and N be the Kripke structures illustrated in Figure 2.7 and Figure 2.8 respectively. Then M and N are similar to each other, i.e. $M \preceq N$ and $N \preceq M$.

The simulation relations are $R = \{(s_i, t_i), (s_1, t_3) \mid i = 1, \dots, 5\}$ and $Q = \{(t_i, t_i) \mid i = 0, \dots, 5\}$ respectively. ◆

Similarity is an equivalence relation. Another important equivalence relation between Kripke structures is *bisimulation* which goes back to David Park [144].

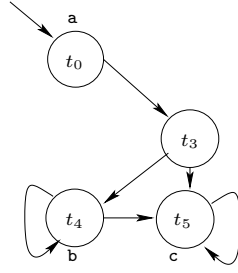


Figure 2.8: A Kripke structure N .

Definition 20 (Bisimulation)

Let $J = (S, R, L)$ and $K = (T, R, L)$ be pointed Kripke structures with initial state s^{in} and t^{in} respectively. A *bisimulation* is a relation $Q \subseteq S \times T$ such that for all $s \in S$ and $t \in T$ with $(s, t) \in Q$:

- (i) $L(s) = L(t)$;
- (ii) whenever $s \rightarrow s'$, there is $t' \in T$ such that $t \rightarrow t'$ and $(s', t') \in Q$;
- (iii) whenever $t \rightarrow t'$, there is $s' \in S$ such that $s \rightarrow s'$ and $(s', t') \in Q$.

Two states s and t are called *bisimilar*, denoted $s \sim t$ iff there is a bisimulation Q such that $(s, t) \in Q$.

Kripke structures J and K are called *bisimilar*, denoted $J \sim K$, iff the initial state of J is bisimilar to the initial state of K . ◆

Bisimulation is well-defined, in particular there exists a greatest relation with these properties.

Example 14

Let K and L be the Kripke structures illustrated in Figure 2.1 (page 33) and Figure 2.9 respectively. Then K and L are bisimilar.

The bisimulation relation is $R = \{(s_i, t_i), (s_1, t_3) \mid i = 0, \dots, 2\}$. ◆

Bisimilarity clearly implies similarity, but interestingly the converse is not true in general. The following example shows two Kripke structures M and N which simulate each other, i.e. $M \preceq N$ and $N \preceq M$, but are not bisimilar.

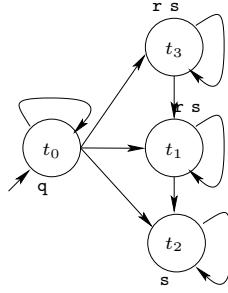


Figure 2.9: A Kripke structure L .

Example 15

The Kripke structures M and N of Figure 2.7 and Figure 2.8 respectively are similar but not bisimilar.

The simulation relations are as in Example 13 above, but there is no bisimulation relation between M and N . ◆

This phenomenon is related to the notion of *label-determinism*, which basically asserts that successor states must be (locally) unique with regard to their labelling.⁷

Definition 21

We call a Kripke structure K *label-deterministic*⁸ if for $s \in S$ whenever

$$s \longrightarrow s' \text{ and } s \longrightarrow s'' \text{ and } L(s') = L(s'')$$

then $s' = s''$. ◆

For label-deterministic Kripke structures bisimilarity and similarity coincide.⁹

⁷The analogous notion for labelled transition systems is called *action-determinism*.
⁸Determinism here means that all successor states s' of a state s are uniquely determined by their labelling. Two successor states $s' \neq s''$ of s may not have the same labelling.
⁹In the label-deterministic case bisimilarity and similarity also coincide with *trace-equivalence* which we do not introduce formally in this thesis.

Lemma 1

Let K and J be label-deterministic¹⁰ Kripke structures, then

$$(K \preceq J \text{ and } J \preceq K) \text{ if and only if } K \sim J . \quad \bullet$$

Simulation is of interest for us as it can be used as an abstraction relation which preserves certain properties. This preservation of properties is formalised in the following lemma.

Lemma 2

Let J and K be Kripke structures. If K simulates J , then J satisfies every ACTL formulae which is satisfied by K . Formally, if $K \preceq J$, then for every ACTL formula ϕ :

$$\text{whenever } K \models \phi \text{ then } J \models \phi . \quad (2.3) \quad \bullet$$

This lemma does in fact also hold for ACTL* and thus in particular for LTL.

We call property (2.3) of an abstraction relation (which is the simulation relation in this case) *soundness*. Informally we say, simulation is *sound* for ACTL. As discussed in Section 1.6, this is not a very precise statement and we should rather say something like “simulation between Kripke structures is sound for the verification of ACTL formulae”. Soundness is actually a property of a relation between model-checking frameworks – i.e. the triple of models; relation between models; and logics – rather than just of the relation between the models.

Example 16

Let A and C be the Kripke structures depicted in Figure 2.10 and Figure 2.11 respectively. Let ϕ be the ACTL formula $\phi := A[\mathbf{qW}(\mathbf{r} \vee \mathbf{s})]$. As A simulates C , i.e. $A \preceq C$, and $A \models \phi$ we know $C \models \phi$ by the soundness of simulation for ACTL. \blacklozenge

¹⁰The definition of label-determinism requires that whenever two successor states s' and s'' of s agree on all propositions, s' and s'' must be the same state. In fact, Lemma 1 already holds if we weaken this condition of equality to bisimilarity, i.e. $s' \sim s''$ rather than $s' = s''$.

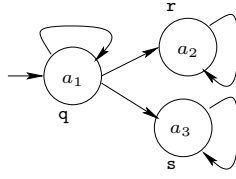


Figure 2.10: A Kripke structure A .

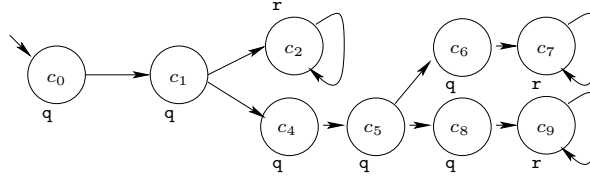


Figure 2.11: A Kripke structure C .

Simulation is not sound for the full logic CTL.

Example 17

Let A and C be the Kripke structures of Figure 2.10 and Figure 2.11 respectively. For $\psi = E[q \cup s]$ we have $A \models \psi$ and $C \not\models \psi$ although $A \preceq C$.

The simulation relation is $R = \{(a_1, c_i) \mid L(c_i) = q\} \cup \{(a_2, c_i) \mid L(c_i) = r\}$. ◆

Simulation is also sound for refutation of ECTL, which follows directly from the soundness for ACTL via negation.

Lemma 3

Let J and K be Kripke structures. If K simulates J , then K satisfies every ECTL formulae which is satisfied by J . Formally, if $K \preceq J$, then for every ECTL formula ϕ

$$\text{whenever } J \models \phi \text{ then } K \models \phi . \tag{2.4}$$



Analogous notions of simulation, similarity and bisimulation can be defined for partial Kripke structures. Their definitions agree with the definitions above on the structural part, e.g. on the transitions, but refer to the

information order for the conditions on the state labelling. This introduces some technical issues which are not relevant for the work in this thesis and hence are not discussed here. With notions such as *property-reflecting* and *property-preserving* simulations one gets similar soundness results for partial Kripke structures as for 2-valued Kripke structures [161]. The formal definitions of these relations and the corresponding technical results are not relevant for the main part of this thesis, and we refer to [161] for further details.

We mention the concept of *nearly bisimilar* (in the terminology of [82]) as it is the relation for partial Kripke structure which corresponds closest to the probabilistic simulation for 3-valued Markov chains which we use in the main part of this thesis.

Definition 22 (Near bisimulation)

Let $J = (S, R, L)$ and $K = (T, R, L)$ be pointed, partial Kripke structures. A relation $Q \subseteq S \times T$ is called *near bisimulation* iff for all $s \in S$ and $t \in T$ we have whenever $(s, t) \in Q$:

- for all $\mathbf{q} \in \mathbf{AP}$ we have: $L(t, \mathbf{q}) \leq L(s, \mathbf{q})$;
- for all $s' \in S$ with $s \longrightarrow s'$ there exists $t' \in T$ with $t \longrightarrow t'$ and $(s', t') \in Q$;
and
- for all $t' \in T$ with $t \longrightarrow t'$ there exists $s' \in S$ with $s \longrightarrow s'$ and $(s', t') \in Q$.

We say “ t is *nearly bisimilar* to s ” iff there is a near bisimulation Q with $(s, t) \in Q$.¹¹

Partial Kripke structures K is nearly bisimilar to J if the initial state of K is nearly bisimilar to the initial state of J . ◆

This relation establishes that two states s and t are “bisimilar except that the atomic propositions in state t may be less defined than in state s ” [82]. Near bisimulation coincides with bisimulation for 2-valued Kripke structures.

A more thorough overview on simulation, bisimulation and their connections to various logics is given in [20, Chapter 7].

¹¹One needs to be careful with the order of s and t here as near bisimulation is not a reflexive relation.

2.4 Markov chains and Markov decision processes

We now introduce labelled, discrete-time Markov chains as the class of models which we use throughout most of this thesis. In analogy to Kripke structures, Markov chains can be seen as graph structures where states are labelled with propositions and the outgoing transitions of each state are decorated with the probabilities of a discrete probability distribution.

There is broad literature on Markov chains. We mostly follow the terminology used in [85] and [149].

Definition 23 (Discrete-time Markov chain)

A *labelled discrete-time Markov chain* M is a tuple (S, \mathbf{P}, L) where

1. S is a countable, non-empty set of *states*;
2. \mathbf{P} is a *stochastic matrix* whose entries define *transition probabilities* $P: S \times S \rightarrow [0, 1]$, such that the countable sum

$$\sum_{s' \in S} P(s, s')$$

exists and equals 1 for all $s \in S$; and

3. L is a total labelling function $L: S \times \mathbf{AP} \rightarrow \{\text{tt}, \text{ff}\}$, where \mathbf{AP} is a finite set of *atomic propositions*. \blacklozenge

The transition probabilities $\mathbf{P}(s, \cdot)$ form a probability *distribution* over S . The set of all distributions over S is denoted by $\text{Dist}(S)$.

As intuition one can think of Markov chains M as serial Kripke structures where the transitions are labelled with probabilities, such that for each state s the probabilities on the outgoing transitions add up to 1. Dually, one can also see each Kripke structure K as a Markov chain by assigning a (e.g. uniform) distribution to the outgoing transitions of each state. Both conversions are just meant as intuition, they do not in general preserve all information and meaning.

In a Markov chain every state needs to have at least one outgoing transition – to itself or to another state – so that the matrix \mathbf{P} actually is a

stochastic matrix, i.e. the entries in each row add up to 1. This is different from (non-serial) Kripke structures which may contain states with no outgoing transitions.

Assumption 3 (Labelled Markov chains)

In this thesis all Markov chains are labelled Markov chains and we often omit the attribute “labelled”. ◆

We now define some useful concepts, notations and naming conventions for Markov chains:

Definition 24

Let $M = (S, \mathbf{P}, L)$ be a Markov chain.

1. M is called *finite-state* iff the set S is finite.
2. M is called *finite-branching* iff for all $s \in S$ the set of successors $\text{Succ}(s) := \{s' \in S \mid \mathbf{P}(s, s') > 0\}$ is finite. A finite-state Markov chain M is necessarily also finite-branching.
3. M is called *pointed* if it contains a designated *initial state* $s^{\text{in}} \in S$. We write (M, s^{in}) to denote the initial state s^{in} of M .
4. A state s is called *absorbing* if it has a self-loop of probability 1, $P(s, s) = 1$, and thus no other outgoing transition. ◆

Assumption 4 (Pointed Markov chains)

In this thesis we assume that all Markov chains M are pointed, i.e. have an initial state s^{in} . ◆

A Markov chain $M = (S, \mathbf{P}, L)$ determines a directed graph (S, E) where the vertices are states S and edges are defined by $(s, s') \in E$ iff $P(s, s') > 0$. Edges (s, s') are decorated with $P(s, s')$. Thus, we sometimes use graph terminology for features of Markov chains, e.g. *cycles* and *strongly connected components*. In this graph sense we talk about a *transition* between two states s and s' of a Markov chain if the probability $P(s, s')$ is strictly positive.

A path π from state s in M is an infinite sequence of states $s_0s_1\dots$ with $s_0 = s$ and $P(s_i, s_{i+1}) > 0$ for all $i \geq 0$. Note that each state s has at least one outgoing transition and therefore every finite sequence of states can be extended to a path in M . We write $\text{Path}(s)$ for the set of all paths from s .

For $Y \subseteq S$, we write $P(s, Y)$ as a shorthand for the (possibly infinite but well defined) sum $\sum_{s' \in Y} P(s, s')$.

The labelling function $L(s, \mathbf{q})$ determines the truth value of atomic propositions $\mathbf{q} \in \mathbf{AP}$ at state s . Thinking of a Markov chain as a graph structure, L assigns (the truth value of) atomic propositions to the vertices s in the graph of the Markov chain M . Here we use the same conventions as for Kripke structures: the set of propositions that hold at s is denoted $L(s) := \{\mathbf{q} \in \mathbf{AP} \mid L(s, \mathbf{q}) = \text{tt}\}$; in figures the label \mathbf{q} indicates $L(s, \mathbf{q}) = \text{tt}$, and the absence of such label means $L(s, \mathbf{q}) = \text{ff}$. As with partial Kripke structures we will generalise Markov chains to 3-valued labelling function L in Definition 26 below.

Example 18

The Markov chain M from Figure 2.12 is pointed, finite-branching and finite-state. It has two absorbing states s_1 and s_2 . ◆

Example 19

Figure 2.12 illustrates a labelled discrete-time Markov chain M with initial state s_0 designated by an incoming arrow. State names are written in vertices, labels indicating the value of $L(s, \mathbf{q})$ are written next to states s . Transition probabilities $P(s, s')$ are written on the edges. Edges with probability 0 are omitted. A probability label 1 on a transition $s \rightarrow s'$ may also be omitted as it is implicit by $s \rightarrow s'$ being the only transition from s . ◆

Time is not explicitly mentioned in a discrete-*time* Markov chain. As we use discrete time (as opposed to continuous time) no parameter t is needed. Time is handled implicitly: one transition is taken at each discrete time step. Thus, the probability for a discrete-time Markov chain to be in a certain state after k time steps is computed by the k -th power of the stochastic matrix \mathbf{P} [149].

For a Markov chain M we are interested in the probability measure of paths which satisfy certain properties; for example in Figure 2.12 the set of

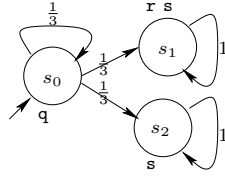


Figure 2.12: A labelled discrete-time Markov chain M .

paths which satisfy $\mathbf{q} \cup \mathbf{r}$. Hence we need to derive a probability measure on path sets from the transition probabilities $P(s, s')$. The standard way to do this is via *cylinders* [115].

Definition 25 (Basic cylinder)

For a finite prefix $\rho = s_0 \dots s_n$ of a path we define the *basic cylinder*:

$$\Delta(\rho) = \{\pi \text{ path} \mid \rho \text{ a prefix of } \pi\} \quad \blacklozenge$$

These cylinders generate a sigma-algebra which gives rise to a unique probability measure \mathbf{Prob} via the product of transition probabilities in the prefix:

$$\mathbf{Prob}(\Delta(s_0 \dots s_n)) = \prod_{i=0}^{n-1} P(s_i, s_{i+1})$$

For more details on this construction and proofs see, for example, [115].

Analogously to partial Kripke structures, we now define Markov chains with 3-valued labelling by replacing the labelling function in Definition 23.

Definition 26 (3-valued Markov chains)

A *3-valued discrete-time Markov chain* M is a tuple (S, \mathbf{P}, L) , where

1. S is a countable, non-empty set of *states*;
2. \mathbf{P} is a *stochastic matrix* such that the countable sum

$$\sum_{s' \in S} P(s, s')$$

exists and equals 1 for all $s \in S$; and

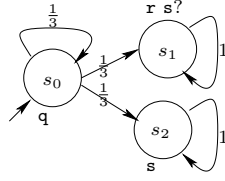


Figure 2.13: A 3-valued discrete-time Markov chain M .

3. L is a total function $L: S \times \text{AP} \longrightarrow \{\text{tt}, \text{?}, \text{ff}\}$ for a finite set AP of atomic propositions. \blacklozenge

In graphical representations of 3-valued Markov chains M the label \mathbf{q} indicates $L(s, \mathbf{q}) = \text{tt}$; label $\mathbf{q}?$ marks $L(s, \mathbf{q}) = \text{?}$; and the absence of any of the other two labels means $L(s, \mathbf{q}) = \text{ff}$. To stress that a Markov chains M is 2-valued we sometimes call it *concrete*.

Example 20

Figure 2.13 illustrates a 3-valued discrete-time Markov chain M . \blacklozenge

Another possible generalisation of (2-valued) Markov chains is the introduction of non-determinism. Instead of having one unique probability distribution at each state one allows non-deterministic choice from a set of distributions. Such models are called *Markov Decision Processes (MDP)*. Kripke structures are in general non-deterministic, Markov chains can be seen as deterministic although probabilistic, and Markov decision processes are non-deterministic and probabilistic. These are crucial distinctions for example with regard to *fairness, optimistic and pessimistic semantics*, and with regard to our completeness question.

Definition 27 (Markov decision process)

A *labelled Markov decision process (MDP)* M is a tuple (S, D, L) where

1. S is a countable, non-empty set of *states*,
2. $D: S \longrightarrow 2^{\text{Dist}(S)}$ is a mapping from states to sets of distributions over S ; and

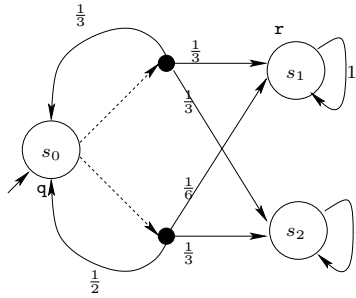


Figure 2.14: A Markov decision process M .

3. L is a *labelling function* $L: S \times \text{AP} \rightarrow \{\text{tt}, \text{ff}\}$, where AP is a finite set of *atomic propositions*. \blacklozenge

The function D assigns a set $D(s)$ of distributions to each state s . The choice between distributions from these sets is non-deterministic. If $D(s)$ is a singleton for every s , the MDP M is equivalent to a standard Markov chain.

Example 21

Figure 2.14 shows a Markov decision process M . We use the same notations as for Markov chains; additionally non-deterministic choice is depicted by dashed arrows to small filled circles where each circle represents one possible choice of distribution. \blacklozenge

2.5 Probabilistic branching-time logic: PCTL

The probabilistic branching-time logic PCTL (*Probabilistic Computation Tree Logic*) was introduced by Hans Hansson and Bengt Jonsson in 1994 [88]. It was subsequently used, sometimes in slightly modified variants, for probabilistic model checking [14, 26] and is now widely used in probabilistic model-checking tools, for example in PRISM [1] and Verus [17, 35]. Hansson and Jonsson define PCTL in [88] without the Next modality X . Leslie Lamport even argues in [128] that the Next modality should be excluded from any temporal modal logic. We use a variant of PCTL which is based

on [88] but includes X . We also include the bounded versions of the Until modalities which are present in [88] but omitted in other variants of PCTL, for example in [14, 26, 17]. A restriction to the finitely bounded Until operators is fairly common in practical applications – for PCTL as well as for CTL – as it yields an upper bound on the search depth for model-checking algorithms. Model checking with limited depth as a means of approximating “unlimited” model checking is known as *bounded model checking* [27].

Definition 28 (PCTL syntax)

The formulae of PCTL are given by the following state formulae ϕ :

$$\begin{aligned} \phi, \psi &::= \mathbf{q} \mid \neg\phi \mid \phi \wedge \psi \mid [\alpha]_{\bowtie p} && \text{(state formulae)} \\ \alpha &::= X\phi \mid \phi U^{\leq k} \psi \mid \phi W^{\leq k} \psi && \text{(path formulae)} \end{aligned}$$

where \mathbf{q} is a proposition from a finite set \mathbf{AP} , $p \in [0, 1]$, $\bowtie \in \{<, \leq, \geq, >\}$ and $k \in \mathbb{N} \cup \{\infty\}$. ◆

Path formulae α are essentially “LTL modalities” *Next* (X); (*strong*) *Until* (U); and *weak Until* (W). They are interpreted as predicates over paths of M . PCTL formulae wrap path formulae with probability thresholds (turning predicates on paths into predicates on states), and may add a propositional logic layer on top of that, which may then be used to build up new path formulae.

We define constants Truth and False: let \mathbf{ff} be an abbreviation for any $[\alpha]_{>1}$, and let \mathbf{tt} denote any $[\alpha]_{\geq 0}$. We write $\phi U \psi$ as a shorthand for $\phi U^{\leq \infty} \psi$, and write $\phi W \psi$ as shorthand for $\phi W^{\leq \infty} \psi$. Further, we use the usual abbreviations for derived operators:

$$\begin{aligned} \phi \vee \psi &\equiv \neg(\neg\phi \wedge \neg\psi) \\ \phi \Rightarrow \psi &\equiv \neg\phi \vee \psi \\ \mathbf{F} \phi &\equiv \mathbf{tt} U \phi \\ \mathbf{G} \phi &\equiv \phi W \mathbf{ff} \end{aligned}$$

For labelled a Markov chain $M = (S, \mathbf{P}, L)$, the denotational semantics of a PCTL formula ϕ is a subset $\llbracket \phi \rrbracket_M$ of S . We write $\llbracket \phi \rrbracket$ if M is clear from the context and define $\llbracket \phi \rrbracket$ by structural induction, as follows.

Definition 29 (PCTL semantics)

Let $M = (S, \mathbf{P}, L)$ be a Markov chain. The semantics of PCTL is defined inductively as:

$$\begin{aligned} \llbracket \mathbf{q} \rrbracket &= \{s \in S \mid L(s, \mathbf{q}) = \mathbf{tt}\} \\ \llbracket \phi \wedge \psi \rrbracket &= \llbracket \phi \rrbracket \cap \llbracket \psi \rrbracket \\ \llbracket \neg \phi \rrbracket &= S \setminus \llbracket \phi \rrbracket \\ \llbracket [\alpha]_{\bowtie p} \rrbracket &= \{s \in S \mid \mathbf{Prob}_M(s, \alpha) \bowtie p\} \end{aligned}$$

where $\mathbf{Prob}_M(s, \alpha)$ is the probability of the measurable set $\mathbf{Path}(s, \alpha)$ of paths in M that begin in s and satisfy the path formula α where the semantics for path formulae is as follows:

- $\pi \models \mathbf{X} \phi$ iff $\pi[1] \in \llbracket \phi \rrbracket_M$;
- $\pi \models \phi \mathbf{U}^{\leq k} \psi$ iff there is a $l \in \mathbb{N}$ such that $0 \leq l \leq k$, $\pi[l] \in \llbracket \psi \rrbracket_M$ and for all $0 \leq j < l$ we have $\pi[j] \in \llbracket \phi \rrbracket_M$;
- $\pi \models \phi \mathbf{W}^{\leq k} \psi$ iff for all $l \in \mathbb{N}$ such that $0 \leq l \leq k$ we have either $\pi[l] \in \llbracket \phi \rrbracket_M$ or there is $0 \leq j \leq l$ with $\pi[j] \in \llbracket \psi \rrbracket_M$.

We say Markov chain (M, s_0) satisfies ϕ , denoted $M \models \phi$, iff $s_0 \in \llbracket \phi \rrbracket_M$. \blacklozenge

Occasionally we need to assert that a PCTL (sub-)formula ϕ holds at a state s of a Markov chain M . We denote this $s \models \phi$, and we write $s \models_M \phi$ to clarify which Markov chain M the state s belongs to.

We say that two PCTL formulae ϕ and ψ are *semantically equivalent* iff for all labelled Markov chains M we have $\llbracket \phi \rrbracket_M = \llbracket \psi \rrbracket_M$.

Note that the semantics of PCTL state and path formulae is mutually recursive, reflecting the mutual recursion of their syntax. Until formulae $\phi \mathbf{U}^{\leq k} \psi$ are *strong* Untils since paths that satisfy such a formula have to maintain temporary invariant ϕ until they reach a state satisfying ψ , and such a state has to be reached within finitely many transitions (and within k transitions if $k \neq \infty$). Weak Until formulae $\phi \mathbf{W}^{\leq k} \psi$ are *weak* Untils since reaching a state satisfying ψ is optional if ϕ is an invariant on the sequence $s_0 s_1 \dots s_k$, which is understood to be π when $k = \infty$. The value $k = \infty$ is being used to express unbounded Untils, whereas $k \in \mathbb{N}$ expresses a proper step bound.

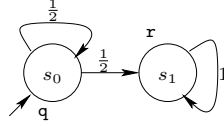


Figure 2.15: A Markov chain M which satisfies the PCTL formula $[\mathbf{q} \mathbf{U} \mathbf{r}]_{\geq 1}$, but – seen as Kripke structure without fairness constraints – does not satisfy the LTL formula $\mathbf{q} \mathbf{U} \mathbf{r}$.

The Until operators of PCTL allow to express useful properties such as reachability with probability 1. More examples of practical specifications expressible in PCTL are given, e.g., in [88]. In general such probabilistic properties are very distinct from their non-probabilistic analogons, as illustrated by the following example.

Example 22

The Markov chain M in Figure 2.15 satisfies the PCTL formula $[\mathbf{q} \mathbf{U} \mathbf{r}]_{\geq 1}$. The underlying Kripke structure – M without the probabilities – neither satisfies the LTL formula $\mathbf{q} \mathbf{U} \mathbf{r}$, nor the CTL formula $\mathbf{A}[\mathbf{q} \mathbf{U} \mathbf{r}]$. \blacklozenge

Example 23

Let M be a discrete-time Markov chain as illustrated in Figure 2.16. The state s_0 of M satisfies the PCTL formula $[\mathbf{q} \mathbf{U} \mathbf{r}]_{\geq 1/2}$ since the probability of all paths which start at s_0 and satisfy the path formula $\alpha = \mathbf{q} \mathbf{U} \mathbf{r}$ is

$$\text{Prob}(s_0, \mathbf{q} \mathbf{U} \mathbf{r}) = \sum_{j=1}^{\infty} \left(\frac{1}{3}\right)^j = \frac{1}{2}. \quad \blacklozenge$$

Next we define a PCTL semantics on 3-valued Markov chains. It is very similar to the semantics on 2-valued Markov chains but based on an optimistic and a pessimistic interpretation of propositions [81, 100]. Optimistically, we interpret a proposition as true if it is not false, i.e.

$$\begin{aligned} \llbracket \mathbf{q} \rrbracket_M^{\circ} &= \{s \in S \mid L(s, \mathbf{q}) \neq \text{ff}\} \\ &= \{s \in S \mid L(s, \mathbf{q}) = \text{tt} \text{ or } L(s, \mathbf{q}) = \text{?}\}; \end{aligned}$$

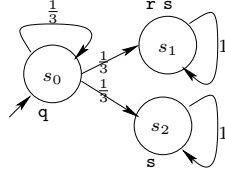


Figure 2.16: A labelled Markov chain M which satisfies $[q \text{ U } r]_{\geq 1/2}$ because $\text{Prob}_M(s_0, q \text{ U } r) = 1/2$.

pessimistically, q is true only if the labelling says so, i.e.

$$\llbracket \mathbf{q} \rrbracket_M^{\mathbf{p}} = \{s \in S \mid L(s, \mathbf{q}) = \mathbf{tt}\} .$$

Definition 30 (PCTL semantics)

Let $m \in \{\mathbf{o}, \mathbf{p}\}$ be two modes of interpretation, $\neg \mathbf{o} = \mathbf{p}$, and $\neg \mathbf{p} = \mathbf{o}$. For ϕ in PCTL, we define $\llbracket \phi \rrbracket_M^m$:

$$\begin{aligned} \llbracket \mathbf{q} \rrbracket_M^{\mathbf{o}} &= \{s \in S \mid L(s, \mathbf{q}) \neq \mathbf{ff}\} \\ \llbracket \mathbf{q} \rrbracket_M^{\mathbf{p}} &= \{s \in S \mid L(s, \mathbf{q}) = \mathbf{tt}\} \\ \llbracket \phi \wedge \psi \rrbracket_M^m &= \llbracket \phi \rrbracket_M^m \cap \llbracket \psi \rrbracket_M^m \\ \llbracket \phi \vee \psi \rrbracket_M^m &= \llbracket \phi \rrbracket_M^m \cup \llbracket \psi \rrbracket_M^m \\ \llbracket \neg \phi \rrbracket_M^m &= S \setminus \llbracket \phi \rrbracket_M^m \\ \llbracket [\alpha]_{\bowtie p} \rrbracket_M^m &= \{s \in S \mid \text{Prob}_M^m(s, \alpha) \bowtie p\} \end{aligned}$$

where $\text{Prob}_M^m(s, \alpha)$ is the probability of the measurable set $\text{Path}(s, \alpha)^m$ of paths $\pi = s_0 s_1 \dots$ in M that begin in $s_0 = s$ and for which $\pi \models^m \alpha$ in mode m :

- $\pi \models^m \mathbf{X} \phi$ iff $\pi[1] \in \llbracket \phi \rrbracket_M^m$
- $\pi \models^m \phi \text{ U}^{\leq k} \psi$ iff there is a $l \in \mathbb{N}$ such that $l \leq k$, $\pi[l] \in \llbracket \psi \rrbracket_M^m$ and for all $0 \leq j < l$ we have $\pi[j] \in \llbracket \phi \rrbracket_M^m$
- $\pi \models^m \phi \text{ W}^{\leq k} \psi$ iff for all $l \in \mathbb{N}$ such that $0 \leq l \leq k$ we have either $\pi[l] \in \llbracket \phi \rrbracket_M^m$ or there is $0 \leq j \leq l$ with $\pi[j] \in \llbracket \psi \rrbracket_M^m$ ◆

Example 24

Let M be the 3-valued discrete-time Markov chain illustrated in Figure 2.13. The above PCTL semantics for 3-valued Markov chains yields

$$\begin{aligned} s_0 &\models^{\circ} [\mathbf{q} \mathbf{U} \mathbf{s}]_{\geq 3/4} \\ s_0 &\not\models^{\mathbf{p}} [\mathbf{q} \mathbf{U} \mathbf{s}]_{\geq 3/4} \\ s_0 &\models^{\circ} [\mathbf{q} \mathbf{U} \mathbf{s}]_{\geq 1/2} \\ s_0 &\models^{\mathbf{p}} [\mathbf{q} \mathbf{U} \mathbf{s}]_{\geq 1/2} \end{aligned}$$

according to the optimistic respectively pessimistic interpretation of $\mathbf{s}?$ on s_1 . For formulae ϕ which do not contain \mathbf{s} , such as $[\mathbf{q} \mathbf{U} \mathbf{r}]_{\geq 1/2}$, the two semantics \models° and $\models^{\mathbf{p}}$ coincide with the standard PCTL semantics \models . \blacklozenge

Recall the duality between (strong) Until and weak Until in LTL as in equation (2.2) on page 40. While in LTL the duality

$$\phi \mathbf{W} \psi \equiv (\phi \mathbf{U} \psi) \vee \neg(\mathbf{tt} \mathbf{U} \neg\phi) \quad (2.5)$$

holds, it does not hold for PCTL. As in CTL, PCTL path formulae are not closed under logical disjunction \vee which appears in the right hand side of the equation. Nevertheless very similar dualities can be established in PCTL. Hansson and Jonsson state in [88] the following dualities between \mathbf{U} and \mathbf{W} :

$$[\phi_1 \mathbf{W}^{\leq k} \phi_2]_{\geq p} = \neg \left([\neg\phi_2 \mathbf{U}^{\leq k} \neg(\phi_1 \vee \phi_2)]_{>1-p} \right) \quad (2.6)$$

$$[\phi_1 \mathbf{W}^{\leq k} \phi_2]_{> p} = \neg \left([\neg\phi_2 \mathbf{U}^{\leq k} \neg(\phi_1 \vee \phi_2)]_{\geq 1-p} \right) \quad (2.7)$$

We now look at the derivation of one of these dualities in detail. Note that our argument uses formulae as intermediate steps which are not in PCTL, i.e. we reason in the logic PCTL* [14] which contains PCTL.

$$\begin{aligned} [\phi_1 \mathbf{W}^{\leq k} \phi_2]_{\geq p} &= [\phi_1 \mathbf{U}^{\leq k} \phi_2 \vee \mathbf{G} \phi_1]_{\geq p} \\ &= [\neg(\phi_1 \mathbf{U}^{\leq k} \phi_2 \vee \mathbf{G} \phi_1)]_{\leq 1-p} \\ &= \neg[\neg(\phi_1 \mathbf{U}^{\leq k} \phi_2 \vee \mathbf{G} \phi_1)]_{>1-p} \\ &= \neg[\neg(\phi_1 \mathbf{U}^{\leq k} \phi_2) \wedge \neg \mathbf{G} \phi_1]_{>1-p} \\ &= \neg[(\neg\phi_2) \mathbf{U}^{\leq k} \neg(\phi_1 \vee \phi_2) \vee \mathbf{G} \phi_1 \wedge \neg \mathbf{G} \phi_1]_{>1-p} \end{aligned}$$

As $\mathbf{G} \phi_1 \wedge \neg \mathbf{G} \phi_1$ is false in all states we omit $\mathbf{G} \phi_1$ from the disjunction:

$$\begin{aligned}
&= \neg[(\neg\phi_2) \mathbf{U}^{\leq k} \neg(\phi_1 \vee \phi_2)) \wedge \neg \mathbf{G} \phi_1]_{>1-p} \\
&= \neg[(\neg\phi_2) \mathbf{U}^{\leq k} \neg(\phi_1 \vee \phi_2)) \wedge \mathbf{F} \neg\phi_1]_{>1-p} \\
&= \neg[(\neg\phi_2) \mathbf{U}^{\leq k} \neg(\phi_1 \vee \phi_2)) \wedge \mathbf{F} \neg\phi_1]_{>1-p} \\
&= \neg[(\neg\phi_2) \mathbf{U}^{\leq k} (\neg\phi_1 \wedge \neg\phi_2)) \wedge \mathbf{F} \neg\phi_1]_{>1-p}
\end{aligned}$$

We know from the Until subformula that $(\neg\phi_1 \wedge \neg\phi_2)$ and hence $\neg\phi_1$ will be realised within finite time. Therefore we can omit the condition $\mathbf{F} \neg\phi_1$ and arrive at

$$[\phi_1 \mathbf{W}^{\leq k} \phi_2]_{\geq p} = \dots = \neg[\neg\phi_2 \mathbf{U}^{\leq k} \neg(\phi_1 \vee \phi_2)]_{>1-p} .$$

The second equation follows analogously.

In our context it is important to note that there are various normal forms possible for PCTL formulae. We present *Greater-Than normal form* and *Greater-Than negation normal form*.

Definition 31 (Greater-Than normal form)

The following subset of PCTL constitutes the *Greater-Than normal form* (GTNF):

$$\begin{aligned}
\phi, \psi &::= \mathbf{q} \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid [\alpha]_{\bowtie p} \\
\alpha &::= \mathbf{X} \phi \mid \phi \mathbf{U}^{\leq k} \psi \mid \phi \mathbf{W}^{\leq k} \psi
\end{aligned}$$

where $\mathbf{q} \in \mathbf{AP}$, $p \in [0, 1]$, $\bowtie \in \{\geq, >\}$ and $k \in \mathbb{N} \cup \{\infty\}$. ◆

Lemma 4

Each PCTL formula ϕ is semantically equivalent to a PCTL formula in Greater-Than normal form. The Greater-Than normal form of ϕ can be obtained by replacing all occurrences of the form $[\alpha]_{<p}$ in ϕ with $\neg[\alpha]_{\geq p}$, and by replacing all occurrences of $[\alpha]_{\leq p}$ in ϕ with $\neg[\alpha]_{>p}$. ●

Alternatively one could also replace formulae $[\alpha]_{<p}$ by $[\neg\alpha]_{>1-p}$ and $[\alpha]_{\leq p}$ by $[\neg\alpha]_{\geq 1-p}$ and then recursively process $\neg\alpha$ (which is in general not a valid

subformula in PCTL but in PCTL* [26]) according to the rules for LTL negation normal form.

For Markov decision processes and 3-valued Markov chains the above lemma on Greater-Than normal form is in general not true, because of the best-case respectively worst-case semantics for PCTL defined via schedulers for Markov decision processes respectively the optimistic and pessimistic PCTL semantics for 3-valued Markov chains.

Example 25

The Greater-Than normal form (GTNF) of the formula

$$[[\mathbf{X}[q \mathbf{U} r]_{<1/3}]_{\leq 1/2} \mathbf{U} r]_{>1/4}$$

is

$$[\neg[\mathbf{X} \neg[q \mathbf{U} r]_{\geq 2/3}]_{>1/2} \mathbf{U} r]_{>1/4} .$$

The two formulae are semantically equivalent; the latter formula uses only the comparison operators \geq and $>$. ◆

Furthermore each PCTL formula ϕ is semantically equivalent (over concrete Markov chains) to a PCTL formula in the following *Greater-Than negation normal form*.

Definition 32 (Greater-Than negation normal form)

The following subset of PCTL constitutes the *Greater-Than negation normal form* (GTNNF):

$$\begin{aligned} \phi, \psi ::= & \mathbf{q} \mid \neg \mathbf{q} \mid \phi \wedge \psi \mid \phi \vee \psi \mid [\alpha]_{\bowtie p} \\ \alpha ::= & \mathbf{X} \phi \mid \phi \mathbf{U}^{\leq k} \psi \mid \phi \mathbf{W}^{\leq k} \psi \end{aligned}$$

where $\mathbf{q} \in \mathbf{AP}$, $p \in [0, 1]$, $\bowtie \in \{\geq, >\}$ and $k \in \mathbb{N} \cup \{\infty\}$. ◆

Lemma 5

Every formula ϕ of PCTL that is not in GTNNF can be transformed to a formula in GTNNF, equivalent in the 2-valued semantics over Markov chains, by

1. replacing each subformula of the form $[\alpha]_{<p}$ and $[\alpha]_{\leq p}$ by $\neg[\alpha]_{\geq p}$ and $\neg[\alpha]_{>p}$ respectively; and then
2. pushing negations inwards according to the following recursive rules:

$$\begin{aligned}
\text{nnf}(\mathbf{q}) &= \mathbf{q} \\
\text{nnf}(\neg\mathbf{q}) &= \neg\mathbf{q} \\
\text{nnf}(\neg\neg\phi) &= \phi \\
\text{nnf}(\phi \wedge \psi) &= \text{nnf}(\phi) \wedge \text{nnf}(\psi) \\
\text{nnf}(\neg(\phi \wedge \psi)) &= \text{nnf}(\neg\phi) \vee \text{nnf}(\neg\psi) \\
\text{nnf}(\phi \vee \psi) &= \text{nnf}(\phi) \vee \text{nnf}(\psi) \\
\text{nnf}(\neg(\phi \vee \psi)) &= \text{nnf}(\neg\phi) \wedge \text{nnf}(\neg\psi) \\
\text{nnf}([\mathbf{X} \phi]_{\boxtimes p}) &= [\mathbf{X} \text{nnf}(\phi)]_{\boxtimes p} \\
\text{nnf}(\neg[\mathbf{X} \phi]_{\boxtimes p}) &= [\mathbf{X} \text{nnf}(\neg\phi)]_{\boxtimes 1-p} \\
\text{nnf}([\phi \mathbf{U}^{\leq k} \psi]_{\boxtimes p}) &= [\text{nnf}(\phi) \mathbf{U}^{\leq k} \text{nnf}(\psi)]_{\boxtimes p} \\
\text{nnf}([\neg(\phi \mathbf{U}^{\leq k} \psi)]_{\boxtimes p}) &= [\text{nnf}(\neg\psi) \mathbf{W}^{\leq k} \text{nnf}(\neg\phi \wedge \neg\psi)]_{\boxtimes 1-p} \\
\text{nnf}([\phi \mathbf{W}^{\leq k} \psi]_{\boxtimes p}) &= [\text{nnf}(\phi) \mathbf{W}^{\leq k} \text{nnf}(\psi)]_{\boxtimes p} \\
\text{nnf}([\neg(\phi \mathbf{W}^{\leq k} \psi)]_{\boxtimes p}) &= [\text{nnf}(\neg\psi) \mathbf{U}^{\leq k} \text{nnf}(\neg\phi \wedge \neg\psi)]_{\boxtimes 1-p}
\end{aligned}$$

where \boxtimes is defined such that:

$$\begin{aligned}
\boxtimes &\text{ equals } \geq \\
\boxtimes &\text{ equals } >
\end{aligned}$$

●

The second step – pushing negations inwards – is possible without breaking the syntactical restrictions of PCTL, only because our definition includes both weak Until and strong Until. An intermediate step in PCTL* explains the transformations:

$$\begin{aligned}
\neg[\mathbf{X} \phi]_{>p} &\equiv [\neg\mathbf{X} \phi]_{\geq 1-p} \equiv [\mathbf{X} \neg\phi]_{\geq 1-p} \\
\neg[\phi \mathbf{U}^{\leq k} \psi]_{>p} &\equiv [\neg(\phi \mathbf{U}^{\leq k} \psi)]_{\geq 1-p} \equiv [(\neg\psi) \mathbf{W}^{\leq k} (\neg\phi \wedge \neg\psi)]_{\geq 1-p} \\
\neg[\phi \mathbf{W}^{\leq k} \psi]_{>p} &\equiv [\neg(\phi \mathbf{W}^{\leq k} \psi)]_{\geq 1-p} \equiv [(\neg\psi) \mathbf{U}^{\leq k} (\neg\phi \wedge \neg\psi)]_{\geq 1-p}
\end{aligned}$$

Swapping the roles of \geq and $>$ in the above equivalences yields the dualities for the remaining combinations of temporal operators and threshold types. The negations $\neg\phi$ and $\neg\psi$ above are then recursively processed in the same manner as before.

Example 26

The Greater-Than negation normal form (GTNNF) of the formula

$$[[X[q \text{ U } r]_{<1/3}]_{\leq 1/2} \text{ U } r]_{>1/4}$$

is derived from its Greater-Than normal form (GTNF) as follows:

$$\begin{aligned} & [[\neg X \neg[q \text{ U } r]_{\geq 2/3}]_{\geq 1/2} \text{ U } r]_{>1/4} \\ \equiv & [[\neg X \neg[q \text{ U } r]_{\geq 2/3}]_{>1/2} \text{ U } r]_{>1/4} \\ \equiv & [[X \neg\neg[q \text{ U } r]_{\geq 2/3}]_{\geq 1-1/2} \text{ U } r]_{>1/4} \\ \equiv & [[X[q \text{ U } r]_{\geq 2/3}]_{\geq 1/2} \text{ U } r]_{>1/4} . \end{aligned}$$

◆

Alternatively, one could achieve a normal form which contains only one type of Until, i.e. U or W , by using negation of arbitrary formulae or all four comparison operators or both:

$$\begin{aligned} [\phi \text{ U } \psi]_{\bowtie p} &= \neg[(\neg\psi) \text{ W } (\neg\phi \wedge \neg\psi)]_{\bowtie p} \\ &= [(\neg\psi) \text{ W } (\neg\phi \wedge \neg\psi)]_{\bowtie 1-p} \\ [\phi \text{ W } \psi]_{\bowtie p} &= \neg[(\neg\psi) \text{ U } (\neg\phi \wedge \neg\psi)]_{\bowtie p} \\ &= [(\neg\psi) \text{ U } (\neg\phi \wedge \neg\psi)]_{\bowtie 1-p} \end{aligned}$$

Another important fragment of PCTL is qualitative PCTL which is PCTL restricted to thresholds of the form > 0 and ≥ 1 . To distinguish the unrestricted logic PCTL from the qualitative fragment we sometimes call the former *full* or *quantitative* PCTL.

There are still some important open questions about PCTL, in particular over infinite models. Some difficulties become already apparent for surprisingly small subsets of PCTL. Tomáš Brázdil, Vojtech Forejt, Jan Kretínský and Antonín Kucera recently showed that already qualitative PCTL does not have the finite model property and that “there are even qualitative PCTL

formulae which have only infinite-state models” [30]. An example from [30] which can only have infinite models is

$$G^{>0}(\neg q \wedge F^{>0} q)$$

in our notation of PCTL.

In the same paper Brázdil et al. show that satisfiability of qualitative PCTL is a decidable though EXPTIME-complete problem. Although the qualitative PCTL fragment does not have the finite model property, the authors show that for every satisfiable formulae ϕ from this fragment there exists a so called *marked graph* which is a “finite description of a model” for ϕ . Furthermore they show that if a quantitative PCTL formula is satisfiable, then it also has a model whose branching degree is bounded by (a function of) the length of the formula [30].

More than twenty years earlier, Sergiu Hart and Micha Sharir already showed the absence of finite model property for two logics which very closely resemble qualitative PCTL [89]. Further they prove that satisfiability of these logics over certain classes of models, namely finite state Markov chains and Markov chains with transition probabilities bounded away from 0, is decidable via a tableau method [89].

The decidability of satisfiability for full PCTL is still an open research problem. Brázdil et al. conjecture the problem to be decidable [30].

The model-checking problem for PCTL over (finite-state) Markov chains is decidable in polynomial time [88]. Model checking PCTL* is decidable in PSPACE (i.e. exponential in the length of the formula and polynomial in the size of the model) over Markov chains and certain generalisations of Markov chains [14, 26].

In contrast to the modal μ -calculus, PCTL allows only very restricted fixpoint operations. While the μ -calculus is closed under arbitrary least and greatest fixpoints, there is no such natural closure for PCTL. In this sense PCTL is similar to CTL which also contains only a few selected fixpoint operators, i.e. the unbounded Untils. The automata presented in Chapter 6 might be a first step toward a probabilistic logics which contains PCTL and is as naturally closed under fixpoint recursions as the modal μ -calculus is.

2.6 Probabilistic simulation and bisimulation

As for Kripke structures we are interested in abstraction (respectively refinement) and equivalence relations between Markov chains. Of particular importance for this thesis are probabilistic bisimulation which was first defined by Kim Larsen and Arne Skou [131], and probabilistic simulation which was defined for probabilistic specifications by Kim Larsen and Bengt Jonsson [108].

Definition 33 (Probabilistic bisimulation)

Let $M = (S, \mathbf{P}, L)$ and $N = (T, \mathbf{P}, L)$ be pointed Markov chains. An equivalence relation $R \subseteq S \times T$ is called *probabilistic bisimulation* iff whenever $(s, t) \in R$ then

- $L(s) = L(t)$ and
- $P(s, \pi_S(Q)) = P(t, \pi_T(Q))$ for all equivalence classes Q in $(S \times T)/R$

where π_S and π_T are the projections from $S \times T$ onto S and T respectively. We say s and t are *bisimilar*, denoted $s \sim t$, iff there exists a probabilistic bisimulation R such that $(s, t) \in R$.

Two pointed Markov chains M and N are bisimilar if their respective initial states are bisimilar. ◆

Intuitively, bisimilar states s and t agree on their labels and have the same transition probability to equivalence classes of R for some suitable projection of such equivalence classes onto S and T .

Jonsson and Larsen introduced a framework to describe *probabilistic specifications* and *probabilistic processes* – the latter being a special case of the former [108]. In the same article they define *probabilistic simulation* which captures refinement between probabilistic specifications as well as satisfaction between processes and specifications. We adopt their probabilistic simulation to our notation and restrict the definition to the special case of Markov chains as probabilistic processes.

Definition 34 (Probabilistic simulation)

Let $M = (S, \mathbf{P}, L)$ and $N = (T, \mathbf{P}, L)$ be concrete Markov chains. A relation $Q \subseteq S \times T$ is called *probabilistic simulation* if whenever $(s, t) \in Q$ then:

1. $L(s) = L(t)$.
2. There is a *weight function* $\rho: S \rightarrow (T \rightarrow [0, 1])$ which yields a probability distribution $\rho(s)$ for each $s \in S$ such that
 - a) $\sum_{s' \in S} (P(s, s') \cdot \rho(s')(t')) = P(t, t')$ for all $t' \in T$;
 - b) $(s', t') \in Q$ whenever $\rho(s')(t') > 0$.

We say t *simulates* s , denoted by $t \preceq s$, if there is a probabilistic simulation Q such that $(s, t) \in Q$. We say N *simulates* M , denoted $N \preceq M$, if the initial state of N simulates the initial state of M . \blacklozenge

Larsen and Jonsson give the following intuition for the weight function ρ in [108]:

Intuitively, the function ρ gives for each transition from s to s' a way of distributing the probability of this transition onto the transitions from t : the transition from t to t' receives the fraction $\rho(s')(t')$. The first condition on ρ checks that for any particular probability distribution that conforms with $\sigma(s)$, the fractions add up correctly.

For probabilistic processes, e.g. for concrete Markov chains, rather than probabilistic specifications this probabilistic simulation is an equivalence relation and actually a probabilistic bisimulation in the sense of Definition 33 [22, 108]. Nevertheless “probabilistic simulation” rather than “probabilistic bisimulation” is often used in the existing literature for the above relation even in the context of probabilistic processes.

Example 27

Let M and N be Markov chains as illustrated in Figure 2.17 and Figure 2.18 respectively. Then N simulates M . \blacklozenge

We extend probabilistic simulation to 3-valued Markov chains by changing the condition on the labelling functions. Intuitively, simulation respects the partial order on the truth values.

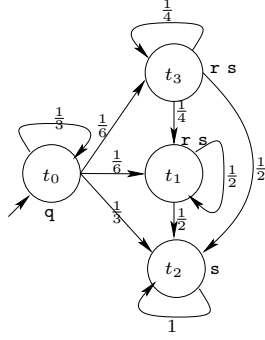


Figure 2.17: A Markov chain M .

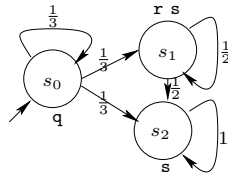


Figure 2.18: A Markov chain N .

Definition 35 (Probabilistic simulation)

Let $M = (S, \mathbf{P}, L)$ and $N = (T, \mathbf{P}, L)$ be 3-valued Markov chains. A relation $Q \subseteq S \times T$ is called *probabilistic simulation* if whenever $(s, t) \in Q$ then

1. $L(t, \mathbf{q}) \leq L(s, \mathbf{q})$ for all $\mathbf{q} \in \text{AP}$.
2. There is a *weight function* $\rho: S \rightarrow (T \rightarrow [0, 1])$ which yields a probability distribution $\rho(s)$ for each $s \in S$ such that
 - a) $\sum_{s' \in S} (P(s, s') \cdot \rho(s')(t')) = P(t, t')$ for all $t' \in T$;
 - b) $(s', t') \in Q$ whenever $\rho(s')(t') > 0$.

We say t *simulates* s , denoted by $t \preceq s$, if there is a probabilistic simulation Q such that $(s, t) \in Q$. Markov chain N simulates M , denoted $N \preceq M$, if the initial state of N simulates the initial state of M . \blacklozenge

For PCTL formulae in Greater-Than negation normal form and probabilistic simulation we can now secure a soundness result:

Lemma 6

Let M and N be 3-valued Markov chains and $N \preceq M$. Then for all formulae ϕ in GTNNF we have

- $N \models^P \phi$ implies $M \models^P \phi$; and
- $M \models^\circ \phi$ implies $N \models^\circ \phi$. ●

Proof

The proof is a structural induction on ϕ , using standard fixed-point and duality arguments for weak and strong Until formulae. ■

For concrete Markov chains, probabilistic bisimulation (and thus our probabilistic simulation) fully characterises PCTL [20, 22, 66, 131]. We quote the corresponding Theorem 10.67 from [20]: For a Markov chain M , the following statements are equivalent:

- s_1 and s_2 are bisimilar;
- s_1 and s_2 are PCTL*-equivalent, i.e. fulfill the same PCTL* formulae;
- s_1 and s_2 are PCTL-equivalent, i.e. fulfill the same PCTL formulae.

2.7 Unfoldings

In this section we define and discuss *unfoldings* of Markov chains. The partial completeness result in Chapter 4 builds on these unfoldings. We start with *full unfoldings*.

Definition 36 ((Full) Unfolding)

Let $M = (S, \mathbf{P}, L)$ be a 3-valued Markov chain. The *full unfolding* of M at s_0 is the Markov chain $M_{\text{full}}^{s_0} = (S_{\text{full}}, \mathbf{P}', L')$ where S_{full} is the set of nonempty sequences σ over S ; P' the transition probability

$$P'(s_0 \dots s_n, s_0 \dots s_n s_{n+1}) := P(s_n, s_{n+1})$$

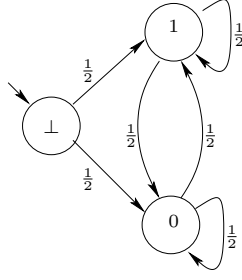


Figure 2.19: A Markov chain B .

and L' the labelling function

$$L'(\sigma \cdot s, \mathbf{q}) := L(s, \mathbf{q}) .$$

We restrict the set S_{full} to the set of sequences reachable from s_0 with positive probability. \blacklozenge

If M is a concrete Markov chain, so is $M_{\text{full}}^{s_0}$. Also, M and $M_{\text{full}}^{s_0}$ are bisimilar.

Example 28

Let B be the Markov chain illustrated in Figure 2.19. Its full unfolding B_{full}^{\perp} is the Markov chain given by the infinite, binary tree with uniform distribution over the two successors of each node. \blacklozenge

We now formalise finite unfoldings.

Definition 37 (Finite Unfolding)

1. Let $M = (S, \mathbf{P}, L)$ be a 3-valued Markov chain. For $i \in \mathbb{N}$ and $s_0 \in S$, the *finite unfolding* $M_i^{s_0} = (S_i, \mathbf{P}_i, L_i)$ is a Markov chain where S_i is the set of nonempty sequences over S of length at most i , plus a designated sink state t_{sink} . As for full unfoldings the transition probabilities are defined as

$$P_i(s_0 \dots s_n, s_0 \dots s_n s_{n+1}) := P(s_n, s_{n+1}) ,$$

for $n < i - 1$, and

$$P_i(s_0 \dots s_{i-1}, t_{\text{sink}}) = 1$$

for all sequences of length i , and additionally

$$P_i(t_{\text{sink}}, t_{\text{sink}}) = 1 .$$

Again, $L_i(\sigma \cdot s) = L(s)$, and $L_i(t_{\text{sink}}, \mathbf{q}) = \mathbf{?}$ for all $\mathbf{q} \in \mathbf{AP}$. We restrict S_i to sequences reachable from s_0 with positive probability.

2. For $j \in \mathbb{N}$, the finite unfolding $M_i^{s_0}$ is further restricted to maximal branching degree j as follows. Finite unfolding $M_{i,j}^{s_0} = (S_{i,j}, \mathbf{P}_{i,j}, L_{i,j})$ is a Markov chain where $S_{i,j} = S_i$ and $L_{i,j} = L_i$. For each $s \in S_i$, let t_1, t_2, \dots be an enumeration of $\{t_k \in S_i \mid P_i(s, t_k) > 0\}$ such that $P_i(s, t_k) \geq P_i(s, t_{k+1})$ for all $k \in \mathbb{N}$. We then define $\mathbf{P}_{i,j}$ by setting $P_{i,j}(s, t_k) = P_i(s, t_k)$ for $k \leq j$ and $P_{i,j}(s, t_{\text{sink}}) = 1 - \sum_{k=1}^j P_i(s, t_k)$. Again we restrict $S_{i,j}$ to sequences reachable from s_0 with positive probability. ◆

Example 29

The unfolding $M_{3,3}^{s_0}$ for the labelled Markov chain M of Figure 2.20 is depicted in Figure 2.21. ◆

Finite unfoldings give rise to simulations.

Lemma 7

For all 3-valued Markov chains M with initial state s_0 and $i, j \in \mathbb{N}$, the finite unfolding $M_{i,j}^{s_0}$ simulates M , i.e. $M_{i,j}^{s_0} \preceq M$. ●

Proof

We show that the relation

$$H = \{(s, \sigma \cdot s) \mid s \in S, \sigma \in S^*\} \cup \{(s, t_{\text{sink}}) \mid s \in S\}$$

is a simulation relation.

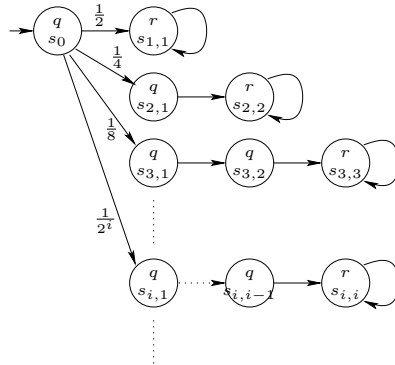


Figure 2.20: A Markov chain M satisfying $[qUr]_{\geq 1}$ and $[qWr]_{\geq 1}$.

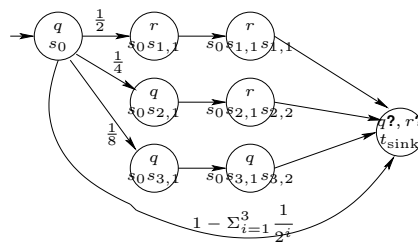


Figure 2.21: The finite unfolding $M_{3,3}^{s_0}$.

First, for all $\sigma \in S^*$, $s \in S$ and $\mathbf{q} \in \text{AP}$ we have $L(s, \mathbf{q}) = L(\sigma \cdot s, \mathbf{q})$. Since $L(t_{\text{sink}}, \mathbf{q}) = ?$ for all $\mathbf{q} \in \text{AP}$ we clearly have $L(t_{\text{sink}}, \mathbf{q}) \leq L(s, \mathbf{q})$ for all $s \in S$.

We now define the weight function $\rho_s: S \times S \rightarrow [0, 1]$:

- $\rho_s(s', t_{\text{sink}}) = 1$ if there is no sequence σ such that $\sigma \cdot s'$ is in $M_{i,j}^{s_0}$ (which equivalently means $|s_0 \dots s'| > i$ for every such sequence in M or $s' = t_{k'}$ with $k' > j$ for the ordering of the successor states t_k of a state s in $M_i^{s_0}$ as described in Definition 37).
- $\rho_s(s', t_{\text{sink}}) = 0$ for all other $s' \in S$.
- $\rho_s(s', \sigma) = 1$ if $\sigma = \sigma' \cdot s \cdot s'$.
- $\rho_s(s', \sigma) = 0$ for all other sequences.

Then the condition $\sum_{s' \in S} (P(s, s') \cdot \rho_s(s', \sigma')) = P(\sigma, \sigma')$ for all $\sigma' \in M_{i,j}^{s_0}$ collapses to $P(s, s') = P(\sigma, \sigma')$ for $\sigma = \sigma'' \cdot s$ and $\sigma' = \sigma'' \cdot s \cdot s'$ and $\sum_{s' \in S} P(s, s') = 1 = P(\sigma, t_{\text{sink}})$ for all other states. Both equations are obviously true by the construction of $M_{i,j}^{s_0}$.

Finally, we need to check the co-inductive condition for simulation: Whenever $\rho_s(s', t_{\text{sink}}) > 0$ we have $(s', t_{\text{sink}}) \in H$; whenever $\rho_s(s', \sigma') > 0$, then $\sigma' = \sigma'' \cdot s \cdot s'$ and hence $(s', \sigma') = (s', \sigma'' \cdot s \cdot s') \in H$. ■

Example 30

The infinite-state Markov chain M depicted in Figure 2.20 is simulated by the finite-state model $M_{3,3}^{s_0}$ in Figure 2.21, i.e. $M_{3,3}^{s_0} \preceq M$. ♦

2.8 Summary of chapter

In this chapter we provided relevant background on models, modal logics and simulation relations.

The chapter started with a brief introduction of non-probabilistic model-checking frameworks. It introduced Kripke structures and labelled transition systems as standard non-probabilistic models. We then discussed linear-time and branching-time logics, e.g. LTL, CTL, the modal μ -calculus and some important fragments of these logics. With regard to non-probabilistic

abstraction relations, we discussed simulation and bisimulation, and how these relations preserve formulae from certain fragments of the previously introduced logics.

We then turned to probabilistic model-checking frameworks with Markov chains and Markov decision processes as models and with PCTL as probabilistic logic. Discrete-time Markov chains and PCTL were identified as the main focus of this thesis. We defined probabilistic simulation and bisimulation and discussed – as in the non-probabilistic setting – how these relations preserve formulae of PCTL. In preparation of our partial completeness result in Chapter 4, we introduced the concept of finite unfoldings of discrete-time Markov chains.

Furthermore, this chapter introduced useful normal forms for formulae of non-probabilistic and probabilistic logics, and defined 3-valued extensions of the standard models in both the non-probabilistic and the probabilistic setting.

3 Background on Automata and Games

In this chapter we provide some background on automata and games which is relevant for the game-based PCTL semantics developed in Chapter 5, and for the class of probabilistic automata that we develop in Chapter 6 to achieve a complete abstraction framework for full PCTL.

Automata are widely used in computer science. The concept of automata, in its current interpretation, was established by Michael Rabin and Dana Scott [156], and Richard Büchi [33]. Since their groundbreaking work in the 1960s a multitude of different automata and supporting formalisms have been developed [168, 84] and automata theory is still a rich and active field of research. In this chapter, we restrict ourselves to the core concepts needed in this thesis. In particular we introduce

- non-deterministic word automata [156];
- probabilistic automata [157]; and
- alternating tree automata [158].

In their simplest form, called *word automata*, automata accept or reject finite words. Thus, each word automaton A defines a language $\mathcal{L}(A)$ of finite words. With suitable acceptance conditions word automata are generalised to accept or reject infinite words, thus defining a language $\mathcal{L}(A)$ of infinite words. Analogously, tree automata A accept or reject trees, thus defining tree languages $\mathcal{L}(A)$. Tree automata, too, can be defined for finite and infinite trees as input.

Probabilistic automata A have inputs as above, i.e. words and trees respectively. A probabilistic automaton maps an input x to a probability of acceptance. This gives rise to probabilistic languages $\mathcal{L}_\mu(A)$ which consist of all those inputs which are accepted with a probability greater than the fixed

threshold μ . The inputs of a probabilistic automaton are not probabilistic in themselves; the probabilities are induced purely by the automaton. Our p-automata, introduced in Chapter 6, are very different in that respect, as they determine a language $\mathcal{L}(A)$ of probabilistic inputs. A p-automaton maps a probabilistic input, i.e. a Markov chain, to a Boolean decision of acceptance or rejection. Thus, p-automata are more like tree automata than probabilistic automata: they accept Markov chains, but do not induce a probability of accepting or rejecting an input.

In this thesis we think of automata as (at least intuitively) similar to labelled transition systems. In particular we follow standard conventions in the literature and use action labels for automata, as opposed to state labels which we use for Kripke structures and Markov chains.

A very natural formalism for handling key notions of automata are games. For example, deciding whether an input is accepted or rejected by an automaton can be determined by establishing who wins a corresponding 2-player game. In this chapter we introduce two notions of games which are needed in this thesis:

- Hintikka Games capture the semantics of a logic in an operational fashion [94]. We define a Hintikka game for PCTL in Chapter 5.
- (Stochastic) Parity Games are a powerful general framework [37, 38, 179]. We use parity games and stochastic parity games to capture acceptance and simulation of p-automata in Chapter 6.

Other games which occur in the context of automata and logics, but which are beyond the scope of this thesis, are e.g. Ehrenfeucht-Fr ass e Games [170, 169, 67] and model-checking games [167, 129].

3.1 Word automata

We start with *deterministic word automata*.

Definition 38 (Deterministic Word Automaton)

A *deterministic word automaton* over a finite, non-empty alphabet Σ is a tuple $A = (S, \delta, s_0, F)$ where

- S is a finite set of states;

- $\delta: S \times \Sigma \longrightarrow S$ a total transition function;
- $s_0 \in S$ a designated initial state; and
- $F \subseteq S$ a designated set of accepting states. ◆

The domain of the transition function δ can be lifted to $S \times \Sigma^*$, i.e. to finite words x over Σ , by applying δ for each letter a_i in $x = a_0 a_1 \dots a_n$ at the state s_i which is determined by the previous application of δ starting from s_0 . We write

$$\delta(s_0, x) := \delta(\dots (\delta(\underbrace{\delta(s_0, a_0)}_{=s_1}, a_1) \dots), a_n)$$

for the lifted function $\delta: S \times \Sigma^* \longrightarrow S$.

A word $x \in \Sigma^*$ is *accepted* by a deterministic word automaton A if and only if $\delta(s_0, x) \in F$.

A deterministic word automaton $A = (S, \delta, s_0, F)$ contains a graph G_A with vertices S and edges $(s, s') \in S \times S$ whenever $\delta(s, \sigma) = s'$ for some $\sigma \in \Sigma$. In graphical representations we mark accepting states $F \subseteq S$ with double outlines and write σ on edge (s, s') if $\delta(s, \sigma) = s'$.

Example 31

Figure 3.1 depicts a deterministic word automaton A over alphabet $\Sigma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ with initial state s_0 and a single accepting state s_3 , i.e. $F = \{s_3\}$.◆

Definition 39 (Language)

The *language* $\mathcal{L}(A)$ of automaton A is the set of all words accepted by A .◆

The language $\mathcal{L}(A)$ of an automaton A is such an important concept in automata theory, that some automata terminology refers to set-theoretic concepts – implicitly applied to the set of all inputs accepted by A . For example:

- Automaton \overline{A} is called the *complement* of automaton A if $\mathcal{L}(\overline{A})$ is the set-theoretical complement of $\mathcal{L}(A)$.

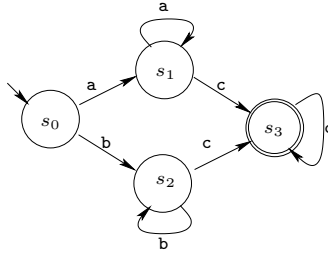


Figure 3.1: A deterministic word automaton A over alphabet $\Sigma = \{a, b, c\}$. Initial state s_0 is marked with an incoming arrow; accepting state s_3 is marked with double outlines; letters $\sigma \in \Sigma$ annotate edges.

- The *intersection* of automaton A and B is the automaton C that accepts the intersection of their languages, i.e. $\mathcal{L}(C) = \mathcal{L}(A) \cap \mathcal{L}(B)$.

Example 32

Automaton A of Figure 3.1 accepts all words over alphabet $\Sigma = \{a, b, c\}$ which start with a finite prefix of a or b and end with one or more c . Its language is

$$\mathcal{L}(A) = \{aa^*cc^*\} \cup \{bb^*cc^*\} .$$

The words consisting of only a 's or only b 's are not in the language of A as neither s_1 nor s_2 is accepting. ◆

The definition of word automata extends naturally to infinite words over Σ . The acceptance of infinite words x is, of course, not defined by the last state which the automaton A reaches in *consuming* the input word x . Instead, acceptance is defined by some other *acceptance condition*. Here we introduce *Büchi acceptance conditions* which essentially require a designated set $F \subseteq S$ to be visited infinitely often in consuming the word x .

Definition 40 (Deterministic Büchi Automaton)

A *deterministic Büchi automaton* A over an alphabet Σ is a tuple $A = (S, \delta, s_0, F)$ where

- S is a finite set of states;

- $\delta: S \times \Sigma \longrightarrow S$ a transition function;
- $s_0 \in S$ a designated initial state; and
- $F \subseteq S$ a designated set of accepting states. ◆

The definition of deterministic Büchi automaton is the same as the definition of deterministic word automaton. The two types of automata differ only in the interpretation of the set of accepting states, which for Büchi automata allows infinite *runs*.

Definition 41 (Run)

For a word $x \in \Sigma^\omega$ and a deterministic automaton A let the *run* π_x be the sequence of states $s_0s_1\dots$ which is visited when A consumes x . Let $\text{Inf}(\pi_x) \subseteq S$ be the set of states which occur infinitely often in the run π_x . For deterministic automaton A every word x yields a unique run π_x , and we sometimes write $\text{Inf}(x)$ instead of $\text{Inf}(\pi_x)$. ◆

Definition 42 (Büchi acceptance condition)

A infinite word $x = a_0a_1a_2\dots \in \Sigma^\omega$ is accepted by a deterministic Büchi automaton A if and only if $\text{Inf}(x) \cap F \neq \emptyset$. Or, equivalently (since S is finite):

$$\text{for all } i \in \mathbb{N} \text{ there exists } j \geq i \text{ such that } s_j := \delta(s_0, a_0 \dots a_j) \in F .$$

We call this requirement a *Büchi acceptance condition*. ◆

Accepting states of a Büchi automaton are sometimes called *fair*. In this sense we then call all other states *unfair*.

We do not assume all input words, respectively runs, of an automaton to be necessarily infinite, as we did for the paths of labelled transition systems and Kripke structures respectively. Therefore an acceptance condition actually requires two clauses, i.e. one for finite runs and one for infinite runs. The clause for finite runs is usually rather simple, such as

- all finite runs are rejecting; or

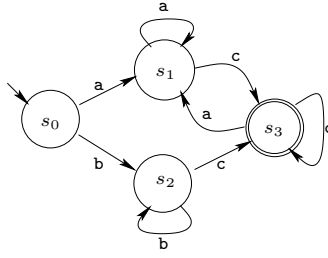


Figure 3.2: A deterministic Büchi automaton A over alphabet $\Sigma = \{a, b, c\}$.

- finite run $\pi = s_0s_1 \dots s_n$ is accepting iff its last state s_n satisfies some condition, e.g. $s_n \in F$.

Thus acceptance of finite runs is relatively straightforward and we often focus on the acceptance of infinite runs.

Example 33

Figure 3.2 depicts a deterministic Büchi automaton A over alphabet $\Sigma = \{a, b, c\}$ with initial state s_0 and a single accepting state, $F = \{s_3\}$. It accepts all infinite words over Σ which have a finite prefix of either a or b ; contain infinitely many c ; and contain only a and c after the first occurrence of c :

$$\mathcal{L}(A) = \{aa^*(cc^*a^*)^\omega\} \cup \{bb^*(cc^*a^*)^\omega\} .$$

The words b^ω and $(a^*c^*)^*a^\omega$ are not accepted by A because neither s_1 nor s_2 is an accepting state. ◆

A *co-Büchi automaton* is a Büchi automaton as above, where the acceptance condition for a word $x = a_0a_1a_2 \dots \in \Sigma^\omega$ is replaced by a *co-Büchi acceptance condition*:

there exists $i \in \mathbb{N}$ such that for all $j \geq i$
we have $s_j := \delta(s_0, a_0 \dots a_j) \in F$.

A word $x \in \Sigma^\omega$ is accepted by a co-Büchi automaton A if and only if there exists $i \in \mathbb{N}$ such that A – after consuming a finite prefix $x[0, \dots, i - 1]$ of x

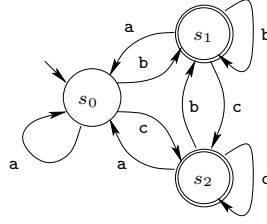


Figure 3.3: A deterministic Büchi automaton B over alphabet $\Sigma = \{a, b, c\}$.

– visits only states in F when consuming the remainder x^i of x .¹ In other words, eventually the run enters F and stays there.

Example 34

We interpret $F = \{s_1, s_2\}$ of automaton B illustrated in Figure 3.3 first as Büchi acceptance condition, then as co-Büchi acceptance condition. With Büchi acceptance condition B accepts all infinite words over $\Sigma = \{a, b, c\}$ which do not end in an a^ω suffix. With co-Büchi acceptance condition B accepts all infinite words over Σ which contain a only in a finite prefix. For example $(abc)^\omega$ is in the language of B with Büchi acceptance condition but not with co-Büchi acceptance condition. \blacklozenge

There are other acceptance conditions, such as *Rabin*, *Müller*, *Streett* and *Parity*. More details on these acceptance conditions can be found, for example, in [84] and [168]. Here we only introduce *parity conditions* as the most general type of acceptance condition used in this thesis.

Definition 43 (Parity condition)

For an automaton A with set of states S , a *parity condition* F is a function

$$F: S \longrightarrow \mathbb{N}$$

which assigns *parity*² $F(s)$ to states s . An infinite run π of A is accepting according to parity condition F if and only if the largest parity which occurs

¹For an infinite run on an input x , Büchi and co-Büchi acceptance are essentially determined by a finite prefix of the run or independent of all such finite prefixes, respectively.

This duality is similar to the duality of safety and liveness discussed on page 47.

²Sometimes also called *priority* by other authors.

infinitely often on π , formally

$$\sup\{F(s) \mid s \in \text{Inf}(\pi)\}$$

is even. ◆

Definition 44

For a paths π of an automaton A with parity function F , let

$$\text{lim sup}(\pi) := \sup_{s \in \text{Inf}(\pi)} F(s)$$

be the highest parity which occurs infinitely often on π . ◆

Alternatively it is also possible to use the smallest occurring value, i.e. $\inf\{F(s) \mid s \in \text{Inf}(\pi)\}$ to define acceptance. One then replaces lim sup with lim inf accordingly. Both variants are commonly found in the literature.

Many other acceptance conditions can be expressed as parity condition. For example the accepting behaviour of a Büchi automaton with accepting set of states E can be expressed by a parity condition F which assigns $F(s) = 2$ to all $s \in E$ and $F(s) = 1$ to all other states.

Another important extension of deterministic automata are the so-called *non-deterministic* automata. In a non-deterministic automaton transition function δ is replaced by

$$\delta: S \times \Sigma \longrightarrow 2^S$$

yielding a set of successor states $\delta(s, \sigma)$ from which a successor of s is chosen non-deterministically.

Definition 45 (Non-deterministic Büchi Automaton)

A *non-deterministic Büchi automaton* over alphabet Σ is a tuple $A = (S, \delta, s_0, F)$ where

- S is a finite set of states;
- $\delta: S \times \Sigma \longrightarrow 2^S$ a transition function;
- $s_0 \in S$ a designated initial state; and

- $F \subseteq S$ a designated set of accepting states. ◆

Acceptance for non-deterministic automata A depends on a more general notion of *run*. Essentially a run is a path in S given by an infinite sequence of non-deterministic choices from sets of successor states $\delta(s_i, a_i)$.

Definition 46 (Run)

Let $A = (S, \delta, s_0, F)$ be a non-deterministic automaton over alphabet Σ . A *run* ρ of A on input $x = a_0a_1 \dots \in \Sigma^\omega$ is an infinite sequence of states $s_0s_1 \dots$ starting from the initial state s_0 , such that $s_{i+1} \in \delta(s_i, a_i)$ for all $i \geq 0$. ◆

A run $\rho = s_0s_1 \dots$ of automaton A is *accepting* if it is accepted according to the acceptance condition of A when interpreted as path in A . For example, a run $\rho = s_0s_1 \dots$ of a non-deterministic Büchi automata $A = (S, \delta, s_0, F)$ is accepting if $\text{Inf}(s_0s_1 \dots) \cap F \neq \emptyset$.

Definition 47

A word $x \in \Sigma^\omega$ is accepted by non-deterministic Büchi automaton A if and only if there exists an accepting run of A on x . ◆

Note that the existence of one accepting run is enough; not all possible runs need to be accepting.

Example 35

Figure 3.4 illustrates a non-deterministic Büchi automaton C over alphabet $\Sigma = \{\mathbf{a}, \mathbf{b}\}$. It is non-deterministic as s_1 has two outgoing transitions for letter \mathbf{a} and s_2 has two outgoing transitions for letter \mathbf{b} . It accepts all infinite words over $\Sigma = \{\mathbf{a}, \mathbf{b}\}$ which start with an even number of \mathbf{a} , and contain infinitely many \mathbf{b} , and if there are further appearances of \mathbf{a} these \mathbf{a} appear in sequences of odd length:

$$(\mathbf{aa})^* \mathbf{aab}^* (\mathbf{b}((\mathbf{aa}^*)\mathbf{a})^* \mathbf{b}^*)^\omega$$

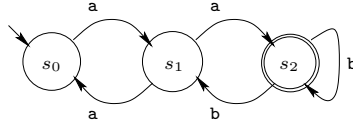


Figure 3.4: A non-deterministic Büchi automaton C over alphabet $\Sigma = \{a, b\}$.

An accepting run for such a word is of the form

$$(s_1 s_0)^* s_1 s_2 s_2^* (s_1 (s_0 s_1)^* s_2 s_2^*)^\omega$$

which visits accepting state s_2 infinitely often. Other words are not accepted, e.g. every run for a word which starts with an odd number of a will either be in s_2 when it needs to consume another a , or will be in s_0 when it needs to consume the first b . In both cases the next letter can not be consumed and the run is rejecting.

Even for words which have the right form there can be rejecting runs, e.g. when the wrong transition is chosen from s_1 or s_2 the run will lead to a situation where no further letter can be consumed. Such rejecting runs pose no problem as the existence of one accepting run is enough for C to accept the word. \blacklozenge

Of course, deterministic Büchi automata are a special case of the more general non-deterministic Büchi automata; and (non-deterministic) word automata can be seen as Büchi automata with a suitable extension of finite words. Büchi automata themselves are a special case of the more general automata with parity acceptance condition [84].

A labelled transition system $T = (S, R, \text{Act})$ with initial state s_0 can be seen as non-deterministic Büchi automaton $A = (S, R, s_0, S)$, where all states S are accepting. Similarly a Kripke structure $K = (S, R, L)$ with initial state s_0 can also be converted into a non-deterministic Büchi automaton.

Example 36

The language $L = (a^* b^*)^* a^\omega$ of words over $\{ab\}$ which end on an infinite suffix of a is not recognisable by any deterministic Büchi automaton [168].

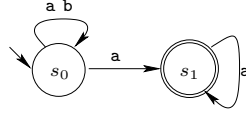


Figure 3.5: A non-deterministic Büchi automaton A over alphabet $\Sigma = \{a, b\}$ that recognises the language $L = (a^*b^*)^*a^\omega$.

Nevertheless L is recognised by the non-deterministic Büchi automaton of Figure 3.5. \blacklozenge

3.2 Probabilistic automata

Probabilistic automata were introduced by Michael Rabin [157]. These automata associate probabilities with input words. A word x is accepted by automaton A if its associated probability exceeds a given threshold μ . The language of A for threshold μ consists of all words which are assigned probability at least μ .

Definition 48 (Probabilistic Automata)

A *probabilistic automaton* over alphabet Σ is a tuple $A = (S, \delta, s_0, F)$ where

- S is a finite set of states;
- $\delta: S \times \Sigma \times S \rightarrow [0, 1]$ a function which assigns transition probabilities such that

$$\sum_{t \in S} \delta(s, \sigma, t) = 1$$

for every $(s, \sigma) \in S \times \Sigma$;

- $s_0 \in S$ a designated initial state; and
- $F \subseteq S$ a designated set of accepting states. \blacklozenge

For simplicity of presentation we only consider the acceptance of finite words $x \in \Sigma^*$. For probabilistic automata A , consuming a word x of length n defines a probabilistic tree of height n . An input word $x \in \Sigma^*$ is accepted

by A with probability p , where p is the aggregated probability of A reaching accepting states $s \in F$ when consuming x . The language $\mathcal{L}_\mu(A)$ of a probabilistic automaton A consists of all words which are accepted by A with probability at least μ .

Example 37

Figure 3.6 illustrates a probabilistic automaton A over alphabet $\Sigma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$. We group the probabilistic transitions (s, α, \cdot) from a state s which read the same letter α by using dashed lines and filled dots as we did for Markov decision processes in Section 2.4.

The automaton reads words $x \in \Sigma^*$ of the form

$$(\mathbf{a}^* \mathbf{b}^*)^* \mathbf{a} \mathbf{c}^*$$

and the probability assigned to word x is

$$\left(\frac{1}{2}\right)^k$$

where k is the number of \mathbf{a} in x . All other words are accepted with probability 0. The language $L_{0.06}(A)$ then is the set of all words over Σ of the form $(\mathbf{a}^* \mathbf{b}^*)^* \mathbf{a} \mathbf{c}^*$ which contain at most four \mathbf{a} . As $(\frac{1}{2})^4 = 0.0625$ only these words, when consumed by A , are assigned a probability which exceeds the threshold 0.06 for this language.

Other thresholds yield different languages. For example, words in $L_{0.5}(A)$ may contain only one \mathbf{a} , and thus must contain exactly one. \blacklozenge

A Markov chain $M = (S, \mathbf{P}, L)$ can be understood as probabilistic automaton with $F = S$ which assigns a probability to a (finite) sequence of states.

Conceptually probabilistic automata are very different from the Büchi automata introduced above, as the “reply” of A for an input word x is not Boolean any more, i.e. it is not just “accept” or “reject” but a value in $[0, 1]$. Further, the language of A is not fully intrinsic to A but dependent on a threshold μ . Thus, potentially there are infinitely many languages $\mathcal{L}_\mu(A)$ associated with a single automaton A . Probabilistic automata are also more expressive than standard automata, i.e. for every non-deterministic

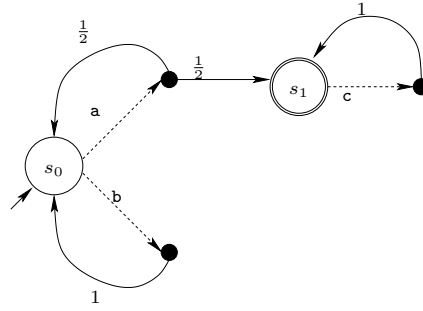


Figure 3.6: A probabilistic automaton A over alphabet $\Sigma = \{a, b, c\}$.

automaton there exists a probabilistic automaton which accepts the same language but there exist probabilistic automata whose language can not be recognised by a non-deterministic automaton [157].³

Example 38

The automaton C of Figure 3.7 has only three distinct languages \mathcal{L}_μ over $\Sigma = \{a, b\}$:

- The empty language, i.e. $\mathcal{L}_\mu = \{\}$ for $\mu > \frac{1}{2}$;
- Words starting with a , i.e. $\mathcal{L}_\mu = a(a^*b^*)^*$ for $\frac{1}{2} \geq \mu > \frac{1}{3}$;
- All words, i.e. $\mathcal{L}_\mu = (a^*b^*)^*$ for $\frac{1}{3} \geq \mu$.

Automaton D of Figure 3.8 has infinitely many distinct languages \mathcal{L}_{μ_k} . It reads all words x over $\Sigma = \{a, b\}$ and assigns a probability to each word which is proportional to its length. The probability of D accepting a word x of length k is

$$\mu_k := \sum_{i=1}^k \left(\frac{1}{10}\right)^i \left(\frac{9}{10}\right)^{(i-1)}.$$

Thus, such thresholds μ_k define infinitely many distinct languages \mathcal{L}_{μ_k} , each of which contains the words of length at least k . These languages are ordered by set inclusion $\mathcal{L}_{\mu_1} \subset \mathcal{L}_{\mu_2} \subset \mathcal{L}_{\mu_3} \dots$ \blacklozenge

³Thus, the language of a probabilistic automaton is not in general a *regular language*.

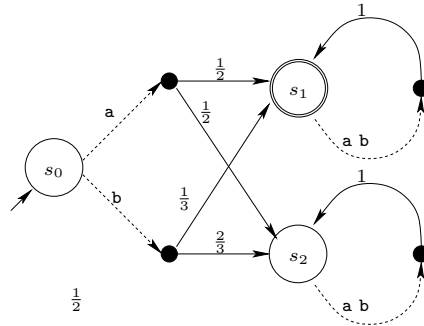


Figure 3.7: A probabilistic automaton C over alphabet $\Sigma = \{a, b\}$.

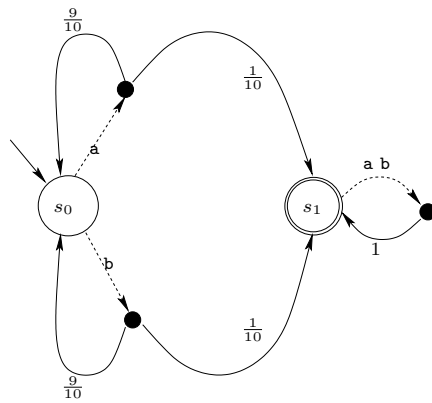


Figure 3.8: A probabilistic automaton D over alphabet $\Sigma = \{a\}$.

As we have already described for Markov chains in Section 2.4, the transition probabilities in a probabilistic automaton can be extended from a probability distribution over finite words to a distribution over infinite words [16]. As in the case of non-probabilistic automata, the set of final states F in the definition of probabilistic automata can be replaced by more general acceptance conditions [16]. We omit a detailed discussion of those extensions as they are not relevant for the work in this thesis.

3.3 Alternating tree automata

Word automata and probabilistic automata both consume words as input. They can be generalised to tree automata [158] which accept trees rather than words. Alternating tree automata then are a further generalisation of tree automata which allows for easy complementation of automata at the price of more complex decision problems [49, 36]. Complementation of automata and languages here are related in that the complement \bar{A} of automaton A has as its language $\mathcal{L}(\bar{A})$ the set-theoretic complement of the language of A :

$$\mathcal{L}(\bar{A}) = \Sigma^\omega \setminus \mathcal{L}(A)$$

Such a complementation is particularly of interest when one uses automata as representatives for logical formulae [179, 84], or for model checking [124, 178, 25]. Since model checking is one of the main motivations of this thesis, and the correspondence between automata and logical formulae plays a crucial role in our technical results in Chapter 6, we introduce alternating tree automata directly without discussing (standard) tree automata and their tree languages first.

Pointed Kripke structures can be understood as trees simply by “unfolding” them from their initial state. Hence alternating tree automata are also acceptors of Kripke structures – much like our p-automata in Chapter 6 are acceptors of Markov chains. Conversely a tree can be seen as (infinite-state) Kripke structure. Thus we use both terms interchangeably for the inputs of alternating tree automata.

Our presentation of alternating tree automata in this chapter follows the notations of [84, Chapter 9] and [179].

The automata in the previous sections had simple transitions from states

to states, e.g. given as transition relation $\delta \subseteq S \times S$, or they had the choice from a set of such transitions. In contrast, the more general transitions of tree automata and alternating tree automata are given by so called *transition conditions*.

Definition 49 (Transition conditions)

For a set Q we define *transition conditions* as follows:

$$\phi ::= \text{tt} \mid \text{ff} \mid \phi \vee \phi \mid \phi \wedge \phi \mid (\square, q) \mid (\diamond, q) \quad (3.1)$$

where $q \in Q$. ◆

The subset of the transition conditions without (\square, q) and (\diamond, q) are the positive Boolean formulae $B^+(Q)$ over Q . In this sense we refer to the transition conditions of equation (3.1) as $B^+(\{\square, \diamond\} \times Q)$.

In this thesis we abbreviate the notation for \square and \diamond as follows:

$$\begin{aligned} q^\square &:= (q, \square) \\ q^\diamond &:= (q, \diamond) \end{aligned}$$

This notation is slightly unusual – other authors use $\square q$ and $\diamond q$ instead of q^\square and q^\diamond respectively – but it aids the presentation of our p-automata in Chapter 6.

Definition 50 (Alternating Tree Automaton)

An *alternating tree automata* over alphabet Σ is a tuple $A = (Q, \delta, \varphi^{\text{in}}, F)$ where

- Q is a finite set of states;
- $\delta: Q \times \Sigma \longrightarrow B^+(\{\square, \diamond\} \times Q)$ a transition function from automaton states reading letters to transition conditions;
- $\varphi^{\text{in}} \in B^+(\{\square, \diamond\} \times Q)$ a designated initial condition; and
- $F: Q \longrightarrow \mathbb{N}$ a parity condition. ◆

At first the notion of transition condition in alternating tree automata looks very different from the transition relation of the word automata which we introduced in the previous sections. Our word automata look very much like labelled transition systems, while the transition condition of alternating tree automata essentially are logical formulae.

To get used to this perspective of “transitions as formulae” it is instructive to look at non-deterministic word automata in the transition condition notation: Restricting the transition conditions of an alternating tree automata to disjunctions yields non-deterministic word automata. Intuitively, a non-deterministic transition $\delta(s, a) = \{s', s''\}$ reads “when reading a at s the automaton A moves to s' or to s'' ”. Disjunction and conjunction together yield tree automata, where the conjunction handles the branching of input trees. To get a first approximative intuition, one could imagine that a tree automaton A in a state s with conjunctive transition condition $s' \wedge s''$, consumes a node of a (binary) tree by moving to automaton states s' and s'' which consume the next node in the left and right successor of the tree node respectively. (This intuition is not quite right though, as it would imply that the automaton distinguishes between directions “left” and “right”, and that e.g. $s' \wedge s''$ would thus be different from $s'' \wedge s'$.) Another intuition for conjunctions in tree automata, which is used frequently in the literature, is that automaton A splits into two copies A' and A'' which then continue to consume the input tree simultaneously.

Alternating tree automata also have the modalities \Box and \Diamond in their transition conditions which are essentially universal and existential quantifiers, and which play a crucial role in the correspondence between alternating tree automata and the modal μ -calculus.

Before we give formal definitions of *runs* and *acceptance* for alternating tree automata, here some intuitive examples adopted from [179].

Example 39

Let $A = (\{q\}, \delta, q, F)$ be an alternating tree automaton with only one state, where

- $\delta(q, \cdot) = q^\Box$;
- $F(q) = 1$.

The single state q has parity 1 and its transition condition is q^\square . Intuitively the transition condition $\delta(q, \cdot) = q^\square$ says that whenever A in q reads a state s of a Kripke structure, then all successors s' of s will also be processed by q .

Therefore any paths π of a Kripke structure can be read by A , and A will remain in q while reading π . Every infinite run of A clearly contains only parity 1 infinitely often and thus is rejecting. Thus A accepts exactly these Kripke structures K which have only finite paths, as these paths end on a state s with empty successor set.⁴ \blacklozenge

Example 40

Let $B = (\{q\}, \delta, q, F)$ be an alternating tree automaton with only one state, where

- $\delta(q, \mathbf{a}) = \mathbf{tt}$;
- $\delta(q, \neg \mathbf{a}) = q^\diamond$;
- $F(q) = 1$.

The single state q has parity 1 and thus all infinite runs are rejecting. Intuitively $\delta(q, \cdot) = q^\diamond$ says that whenever A in q reads a state s , then at least one successor s' of s will be processed by q . Therefore B accepts these Kripke structures K in which a state s which satisfies \mathbf{a} is reachable in finitely many steps.

The automaton B corresponds to the μ -calculus formula $\mu q.(\mathbf{a} \vee \diamond q)$. \blacklozenge

While runs of word automata were sequences of (automata) states, the runs of alternating tree automata are trees labelled with pairs of automata states and states of a Kripke structure.

For a pair (q, s) of an automata state q and a Kripke structure state s the transition condition $\delta(q, L(s))$ can be understood as “In q reading the labelling of s the automaton moves to $\delta(q, L(s))$ ”, which in turn enforces conditions on the successors of s .

⁴In particular A does not accept serial Kripke structures. We include the example nevertheless, as it is intuitive and insightful enough, although we assumed all Kripke structures in this thesis to be serial.

Definition 51 (Run of an alternating tree automaton)

Let $A = (Q, \delta, \varphi^{\text{in}}, F)$ be an alternating tree automaton and $K = (S, \longrightarrow, L)$ be a Kripke structure. A *run* $R = (V, E, \lambda)$ of A on K is a tree with

- vertices V ;
- edges $E \subseteq V \times V$; and
- labelling function $\lambda: V \longrightarrow Q \times S$.

For vertex v with $\lambda(v) = (q, s)$ let $\pi_1(v)$ and $\pi_2(v)$ be the projection on the first and second component of $\lambda(v)$ respectively.

The root of R is labelled $(q^{\text{in}}, s^{\text{in}})$ for the initial states of A and K respectively. Every vertex v with $\lambda(v) = (q, s)$ satisfies the corresponding transition condition $\tau = \delta(q, L(s))$ of A . Satisfaction of transition condition τ is denoted $v \models \tau$ and defined as follows:

$v \models \text{tt}$ for all $v \in R$

$v \not\models \text{ff}$ for all $v \in R$

$v \models q^\diamond$ iff there exists $v' \in \text{Succ}_R(v)$ such that

$$\pi_1(v') = q \text{ and } \pi_2(v') \in \text{Succ}_K(\pi_2(v))$$

$v \models q^\square$ iff for all $s \in \text{Succ}_K(\pi_2(v))$ there exists $v' \in \text{Succ}_R(v)$ such that

$$\lambda(v') = (q, s)$$

$v \models \phi \vee \psi$ iff $v \models \phi$ or $v \models \psi$

$v \models \phi \wedge \psi$ iff $v \models \phi$ and $v \models \psi$ ◆

A run R is accepting if and only if all infinite paths $v_0 v_1 \dots$ in R are accepting as determined by the parity acceptance condition F of A on $\pi_1(v_0) \pi_1(v_1) \dots$. A Kripke structure K is accepted by automaton A if there exists an accepting run R of A on K . This notion of acceptance can be phrased as a parity game, and indeed it can be useful to define acceptance directly as a parity game [84, Chapter 9]. In Chapter 6 we similarly define acceptance for p-automata as a two-player game.

With the notion of accepting run in place we now re-visit Example 40, and look at an accepting run of the automaton B for a concrete Kripke structure.

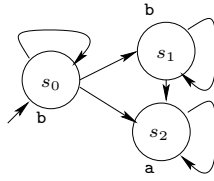


Figure 3.9: A Kripke structure M over alphabet $\Sigma = \{\mathbf{a}, \mathbf{b}\}$.

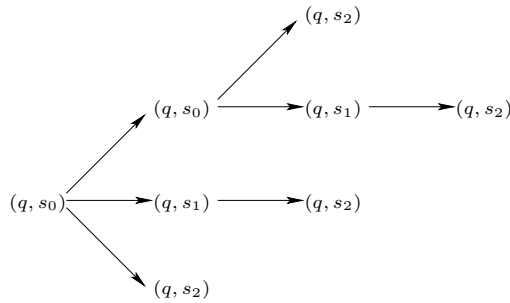


Figure 3.10: An accepting run of B on M .

Example 41

Let B the automaton of Example 40 and let M be the Kripke structure depicted in Figure 3.9. Automaton B accepts M . To simplify presentation we identify vertices v of a run with their label $\lambda(v) = (q, s)$. The shortest accepting run is

$$(q, s_0), (q, s_2) .$$

Another accepting run is shown in Figure 3.10. There are also rejecting runs, e.g.

$$(q, s_0), (q, s_0), (q, s_0), \dots$$

but as for non-deterministic word automata the existence of one accepting run is sufficient for M to be accepted by B . \blacklozenge

The language $\mathcal{L}(A)$ of an alternating tree automaton A consists of all Kripke structures whose infinite unfoldings are accepted by A .

Example 42

Let B be the automaton of Example 40. Its language is the set of all Kripke structures K in which a state s which satisfies \mathbf{a} is reachable in finitely many steps from the initial state of K . \blacklozenge

As preparation for the evaluation of our p-automata we state some classical decision problems of automata theory and their complexity for alternating tree automata. Let \mathbf{K} be the set of all Kripke structures.

Membership Given alternating tree automaton A and Kripke structure K , does A accept K , i.e. $K \in \mathcal{L}(A)$? Membership is decidable in $\text{UP} \cap \text{co-UP}$ [179].

Non-emptiness Given alternating tree automaton A , is its language empty, i.e. $\mathcal{L}(A) = \{\}$? Deciding Non-emptiness is EXPTIME-complete [49, 124].

Universality Given alternating tree automaton A , does it accept every input, i.e. $\mathcal{L}(A) = \mathbf{K}$? As alternating tree automata can be complemented in linear time [49], Universality is equivalent to Non-emptiness. Thus it is also EXPTIME-complete [49].

Language inclusion Given alternating tree automata A and B , is the language of A subset of the language of B , i.e. $\mathcal{L}(A) \subseteq \mathcal{L}(B)$? Language inclusion can be reduced to complementation, intersection and non-emptiness. Therefore it is also EXPTIME-complete [49].

Further explanations of the formalism and the intuition of alternating tree automata can be found for example in [179]; for a detailed discussion including the complexity of relevant decision problems see [49, Chapter 7] and [84, Chapter 9]. As mentioned above, there is a close connection between alternating tree automata and formulae of certain (branching time) logics: e.g. alternating tree automata and the modal μ -calculus are equally expressive [71, 148, 179, 84]. In particular every formula ϕ of the modal μ -calculus gives rise to an alternating tree automaton A that accepts exactly those Kripke structures that satisfy the formula ϕ . See [179] for the conversion between formulae and automata and for a detailed discussion of the correspondence between the two formalisms. Here we only sketch one

direction of the correspondence: translating a formula into an alternating tree automaton.

Every formula ϕ of the modal μ -calculus gives rise to an alternating tree automaton A_ϕ such that

$$\mathcal{L}(A_\phi) = \{K \in \mathbf{K} \mid K \models \phi\} .$$

Let ϕ be in positive normal form, then $A_\phi = (Q, \delta, [\phi], F)$ can be defined as follows:

- For each subformula ψ of ϕ (including ϕ itself) Q contains a state $[\psi]$.
- The initial state of A_ϕ is $[\phi]$.
- The transition function δ is as follows:

$$\begin{aligned} \delta([\text{ff}], \cdot) &= \text{ff} \\ \delta([\text{tt}], \cdot) &= \text{tt} \\ \delta([q], \cdot) &= q \text{ if } q \in \text{free}(\phi) \\ \delta([q], \cdot) &= [\eta q. \psi] \text{ if } q \notin \text{free}(\phi) \\ \delta([\psi \wedge \chi], \cdot) &= [\psi] \wedge [\chi] \\ \delta([\psi \vee \chi], \cdot) &= [\psi] \vee [\chi] \\ \delta([\diamond \psi], \cdot) &= [\psi]^\diamond \\ \delta([\square \psi], \cdot) &= [\psi]^\square \\ \delta([\eta q. \psi], \cdot) &= [\psi] \end{aligned}$$

where $\text{free}(\phi)$ is the set of free propositional variables in ϕ , and $\eta \in \{\mu, \nu\}$ as determined by the structure of ϕ .

- The parity $F([\psi])$ is essentially the *alternation depth* $\Omega(\psi)$ [179, 29] of the subformulae ψ of ϕ , shifted to an even number if the outermost fixpoint operator of ϕ is a greatest fixpoint and shifted to an odd number if the outermost fixpoint operator is a least fixpoint. See [179] for details on this construction.

For a formulae ϕ the construction sketched above yields an automaton A_ϕ which accepts exactly those Kripke structures that satisfy ϕ . For the correctness of this construction see e.g. [179].

In Chapter 6 we embed PCTL formulae into our p-automata but there is no one-to-one correspondence between formulae and automata as in the above case of the modal μ -calculus and alternating tree automata. In fact, we show that our automata are strictly more expressive than PCTL. Thus, PCTL corresponds to a proper subset of p-automata as, e.g., CTL does for alternating tree automata. It is presently not known whether there is a logic that plays the role for p-automata that the modal μ -calculus plays for alternating tree automata. That is to say, there is currently no canonical, probabilistic fixed-point logic that contains PCTL as a fragment. The question of whether such a suitable probabilistic logic exists is not addressed in this thesis and left as future work.

3.4 Hintikka games

Hintikka games provide an operational definition of truth, in contrast with Alfred Tarski's definition of truth as a predicate [94].⁵ Hintikka games can be used to define the semantics of a logic in an operational fashion.

A Tarskian notion of truth is a formally defined predicate \models between models and formulae of first-order logic, where “property ϕ is true in model M ” is defined as “predicate $M \models \phi$ holds”. For example, if M is the set of natural numbers $\{0, 1, \dots\}$ and ϕ is $\neg\exists x((x * x < x + 1) \wedge (x > 1))$, then $M \models \phi$ holds since all natural numbers are either at most 1 or their square is bigger than their successor number.

We introduce Hintikka games by example, i.e. by sketching a Hintikka game for first-order logic. We keep this example fairly brief and omit a formal definition of the game moves etc. For model M and formula ϕ , a Hintikka game $G(M, \phi)$ involves two players, Verifier V (who wants to prove that M satisfies ϕ) and Refuter R (who wants to prove that M does not satisfy ϕ). Game $G(M, \phi)$ has *configurations* which are triples of form

$$\langle M[\vec{x} \mapsto \vec{a}], \psi, \mathbf{C} \rangle$$

where $[\vec{x} \mapsto \vec{a}]$ binds a set of variables x_i to natural numbers a_i , \mathbf{C} is either

⁵Roughly at the same time as the work by Hintikka, but independently and in a different context, a definition of truth via dialogue games was developed by Paul Lorenzen and Kuno Lorenz [135, 136]. A more general overview on dialogue games in logics can be found in [159].

Refuter \mathbf{R} or Verifier \mathbf{V} , and ψ is either ϕ or a strict sub-formula of ϕ . The initial configuration is $\langle M, \phi, \mathbf{V} \rangle$, saying that \mathbf{V} claims that ϕ is true in M . Game $\mathbf{G}(M, \phi)$ generates a game tree whose paths are plays – finite sequences of configurations. Below, we write $\bar{\mathbf{C}}$ for the player other than \mathbf{C} , i.e. $\bar{\mathbf{V}} = \mathbf{R}$, $\bar{\mathbf{R}} = \mathbf{V}$ and $\bar{\mathbf{C}} = \mathbf{C}$.

For sake of illustration, consider the game for our example. The formula $\phi = \neg \exists x ((x * x < x + 1) \wedge (x > 1))$ is a negation, so the initial configuration has $\langle M, \psi_0, \mathbf{R} \rangle$ as sole next configuration with $\psi_0 = \exists x ((x * x < x + 1) \wedge (x > 1))$. Thus, we just swap the player – if \mathbf{C} claims $\neg \psi$ then $\bar{\mathbf{C}}$ claims ψ – and remove the negation symbol when processing a negation. At configuration $\langle M, \psi_0, \mathbf{R} \rangle$, the formula is an existential one and so the player that claims its truth (here \mathbf{R}) can choose a natural number, say 5, and bind it to x , resulting in the next configuration $\langle M[x \mapsto 5], \psi_1, \mathbf{R} \rangle$, where ψ_1 is $(x * x < x + 1) \wedge (x > 1)$. In particular, configuration $\langle M, \psi_0, \mathbf{R} \rangle$ has infinitely many next configurations, one for each natural number a bound to x . Now that formula ψ_1 is a conjunction claimed to be true by \mathbf{R} , and so his opponent \mathbf{V} can choose a conjunct.

If \mathbf{V} chooses conjunct $x > 1$, the next configuration is $\langle M[x \mapsto 5], x > 1, \mathbf{R} \rangle$. Formula $x > 1$ is atomic and x is bound to 5 so we simply evaluate this to $5 > 1$, which is true. Refuter has won this play. But if \mathbf{V} chooses $x * x < x + 1$, the next configuration is $\langle M[x \mapsto 5], x * x < x + 1, \mathbf{R} \rangle$ and now Verifier wins since $5 * 5 = 25 \not< 6 = 5 + 1$.

This yields a winning strategy for \mathbf{V} : if player \mathbf{V} always chooses $x * x < x + 1$ whenever a is greater than 1, and chooses $x > 1$ whenever a is at most 1, she wins all plays in the game tree of $\mathbf{G}(M, \phi)$. The existence of a winning strategy for \mathbf{V} establishes that ϕ is true in M .

To summarise, negation in a configuration $\langle M, \neg \psi, \mathbf{C} \rangle$ determines a swap of players and the removal of the negation with next configuration $\langle M, \psi, \bar{\mathbf{C}} \rangle$. Existential quantifiers in a configuration $\langle M, \exists x \psi, \mathbf{C} \rangle$ require binding its quantified variable x to an element a of the model, chosen by player \mathbf{C} , with next configuration $\langle M[x \mapsto a], \psi, \mathbf{C} \rangle$. Conjunction in a configuration $\langle M, \psi_1 \wedge \psi_2, \mathbf{C} \rangle$ means that player $\bar{\mathbf{C}}$ chooses a conjunct ψ_i for the next configuration $\langle M, \psi_i, \mathbf{C} \rangle$. Atomic configurations $\langle M, a, \mathbf{C} \rangle$ are simply evaluated, using the binding information of the model: player \mathbf{C} wins if a is true in M , otherwise player $\bar{\mathbf{C}}$ wins.

Strategies for both players are objects σ that allow them to make necessary

choices for determining continuation plays. For example, Verifier needs to make choices for existential quantifiers in configurations of form $\langle M, \exists x \psi, \mathbf{V} \rangle$, and for conjunctions in configurations of form $\langle M, \psi_1 \wedge \psi_2, \mathbf{R} \rangle$. A strategy σ is winning for a player if all plays played according to the choices offered by strategy σ are won by that player. Since all plays for first-order logic are finite, classical game theory guarantees that games $\mathbf{G}(M, \phi)$ are determined: exactly one of the two players has a winning strategy for that game.

In ordinary set theory ZF the assumption of the Axiom of Choice is equivalent to the following principle

(Correspondence) “Verifier wins game $\mathbf{G}(M, \phi)$ if, and only if, predicate $M \models \phi$ holds.”

holds. So one gets an operational account of truth in first-order logic from the Axiom of Choice.

3.5 Stochastic parity games

In this section we introduce parity games and stochastic parity games. Again we restrict ourselves to only those concepts which are relevant for this thesis, e.g. we define only pure memoryless strategies as they are sufficient for our needs.

Definition 52 ((Stochastic) Parity Game)

A *stochastic parity game* is a tuple $G = ((V, E), (V_0, V_1, V_p), \delta, \alpha)$ where

- (V, E) is a directed graph with non-empty set V ;
- (V_0, V_1, V_p) is a partition of V ;⁶
- the set V_0 is the set of Player 0 configurations;
- V_1 is the set of Player 1 configurations; and
- V_p is the set of probabilistic configurations.
- $\alpha: V \rightarrow [0 \dots k]$ a parity condition which associates a priority $\alpha(v)$ in $[0 \dots k] := \{0, \dots, k\}$ with each configuration v .

⁶One or two sets V_i of the partition may be empty.

A stochastic parity game is simply a *parity game* if $V_p = \emptyset$. The function δ associates with every $v \in V_p$ a distribution $\delta(v)$ over $E(v) = \{v' \mid (v, v') \in E\}$. We assume that $(v, v') \in E$ iff $\delta(v)(v') \neq 0$ and write $\delta(v, v')$ instead of $\delta(v)(v')$. \blacklozenge

If one or two sets in the partition (V_0, V_1, V_p) are empty the stochastic parity game $G = ((V, E), (V_0, V_1, V_p), \delta, \alpha)$ collapses to well known special cases, i.e. one-player game, two-player game, one-player stochastic game, stochastic process. Markov chains can be thought of as stochastic parity games where $V_0 = V_1 = \emptyset$.

Definition 53 (Play)

A *play* of a stochastic parity game $G = ((V, E), (V_0, V_1, V_p), \delta, \alpha)$ is a maximal path in (V, E) , i.e. an infinite path or one that reaches a dead end. \blacklozenge

We write Ω for the set of all plays, and Ω_v for the set of plays that start in location v . A play is *winning* for Player 0 if it is finite and ends in a Player 1 configuration (meaning that this Player 1 configuration has no successor configuration in E) or it is infinite and $\limsup(\alpha(v_i))$ is even. Otherwise, it is winning for Player 1.

Definition 54 (Strategy)

A (*pure memoryless*) *strategy* for Player 0 is a partial function $\sigma: V_0 \rightarrow V$ that chooses a successor for every configuration in V_0 , i.e. $(v, \sigma(v)) \in E$. A play v_0, v_1, \dots is *consistent* with strategy σ if whenever $v_i \in V_0$ we have $v_{i+1} = \sigma(v_i)$. \blacklozenge

Strategies for Player 1 are defined analogously. We denote by Σ and Π the set of all strategies for Player 0 and Player 1, respectively.

Fixing strategies σ and π for Player 0 and Player 1 respectively removes all non-probabilistic, non-deterministic choice from game

$$G = ((V, E), (V_0, V_1, V_p), \delta, \alpha) .$$

This removal determines a Markov chain M whose paths are plays in G consistent with σ and π . Thus, for fixed initial location and strategies $\sigma \in \Sigma$ and $\pi \in \Pi$ for the players, the game specialises to a Markov chain $M_s^{\sigma, \pi}$ for which the probabilities of events are uniquely defined, where events are measurable sets of plays in the game (respectively measurable sets of paths in the corresponding Markov chain). We define $\text{val}_0^{\sigma, \pi}(v)$ to be the measure of plays winning for Player 0 in $M_v^{\sigma, \pi}$ and $\text{val}_1^{\sigma, \pi}(v) = 1 - \text{val}_0^{\sigma, \pi}(v)$. We define

$$\text{val}_0(v) = \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \text{val}_0^{\sigma, \pi}(v)$$

and

$$\text{val}_1(v) = \sup_{\pi \in \Pi} \inf_{\sigma \in \Sigma} \text{val}_1^{\sigma, \pi}(v) .$$

Strategies that achieves these values are called *optimal*.

Definition 55 (Strongly Connected Component)

For (directed) graph $G = (V, E)$, a *strongly connected component* (SCC) is a subgraph (S, E') of G , in which there exists a paths from every vertex $s \in S$ to every other vertex $s' \in S$.

A *maximal* strongly connected component is a SCC of which no proper subset is strongly connected.

Reachability in the directed graph $G = (V, E)$ induces a partial order on its maximal strongly connected components:

$$S \leq S'$$

iff there exists $s \in S$ and $s' \in S'$ and a paths from s to s' in E . The maximal elements of this partial order are called *bottom* strongly connected component (BSCC). \blacklozenge

Definition 56 (Weak Game)

A stochastic parity game $G = ((V, E), (V_0, V_1, V_p), \delta, \alpha)$ is called a *weak stochastic game* if for every maximal, strongly connected component (SCC)

$V' \subseteq V$ either

$$V' \subseteq \alpha \text{ or } V' \cap \alpha = \{\}$$

If $V_p = \{\}$, we call G simply a *weak game*. ◆

Markov chains can be thought of as stochastic weak games where $V_0 = V_1 = \{\}$ and $\alpha = V$.

We state an important result on optimal strategies for stochastic weak games [50, 160]:

Theorem 1

Let $G = ((V, \cdot), \dots)$ be a stochastic weak game and $v \in V$. Then $\text{val}_0(v) + \text{val}_1(v) = 1$. If G is finite, $\text{val}_0(v)$ is computable in $\text{NP} \cap \text{co-NP}$; and optimal strategies exist for both players. If G is a weak game, $\text{val}_0(v)$ is in $\{0, 1\}$ and linear-time computable. ●

One can generalise these results to the setting in which some configurations have pre-seeded game values (in $[0, 1]$ for stochastic weak games, and in $\{0, 1\}$ for weak games). One possibility for such a generalisation of stochastic weak games is to replace every node v with pre-seeded value $\text{val}_0(v) \in [0, 1]$ by a probabilistic node v' which has one transition of probability $\text{val}_0(v)$ to a sink state with value 1, and one transition of probability $1 - \text{val}_0(v)$ to a sink state with value 0. We use such a generalisation of stochastic weak games in Chapter 6.

3.6 Summary of chapter

This chapter presented relevant background on automata and games, which is used in the main part of this thesis. We use the concepts from this background chapter for our Hintikka games in Chapter 5 and for our p-automata in Chapter 6.

The chapter introduced three important types of automata, i.e. word automata, (alternating) tree automata and probabilistic automata. These automata define languages according to their acceptance condition. Acceptance conditions, such as Büchi or parity conditions, are based on the notion

of a run of an automata and can be conveniently expressed as two-player games. The chapter also discussed the close relation between alternating tree automata and the modal μ -calculus as we use similar techniques for the construction of our p-automata in Chapter 6.

The chapter then introduced relevant game formalisms. Hintikka games are used to define the semantics of a logic in an operational fashion as a two-player game. Parity games and stochastic parity games are used to capture, e.g., acceptance of automata, or simulation and bisimulation relations.

4 Completeness for a Fragment of PCTL

With the technical background from the previous two chapters in place we now turn back to the completeness question which we stated rather informally in Section 1.3:

We call an abstraction framework – a class of abstract models, their abstract semantics, and an abstraction via some kind of simulation relation – *complete* for a formula ϕ iff for all concrete models M that satisfy ϕ there is a finite-state abstract model A such that A abstracts M and A satisfies ϕ in the abstract semantics, denoted as $A \preceq M$ and $A \models \phi$ respectively.

We briefly state existing completeness results for linear-time logics and branching-time logics in Section 4.1. Section 4.2 formalises the specific instance of the completeness question, namely completeness for PCTL over Markov chains, which we are concerned with in this thesis. In Section 4.3 we present a first attempt to address completeness for PCTL. This attempt achieves only partial success for a fragment of PCTL, which shows the limitation of “conventional” approaches in the probabilistic context and thus motivates the approach via automata and games taken in Chapters 5 and 6.

4.1 Existing results

In this thesis we are concerned with qualitative, probabilistic model checking. Previous work on complete abstractions was mostly done in the context of non-probabilistic model checking [31, 117, 61, 58, 59, 73, 74]. In particular, the completeness question for qualitative, non-probabilistic model checking of linear-time and branching-time logics have been successfully answered and we summarise the corresponding completeness results in the sections below.

Quantitative model checking is very different from qualitative model checking, both in an probabilistic and non-probabilistic setting. Completeness results there are still an open research question. Actually not even the definition of completeness for quantitative model checking is straight-forward or even unique. In quantitative model checking the answer to a query $M \models \phi$ is in general not Boolean but a real value, such as a cost or performance measure, or a probability. For a probabilistic answer domain quantitative model checking would yield answers of the form $(M \models \phi) \in [0, 1]$. Soundness and completeness then need to be defined as relations between real values. Soundness, for example, could mean that the abstract model yields lower bounds on the probabilities in the concrete model:

$$(A \models \phi) \leq (M \models \phi) \text{ whenever } A \prec M$$

But quantitative soundness could also be defined differently as a non-Boolean answer domain obviously allows for various alternative definitions. Thus it also not clear in general how to define quantitative completeness. In this thesis we focus on qualitative probabilistic model checking and do not address the completeness problem for quantitative model checking.

4.1.1 Completeness for Linear-time Temporal Logic

For linear-time temporal logic (LTL) soundness and completeness of abstraction can be achieved. Yonit Kesten and Amir Pnueli show in [117] that abstraction of infinite Kripke structures by finite-state Kripke structures with fairness constraints via *augmented abstractions* is sound and complete for LTL. They propose a complete abstraction framework which they call *verification by finitary abstraction (VFA)* and *verification by augmented finitary abstraction*. The augmentation with a *progress monitor* is needed to preserve liveness properties in the abstraction. The idea of progress monitors or *ranking monitors* is to make fairness explicit in the concrete system. The monitors are added to the concrete system as finitary parallel components. Those components are carried over into the abstractions where they enable model checks for liveness properties. Although their existence is proved, finding an “adequate augmentation”, i.e. the progress monitors, still requires “ingenuity and deep understanding of the analyzed system” [117].

The notion of fairness is a crucial point in Kesten and Pnueli’s complete-

ness result for LTL. As the standard notion of fairness can not be transferred to Markov chains, the techniques used to achieve completeness for LTL over Kripke structures can not be lifted to LTL over Markov chains [175, 52]. Nevertheless, p-automata – the abstraction framework developed in Chapter 6 of this thesis – can express regular path properties and thus in particular LTL formulae with enveloping probabilistic thresholds. Our p-automata framework is thus not only complete for PCTL but also for probabilistic interpretations of LTL over Markov chains. This is in some sense similar to the alternating tree automata framework which – as described in the next section – achieves completeness not only for branching time but for the whole modal μ -calculus and thus also for LTL.

4.1.2 Completeness for Branching-time Temporal Logic

Dennis Dams and Kedar Namjoshi show in [58] that completeness for branching time can not be achieved by using the same methods as for linear time. More complex abstraction frameworks with more expressive abstract models are needed.

Dams and Namjoshi present such a complete abstraction framework for branching time in [58]. The same article [58] also inspired our incompleteness examples in Section 4.3.2. Dams and Namjoshi’s abstraction framework builds around *focused transition systems (FTS)* as abstract models. These models use a focusing and de-focusing mechanism to achieve finitary abstract models. Through their focus and de-focus mechanism focused transition systems become alternating in their nature, which allows for a finite representation of potentially unbounded fixpoints. In the satisfaction game and the simulation game for focused transition systems this alternation essentially means that for all configurations with focus and de-focus transitions, the players choose alternatingly sets and elements of these sets.

Some weaknesses of focused transition systems are pointed out by Harald Fecher and Michael Huth in [73]. They propose and discuss *disjunctive modal transition systems (DMTS)* [73], *hypermixed Kripke structures* and ranked predicate abstraction [74] as complete abstraction frameworks with advantages over FTS. In particular they argue that it should be easier to construct finite DMTS from given concrete models, and that DMTS would thus be better suited for practical applications.

Expanding on the alternating structure of focused transition systems, Dams and Namjoshi present a complete abstraction framework for the full μ -calculus based on alternating tree automata in [59]. Their automata framework makes the presentation very elegant and offers a unified view on various other abstraction frameworks. The article [59] solves the completeness question for the μ -calculus and thus of course also for branching-time logics CTL and CTL* which are subsets of the μ -calculus. The article illustrates that automata and games are a useful means which can unify verification and abstraction in the same formal framework. In this sense Dams and Namjoshi’s work inspired our automata-based approach in Chapter 6.

4.2 Our setting: Markov chains and PCTL

With discrete-time Markov chains as concrete models and PCTL as our logic of choice, we now define the specific completeness problem which forms the core of this thesis. This notion of completeness is relative to a class of abstract models \mathcal{A} , an abstract PCTL semantics \models^α , and the abstraction between concrete and abstract models $A \preceq M$. We refer to this triad of abstract models, abstract semantics and abstraction as “abstraction framework” subsequently.

A first example of such an abstraction framework for discrete-time Markov chains is (i) finite Markov chains as abstract models; (ii) standard PCTL semantics \models as abstract semantics; and (iii) probabilistic simulation as abstraction. In this example the abstract semantics coincides with the concrete semantics and the abstract models are a sub-class of the concrete models. It is in fact often desirable to design an abstraction framework such that one or more component of the abstraction frameworks relates rather naturally to the concrete models and their semantics.

Definition 57 (Completeness)

An abstraction framework $(\mathcal{A}, \models^\alpha, \preceq)$ is *complete* for a PCTL formula ϕ iff for every Markov chains M that satisfy ϕ there is a model $A \in \mathcal{A}$ such that $A \preceq M$ and $A \models^\alpha \phi$. The abstraction framework is complete for a set of PCTL formulae Γ if it is complete for each $\phi \in \Gamma$. \blacklozenge

Assumption 5

All abstraction frameworks in this thesis are such that the class of abstract models \mathcal{A} is implicitly restricted to contain only finite-state structures. \blacklozenge

Completeness for ϕ thus means that all Markov chains that satisfy ϕ ($M \models \phi$) have a finite-state abstraction that also satisfies ϕ in the abstract semantics \models^α . In this chapter we choose the following abstraction framework:

- abstract models \mathcal{A} are 3-valued Markov chains;
- abstract semantics \models^α is the pessimistic semantics \models^P of Definition 30; and
- abstraction \preceq is probabilistic simulation of Definition 35.

We choose \models^P as abstract semantics since it is, unlike \models° , sound for verifications (see Section 2.5).

4.3 Completeness for a fragment of PCTL via 3-valued unfoldings

Our first attempt to solve the completeness question for PCTL uses 3-valued abstractions, which are a well established method of abstraction for model checking. With 3-valued Markov chains as abstract models we achieve completeness for a sizeable fragment of PCTL but not for the full logic as we show by giving some concrete counterexamples.

Another approach which – from a model-checking perspective – could also have been a good candidate for a first attempt at our completeness problem, is to use Markov decision processes as abstract models. Markov decision processes incorporate non-determinism into the abstract model, which is somehow similar to *hyper-transitions*, which are used in the abstract models of some complete abstraction frameworks for non-probabilistic logics with [77, 74]. However work by Mark Kattenbelt and Michael Huth [112, 111] showed that abstraction by Markov decision processes is incomplete even for formulae for which the 3-valued abstractions presented in this chapter yields completeness. Thus we do not pursue abstraction by Markov decision processes in this thesis.

4.3.1 Complete fragment of PCTL

We now present a complete fragment of PCTL. The fragment consists of PCTL formulae with certain combinations of negation polarity and threshold type. The technical details of this definition, and its characterisation via a normal form are formalised below. We then show that for this fragment the desired finite abstractions can be obtained by unfolding the infinite model up to a bounded height and width.

We achieve completeness for this PCTL fragment by using (finite) unfoldings of Markov chains as defined in Section 2.7. Here we use finite unfoldings as abstractions, which obviously is a most important reduction of the state space if the concrete Markov chain has infinitely many states. If the concrete Markov chain is already finite-state, its finite unfoldings will in general not be smaller but might actually have a larger state space as its construction may introduce duplicate states through the unfolding of loops.

Ideally, one would like to have an abstraction which always decreases the size of the state space, e.g. which reduces infinite models to finite ones and finite models to models with fewer states. Alternatively the abstraction could leave models which are already finite-state unchanged, and thus not increase the size of the state space. Finite unfoldings do not have these properties and there is certainly potential for optimising the size of a finite unfolding, e.g. by taking the bisimulation quotient of its state space.

Although our motivation is abstraction as state space reduction, we do not pursue such an optimisation here. Our primary concern in this thesis is finitary completeness, i.e. the reduction of infinite-state models to finite-state models. Any further reduction (or even an increase) of the size of finite-state models is thus only of secondary interest with regard to this research question.

First we show that Next and Strong Until with $> p$ bounds have “finite” unfoldings of Markov chains as witnesses.

Lemma 8

Let M be a 3-valued Markov chain, $\mathbf{q}, \mathbf{r} \in \mathbf{AP}$ be propositions, and $M \models^{\mathbf{P}} [\alpha]_{>p}$ for $\alpha \in \{\mathbf{X}\mathbf{q}, \mathbf{q}\mathbf{U}\mathbf{r}\}$. There exist $i_0, j_0 \in \mathbb{N}$ with $M_{i,j}^{s_0} \models^{\mathbf{P}} [\alpha]_{>p}$ for all $i \geq i_0$ and $j \geq j_0$. ●

Proof

Let α be \mathbf{Xq} . By assumption $M \models^{\mathbf{P}} [\mathbf{Xq}]_{>p}$. If s_0 has finitely many successors, the claim is obviously true. Otherwise, let t_1, t_2, \dots be the successors of s_0 ordered such that $\mathbf{P}(s_0, t_l) \geq \mathbf{P}(s_0, t_{l+1})$ for every $l \geq 1$. Let t_{m_1}, t_{m_2}, \dots be the sub-sequence of those states t_i with $t_i \models_M^{\mathbf{P}} \mathbf{q}$. Then $M \models^{\mathbf{P}} [\mathbf{Xq}]_{>p}$ implies $\sum_{l=1}^{\infty} \mathbf{P}(s_0, t_{m_l}) > p$. Thus there is some l_0 with $\sum_{l=1}^{l_0} \mathbf{P}(s_0, t_{m_l}) > p$. Let $j_0 = m_{l_0}$. For every $i \geq 1$ and $j \geq j_0$ it is then easily seen that $M_{i,j}^{s_0} \models^{\mathbf{P}} [\mathbf{Xq}]_{>p}$.

Now let α be \mathbf{qUr} . Consider first the case that M is finitely branching. It is simple to see that for all $i \geq 0$ we have

$$\text{Prob}_{M_i^{s_0}}^{\mathbf{P}}(s_0, \mathbf{qUr}) \leq \text{Prob}_{M_{i+1}^{s_0}}^{\mathbf{P}}(s_0, \mathbf{qUr})$$

and that

$$\lim_{i \rightarrow \infty} \text{Prob}_{M_i^{s_0}}^{\mathbf{P}}(s_0, \mathbf{qUr}) = \text{Prob}_M^{\mathbf{P}}(s_0, \mathbf{qUr}) .$$

Hence, for some i_0 we have that $\text{Prob}_{M_{i_0}^{s_0}}^{\mathbf{P}}(s_0, \mathbf{qUr}) > p$ and for every $i \geq i_0$ we have $M_i^{s_0} \models^{\mathbf{P}} [\mathbf{qUr}]_{>p}$.

In the case that M has infinite branching the proof is similar. As before, there is some i_0 such that $M_{i_0}^{s_0} \models^{\mathbf{P}} [\mathbf{qUr}]_{>p}$. We notice that for every $j \in \mathbb{N}$ we have

$$\text{Prob}_{M_{i_0,j}^{s_0}}^{\mathbf{P}}(s_0, \mathbf{qUr}) \leq \text{Prob}_{M_{i_0,j+1}^{s_0}}^{\mathbf{P}}(s_0, \mathbf{qUr})$$

and that

$$\lim_{j \rightarrow \infty} \text{Prob}_{M_{i_0,j}^{s_0}}^{\mathbf{P}}(s_0, \mathbf{qUr}) = \text{Prob}_{M_{i_0}^{s_0}}^{\mathbf{P}}(s_0, \mathbf{qUr}) .$$

Hence, for some j_0 we have $\text{Prob}_{M_{i_0,j_0}^{s_0}}^{\mathbf{P}}(s_0, \mathbf{qUr}) > p$ and the claim follows. ■

In a similar fashion we show that weak Until and Next with $\geq p$ bounds have finite counter-examples.

Corollary 1

Let $M \not\models^{\circ} [\alpha]_{\geq p}$ for $\alpha \in \{\mathbf{Xq}, \mathbf{qWr}\}$ and a 3-valued Markov chain M . Then there exist i_0 and j_0 such that for all $i \geq i_0$ and $j \geq j_0$ we have $M_{i,j}^{s_0} \not\models^{\circ} [\alpha]_{\geq p}$. ●

Proof

For $\alpha \equiv \mathbf{X} \mathbf{q}$ this follows from $[\mathbf{X} \varphi]_{>p} \equiv \neg[\mathbf{X} \neg\varphi]_{\geq 1-p}$ over 2-valued Markov chains and from the duality of the optimistic and pessimistic semantics in 3-valued Markov chains. For $\alpha \equiv \mathbf{q} \mathbf{W} \mathbf{r}$, we similarly exploit that $[\varphi_1 \mathbf{W} \varphi_2]_{\geq p}$ is equivalent to $\neg[\neg\varphi_2 \mathbf{U}(\neg\varphi_1 \wedge \neg\varphi_2)]_{> 1-p}$ over 2-valued Markov chains and again use the duality of the optimistic and pessimistic semantics. \blacksquare

For the abstraction via finite unfoldings we have to carefully distinguish between formulae with $>$ and \geq threshold respectively. To this end we now introduce a finite-state approximation lemma (Lemma 9) for the validity of Until formulae with non-strict (i.e. \geq) probability thresholds at states of labelled Markov chains. As we are dealing with discrete-time Markov chains M (which have countable branching and countable sets of states) we know that if a state s of M satisfies $\phi = [\mathbf{q} \mathbf{U} \mathbf{r}]_{>p}$, then it is sufficient to explore M from s up to a finite depth in order to witness $s \models \phi$. The lemma then says, that a state s of M satisfies a formula $\phi = [\mathbf{q} \mathbf{U} \mathbf{r}]_{\geq p}$ if and only if it satisfies $\phi' = [\mathbf{q} \mathbf{U} \mathbf{r}]_{>p-\frac{1}{n}}$ for each $n > 0$, which again is witnessed by a finite-depth exploration of M . (This lemma will also be crucial in Chapter 5 to prove that our game semantics for PCTL captures exactly the denotational semantics as defined in Section 2.5.)

Lemma 9 (Finite-state approximation)

Let M be a labelled Markov chain (S, \mathbf{P}, L) , let $\mathbf{q}, \mathbf{r} \in \mathbf{AP}$, and $p \in [0, 1]$. Then $s \in \llbracket [\mathbf{q} \mathbf{U} \mathbf{r}]_{\geq p} \rrbracket_M$ iff for all $n \in \mathbb{N}$ there are $k, l \in \mathbb{N}$ such that $s \in \llbracket [\mathbf{q} \mathbf{U} \mathbf{r}]_{>p-\frac{1}{n}} \rrbracket_{M_{k,l}^s}$, where $M_{k,l}^s$ is a finite unfolding. \bullet

Proof

Consider first the case that M is finite-branching. Recall that $\text{Path}(s, \mathbf{q} \mathbf{U} \mathbf{r})$ denotes the set of paths beginning in s that satisfy $\mathbf{q} \mathbf{U} \mathbf{r}$. Let

$$\text{Path}_i(s, \mathbf{q} \mathbf{U} \mathbf{r}) = \text{Path}(s, (\mathbf{q} \mathbf{U}^{\leq i} \mathbf{r}) \wedge \bigwedge_{0 \leq j < i} \neg(\mathbf{q} \mathbf{U}^{\leq j} \mathbf{r})), \quad (4.1)$$

i.e. the paths in which \mathbf{q} holds until location i where \mathbf{r} holds and \mathbf{r} does not hold in locations smaller than i . We set $\text{Path}_0(s, \mathbf{q} \mathbf{U} \mathbf{r})$ to be $\text{Path}(s, \mathbf{q} \mathbf{U}^{\leq 0} \mathbf{r})$.

- For the “if” part, assume that for all $n \in \mathbb{N}$ there is $k \geq 0$ such that $s \in \llbracket [\mathbf{qUr}]_{>p-\frac{1}{n}} \rrbracket_{M_k^s}$. Then, $s \in \llbracket [\mathbf{qUr}]_{>p-\frac{1}{n}} \rrbracket_M$ by the monotonicity of the denotational semantics for “Greater-Than” thresholds. Thus, $s \in \bigcap_{n \in \mathbb{N}} \llbracket [\mathbf{qUr}]_{>p-\frac{1}{n}} \rrbracket_M = \llbracket [\mathbf{qUr}]_{\geq p} \rrbracket_M$.
- For the “only if” part, let $s \in \llbracket [\mathbf{qUr}]_{\geq p} \rrbracket_M$ and $n \in \mathbb{N}$. It suffices to find some $k \geq 0$ with $s \in \llbracket [\mathbf{qUr}]_{>p-\frac{1}{n}} \rrbracket_{M_k^s}$. As $\text{Path}_i(s, \mathbf{qUr})$ is of form $\text{Path}(s, \alpha)$ for a path formula α , that set of paths is measurable. For all $i \neq j$ note that sets $\text{Path}_i(s, \mathbf{qUr})$ and $\text{Path}_j(s, \mathbf{qUr})$ are disjoint. Since

$$\text{Path}(s, \mathbf{qUr}) = \bigcup_{i \geq 0} \text{Path}_i(s, \mathbf{qUr})$$

and as the latter is a disjoint union, we know that

$$\text{Prob}_M(s, \text{Path}(s, \mathbf{qUr})) = \sum_{i \geq 0} \text{Prob}_M(s, \text{Path}_i(s, \mathbf{qUr}))$$

By definition of convergence for that infinite sum, for every $n \in \mathbb{N}$ there exists $k \geq 0$ such that

$$\sum_{i=0}^k \text{Prob}_M(s, \text{Path}_i(s, \mathbf{qUr})) \geq \text{Prob}_M(s, \text{Path}(s, \mathbf{qUr})) - \frac{1}{n}.$$

As $\sum_{i=1}^k \text{Prob}_M(s, \text{Path}_i(s, \mathbf{qUr}))$ equals $\text{Prob}_{M_k^s}(s, \mathbf{qUr})$ we obtain $s \in \llbracket [\mathbf{qUr}]_{>p-\frac{1}{n}} \rrbracket_{M_k^s}$ and we are done.

As M is finitely branching, there exists l such that l is an upper bound on the branching degree for all states in M_k^s . It follows that $\text{Prob}_{M_k^s}(s, \mathbf{qUr}) = \text{Prob}_{M_{k,l}^s}(s, \mathbf{qUr})$.

In the case that M has infinite branching the proof is similar. We have to be more careful in noticing that every path set $\text{Path}_i(s, \mathbf{qUr})$ is still measurable and have to be careful in the way in which we sum up the probability of the set $\text{Path}(s, \mathbf{qUr})$. But this works out since all involved infinite sums are absolute convergent, which establishes that for some k we have $s \in \llbracket [\mathbf{qUr}]_{>p-\frac{1}{n}} \rrbracket_{M_k^s}$. The existence of $M_{k,l}^s$ as required follows from convergence of $\text{Prob}_{M_{k,l}^s}(s, \mathbf{qUr})$ to $\text{Prob}_{M_k^s}(s, \mathbf{qUr})$ in l . ■

Lemma 9 has a dual version, required in the proof of Theorem 4 in Chap-

ter 5:

Corollary 2

For a labelled Markov chain $M = (S, \mathbf{P}, L)$, $\mathbf{q}, \mathbf{r} \in \mathbf{AP}$, and $p \in [0, 1]$ we have $s \notin \llbracket [\mathbf{q} \mathbf{W} \mathbf{r}]_{>p} \rrbracket_M$ iff for all $n \in \mathbb{N}$ there are $k, l \in \mathbb{N}$ with $s \notin \llbracket [\mathbf{q} \mathbf{W} \mathbf{r}]_{\geq p + \frac{1}{n}} \rrbracket_{M_{k,l}^s}$. ●

Proof

This follows from Lemma 9 and the duality of weak and strong Until. We have

$$s \notin \llbracket [\mathbf{q} \mathbf{W} \mathbf{r}]_{>p} \rrbracket_M \text{ iff } s \in \llbracket [\neg \mathbf{r} \mathbf{U}(\neg \mathbf{q} \wedge \neg \mathbf{r})]_{\geq 1-p} \rrbracket_M,$$

since (semantically) $\neg \mathbf{r} \mathbf{U}(\neg \mathbf{q} \wedge \neg \mathbf{r})$ is the negation of $\mathbf{q} \mathbf{W} \mathbf{r}$ by (2.2) on page 40. By Lemma 9, $s \in \llbracket [\neg \mathbf{r} \mathbf{U}(\neg \mathbf{q} \wedge \neg \mathbf{r})]_{\geq 1-p} \rrbracket_M$ holds iff for every $n \in \mathbb{N}$ there is $k, l \in \mathbb{N}$ with $s \in \llbracket [\neg \mathbf{r} \mathbf{U}(\neg \mathbf{q} \wedge \neg \mathbf{r})]_{> 1-p-\frac{1}{n}} \rrbracket_{M_{k,l}^s}$, i.e. $s \notin \llbracket [\mathbf{q} \mathbf{W} \mathbf{r}]_{\geq p + \frac{1}{n}} \rrbracket_{M_{k,l}^s}$. ■

Example 43

Consider the labelled Markov chain M in Figure 4.1 and its finite unfolding $M_2^{s_0}$ in Figure 4.2. Probability $\text{Prob}_M(s_0, \mathbf{q} \mathbf{U} \mathbf{r}) = \frac{1}{2}$ is attained by paths of increasing length as the value of the infinite sum $\sum_{j=1}^{\infty} (\frac{1}{3})^j$. However, for every $n \in \mathbb{N}$ there exists some $i \in \mathbb{N}$ such that $\sum_{j=1}^i (\frac{1}{3})^j > \frac{1}{2} - \frac{1}{n}$ and where that finite sum is attainable in a finite unfolding of M . For example, for $M_2^{s_0}$ in Figure 4.2 the probability of $\mathbf{q} \mathbf{U} \mathbf{r}$ at s_0 is $\frac{4}{9}$ so for every $n < 18$ we have $s_0 \in \llbracket [\mathbf{q} \mathbf{U} \mathbf{r}]_{> \frac{1}{2} - \frac{1}{n}} \rrbracket_{M_2^{s_0}}$. In $M_4^{s_0}$ the probability of $\mathbf{q} \mathbf{U} \mathbf{r}$ at s_0 is $\frac{13}{27}$ and so for every $n < 54$ we have $s_0 \in \llbracket [\mathbf{p} \mathbf{U} \mathbf{q}]_{> \frac{1}{2} - \frac{1}{n}} \rrbracket_{M_4^{s_0}}$. Lemma 9 promises that for every labelled Markov chain there is a similar approximation. ◆

We state and prove the main result of this chapter, the completeness of $\text{PCTL}_{>}$, which is defined below. The GTNMF, i.e. the normal form according to Definition 32, of $\text{PCTL}_{>}$ allows only $[\mathbf{U}]_{>p}$ and $[\mathbf{X}]_{>p}$ type operators. That is, it disallows weak Until and the comparison $\geq p$, which justifies the name $\text{PCTL}_{>}$.

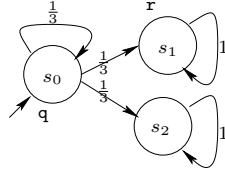


Figure 4.1: A labelled Markov chain M with $s_0 \in \llbracket [\mathbf{q} \cup \mathbf{r}]_{\geq \frac{1}{2}} \rrbracket_M$.

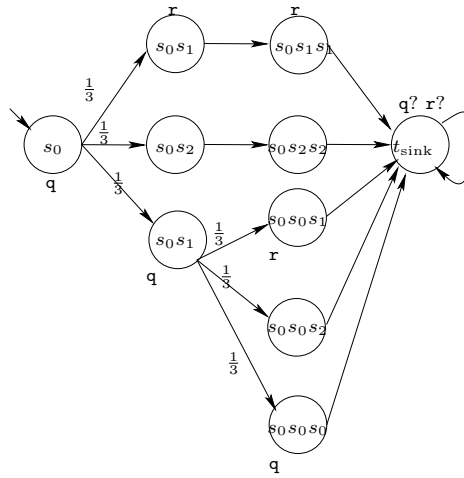


Figure 4.2: The finite unfolding $M_2^{s_0}$ of the labelled Markov chain of Figure 4.1 up to depth 2 for $\mathbf{AP} = \{\mathbf{q}, \mathbf{r}\}$.

Definition 58 (PCTL_>)

The fragment PCTL_> of PCTL is given as all ϕ_{pos} as follows

$$\begin{aligned} \phi_{\text{pos}} &::= \mathbf{q} \mid \neg \mathbf{q} \mid \phi_{\text{pos}} \wedge \phi_{\text{pos}} \mid \phi_{\text{pos}} \vee \phi_{\text{pos}} \mid \neg \phi_{\text{neg}} \mid [\alpha_{\text{pos}}]_{>p} \mid [\alpha_{\text{neg}}]_{<p} \\ \phi_{\text{neg}} &::= \mathbf{q} \mid \neg \mathbf{q} \mid \phi_{\text{neg}} \wedge \phi_{\text{neg}} \mid \phi_{\text{neg}} \vee \phi_{\text{neg}} \mid \neg \phi_{\text{pos}} \mid [\alpha_{\text{neg}}]_{\geq p} \mid [\alpha_{\text{pos}}]_{\leq p} \\ \alpha_{\text{pos}} &::= \mathbf{X} \phi_{\text{pos}} \mid \phi_{\text{pos}} \mathbf{U}^{\leq k} \phi_{\text{pos}} \\ \alpha_{\text{neg}} &::= \mathbf{X} \phi_{\text{neg}} \mid \phi_{\text{neg}} \mathbf{W}^{\leq k} \phi_{\text{neg}} \end{aligned}$$

where $\mathbf{q} \in \text{AP}$, $k \in \mathbb{N} \cup \{\infty\}$ and $p \in [0, 1]$. ◆

Although any finite-state abstraction would be sufficient for completeness we show a stronger result: the abstraction can be chosen as finite unfolding.

Theorem 2 (Completeness of PCTL_>)

Let M be a 3-valued Markov chain with initial state s_0 , ϕ a formula in PCTL_>, and $M \models \phi$. Then there exist i, j such that the finite unfolding $M_{i,j}^{s_0}$ of M pessimistically satisfies ϕ , i.e. $M_{i,j}^{s_0} \models^{\text{P}} \phi$. ●

The finite unfolding is also an abstraction of the Markov chain since $M \preceq M_{i,j}^{s_0}$ by the definition of finite unfolding.

Proof

We strengthen the claim with a dual claim for formulae in the negative part of PCTL_> and for the optimistic semantics: For ϕ in the negative part ϕ_{neg} of PCTL_>, if $s \not\models_M \phi$ then there exist i, j such that $M_{i,j}^s \not\models^{\text{O}} \phi$. We show this extended claim by structural induction on ϕ , simultaneously for all states s .

- Let ϕ be \mathbf{q} . If $s \models_M \mathbf{q}$ then for every $i \geq 0$ and $j \geq 0$ we have $M_{i,j}^s \models^{\text{P}} \mathbf{q}$. Dually, if $s \not\models_M \mathbf{q}$ then for every $i \geq 0$ and $j \geq 0$ we have $M_{i,j}^s \not\models^{\text{O}} \mathbf{q}$.
- For the Boolean connectives $\phi_1 \wedge \phi_2$ and $\phi_1 \vee \phi_2$ and a state s , we take as bounds the maximum of the bounds i_k and j_k for subformulae ϕ_k obtained by induction for state s . These bounds work for the dual case as well.
- For a negation $\varphi = \neg \psi_{\text{neg}}$ and a state s , if $s \models_M \neg \psi_{\text{neg}}$, then $s \not\models_M \psi_{\text{neg}}$. By induction, there are i and j with $M_{i,j}^s \not\models^{\text{O}} \psi_{\text{neg}}$. Thus $M_{i,j}^s \models^{\text{P}} \neg \psi_{\text{neg}}$.

Dually, for a negation $\varphi = \neg\psi_{\text{pos}}$ and a state s , if $s \not\models_M \neg\psi_{\text{pos}}$, then $s \models_M \psi_{\text{pos}}$. By induction, there are i and j with $M_{i,j}^s \models^{\text{P}} \psi_{\text{pos}}$, so $M_{i,j}^s \not\models^{\text{O}} \neg\psi_{\text{pos}}$.

- We now consider the path modalities \mathbf{X} , \mathbf{U} , and \mathbf{W} .
 - For formula $\varphi = [\mathbf{X} \psi_{\text{pos}}]_{>p}$ and a state s such that $s \models_M \varphi$, we treat ψ_{pos} as a proposition that labels the states of M . By (the proof of) Lemma 8, there is some j'_0 such that for every $i \geq 1$ and $j \geq j'_0$ we have $M_{i,j}^s \models^{\text{P}} \varphi$. Now we no longer treat the ψ_i as atoms: Let $t_1, \dots, t_{j'_0}$ be the first j'_0 successors of s . For t_k there exists i_0^k and j_0^k such that if $t_k \models_M \psi_{\text{pos}}$ we have $M_{i_0^k, j_0^k}^{t_k} \models^{\text{P}} \psi_{\text{pos}}$. Let $i_0 = 1 + \max_k(i_0^k)$ and $j_0 = \max(j'_0, \max_k(j_0^k))$. It follows that $M_{i_0, j_0}^s \models^{\text{P}} \varphi$.
 - Let $\varphi = [\mathbf{X} \psi_{\text{neg}}]_{\geq p}$ with $s \not\models_M \varphi$. The proof is similar to the one in the previous item and uses Corollary 1.
 - For $\varphi = [\psi_1 \mathbf{U} \psi_2]_{>p}$, with ψ_1 and ψ_2 in the positive fragment ϕ_{pos} , and a state s with $s \models_M [\psi_1 \mathbf{U} \psi_2]_{>p}$, we initially treat ψ_1 and ψ_2 as propositions that label the states of M . By Lemma 8 there are i'_0 and j'_0 such that for every $i \geq i'_0$ and $j \geq j'_0$ we have $M_{i,j}^s \models^{\text{P}} [\psi_1 \mathbf{U} \psi_2]_{>p}$. Now we no longer treat the ψ_i as atoms: Let t_1, \dots, t_m be all the states appearing in $M_{i'_0, j'_0}^s$. For $\alpha \in \{1, 2\}$ and every t_k there exists $i_0^{k,\alpha}$ and $j_0^{k,\alpha}$ such that if $t_k \models_M \psi_\alpha$ we have $M_{i_0^{k,\alpha}, j_0^{k,\alpha}}^{t_k} \models^{\text{P}} \psi_\alpha$. Let $i_0 = i'_0 + \max_{k,\alpha}(i_0^{k,\alpha})$ and $j_0 = \max(j'_0, \max_{k,\alpha}(j_0^{k,\alpha}))$ (see Figure 4.3). It follows that $M_{i_0, j_0}^s \models^{\text{P}} \varphi$.
 - The proof for $\varphi = [\psi_1 \mathbf{W} \psi_2]_{\geq p}$, with ψ_1 and ψ_2 in the fragment ϕ_{neg} , and a state s such that $s \not\models_M [\psi_1 \mathbf{U} \psi_2]_{\geq p}$ is similar to the one in the previous item and uses Corollary 1.
 - Formula $[\alpha_{\text{neg}}]_{<p}$ is equivalent to $\neg[\alpha_{\text{neg}}]_{\geq 1-p}$ of form $\neg\varphi_{\text{pos}}$. Formula $[\alpha_{\text{pos}}]_{\leq p}$ is equivalent to $\neg[\alpha_{\text{pos}}]_{>1-p}$ of form $\neg\varphi_{\text{neg}}$. Thus this case follows by induction. For example, for state s , we have e.g. $M_{i_0, j_0}^s \models^{\text{P}} [\alpha_{\text{neg}}]_{<p}$ iff $M_{i_0, j_0}^s \models^{\text{P}} \neg[\alpha_{\text{neg}}]_{\geq 1-p}$ iff $M_{i_0, j_0}^s \not\models^{\text{O}} [\alpha_{\text{neg}}]_{\geq 1-p}$. ■

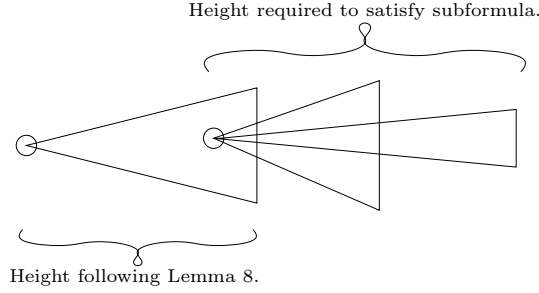


Figure 4.3: Intuitively an unfolding for a sub-formula can be attached to every inner state of the unfolding of the formula. The resulting maximal height is still finite.

4.3.2 Incompleteness of PCTL

We show that 3-valued abstractions are incomplete for full PCTL by giving several counterexamples which consist of a concrete Markov chain M and a PCTL formula φ such that no finite-state model A can exist, which simulates M and for which $A \models^P \varphi$. These examples are strongly inspired by Dams and Namjoshi's work on completeness for Kripke structures and the modal μ -calculus [58].

As these incompleteness results are driven by the structure of formulae ϕ rather than by a specific notion of 3-valued abstraction, we say “ ϕ is incomplete” instead of “3-valued abstractions are incomplete with regard to ϕ ”.

Lemma 10

Not all formulae of form $[\phi \mathbf{U} \psi]_{\geq p}$ and $[\phi \mathbf{W} \psi]_{\geq p}$ are complete. ●

Proof

We consider $[\mathbf{q} \mathbf{U} \mathbf{r}]_{\geq 1}$ and $[\mathbf{q} \mathbf{W} \mathbf{r}]_{\geq 1}$. Let M be the Markov chain illustrated in Figure 2.20 on page 82: The initial state s_0 is labelled \mathbf{q} and is infinitely branching with $\mathbf{P}(s_0, s_{i,1}) > 0$ for all $i \geq 1$; its i -th successor $s_{i,1}$ has probability $\frac{1}{2^i}$, all other transition probabilities are 1; the i -th path $s_0 s_{i,1} \dots s_{i,i}$ consists of $i - 1$ states labelled \mathbf{q} and ends in an absorbing state $s_{i,i}$ labelled \mathbf{r} . The Markov chain M obviously satisfies $[\mathbf{q} \mathbf{U} \mathbf{r}]_{\geq 1}$ and $[\mathbf{q} \mathbf{W} \mathbf{r}]_{\geq 1}$.

Now assume there is a finite-state model A with $n > 0$ states and initial state a_0 , such that $A \models^P \varphi$ and $A \preceq M$. Since A is finite-state there exists a

state a_1 in A (a successor of a_0) which simulates infinitely many successors $s_{i_j,1}$ ($j > 0$) of s_0 in M . Of these states $s_{i_j,1}$ there must be a state $s_{n_0,1}$ which is the starting point of a path $s_{n_0,1} \dots s_{n_0,n_0}$ with $n_0 > n + 1$ states labelled \mathbf{q} before reaching its absorbing \mathbf{r} state. By the definition of simulation this path must be matched by a path $a_1 \dots a_{n_0}$ in A such that $a_j \preceq s_{n_0,j}$ for all $1 \leq j \leq n_0$. Since A is of finite size n there must be a state $a_{j'}$ repeating along this path, and thus there is a loop in A . As the states $s_{n_0,1} \dots s_{n_0,n_0-1}$ are labelled \mathbf{q} , all states of the path $a_1 \dots a_{n_0}$, and on the loop in this path, must be labelled \mathbf{q} or $\mathbf{q}?$. Similarly, as the states $s_{n_0,1} \dots s_{n_0,n_0-1}$ are not labelled \mathbf{r} , for all states a_j of the loop we get $L(a_j, \mathbf{r}) = \text{ff}$ or $L(a_j, \mathbf{r}) = ?$. Now, since $A \models^{\mathbf{P}} \varphi$ by assumption, the states a_j must actually be labelled with \mathbf{q} . (Otherwise, let $\alpha \in \{\mathbf{q} \cup \mathbf{r}, \mathbf{q} \mathbf{W} \mathbf{r}\}$. If one state a_{i_0} in the loop were labelled $\mathbf{q}?$, and so $A^{a_{i_0}} \not\models^{\mathbf{P}} \mathbf{q}$, there would be a finite prefix $a_0 a_1 \dots a_{i_0}$, and thus a measurable cylinder path set with positive probability which does not pessimistically satisfy α . Thus $\text{Prob}_M(a_0, \alpha) < 1$, contradicting $A \models^{\mathbf{P}} \varphi$.)

But now we have an overall contradiction: no model that contains a loop of states labelled \mathbf{q} can simulate M because this would imply (by the weight function of a probabilistic simulation) that M contains an infinite path of states labelled \mathbf{q} , which the given M clearly does not. Hence there cannot be a finite-state model A such that $A \models^{\mathbf{P}} \varphi$ and $A \preceq M$. Further, any such A would satisfy $[\mathbf{q} \mathbf{W} \text{ff}]_{>0}$ which M clearly does not. ■

Lemma 11

Not all formulae of form $[[\phi \cup \psi]_{>p} \mathbf{W} \rho]_{>p'}$ are complete. ●

Proof

Let φ be $[[\mathbf{q} \cup \mathbf{r}]_{>0} \mathbf{W} \text{ff}]_{>0}$ and M be as in Figure 4.4. It is simple to see that $M \models \varphi$. Suppose there is a finite-state model A such that $A \preceq M$ and $A \models^{\mathbf{P}} \varphi$. Let a be the initial state of A such that $a \preceq s_0$. As $A \models^{\mathbf{P}} \varphi$, there is a bottom strongly connected component in A such that every state in this SCC satisfies pessimistically $[\mathbf{q} \cup \mathbf{r}]_{>0}$. By a pigeon-hole principle, we can find a state a' in this SCC that is labelled by r and is simulated by states s_i for infinitely many i . Consider a cycle from a' to itself. This cycle has some fixed length n . As for every $i > 0$ the distance from s_i to s_{i+1} is $i + 1$, this is a contradiction. ■

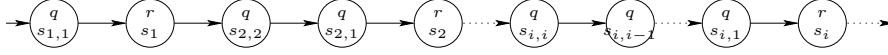


Figure 4.4: A concrete Markov chain M that satisfies $[[\mathbf{q} \mathbf{U} \mathbf{r}]_{>0} \mathbf{W} \mathbf{ff}]_{>0}$.

We note that sub-formula $[\mathbf{q} \mathbf{U} \mathbf{r}]_{>0}$ of the lemma above, is not in and of itself incomplete but belongs to the complete fragment $\text{PCTL}_{>}$.

We can use the concrete Markov chain M from Figure 2.20 on page 82 and a similar proof structure to show the incompleteness of $[\mathbf{X} \phi]_{\geq p}$.

Lemma 12

Not all formulae of form $[\mathbf{X} \phi]_{\geq p}$ are complete. ●

Proof

We consider $\varphi = [\mathbf{X} [\mathbf{q} \mathbf{U} \mathbf{r}]_{>0}]_{\geq 1}$ and the Markov chain M from Figure 2.20 on page 82 which satisfies φ . Again, assume there is a finite-state model A with n states and initial state a_0 , such that $A \models^{\mathbf{P}} \varphi$ and $A \preceq M$.

Since A is finite-state there exists a state a_1 in A (a successor of a_0) which simulates infinitely many successors $s_{i,1}$ of s_0 in M . Since $A \models \varphi$ the state a_1 needs to satisfy $[\mathbf{q} \mathbf{U} \mathbf{r}]_{>0}$. Hence there must be a path $\pi = a_1 \dots a_k$ where the states a_1, \dots, a_{k-1} are labelled \mathbf{q} and a_k is labelled \mathbf{r} . If this path were loop-free, then every one of the infinitely many states $s_{i,1}$ would be a starting point of a path which reaches an \mathbf{r} state after at most k steps. This is a contradiction to the definition of M . Thus π must contain a loop of states labelled \mathbf{q} . But this would force M to contain an infinite path $s_{i,1} \dots$ where all states are labelled \mathbf{q} . Again we have a contradiction because M does not contain such a path. ■

Incompleteness of formulae of the form $[\mathbf{X} [\phi \mathbf{U} \psi]_{>p}]_{\geq p'}$ requires infinite branching, as in the Markov chain in Figure 2.20. For finitely branching Markov chains this type of formula is complete, because then only a finite number of successor states needs to be considered, on each of which sub-formula $[\phi \mathbf{U} \psi]_{>p}$ can be finitely verified. Formula of type $[\phi \mathbf{U} \psi]_{\geq 1}$ and $[\phi \mathbf{W} \psi]_{\geq 1}$ are still incomplete even for finitely branching models (which can be shown with slightly different Until formulae in the proofs). We summarise:

Corollary 3

Full PCTL is incomplete for the abstraction framework considered in this chapter. ●

Our incompleteness proofs above work for every 3-valued finite-state abstraction with pessimistic semantics and a simulation notion \preceq as abstraction that satisfies

1. $L(t, \mathbf{q}) \leq L(s, \mathbf{q})$ for all $\mathbf{q} \in \text{AP}$
2. $\mathbf{P}(s, s') > 0$ implies $\mathbf{P}(t, t') > 0$ for some t' with $s' \preceq t'$
3. $\mathbf{P}(t, t') > 0$ implies $\mathbf{P}(s, s') > 0$ for some s' with $s' \preceq t'$

whenever $t \preceq s$. In their bi-directionality, these three conditions are reminiscent of Larsen and Skou's probabilistic $2/3$ -bisimulation [131] and of Dams and Namjoshi's notion of (mixed) reverse simulation for labelled transition systems [58]: conditions (1) and (2) together constrain the abstract model in terms of the concrete model (and are necessary but not sufficient for sound abstraction as in Lemma 6); conditions (1) and (3) constrain the concrete model in terms of the abstract one (and are necessary for securing our incompleteness results).

4.3.3 Maximality of $\text{PCTL}_{>}$

In Section 4.3.1 we define a benign PCTL fragment $\text{PCTL}_{>}$ and show that it is complete. Section 4.3.2 then shows that full PCTL can not be complete with regard to 3-valued Markov chains as abstract models. We now use these incompleteness results for full PCTL to show that the complete fragment $\text{PCTL}_{>}$ is maximal for 3-valued Markov chains; i.e. every static extension of $\text{PCTL}_{>}$ is incomplete.

Theorem 3

Consider a PCTL fragment κ that contains one of the following combinations of PCTL operators:

- (i) $[\phi \mathbf{W} \psi]_{\geq p}$; or
- (ii) $[\phi \mathbf{U} \psi]_{\geq p}$; or

(iii) $[\mathbf{X}\phi]_{\geq p}$ and $[\phi \mathbf{U}\psi]_{> p}$; or

(iv) $[\phi \mathbf{W}\psi]_{> p}$ and $[\phi \mathbf{U}\psi]_{> p}$.

Then κ is incomplete. ●

Proof

The first three items follow from Lemmas 10 and 12. The last item follows from Lemma 11. ■

4.4 Summary of chapter

In this chapter we formalised the completeness question which we set out to solve in this thesis.

First we mentioned existing completeness results for non-probabilistic logics and highlighted contributions which inspired the work in this thesis. In particular completeness was previously achieved for linear time and branching time. An abstraction formwork based on alternating tree-automata achieved completeness for the full modal μ -calculus and inspired some of the automata-based techniques used in this thesis.

The chapter then presented a first result, which achieves completeness for a fragment $\text{PCTL}_{>}$ of PCTL via 3-valued unfoldings, a pessimistic PCTL semantics, and probabilistic simulation. We showed that full PCTL is incomplete with regard to this abstraction framework and that $\text{PCTL}_{>}$ is maximal. Thus, this chapter showed that to achieve completeness beyond $\text{PCTL}_{>}$ requires a richer abstraction framework, such as automata and games.

5 A Game Semantics for PCTL

In this chapter we develop an operational semantics for the probabilistic branching-time logic PCTL. We define a Hintikka game, as introduced in Section 3.4, which captures the semantics of PCTL as a two-player game. The game is played by two adversarial players who want to prove, respectively refute, the truth of a PCTL formula ϕ on a Markov chain M . The game is played on configurations which are formed from states s of M and subformulae of the disputed query ϕ . The winning player is determined by certain (co-)Büchi acceptance conditions.

This operational definition of semantics for PCTL helps us to gain a deeper understanding of the problems which have to be solved in order to find a complete abstraction framework for full PCTL.

Assumption 6

In this chapter we assume every PCTL formula to be in Greater-Than normal form (GTNF). By Lemma 4 this assumption causes no loss of generality. \blacklozenge

5.1 Game semantics

Let $M = (S, \mathbf{P}, L)$ be a labelled Markov chain over a countable set of atomic propositions AP. For each state $s \in S$ and PCTL formula ϕ we define a 2-person Hintikka game $\mathbf{G}_M(s, \phi)$. As already mentioned, these games are played between two players \mathbf{V} (the Verifier) and \mathbf{R} (the Refuter). After having defined these games and their winning conditions, we show that the game $\mathbf{G}_M(s, \phi)$ is won by player \mathbf{V} iff $s \in \llbracket \phi \rrbracket_M$; and won by player \mathbf{R} iff $s \notin \llbracket \phi \rrbracket_M$. In particular, each game $\mathbf{G}_M(s, \phi)$ is *determined*, i.e. exactly one of the players \mathbf{V} and \mathbf{R} wins that game.

The game $G_M(s, \phi)$ has as set of configurations

$$\text{Cf}_M(s, \phi) = \{\langle s', \psi, \mathbf{C} \rangle \mid s' \in S, \psi \in \text{cl}(\phi), \mathbf{C} \in \{\mathbf{R}, \mathbf{V}\}\}$$

where we define the set of PCTL formulae $\text{cl}(\phi)$, the *closure* of ϕ , further below. Plays in game $G_M(s, \phi)$ are finite or infinite sequences of elements in $\text{Cf}_M(s, \phi)$ starting in the distinguished initial configuration $\langle s, \phi, \mathbf{V} \rangle$. A play is generated by game moves, specified in detail below.

Our game semantics treats Boolean connectives in the same manner as Hintikka games for first-order logic (here we take the point of view of Verifier): proving truth of formula ϕ at state s amounts to winning the game from configuration $\langle s, \phi, \mathbf{V} \rangle$.

- To prove a conjunction we allow Refuter to choose which branch of the conjunction to challenge.
- To prove a disjunction we allow Verifier to choose which branch of the disjunction to prove.
- To handle negation, the game continues in the same state but with the unnegated formula and a swap of the role of players, thus attempting to show that Refuter cannot win from the unnegated formula.

In games for branching-time logics, such as CTL or the μ -calculus, the universal quantification in $\forall X$ is resolved by Refuter's choice of a successor state and the existential quantification in $\exists X$ is resolved by Verifier supplying one successor state [179]. This is familiar from the treatment of quantifiers in Hintikka games for first-order logic. For PCTL, however, things are more complicated. The next operator $[X\phi]_{\bowtie p}$ includes a promised probability $\bowtie p$, e.g. “at least p ” or “more than p ”. Verifier now resolves this “probabilistic quantification” by showing how to re-distribute the required probability between the successors of the current state.

In *qualitative* games, Until operators are resolved by using the logical equivalence

$$\mathbf{qUr} \equiv r \vee (q \wedge X(\mathbf{qUr})) \tag{5.1}$$

and similarly for weak Until operators. The problem in adopting this for PCTL is in the possibility of deferring promises forever. For games in qual-

itative settings this is typically handled by fairness, but for PCTL fairness is not strong enough:

Example 44

PCTL formula $[\mathbf{qU}\mathbf{r}]_{\geq 0.5}$ holds at state s_0 in the labelled Markov chain shown in Figure 4.1 on page 124. In order to prove this we have to appeal to the entire infinite sum $\sum_{i=1}^{\infty} (\frac{1}{3})^i$. Any fairness constraint forcing a transition from s_0 into $\{s_1, s_2\}$ would cut that infinite sum down to a finite one, failing to prove the formula for state s_0 .

However, allowing to defer the satisfaction of the strong Until indefinitely is unsound. The PCTL formula $[\mathbf{qU}\mathbf{r}]_{>0.5}$ does not hold at s_0 but if Verifier is allowed to defer her promises forever, she could supply the promise $\frac{1}{3}$ immediately and promise more than $\frac{1}{6}$ in the future, and thus – by deferring the promise indefinitely – Verifier could *unsoundly* win game $\mathbf{G}_M(s_0, [\mathbf{qU}\mathbf{r}]_{>0.5})$. ◆

To address this problem we add a special ϵ -move as well as acceptance conditions for infinite plays:

- If the probability is at least p , player V should be able to prove that it is greater than $p - \epsilon$ for every $\epsilon > 0$.
- On the other hand, if the probability is strictly less than p then there exists an ϵ for which it is at most $p - \epsilon$.

Thus, player R chooses the ϵ and player V proves in finite time (appealing to Lemma 9) that she can get as close as needed to the threshold. The same intuition (but dual) works for *weak* Until, when the weak Until formula in question does *not* hold.

We next define the notion of the closure $\mathbf{cl}(\phi)$ of a PCTL formula ϕ . This closure contains all PCTL formulae that can occur in game configurations reached from a configuration of the form $\langle s, \phi, \mathbf{C} \rangle$.

Definition 59 (Closure $\mathbf{cl}(\phi)$)

The closure $\mathbf{cl}(\phi)$ of a PCTL formula ϕ is the union of two sets of PCTL formulae.

The first set $\text{cl}_1(\phi)$ is the actual set of PCTL subformulae of ϕ , including ϕ itself.

The second set $\text{cl}_2(\phi)$ consists of all formulae $[\alpha]_{\bowtie p'}$ such that either

- (a) α is $\psi_1 \mathbf{U} \psi_2$, \bowtie is $>$, and for some $p \in [0, 1]$ and $\bowtie' \in \{>, \geq\}$ we have $[\alpha]_{\bowtie' p} \in \text{cl}_1(\phi)$;
- (b) α is $\psi_1 \mathbf{W} \psi_2$, \bowtie is \geq , and for some $p \in [0, 1]$ and $\bowtie' \in \{>, \geq\}$ we have $[\alpha]_{\bowtie' p} \in \text{cl}_1(\phi)$;
- (c) α is $\psi_1 \mathbf{U}^{\leq k'} \psi_2$ and for some $p \in [0, 1]$ and a finite $k > k'$ we have $[\psi_1 \mathbf{U}^{\geq k} \psi_2]_{\bowtie p} \in \text{cl}_1(\phi)$;
- (d) α is $\psi_1 \mathbf{W}^{\leq k'} \psi_2$ and for some $p \in [0, 1]$ and a finite $k > k'$ we have $[\psi_1 \mathbf{U}^{\geq k} \psi_2]_{\bowtie p} \in \text{cl}_1(\phi)$. \blacklozenge

The second set $\text{cl}_2(\phi)$ allows us to replace any probability thresholds p with other values $p' \in [0, 1]$ and finite time bounds with smaller ones, but to allow this in such a manner that it is consistent with the above intuition behind ϵ -moves:

- strong Until formulae with non-strict thresholds may change to strong Until formulae with strict thresholds;
- weak Until formulae with strict thresholds may change to weak Until formulae with non-strict thresholds; and
- the finite time bounds in bounded Untils should be allowed to decrease.

The difference between the Until and weak Untils stems from their duality: the negation of a weak Until formula is a (strong) Until formula and vice-versa as seen, e.g., in (2.2) on page 40. Thus, a weak Until formula with strict threshold is the negation of a (strong) Until formula with non-strict threshold. When Refuter is trying to disprove a weak Until formula with strict threshold, Refuter is in fact trying to *prove* the dual (strong) Until formula with non-strict threshold.

Example 45

Consider the following formula:

$$\phi = [(r \wedge [\mathbf{X}[(p \wedge \neg r) \mathbf{W}(q \wedge \neg r)]_{\geq 1}]_{> 0}) \mathbf{W} \text{ff}]_{> 0} \quad (5.2)$$

Intuitively, ϕ says there is an infinite path labelled by r such that every state on this path has a successor for which $p \mathbf{W} q$ holds on (almost) all paths and r does not hold along the paths that witness $p \mathbf{W} q$. Let $\alpha = (p \wedge \neg r) \mathbf{W} (q \wedge \neg r)$, $\beta = \mathbf{X}[\alpha]_{>0}$, and $\gamma = (r \wedge [\beta]_{>0}) \mathbf{W} \text{ff}$. The closure of ϕ is:

$$\text{cl}(\phi) = \left\{ \begin{array}{l} \phi, [\gamma]_{\geq b}, \text{ff}, (r \wedge [\beta]_{>0}), \\ [\beta]_{>0}, [\alpha]_{\geq b}, (p \wedge \neg r), \\ p, \neg r, r, (q \wedge \neg r), q \end{array} \middle| b \in [0, 1] \right\}$$

As γ appears in ϕ with a strict threshold, it is in the closure of ϕ with its original threshold as well as with all possible non-strict thresholds. As α appears in ϕ with a non-strict threshold, it appears in the closure of ϕ only with non-strict thresholds.

Similarly, for formula $\phi = [q \mathbf{U} r]_{\geq \frac{1}{2}}$ we have

$$\text{cl}(\phi) = \{\phi, q, r, [q \mathbf{U} r]_{>b} \mid b \in [0, 1]\}. \quad (5.3)$$

As ϕ is a strong Until with non-strict threshold it is part of $\text{cl}_1(\phi)$ and for every possible threshold b its strict counterpart $[q \mathbf{U} r]_{>b}$ is in $\text{cl}_2(\phi)$. \blacklozenge

Subsequently, we write $\bar{\mathbf{C}}$ for the player other than \mathbf{C} , i.e. $\bar{\mathbf{V}} = \mathbf{R}$ and $\bar{\mathbf{R}} = \mathbf{V}$. The possible moves of game $\mathbf{G}_M(s_0, \phi)$ are defined through the moves of games $\mathbf{G}_M(s, \psi)$ by structural induction on $\psi \in \text{cl}(\phi)$, simultaneously for all $s \in S$.

M1. At configurations $\langle s, [\alpha]_{>1}, \mathbf{C} \rangle$, player $\bar{\mathbf{C}}$ wins

M2. At configurations $\langle s, [\alpha]_{\geq 0}, \mathbf{C} \rangle$, player \mathbf{C} wins

We may therefore assume that in subsequent moves configurations of the form $\langle s, [\alpha]_{\bowtie p}, \mathbf{C} \rangle$ never satisfy that $\bowtie p$ equals ≥ 0 or > 1 .

M3. At configurations $\langle s, q, \mathbf{C} \rangle$:

- player \mathbf{C} wins if $s \in L(q)$
- player $\bar{\mathbf{C}}$ wins if $s \notin L(q)$

M4. At configuration $\langle s, \neg\psi, \mathbf{C} \rangle$, the next configuration is $\langle s, \psi, \bar{\mathbf{C}} \rangle$

So move M4 removes the negation from the formula but also swaps the role of players.

M5. At configuration $\langle s, \psi_1 \wedge \psi_2, \mathcal{C} \rangle$, player $\bar{\mathcal{C}}$ chooses as next configuration either $\langle s, \psi_1, \mathcal{C} \rangle$ or $\langle s, \psi_2, \mathcal{C} \rangle$

At configuration $\langle s, \psi_1 \vee \psi_2, \mathcal{C} \rangle$, player \mathcal{C} chooses as next configuration either $\langle s, \psi_1, \mathcal{C} \rangle$ or $\langle s, \psi_2, \mathcal{C} \rangle$

For conjunctions player $\bar{\mathcal{C}}$ chooses a conjunct and the game continues with that conjunct instead of the conjunction. Similarly, for disjunctions player \mathcal{C} chooses a disjunct to continue the game.

M6. At configuration $\langle s, [X \psi]_{\bowtie p}, \mathcal{C} \rangle$, player \mathcal{C} chooses a subset $Y \subseteq S$ satisfying $P(s, Y) \bowtie p$; then player $\bar{\mathcal{C}}$ chooses some $s' \in Y$:

- if $P(s, s') = 0$, player $\bar{\mathcal{C}}$ wins
- otherwise, $P(s, s') > 0$ and the next configuration is $\langle s', \psi, \mathcal{C} \rangle$

Move M6 is well defined. There is a non-empty set Y with $P(s, Y) \bowtie p$ as $p \in [0, 1]$, $P(s, \cdot)$ has mass one, and $\bowtie p$ is neither equal to > 1 nor to ≥ 0 .

M7. At configuration $\langle s, [\psi_1 \mathbf{U} \psi_2]_{\geq p}, \mathcal{C} \rangle$, player $\bar{\mathcal{C}}$ chooses some $n \in \mathbb{N}$ such that $p - \frac{1}{n} \geq 0$ with resulting next configuration $\langle s, [\psi_1 \mathbf{U} \psi_2]_{> p - \frac{1}{n}}, \mathcal{C} \rangle$

In move M7 such a choice is possible since p cannot be 0. The intuition is that $[p, 1] = \bigcap_{n \in \mathbb{N}} (p - \frac{1}{n}, 1]$ so this behaves like a *universal* quantification over $n \in \mathbb{N}$.

M8. Dually, at configuration $\langle s, [\psi_1 \mathbf{W} \psi_2]_{> p}, \mathcal{C} \rangle$, player \mathcal{C} chooses $n \in \mathbb{N}$ such that $p + \frac{1}{n} \leq 1$ with resulting next configuration $\langle s, [\psi_1 \mathbf{W} \psi_2]_{\geq p + \frac{1}{n}}, \mathcal{C} \rangle$

In move M8 such a choice is possible since $p < 1$. The intuition is that a weak Until with a $>$ threshold is the dual of a strong Until with a \geq threshold (based on (2.2)), so it is like an *existential* quantification over $n \in \mathbb{N}$.

M9. At configuration $\langle s, [\alpha]_{\bowtie p}, \mathcal{C} \rangle$ where either α is $\psi_1 \mathbf{U} \psi_2$ and \bowtie is $>$; or α is $\psi_1 \mathbf{W} \psi_2$ and \bowtie is \geq

- player \mathcal{C} is able to move to next configuration $\langle s, \psi_2, \mathcal{C} \rangle$
- if player \mathcal{C} did not move, player $\bar{\mathcal{C}}$ is able to move to next configuration $\langle s, \psi_1, \mathcal{C} \rangle$
- if neither player moved above, the play must proceed as follows:

Player C chooses a sub-distribution $d: S \rightarrow [0, 1]$ such that

$$\sum_{s' \in S} d(s') \geq p \quad (5.4)$$

$$\forall s' \in S: d(s') \leq P(s, s') \quad (5.5)$$

Next, player \bar{C} chooses some state $s' \in S$ with $d(s') > 0$ and the next configuration is $\langle s', [\alpha]_{\bowtie d(s') \cdot P(s, s')^{-1}}, C \rangle$.

In move M9, sub-distribution d has positive mass, is bounded pointwise by the probability distribution $P(s, \cdot)$, and specifies the re-distribution of promise $\bowtie p$ into promised probabilities at successor states. Since $d(s') > 0$, we also have $d(s') \cdot P(s, s')^{-1} \in [0, 1]$ in move M9 by (5.5).

M10. At configuration $\langle s, [\alpha]_{>p}, C \rangle$ where α is either $\psi_1 U^{\leq k} \psi_2$ or $\psi_1 W^{\leq k} \psi_2$ with $k \in \mathbb{N}$:

- if $k = 0$ and α is $\psi_1 U^{\leq k} \psi_2$, the next configuration is $\langle s, \psi_2, C \rangle$
- if $k = 0$ and α is $\psi_1 W^{\leq k} \psi_2$, player C chooses as next configuration either $\langle s, \psi_1, C \rangle$ or $\langle s, \psi_2, C \rangle$
- if $k > 0$, the moves are defined as in M9 above; except when the last item of M9 applies, in which case the counter k in α is decreased to $k - 1$ for that next configuration $\langle s', [\alpha]_{\bowtie d(s') \cdot P(s, s')^{-1}}, C \rangle$

In move M10, a bounded Until with bound 0 has to realise ψ_2 right away; and a bounded weak Until with bound zero has to realise at least one of ψ_1 or ψ_2 right away.

A finite play is won as explained in M1-M10 above. In most moves, the play either ends or moves to configurations where the formula is a *proper* sub-formula in the closure. In a configuration with strong Until with non-strict threshold or weak Until with strict threshold the next configuration changes from non-strict to strict threshold or vice versa. In a configuration with strong Until with strict threshold or weak Until with non-strict threshold the next configuration has the same path formula and threshold type, or moves to a proper sub-formula.

It follows that every infinite play ends with an infinite suffix of configurations that are

A1. all of the form $\langle s_i, [\psi_1 \text{ W } \psi_2]_{\geq p_i}, \mathbf{C} \rangle$ or

A2. all of the form $\langle s_i, [\psi_1 \text{ U } \psi_2]_{> p_i}, \mathbf{C} \rangle$

Configurations of these suffixes are either labelled by strong Until with strict threshold or weak Until with non-strict threshold, where the states and the exact probability threshold may still change, but where neither the player \mathbf{C} nor the sub-formulae ψ_1 and ψ_2 change.

Definition 60 (Acceptance conditions)

Player \mathbf{V} wins all infinite plays with an infinite suffix either of type A1 above with $\mathbf{C} = \mathbf{V}$, or of type A2 above with $\mathbf{C} = \mathbf{R}$. Player \mathbf{R} wins all other infinite plays: those with an infinite suffix either of type A1 when $\mathbf{C} = \mathbf{R}$, or of type A2 when $\mathbf{C} = \mathbf{V}$. \blacklozenge

These winning conditions are Büchi acceptance conditions, and so our games are known to be determined [140]. We use the notion of strategy for player \mathbf{C} informally. Such strategies contain, for each configuration of a game, at most one set of choices as required by the applicable move from M1-M10.

Example 46

Consider the game $\mathbf{G}_M(s_0, [\mathbf{q} \text{ U } \mathbf{r}]_{\geq 0.5})$, where M is as in Figure 4.1 on page 124, and let $\alpha = \mathbf{q} \text{ U } \mathbf{r}$. The initial configuration is $\langle s_0, [\alpha]_{\geq 0.5}, \mathbf{V} \rangle$. In the first move player \mathbf{R} chooses an $n \in \mathbb{N}$ and the next configuration is $\langle s_0, [\alpha]_{> 0.5 - \frac{1}{n}}, \mathbf{V} \rangle$. Then, as long as the play $\Gamma_0 \Gamma_1 \dots$ remains in configurations Γ_i of the form $\langle s_0, [\alpha]_{> p_i}, \mathbf{V} \rangle$, player \mathbf{V} is going to choose the sub-distribution d with constant values $d(s_2) = 0$ and $d(s_1) = \frac{1}{3} - \frac{1}{2n}$, and dynamic value $d(s_0) = p_i - d(s_1)$. A simple calculation shows that as long as player \mathbf{R} chooses s_0 as the next state (clearly, if she chooses s_1 she is going to lose as $s_1 \in L(r)$) the promised probability $> p_i$ is going to decrease

according to the following sequence:

$$\begin{aligned} p_0 &= 0.5 - \frac{1}{n}, \\ p_1 &= 0.5 - \frac{3}{2n}, \\ p_2 &= 0.5 - \frac{6}{2n}, \\ p_3 &= 0.5 - \frac{15}{2n}, \end{aligned}$$

and in general

$$p_i = 0.5 - \frac{3^i + 3}{4n} \text{ for } i \in \mathbb{N}.$$

When p_i decreases below $\frac{1}{3}$ (and there is some $i \in \mathbb{N}$ for which this happens), player V still chooses d with $d(s_2) = 0$ as above but now chooses $d(s_1) = p_i$ and $d(s_0) = 0$, thereby forcing player R to move to s_1 and lose. This describes a winning strategy for player V in game $\mathbf{G}_M(s_0, [\mathbf{q} \mathbf{U} \mathbf{r}]_{\geq 0.5})$. ♦

Example 47

Although the choice of d in Example 46 may seem arbitrary, it meshes well with the use of Lemma 9. Consider again the game $\mathbf{G}_M(s_0, [\alpha]_{\geq \frac{1}{2}})$ from Example 46. Suppose that in the first move player R chooses $9 \in \mathbb{N}$, and the next configuration is $\langle s_0, [\alpha]_{> \frac{7}{18}}, \mathbf{V} \rangle$. Since for $M_2^{s_0}$ in Figure 4.2 on page 124, $\text{Prob}_{M_2^{s_0}}(s_0, \alpha) = \frac{4}{9} > \frac{7}{18}$, player V can use $M_2^{s_0}$ to guide her choices. In $M_2^{s_0}$ we have $\text{Prob}_{M_2^{s_0}}(s_0 s_1, \alpha) = 1$ and $\text{Prob}_{M_2^{s_0}}(s_0 s_0, \alpha) = \frac{1}{3}$. Player V uses the gap of $\frac{1}{18}$ and re-distributes it between the successors of s_0 . She can choose, for example, $d(s_1) = \frac{1}{3} - \frac{1}{54}$ and $d(s_0) = \frac{1}{9} - \frac{1}{54}$. The next possible configurations are then $\langle s_1, [\alpha]_{> \frac{17}{18}}, \mathbf{V} \rangle$ and $\langle s_0, [\alpha]_{> \frac{5}{18}}, \mathbf{V} \rangle$. Player V identifies the resulting states with those obtained in $M_2^{s_0}$, here $s_0 s_1$ and $s_0 s_0$ respectively. As $s_0 s_1 \in \llbracket r \rrbracket_{M_2^{s_0}}$ the first is clearly a winning configuration. From $\langle s_0, [\alpha]_{> \frac{5}{18}}, \mathbf{V} \rangle$ and the corresponding location $s_0 s_0$ in $M_2^{s_0}$, player V notices that $\text{Prob}_{M_2^{s_0}}(s_0 s_0 s_1, \alpha) = 1$ and chooses $d(s_1) = \frac{5}{18}$. The next configuration is $\langle s_1, [\alpha]_{> \frac{15}{18}}, \mathbf{V} \rangle$ (with corresponding $s_0 s_0 s_1$ in $M_2^{s_0}$) and won by supplying r . ♦

Definition 61

1. A strategy σ for player \mathbf{C} in game $\mathbf{G}_M(s, \phi)$ is *winning from a configuration* Γ in that game iff player \mathbf{C} wins all plays beginning in configuration Γ when player \mathbf{C} plays according to her strategy σ – regardless of how player $\bar{\mathbf{C}}$ plays.
2. Player \mathbf{C} *wins* game $\mathbf{G}_M(s, \phi)$ iff player \mathbf{C} has a strategy that is winning from configuration $\langle s, \phi, \mathbf{V} \rangle$. \blacklozenge

We can now formalise the main theorem of this chapter, i.e. that the denotational semantics of PCTL is captured exactly by the existence of winning strategies in games $\mathbf{G}_M(s, \phi)$.

Theorem 4 (Equivalence of game and semantics)

Let $M = (S, \mathbf{P}, L)$ be a labelled Markov chain over \mathbf{AP} , $s \in S$, and ϕ a PCTL formula. Then we have:

1. $s \in \llbracket \phi \rrbracket_M$ iff player \mathbf{V} wins game $\mathbf{G}_M(s, \phi)$; and
2. $s \notin \llbracket \phi \rrbracket_M$ iff player \mathbf{R} wins game $\mathbf{G}_M(s, \phi)$.

In particular, game $\mathbf{G}_M(s, \phi)$ is determined. \bullet

Proof

Given PCTL formula ϕ , we show the two claims by structural induction on the PCTL formulae ψ in the closure of ϕ , simultaneously on all states of M . Since exactly one of $s \in \llbracket \psi \rrbracket_M$ and $s \notin \llbracket \psi \rrbracket_M$ holds, it suffices to show both items in Theorem 4 for such a ψ in their “only if” versions, which we do by splitting ψ into six cases:

Case 1. The cases when ψ equals \mathbf{tt} or \mathbf{ff} are trivial. For example, no state satisfies \mathbf{ff} and all plays beginning in $\langle s, \mathbf{ff}, \mathbf{V} \rangle$ are won by player \mathbf{R} . So we may implicitly assume in subsequent cases that $\boxtimes p$ equals neither > 1 nor ≥ 0 .

Case 2. The cases when ψ equals \mathbf{q} , $\neg\psi_1$, $\psi_1 \wedge \psi_2$ and $\psi_1 \vee \psi_2$ are proved as in the case of Hintikka games for propositional logic. We illustrate this for the case of conjunction and player \mathbf{V} :

- Let $s \in \llbracket \psi_1 \wedge \psi_2 \rrbracket_M$. Then $s \in \llbracket \psi_i \rrbracket_M$ for $i = 1, 2$. By induction, there is a winning strategy w_i for player **V** from configuration $\langle s, \psi_i, \mathbf{V} \rangle$ for $i = 1, 2$. Consider the strategy w for player **V** which composes her strategies w_1 and w_2 as follows: at configuration $\langle s, \psi, \mathbf{V} \rangle$, player **R** has to choose as next configuration some $\langle s, \psi_i, \mathbf{V} \rangle$ with $i = 1, 2$. But then player **V** simply responds according to her winning strategy w_i . This describes a winning strategy w for player **V** from configuration $\langle s, \psi, \mathbf{V} \rangle$.
- Let $s \notin \llbracket \psi_1 \wedge \psi_2 \rrbracket_M$. Then there is some $j \in \{1, 2\}$ such that $s \notin \llbracket \psi_j \rrbracket_M$. By induction, there is a winning strategy w_j for player **R** from configuration $\langle s, \psi_j, \mathbf{V} \rangle$. Consider the strategy w for player **R** which composes his strategy w_j with an initial choice as follows: at configuration $\langle s, \psi, \mathbf{V} \rangle$, player **R** simply chooses $\langle s, \psi_j, \mathbf{V} \rangle$ as next configuration and then plays according to her winning strategy w_j . This describes a winning strategy w for player **R** from configuration $\langle s, \psi, \mathbf{V} \rangle$.

Case 3. The case when ψ equals $[\mathbf{X} \psi_1]_{\bowtie p}$, where $\bowtie \in \{\geq, >\}$:

- Let $s \in \llbracket [\mathbf{X} \psi_1]_{\bowtie p} \rrbracket_M$. Let Y be the set of states s' such that $P(s, s') > 0$ and $s' \in \llbracket \psi_1 \rrbracket_M$. From the latter and induction we infer that player **V** has a winning strategy $w_{s'}$ for the configuration $\langle s', \psi_1, \mathbf{V} \rangle$, for all $s' \in Y$. We construct from all of these $w_{s'}$ a winning strategy w for player **V** from configuration $\langle s, [\mathbf{X} \psi_1]_{\bowtie p}, \mathbf{V} \rangle$ as follows: Since $s \in \llbracket [\mathbf{X} \psi_1]_{\bowtie p} \rrbracket_M$, we know that $P(s, Y) \bowtie p$ holds and that Y is non-empty as $\bowtie p$ is not ≥ 0 . So at configuration $\langle s, [\mathbf{X} \psi_1]_{\bowtie p}, \mathbf{V} \rangle$ player **V** chooses this set Y . Now no matter what next configuration $\langle s', \psi_1, \mathbf{V} \rangle$ player **R** chooses, we have $s' \in Y$ and so player **V** plays according to her winning strategy $w_{s'}$. In particular, w is a winning strategy for player **V** from configuration $\langle s, [\mathbf{X} \psi_1]_{\bowtie p}, \mathbf{V} \rangle$.
- Let $s \notin \llbracket [\mathbf{X} \psi_1]_{\bowtie p} \rrbracket_M$. Player **V** must choose a set Y such that $P(s, Y) \bowtie p$. In making this choice, player **V** would – by induction – lose from configuration $\langle s, [\mathbf{X} \psi_1]_{\bowtie p}, \mathbf{V} \rangle$ if Y contained some s' with $s' \notin \llbracket \psi_1 \rrbracket_M$ (for then player **R** could respond with configuration $\langle s', \psi_1, \mathbf{V} \rangle$ and win the resulting game). Dually, player **V** can only increase her chances of winning from configuration $\langle s, [\mathbf{X} \psi_1]_{\bowtie p}, \mathbf{V} \rangle$ if she adds to Y all states s' with $s' \in \llbracket \psi_1 \rrbracket_M$ and $P(s, s') > 0$. Finally, player **V** has no incentive

to add an $s' \in \llbracket \psi_1 \rrbracket_M$ to Y if $P(s, s') = 0$: this does not contribute to $P(s, Y) \bowtie p$ and only exposes player V to a threat of player R to move to s' . To summarise, if player V has a winning strategy from that configuration, then she also has a winning strategy from that same configuration where she chooses Y as in the previous item. But then $s \notin \llbracket [\text{X } \psi_1]_{\bowtie p} \rrbracket_M$ means that $P(s, Y) \bowtie p$ is false. So player V can only choose a set Y for which player R can respond with a winning strategy.

Case 4. The cases when ϕ equals $[\alpha]_{\geq p}$ where α is $\psi_1 \text{ U } \psi_2$:

- Let $s \in \llbracket \phi \rrbracket_M$. Then $\text{Prob}_M(s, \alpha) \geq p$ and so for each $n \in \mathbb{N}$ with $p - \frac{1}{n} \geq 0$ we have $\text{Prob}_M(s, \alpha) > p - \frac{1}{n}$ and $s \in \llbracket [\alpha]_{> p - \frac{1}{n}} \rrbracket_M$. By induction, player V has a winning strategy w_n from configuration $\langle s, [\alpha]_{> p - \frac{1}{n}}, V \rangle$, for each such $n \in \mathbb{N}$. Player V can synthesise from these countably many strategies a winning strategy w from configuration $\langle s, \phi, V \rangle$ as follows: if player R chooses $n \in \mathbb{N}$, then the next configuration is $\langle s, [\alpha]_{> p - \frac{1}{n}}, V \rangle$ and player V plays according to w_n .
- Let $s \notin \llbracket \phi \rrbracket_M$. Then $\text{Prob}_M(s, \alpha) < p$. Thus, there is some $n_0 \in \mathbb{N}$ with $\text{Prob}_M(s, \alpha) \leq p - \frac{1}{n_0} < 1$. But then $\text{Prob}_M(s, \alpha) \not> p - \frac{1}{n_0}$ implies $s \notin \llbracket [\alpha]_{> p - \frac{1}{n_0}} \rrbracket_M$. By induction, player R has a winning strategy w_{n_0} from configuration $\langle s, [\alpha]_{> p - \frac{1}{n_0}}, V \rangle$. So player R gets a winning strategy w from configuration $\langle s, \phi, V \rangle$ by first choosing that n_0 and then playing according to w_{n_0} .

Case 5. The cases when ϕ equals $[\alpha]_{> p}$ where α is $\psi_1 \text{ W } \psi_2$:

- Let $s \in \llbracket \phi \rrbracket_M$. Then $\text{Prob}_M(s, \alpha) > p$. Thus, there is some $n_0 \in \mathbb{N}$ with $p + \frac{1}{n_0} \leq 1$ and $\text{Prob}_M(s, \alpha) \geq p + \frac{1}{n_0}$. But then $\text{Prob}_M(s, \alpha) \geq p + \frac{1}{n_0}$ implies $s \in \llbracket [\alpha]_{\geq p + \frac{1}{n_0}} \rrbracket_M$. By induction, player V has a winning strategy w_{n_0} from configuration $\langle s, [\alpha]_{\geq p + \frac{1}{n_0}}, V \rangle$. So player V gets a winning strategy w from configuration $\langle s, \phi, V \rangle$ by first choosing that n_0 and then playing according to w_{n_0} .
- Let $s \notin \llbracket \phi \rrbracket_M$. Then $\text{Prob}_M(s, \alpha) \leq p$. Thus, for every $n \in \mathbb{N}$ with $p + \frac{1}{n} \leq 1$ we have $\text{Prob}_M(s, \alpha) < p + \frac{1}{n}$. By induction, player R has a winning strategy w_n from configuration $\langle s, [\alpha]_{> p + \frac{1}{n}}, V \rangle$, for each

$n \in \mathbb{N}$ with $p + \frac{1}{n} \leq 1$. Player R can synthesise from these countable strategies a winning strategy from configuration $\langle s, \phi, \mathbf{V} \rangle$ as follows: if player V chooses $n \in \mathbb{N}$, then the next configuration is $\langle s, [\alpha]_{>p+\frac{1}{n}}, \mathbf{V} \rangle$ and player R plays according to w_n .

Case 6. The cases when ϕ equals $[\alpha]_{\bowtie p}$ where either

- (a) α is $\psi_1 \mathbf{U} \psi_2$ and \bowtie is $>$
 - (b) α is $\psi_1 \mathbf{W} \psi_2$ and \bowtie is \geq or
 - (c) α is $\psi_1 \mathbf{U}^{\leq k} \psi_2$ or $\psi_1 \mathbf{W}^{\leq k} \psi_2$ with $k \in \mathbb{N}$ and \bowtie is either $>$ or \geq :
- Let $s \in \llbracket \phi \rrbracket_M$.

The formula α is logically equivalent to $\psi_2 \vee (\psi_1 \wedge \mathbf{X} \alpha)$ and in case that α is bounded the bound decreases by 1. It follows that it is either the case that $s \in \llbracket \psi_2 \rrbracket_M$ or $s \in \llbracket \psi_1 \wedge [\mathbf{X} \alpha]_{\bowtie p} \rrbracket_M$. In the first case, player V chooses to move to configuration $\langle s, \psi_2, \mathbf{V} \rangle$ and by induction she has a winning strategy from this configuration. In the second case, by induction there is a winning strategy for player V from configuration $\langle s, \psi_1, \mathbf{V} \rangle$, so if player R chooses to go to this configuration, player V wins. If player R does not move to ψ_1 , then M9 demands that player V chooses a sub-distribution $d : S \rightarrow [0, 1]$ satisfying (5.4) and (5.5). By assumption $s \in \llbracket [\mathbf{X} \alpha]_{\bowtie p} \rrbracket_M$. Let T be the set of states t such that $\text{Prob}_M(t, \alpha) > 0$ and $P(s, t) > 0$. Player V chooses d such that $d(s') = 0$ for all $s' \in S \setminus T$.

So it suffices to specify d on set T . For that, let $p' = \sum_{t \in T} P(s, t) \cdot \text{Prob}_M(t, \alpha)$.

- Consider the case that \bowtie is $>$. By assumption $p' > p$. In the case that $p = 0$, V chooses some state $t \in T$ such that $\text{Prob}_M(t, \alpha) > 0$, V sets $d(t) = \text{Prob}_M(t, \alpha) \cdot P(s, t)$, and $d(t') = 0$ for all $t' \neq t$. In the case that $p > 0$, let δ be $p' - p$. We are going to distribute this gap δ between all the states in T according to the distribution $P(s, \cdot)$. That is, for all $t \in T$

$$d(t) = \max(0, (\text{Prob}_M(t, \alpha) - \delta)P(s, t))$$

In case that $\text{Prob}_M(t, \alpha) \leq \delta$ we thus have $d(t) = 0$ (and so effectively remove t from set T above). As

$$p' = \sum_{t \in S} \text{Prob}_M(t, \alpha) P(s, t) \text{ and } p > 0$$

there must be at least one state t such that $\text{Prob}_M(t, \alpha) \geq p'$ and hence $\text{Prob}_M(t, \alpha) - \delta > 0$, implying $d(t) > 0$. It follows that $\sum_{t \in T} d(t) \geq p' - \delta \geq p$.

- Consider the case that \bowtie is \geq . By assumption $p' \geq p$. Let δ be $p' - p$. For all $t \in T$, let

$$d(t) = \max(0, \text{Prob}_M(t, \alpha) - \delta \cdot P(s, t)) .$$

Again, if $\text{Prob}_M(t, \alpha) \leq \delta$ we set $d(t) = 0$.

This completes the specification of sub-distribution d chosen by player **V**.

Now regardless of the choice of player **R**, the next configuration is $\langle t, [\alpha]_{\bowtie p'}, \mathbf{V} \rangle$ such that $t \in \llbracket [\alpha]_{\bowtie p'} \rrbracket_M$. So player **V** maintains the truth value of the configuration. Notice that also the distance from the promised threshold p' and the real probability is maintained.

Case (a): For (strong) Until, we appeal to Lemma 9. We treat subformulae ψ_1 and ψ_2 as propositions (respectively, the \mathbf{q} and \mathbf{r} in that lemma) and annotate states of M by ψ_1 and ψ_2 . Let

$$p' = \text{Prob}_M(s, \psi_1 \mathbf{U} \psi_2) . \tag{5.6}$$

By assumption $p' > p$. In particular, $s \in \llbracket [\psi_1 \mathbf{U} \psi_2]_{\geq p'} \rrbracket_M$. Let $n \in \mathbb{N}$ be such that $p' > p' - \frac{1}{n} > p$. By Lemma 9 (applied to p' instead of p), there are $k, l \geq 0$ with $s \in \llbracket [\psi_1 \mathbf{U} \psi_2]_{> p' - \frac{1}{n}} \rrbracket_{M_{k,l}^s}$ and so the probability of $\psi_1 \mathbf{U} \psi_2$ in $M_{k,l}^s$ at s is greater than p . Player **V**'s strategy is to consider this system $M_{k,l}^s$. She chooses sub-distributions $d: S \rightarrow [0, 1]$ according to the probabilities $\text{Prob}_{M_{k,l}^s}(t, \alpha)$ (instead of $\text{Prob}_M(t, \alpha)$ but as explained above). By definition of $M_{k,l}^s$ there can be only finite sequences of configurations of the form $\langle s', [\alpha]_{> p}, \mathbf{V} \rangle$, and so player **V** wins (cf. Example 47).

Case (b): For weak Until $\psi_1 \mathbf{W} \psi_2$, all infinite plays have a suffix of configurations of form $\langle s', [\psi_1 \mathbf{W} \psi_2]_{\geq p}, \mathbf{V} \rangle$ and are thus winning for player \mathbf{V} . Finite plays again reach configurations of the form $\langle s', \psi_i, \mathbf{V} \rangle$ for $i \in \{1, 2\}$, where induction applies directly.

Case (c): For bounded operators, as the bound decreases, in a finite number of steps the play moves to configurations of the form $\langle s', \psi_i, \mathbf{V} \rangle$ for $i \in \{1, 2\}$, where induction applies directly, and in the desired manner.

- Let $s \notin \llbracket \phi \rrbracket_M$.

It follows that $\mathbf{Prob}_M(s, \alpha) \leq p$ in case that \bowtie is $>$; and $\mathbf{Prob}_M(s, \alpha) < p$ in case that \bowtie is \geq . As above, α is logically equivalent to $\psi_2 \vee (\psi_1 \wedge \mathbf{X}\alpha)$ and in case that α is bounded the bound decreases by 1. It follows that $s \notin \llbracket \psi_2 \rrbracket_M$ and hence there is a winning strategy for player \mathbf{R} from configuration $\langle s, \psi_2, \mathbf{V} \rangle$. Also, it is either the case that $s \notin \llbracket \psi_1 \rrbracket_M$ or $s \notin \llbracket [\mathbf{X}\alpha]_{\bowtie p} \rrbracket_M$. In the first case player \mathbf{R} has a winning strategy from configuration $\langle s, \psi_1, \mathbf{V} \rangle$ and chooses this configuration. In the second case, player \mathbf{V} chooses a sub-distribution $d: S \rightarrow [0, 1]$ such that (5.4) and (5.5) hold.

We claim that there is some $s' \in S$ with $d(s') > 0$ and $\mathbf{Prob}_M(s', \alpha) \not\bowtie d(s')P(s, s')^{-1}$. Proof by contradiction: otherwise, $\mathbf{Prob}_M(s', \alpha) \bowtie d(s')$ for all s' with $d(s') > 0$ implies that

$$\sum_{s' | d(s') > 0} \mathbf{Prob}_M(s', \alpha) \bowtie \sum_{s' \in S} d(s') \geq p$$

by (5.4). But this renders

$$\sum_{s' | d(s') > 0} \mathbf{Prob}_M(s', \alpha) \bowtie p$$

which directly contradicts $s \notin \llbracket [\mathbf{X}\alpha]_{\bowtie p} \rrbracket_M$.

Thus, player \mathbf{R} can choose such an s' and maintain the play in configurations of the form $\langle s', [\alpha]_{\bowtie p'}, \mathbf{V} \rangle$ such that $s' \notin \llbracket [\alpha]_{\bowtie p'} \rrbracket_M$. Notice that

player **R** can choose a successor s' such that

$$p' - \text{Prob}_M(s', \alpha) \geq p - \text{Prob}_M(s, \alpha) ,$$

i.e. the gap between the promise and the actual probability does not decrease.

We now study the consequences of this capability of player **R** for the different forms of path formula α in this case:

Case (a): For weak Until formulae, we appeal to Corollary 2. As before, we treat ψ_1 and ψ_2 as propositions and annotate states of M by them. Let $p' = \text{Prob}_M(s, \psi_1 \mathbf{W} \psi_2)$. By assumption $p' \leq p$. In particular, $s \notin \llbracket [\psi_1 \mathbf{W} \psi_2]_{>p'} \rrbracket_M$. Let $n \in \mathbb{N}$ be such that $p' < p + \frac{1}{n} < p$. By Corollary 2 there are $k, l \geq 0$ with $s \notin \llbracket [\psi_1 \mathbf{W} \psi_2]_{\geq p' + \frac{1}{n}} \rrbracket_{M_{k,l}^s}$ and so the probability of $\psi_1 \mathbf{W} \psi_2$ in $M_{k,l}^s$ at s is less than p . Player **R**'s strategy is to consider this system $M_{k,l}^s$. Let $d: S \rightarrow [0, 1]$ be the sub-distribution chosen by player **V**. As $s \notin \llbracket [\psi_1 \mathbf{W} \psi_2]_{\geq p} \rrbracket_{M_{k,l}^s}$, there is some $s' \in S$ such that

$$s' \notin \llbracket [\psi_1 \mathbf{W} \psi_2]_{\geq d(s')P(s,t)^{-1}} \rrbracket_{M_{k,l}^s} .$$

So player **R** chooses this s' . By definition of $M_{k,l}^s$ there can be only finite sequences of configurations of the form $\langle s', [\alpha]_{\geq p}, \mathbf{V} \rangle$, and so player **R** wins. This is dual to the strategy depicted for **V** in Example 47.

Case (b): For (strong) Until formulae, infinite plays of configurations of the form $\langle s', [\psi_1 \mathbf{U} \psi_2]_{\bowtie p}, \mathbf{V} \rangle$ are winning for player **R** by the winning conditions for infinite plays. Any finite play reduces to configurations of the form $\langle s', \psi_i, \mathbf{V} \rangle$ for $i \in \{1, 2\}$, where induction applies directly, and in the desired manner.

Case (c): For bounded operators, as the bound decreases, in a finite number of steps the play moves to configurations of the form $\langle s', \psi_i, \mathbf{V} \rangle$ for $i \in \{1, 2\}$ and so player **R** wins by induction. ■

Game $G_M(s, \phi)$ is defined such that its initial configuration $\langle s, \phi, \mathbf{V} \rangle$ is owned by player **V**. We can define a dual game with the same moves but

with initial configuration $\langle s, \phi, \mathbf{R} \rangle$. Theorem 4 and its proof then remain valid if we swap the role of players in both.

Example 48

Consider game $G_M(s_0, [\mathbf{q} \mathbf{U} \mathbf{r}]_{> \frac{1}{2}})$, where M is as in Figure 4.1 on page 124, and let $\alpha = \mathbf{q} \mathbf{U} \mathbf{r}$. From configuration $\langle s_0, [\alpha]_{> \frac{1}{2}}, \mathbf{V} \rangle$, player \mathbf{V} will not move to $\langle s_0, r, \mathbf{V} \rangle$ as she would then lose. For the same reason, player \mathbf{R} will not move to $\langle s_0, q, \mathbf{V} \rangle$. So if both players play strategies that are ‘optimal’ for them, player \mathbf{V} has to choose a sub-distribution d at the initial configuration.

If $d(s_2) > 0$, player \mathbf{V} loses as player \mathbf{R} can then choose s_2 . So $d(s_2) = 0$ for any ‘optimal’ strategy of player \mathbf{V} . But both $d(s_1)$ and $d(s_0)$ have to be positive since otherwise the mass of d can be at most $\frac{1}{3}$ by (5.5), which would violate (5.4). Since player \mathbf{V} plays an ‘optimal’ strategy, $d(s_1) \neq \frac{1}{3}$, as otherwise player \mathbf{R} could choose as next configuration $\langle s_1, [\alpha]_{> (\frac{1}{3}) \cdot (\frac{1}{3})^{-1}}, \mathbf{V} \rangle$ and would then win by move M1. By (5.5) there is therefore $\epsilon > 0$ such that $d(s_1) = \frac{1}{3} - \epsilon$. In particular, player \mathbf{R} will not choose s_1 as she would lose the next configuration $\langle s_1, [\alpha]_{> 1-3\epsilon}, \mathbf{V} \rangle$ (since $s_1 \in L(r)$). So player \mathbf{R} chooses s_0 and the next configuration is $\langle s_0, [\alpha]_{> 3d(s_0)}, \mathbf{V} \rangle$. By (5.4), $3d(s_0)$ must be at least $\frac{1}{2} + 3\epsilon$ and so player \mathbf{V} promises *more* in $> 3d(s_0)$ than she promised in the previous configuration.

At configuration $\langle s_0, [\alpha]_{> 3d(s_0)}, \mathbf{V} \rangle$, player \mathbf{V} avoids losing only by choosing a sub-distribution d that maps s_0 to 0 and all other states to positive mass as before, and for the same reasons. Similarly, $d(s_1) < \frac{1}{3}$ has to hold. So although a new sub-distribution d with a new value of ϵ may be chosen, the next configuration is still of the same type $\langle s_0, [\alpha]_{> p'}, \mathbf{V} \rangle$ with $p' > \frac{1}{2}$. Thus, either the play is finite and so lost for player \mathbf{V} as described above; or the play is infinite and so lost for player \mathbf{V} by the acceptance conditions A1 on infinite plays.

We conclude that player \mathbf{R} wins that game. A winning strategy for her from the initial configuration only needs to be specified for move M9:

- player \mathbf{R} will never choose a configuration of form $\langle s_0, q, \mathbf{V} \rangle$, should such an opportunity arise;
- whenever player \mathbf{V} chooses sub-distribution d with $d(s_2) > 0$, player \mathbf{R} will choose s_2 ;
- otherwise, it must be the case that both $d(s_1)$ and $d(s_2)$ are positive;

- if $d(s_1) \geq \frac{1}{3}$, player R chooses s_1 ;
- if $d(s_1) < \frac{1}{3}$, player R chooses s_0 . ◆

5.2 Winning strategies

We show that when a player can win game $G_M(s, \phi)$ she can use winning strategies that are of a very specific type. In addition to being memoryless in the classical sense, they choose very structured distributions when re-visiting a state in a configuration with a strong or weak Until operator.

As before we use the notion of strategy informally. A strategy is *memoryless* if its choices depend solely on the current configuration, not on the finite history of configurations that preceded the current one in a play.

In our games, there can be configurations of type $\langle s, [\alpha]_{\bowtie p}, \mathbf{C} \rangle$ for the same state s and the same path formula α (e.g., $\psi_1 \mathbf{U} \psi_2$) but with different thresholds $\bowtie p$. We show that it is enough to consider winning strategies which induce thresholds that change monotonically, as defined below. Subsequently, for sub-distributions $d, d' : S \rightarrow [0, 1]$, we write

- $d \leq d'$ iff for all $s \in S$ we have $d'(s) \leq d(s)$
- $d' < d$ iff $d' \leq d$ and $d'(s) < d(s)$ for some $s \in S$

First we define *locally monotone* strategies, i.e. strategies in which the choice of sub-distribution d at configuration $\langle s, [\alpha]_{\bowtie p}, \mathbf{C} \rangle$ is monotone in $\bowtie p$ regardless of the history of a play.

Definition 62 (Locally Monotone Strategies)

A strategy σ for player \mathbf{C} in game $G_M(s, \phi)$ is *locally monotone* iff for any two configurations $\langle s, [\alpha]_{\bowtie p}, \mathbf{C} \rangle$ and $\langle s, [\alpha]_{\bowtie p'}, \mathbf{C} \rangle$ that occur in plays consistent with σ (but not necessarily in the same play) where d and d' are the sub-distributions chosen according to σ at these two configurations respectively, then $p \geq p'$ implies $d \geq d'$ and $p > p'$ implies $d > d'$. ◆

We introduce a second notion of monotonicity: A *cyclically monotone* strategy is monotone on cyclic paths within single plays; its player can force

a decrease or increase of the thresholds depending on the path formula and on whether it is a V or R configuration.

Definition 63 (Cyclically Monotone Strategies)

A strategy σ for player C in game $G_M(s, \phi)$ is *cyclically monotone* iff for any two configurations $\langle s, [\alpha]_{\bowtie p}, C' \rangle$ and $\langle s, [\alpha]_{\bowtie p'}, C' \rangle$ that occur in this order on some play consistent with σ , then

- $\alpha = \psi_1 \text{ U } \psi_2$ and $C = C'$ imply $p' < p$,
- $\alpha = \psi_1 \text{ W } \psi_2$ and $C = C'$ imply $p' \leq p$,
- $\alpha = \psi_1 \text{ U } \psi_2$ and $\bar{C} = C'$ imply $p' \geq p$,
- $\alpha = \psi_1 \text{ W } \psi_2$ and $\bar{C} = C'$ imply $p' > p$. ◆

The existence of winning strategies implies the existence of winning strategies that are locally monotone and cyclically monotone.

Theorem 5 (Monotonicity of winning strategies)

For every game $G_M(s, \phi)$, there exists a winning strategy for player C iff there exists a memoryless winning strategy for player C that is also locally monotone and cyclically monotone. ●

Proof

Assuming that there exists some winning strategy for player C in game $G_M(s, \phi)$, it suffices to show that a slight modification of the winning strategy synthesised in the proof of Theorem 4 is memoryless, locally monotone, and cyclically monotone. That slightly modified strategy will clearly be memoryless by construction. We now describe this modified winning strategy and first prove its local monotonicity, by induction as in the proof of Theorem 4. Then we prove that it is cyclically monotone.

Modified winning strategy and its local monotonicity. The only configurations where player \mathbf{C} needs to make choices are

$$\begin{aligned} &\langle s, [\alpha]_{\bowtie p}, \mathbf{C}' \rangle ; \\ &\langle s, \psi_1 \vee \psi_2, \mathbf{C} \rangle ; \text{ and} \\ &\langle s, \psi_1 \wedge \psi_2, \bar{\mathbf{C}} \rangle . \end{aligned}$$

With the latter two, we restrict \mathbf{C} 's strategy to choose ψ_1 whenever possible and only when that is impossible to choose ψ_2 . This is a memoryless choice similar to what one could do in Hintikka games for first-order logic. We show that the way configurations of the form $\langle s, [\alpha]_{\bowtie p}, \mathbf{C}' \rangle$ are handled induces a memoryless and monotone strategy.

First we prove three cases which are fairly straightforward.

1. For configurations where $\alpha = X\psi$, the strategy defined in the proof of Theorem 4 chooses the set of successors according to the state s , and is clearly memoryless.
2. For configurations where $\bar{\mathbf{C}} = \mathbf{C}'$ and either $\alpha = \psi_1 \mathbf{U} \psi_2$ and $\bowtie = \geq$ or $\alpha = \psi_1 \mathbf{W} \psi_2$ and $\bowtie = >$, then player \mathbf{C} has to choose a value n . By choosing the minimal possible n she ensures that the strategy is memoryless.
3. For configurations $\langle s, \psi_1 \vee \psi_2, \mathbf{C} \rangle$ and $\langle s, \psi_1 \wedge \psi_2, \mathbf{C} \rangle$: Whenever the play moves to configurations of the form $\langle s', \psi_i, \mathbf{V} \rangle$ for $i \in \{1, 2\}$, the strategy is memoryless, locally monotone, and cyclically monotone by induction.

We now start with proving local monotonicity for moves that may choose sub-distributions.

1. For configurations where

$$\begin{aligned} &\alpha = \psi_1 \mathbf{W} \psi_2 , \\ &\alpha = \psi_1 \mathbf{W}^{\leq k} \psi_2 , \text{ or} \\ &\alpha = \psi_1 \mathbf{U}^{\leq k} \psi_2 , \end{aligned}$$

and $\mathbf{C} = \mathbf{C}'$ we claim that the strategy composed in the proof of Theorem 4 is locally monotone by induction. Intuitively, this can be seen by the strategy using the gap δ between the probability of the formula and the required threshold. The strategy partitions this gap between all successors, so if the same state is visited with different thresholds the partition of the gap implies that the chosen distribution decreases.

Let $p' = \text{Prob}_M(s, \alpha)$ and $\delta_i = p' - p_i$ for $i \in \{1, 2\}$. According to the proof of Theorem 4 in configuration $\langle s, [\alpha]_{\bowtie p_i}, \mathbf{C} \rangle$ player \mathbf{C} chooses the distribution

$$d_i(t) = \max(0, (\text{Prob}_M(t, \alpha) - \delta_i)P(s, t)) .$$

It follows that if $p_1 \geq p_2$ then for every $t \in S$ we have $d_1(t) \geq d_2(t)$. It follows that if $p_1 = p_2$ then $d_1 = d_2$. Consider the case that $p_1 > p_2$. Then $p_1 > 0$ and for some t we have $d_1(t) > 0$ and $d_1(t) = \text{Prob}_M(t, \alpha) - \delta_1$. As $\delta_1 < \delta_2$ and $d_2(t) = \text{Prob}_M(t, \alpha) - \delta_2$ it follows that $d_1(t) > d_2(t)$.

2. For the case where

$$\alpha = \psi_1 \cup \psi_2 \text{ and } \mathbf{C} = \mathbf{C}' ,$$

the strategy as defined in the proof of Theorem 4 is not locally monotone. (Basically, the strategy in that proof is synthesised according to some finite unfolding M_n where n can be chosen differently for every game configuration so that the resulting choice of sub-distributions does not necessarily yield a locally monotone strategy.)

We modify this construction as follows: For every configuration of the form $\langle s, [\psi_1 \cup \psi_2]_{>p}, \mathbf{C} \rangle$ the sub-distribution d is chosen according to the minimal k such that some fraction of $\text{Prob}_{M_k^s}(s, \alpha)$ is greater than p . The exact definition of this fraction is given below. Furthermore, we use the gap between $\text{Prob}_{M_k^s}(s, \alpha)$ and $\text{Prob}_{M_{k-1}^s}(s, \alpha)$ to ensure local (and later cyclic) monotonicity. The definition of the sub-distribution d and the proof itself are quite technical.

Consider the configuration $\langle s, [\alpha]_{>p}, \mathbf{C} \rangle$. We assume, without loss of generality, that $s \notin \llbracket \psi_2 \rrbracket_M$. We measure the exact probability to satisfy α within

i steps. For every $t \in S$ let

$$\begin{aligned} n_0^t &= \text{Prob}_{M_0^t}(t, \alpha) \\ n_i^t &= \text{Prob}_{M_i^t}(t, \alpha) - \text{Prob}_{M_{i-1}^t}(t, \alpha) \end{aligned}$$

Consider the following increasing sequence:

$$\begin{aligned} N_0^t &= \frac{n_0^t}{2} \\ N_i^t &= N_{i-1}^t + \sum_{j=0}^i \frac{1}{2^{i+1-j}} n_j^t \quad (i > 0) \end{aligned}$$

That is, $N_1^t = \frac{3}{4}n_0^t + \frac{1}{2}n_1^t$, $N_2^t = \frac{7}{8}n_0^t + \frac{3}{4}n_1^t + \frac{1}{2}n_2^t$, $N_3^t = \frac{15}{16}n_0^t + \frac{7}{8}n_1^t + \frac{3}{4}n_2^t + \frac{1}{2}n_3^t$, and so on. Notice that

$$\lim_{i \rightarrow \infty} N_i^t = \text{Prob}_{M_k^t}(t, \alpha)$$

Let i_0 be the minimal such that

$$\sum_{t \in S} N_{i_0}^t P(s, t) > p$$

By abuse of notation for $i \geq 0$, we denote

$$N_{i+1}^s = \sum_{t \in S} N_i^t P(s, t)$$

That is, N_i^s is the sum of the different N_{i-1}^t normalised by the probability to get from s to t . To simplify notations, for $i < 0$ and for all t we set

$$N_i^t = N_{i+1}^s = 0 .$$

The value $N_{i_0}^t P(s, t)$ is going to be the basis for defining $d(t)$. Notice that it must be the case that $N_{i_0}^s \leq p$ and that $N_{i_0}^t - N_{i_0-1}^t > 0$. In order to maintain local monotonicity we distribute the gap between the required threshold p and $N_{i_0}^s$ between all the states t where $N_{i_0+1}^t > 0$. We have to be extremely careful with the states s for which $N_{i_0}^s = p$. For these states, we take a constant fraction of $N_{i_0}^t - N_{i_0-1}^t$ and distribute it among the successors of t . We then have to scale the distribution d for all states s for which this

constant fraction surpasses the required threshold.

We set $d(t)$ as follows:

$$d(t) = \left(N_{i_0-1}^t + \left(\frac{1}{4} + \frac{3}{4} \frac{p - N_{i_0}^s}{N_{i_0+1}^s - N_{i_0}^s} \right) (N_{i_0}^t - N_{i_0-1}^t) \right) P(s, t) .$$

It is simple to see that $\sum_{t \in S} d(t) > p$. Indeed, $\sum_{t \in S} d(t)$ is the sum of the following three expressions:

$$\begin{aligned} \sum_{t \in S} N_{i_0-1}^t P(s, t) &= N_{i_0}^s \\ \sum_{t \in S} \frac{N_{i_0}^t - N_{i_0-1}^t}{4} P(s, t) &= \frac{N_{i_0+1}^s - N_{i_0}^s}{4} \\ \sum_{t \in S} \frac{3}{4} \frac{p - N_{i_0}^s}{N_{i_0+1}^s - N_{i_0}^s} (N_{i_0}^t - N_{i_0-1}^t) P(s, t) &= \frac{3}{4} (p - N_{i_0}^s) \end{aligned}$$

As $N_{i_0+1}^s > p$ the result follows.

Furthermore, when going to some successor t of s the choice of i_0 for s implies that for the choice of the sub-distribution d for t some value $i'_0 < i_0$ is going to be used. Thus, the sequence of configurations of the form $\langle t', [\alpha]_{>p'}, \mathbf{C} \rangle$ is finite and player \mathbf{C} is winning.

We show that this definition of the sub-distribution d implies local monotonicity. Consider two configurations $\langle s, [\alpha]_{>p_1}, \mathbf{C} \rangle$ and $\langle s, [\alpha]_{>p_2}, \mathbf{C} \rangle$. Let d_1 and d_2 be the sub-distributions chosen by σ in these configurations and let i_0^1 and i_0^2 be the values used to define d_1 and d_2 , respectively. By definition $d_j(t)$ is in the open interval

$$(N_{i_0^j-1}^t P(s, t), N_{i_0^j}^t P(s, t))$$

for $j \in \{1, 2\}$. By definition if $p_1 = p_2$ then $i_0^1 = i_0^2$ and it follows that $d_1 = d_2$. Similarly, if $p_1 > p_2$ then $i_0^1 \geq i_0^2$. If $i_0^1 > i_0^2$ the strictness of $d_1 > d_2$ follows from the strictness of the sequence N_i^t . If $i_0^1 = i_0^2$ then $d_1 > d_2$ as $p_1 > p_2$.

Cyclic monotonicity of modified winning strategy. We now turn to cyclic monotonicity. Consider the two configurations $\langle s, [\alpha]_{\bowtie p_1}, \mathbf{C}' \rangle$ and

$\langle s, [\alpha]_{\bowtie p_2}, \mathbf{C}' \rangle$ that appear in this order in a play consistent with σ .

1. Consider the case where $\mathbf{C} = \mathbf{C}'$ and

$$\begin{aligned} \alpha &= \psi_1 \mathbf{W} \psi_2 , \\ \alpha &= \psi_1 \mathbf{W}^{\leq k} \psi_2 , \text{ or} \\ \alpha &= \psi_1 \mathbf{U}^{\leq k} \psi_2 . \end{aligned}$$

The strategy defined in the proof of Theorem 4 is already cyclically monotone. Indeed, from configuration $\langle s, [\alpha]_{\bowtie p}, \mathbf{C} \rangle$ where

$$\text{Prob}_M(s, \alpha) - p = \delta$$

we pass to configuration $\langle t, [\alpha]_{\bowtie p'}, \mathbf{C} \rangle$ and we know that

$$\text{Prob}_M(t, \alpha) - p' = \delta$$

Hence, if configurations $\langle s, [\alpha]_{\bowtie p_1}, \mathbf{C} \rangle$ and $\langle s, [\alpha]_{\bowtie p_2}, \mathbf{C} \rangle$ appear in the same play we have $p_1 \geq p_2$.

2. Consider the case where

$$\alpha = \psi_1 \mathbf{U} \psi_2 \text{ and } \mathbf{C} = \mathbf{C}'$$

and the strategy defined as above. Let i_0^1 be the threshold used for choosing the sub-distribution d in configuration $\langle s, [\alpha]_{> p_1}, \mathbf{C} \rangle$. By construction values smaller than i_0^1 are going to be used to define the sub-distributions in successor configurations. It follows that if configuration $\langle s, [\alpha]_{> p_2}, \mathbf{C} \rangle$ is visited, a value $i_0^2 < i_0^1$ is going to be used to define its sub-distribution. From the strictness of the sequence N_i^t (and N_i^s) and as $N_{i_0^j}^s \leq p_j < N_{i_0^j+1}^s$ it follows that $p_2 < p_1$.

3. Consider the case where $\bar{\mathbf{C}} = \mathbf{C}'$ and

$$\begin{aligned}\alpha &= \psi_1 \mathbf{U} \psi_2 ; \\ \alpha &= \psi_1 \mathbf{U}^{\leq k} ; \text{ or} \\ \alpha &= \psi_1 \mathbf{W}^{\leq k} \psi_2 .\end{aligned}$$

Let $p' = \text{Prob}_M(s', \alpha)$ and $\delta_i = p_i - p'$ for $i \in \{1, 2\}$. Let d be the distribution suggested by player $\bar{\mathbf{C}}$ in configuration $\langle s, [\alpha]_{\times p_1}, \bar{\mathbf{C}} \rangle$. By definition of d we have $\sum_{t \in S} d(t) \geq p_1$. By assumption $\langle s, [\alpha]_{\times p_2}, \bar{\mathbf{C}} \rangle$ is reachable from $\langle s, [\alpha]_{\times p_1}, \bar{\mathbf{C}} \rangle$, so neither player chooses to go to configurations of the form $\langle t, \psi_i, \bar{\mathbf{C}} \rangle$ for $i \in \{1, 2\}$. It follows that

$$\text{Prob}_M(s, \alpha) = \sum_{t \in S} P(s, t) \text{Prob}_M(t, \alpha)$$

We know that

$$\sum_{t \in S} d(t) \geq p' + \delta_1$$

Then, there must exist some $t \in S$ such that

$$d(t) \cdot P(s, t)^{-1} \geq \text{Prob}_M(t, \alpha) + \delta_1$$

It follows that if player \mathbf{C} chooses this state t the gap between the actual probability and the threshold does not decrease. Thus $p_1 \leq p_2$.

4. Consider the case where

$$\alpha = \psi_1 \mathbf{W} \psi_2 \text{ and } \bar{\mathbf{C}} = \mathbf{C}' .$$

Then the proof is similar to the previous item. By assumption \mathbf{C} wins from $\langle s, [\alpha]_{\geq p_1}, \bar{\mathbf{C}} \rangle$ and hence $s \notin \llbracket [\alpha]_{\geq p_1} \rrbracket_M$. Let $p' = \text{Prob}_M(s, \alpha)$. As player \mathbf{C} wins from $\langle s, [\alpha]_{\geq p_1}, \bar{\mathbf{C}} \rangle$ we conclude that $p' < p_1$. In particular, $s \notin \llbracket [\psi_1 \mathbf{W} \psi_2]_{> p'} \rrbracket_M$. Let $n \in \mathbb{N}$ be such that $p' < p + \frac{1}{n} < p$. By Corollary 2 there are $k, l \geq 0$ with $s \notin \llbracket [\psi_1 \mathbf{W} \psi_2]_{\geq p' + \frac{1}{n}} \rrbracket_{M_{k,l}^s}$ and so the probability of $\psi_1 \mathbf{W} \psi_2$ in $M_{k,l}^s$ at s is less than p_1 . Player \mathbf{C} is going to use system $M_{k,l}^s$ to

guide her decisions. As usual

$$\text{Prob}_{M_{k,l}^s}(s, \alpha) = \sum_{t \in S_{k,l}} P(s, t) \text{Prob}_{M_{k,l}^s}(t, \alpha)$$

Let

$$p'' = \text{Prob}_{M_{k,l}^s}(s, \alpha)$$

As mentioned $p'' < p_1$. Let $\delta_1 = p_1 - p''$ and let d be the distribution suggested by player \bar{C} in configuration $\langle s, [\alpha]_{\geq p_1}, \bar{C} \rangle$. By definition of d we have

$$\sum_{t \in S} d(t) \geq p_1 = \delta_1 + p''$$

Then, there must exist some $t \in S$ such that

$$d(t) \cdot P(s, t)^{-1} \geq \text{Prob}_{M_{k,l}^s}(t, \alpha) + \delta_1$$

It follows that if player C chooses this state t the gap between the actual probability in $M_{k,l}^s$ and the threshold does not decrease. We show below in Lemma 13 that when visiting the same state again in $M_{k,l}^s$ the probability of α increases. Hence, $p_2 > p_1$. \blacksquare

Lemma 13

Let M be a labelled Markov chain, q and r in AP , α the path formula qWr , and $M_{k,l}^s$ given for some state s of M and $k, l \in \mathbb{N}$. Let t and t' be different states in $M_{k,l}^s$ that both correspond to some state s' of M such that

- there is a path from t to t' in $M_{k,l}^s$, and
- q holds throughout the unique and finite path from the root of $M_{k,l}^s$ to t' .

If we have $\text{Prob}_{M_k^s}(t, \alpha) < 1$, then $\text{Prob}_{M_k^s}(t', \alpha) > \text{Prob}_{M_k^s}(t, \alpha)$ follows. \bullet

Proof

As $\text{Prob}_{M_k^s}(t, qWr) < 1$ it follows that there is some “leaf” t'' in $M_{k,l}^s$ that is reachable from t in $M_{k,l}^s$ such that the unique finite path from t to t'' in $M_{k,l}^s$ does not satisfy qWr . As $M_{k,l}^s$ is an unwinding of M , it follows that the

subtree reachable from t' in $M_{k,l}^s$ is contained in the subtree reachable from t in $M_{k,l}^s$. Clearly, $\text{Prob}_{M_{k,l}^s}(t', \alpha) \geq \text{Prob}_{M_{k,l}^s}(t, \alpha)$. Indeed, if a path satisfies \mathbf{qWr} then every prefix of the path also satisfies \mathbf{qWr} . We use proof by contradiction to argue that there is a path from t that does not satisfy \mathbf{qWr} and does not pass through t' . Assume such a path does not exist. Then every path beginning in t that does not satisfy \mathbf{qWr} has to pass through t' . However, both t and t' correspond to state s' in M . It follows that the only option to falsify \mathbf{qWr} in game $\mathbf{G}_M(s', \alpha)$ is by “going in a loop” from state s' to itself. But by assumption all states on the path between t and t' satisfy \mathbf{q} , which yields a contradiction and thus concludes the proof. ■

Example 49

The winning strategy for Refuter in Example 48 is locally monotone as Refuter never encounters a pair of configurations that need to be checked for local monotonicity. That strategy is also cyclically monotone: From configuration $\langle s_0, [\mathbf{qUr}]_{>p}, \mathbf{V} \rangle$ the only possible cycles lead to configurations $\langle s_0, [\mathbf{qUr}]_{>p'}, \mathbf{V} \rangle$. As explained already, Verifier is restricted to $d(s_2) = 0$ and $d(s_1) < \frac{1}{3}$ or she loses in the next step. Let $p > \frac{1}{2}$ and $\epsilon = \frac{1}{3} - d(s_1)$. Then $d(s_0) \geq \frac{1}{6} + (p - \frac{1}{2}) + \epsilon$. Thus, in the next configuration $\langle s_0, [\mathbf{qUr}]_{>p'}, \mathbf{V} \rangle$ we have $p' \geq \frac{1}{2} + 3(p - \frac{1}{2}) + 3\epsilon$. As $\epsilon > 0$ and $p - \frac{1}{2} > 0$ we have $p' > p$. Finally, if p_1, p_2, \dots is the sequence of thresholds obtained in this manner, then $p_{i+2} - p_{i+1} > p_{i+1} - p_i$ for all $i \geq 1$. ♦

5.3 Summary of chapter

This chapter defined an operational semantics, i.e. a Hintikka game, for PCTL over Markov chains. This game-based semantics was shown to be equivalent to the standard semantics for PCTL, but is operational in nature.

The contribution of this chapter is twofold: First, a game-based formulation of the standard PCTL semantics is a useful result in its own rights as it provides solid foundation for the computation of finitary evidence for the truth and falsity of PCTL formulae, and for the application of techniques from game theory to probabilistic verification. Second, a game-based semantics provides suitable machinery for an automata-based model-checking framework such as our p-automata in Chapter 6.

6 Completeness for full PCTL via p-Automata

We have seen in Chapter 4 that (3-valued) Markov chains and Larsen-Skou simulation are not expressive enough to yield a complete abstraction framework for full PCTL. On the positive side, we already achieved completeness for a sizable fragment of PCTL via 3-valued unfoldings.

To achieve completeness for full PCTL, which is the primary objective of this thesis, we need a richer formalism. As the work by Mark Kattenbelt and Michael Huth [111] shows, the non-determinism of Markov decision processes does not help to achieve a complete abstraction framework. In fact, their game-based abstraction framework is already not complete for some formulae of $\text{PCTL}_{>}$, i.e. the fragment for which we achieved completeness in Chapter 4.

A complete abstraction framework for non-probabilistic branching-time logics was achieved through alternating tree automata [59]. These automata provide the necessary expressiveness which is needed to solve the completeness problem for branching time. As PCTL is a probabilistic branching-time logic, this success of alternating tree automata suggests that an automata-based approach might be suitable for the PCTL completeness problem, too. In this chapter we develop a framework of automata-like acceptors of Markov chains, called *p-automata*, which captures Markov chains and PCTL formulae in the same formalism. These p-automata then indeed are a complete abstraction framework for full PCTL as we will show in Section 6.4.

The p-automata framework that we develop in this chapter uses two-player games to define core concepts such as acceptance and simulation. These techniques draw on insights from the Hintikka game presented in Chapter 5. In particular, for a p-automaton A_ϕ associated with a PCTL formula ϕ as explained in Section 6.3.3, the acceptance game defined in Section 6.2 which determines whether or not Markov chain M is accepted

by A , closely resembles the Hintikka game of Section 5.1 which determines whether or not M satisfies ϕ . In this sense the p-automata framework builds on theoretical insights from the game-based PCTL semantics. Nevertheless, the presentation of our p-automata framework in this chapter is fully self-contained and does not require any technical material from Chapter 5.

6.1 p-Automata

Traditional probabilistic automata [157] map an input to a probability of accepting it. Such an automaton A then gives rise to a mapping from inputs to the probability of their acceptance, and thus to probabilistic languages \mathcal{L}_μ of inputs which are accepted with probability above a fixed threshold μ . In contrast, our p-automata either accept or reject an entire Markov chain. In particular, a p-automaton determines a language of Markov chains.

For this chapter we assume familiarity with basic notions of trees and (alternating) tree automata as introduced in Chapter 3. To clarify the technical presentation in this chapter, automata have *states*, Markov chains have *locations*, and games have *configurations*.

Definition 64 (Positive Boolean formulae)

For set T , let $B^+(T)$ be the set of *positive Boolean formulae* generated from elements $t \in T$, constants **ff** and **tt**, and disjunctions and conjunctions:

$$\varphi ::= t \mid \mathbf{ff} \mid \mathbf{tt} \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \quad (6.1)$$

Formulae in $B^+(T)$ are finite even if T is not. ◆

Definition 65 (Term sets)

For set Q , we define *term sets* as follows. This uses n -ary operators $*_n$ and $\check{*}_n$ for every $n \in \mathbb{N}$, which we write as $*$ and $\check{*}$ throughout as n will be clear from context.

$$\begin{aligned} \llbracket Q \rrbracket_{>} &= \{ \llbracket q \rrbracket_{\bowtie p} \mid q \in Q, \bowtie \in \{ \geq, > \}, p \in [0, 1] \} \\ \llbracket Q \rrbracket^* &= \{ *(t_1, \dots, t_n) \mid n \in \mathbb{N}, \forall i: t_i \in \llbracket Q \rrbracket_{>} \} \\ \llbracket Q \rrbracket^{\check{*}} &= \{ \check{*}(t_1, \dots, t_n) \mid n \in \mathbb{N}, \forall i: t_i \in \llbracket Q \rrbracket_{>} \} \\ \llbracket Q \rrbracket &= \llbracket Q \rrbracket^* \cup \llbracket Q \rrbracket^{\check{*}} \end{aligned} \quad \blacklozenge$$

Intuitively, Q is the set of states of a p-automaton and the transition structure of state $q \in Q$ (given by $\phi \in B^+(Q \cup \llbracket Q \rrbracket)$ as defined below) models a probabilistic path set. So $\llbracket q \rrbracket_{\bowtie p}$ holds in location s of a Markov chain if the measure of paths that begin in s and satisfy q is $\bowtie p$.

The operator $*$ is a novel probabilistic separation operator and \checkmark is its dual: for example $*(\llbracket q_1 \rrbracket_{>p_1}, \llbracket q_2 \rrbracket_{\geq p_2})$ means q_1 and q_2 hold with probability greater than p_1 and greater than or equal to p_2 , respectively; and that the sets supplying these probabilities are disjoint. Dually, $\checkmark(\llbracket q_1 \rrbracket_{\geq p_1}, \llbracket q_2 \rrbracket_{\geq p_2})$ means that either

- (i) there is $i \in \{1, 2\}$ such that q_i holds with probability at least p_i or
- (ii) the intersection of q_1 and q_2 holds with probability at least $\max(p_1 + p_2 - 1, 0)$.

So $*$ and \checkmark model a “disjoint and” and “intersecting or” operator, respectively. We may write $\llbracket q \rrbracket_{\bowtie p}$ for $*(\llbracket q \rrbracket_{\bowtie p})$ and similarly for \checkmark .

The dual nature of $*$ and \checkmark is used explicitly in the dualisation of p-automata in Section 6.3.1.

With this first intuitive understanding of $*$ and \checkmark and the definition of term sets in place, we have that an element of $Q \cup \llbracket Q \rrbracket$ is either a state of a p-automaton, a $*$ composition of terms $\llbracket q_i \rrbracket_{\bowtie p_i}$, or a \checkmark composition of such terms. Before we turn to the formal definition of p-automata, we define a closure operator for formulae in $B^+(Q \cup \llbracket Q \rrbracket)$.

Definition 66 (Closure cl_p)

Given $\varphi \in B^+(Q \cup \llbracket Q \rrbracket)$, its closure $\text{cl}_p(\varphi)$ is the set of all subformulae of φ according to (6.1) of Definition 64. In particular, $*(t_1, t_2) \in \text{cl}_p(\varphi)$ does not imply $t_1, t_2 \in \text{cl}_p(\varphi)$. For a set Φ of formulae, let $\text{cl}_p(\Phi) = \bigcup_{\varphi \in \Phi} \text{cl}_p(\varphi)$. \blacklozenge

Next we define p-automata.

Definition 67 (p-Automata)

A *p-automaton* A is a tuple $\langle \Sigma, Q, \delta, \varphi^{\text{in}}, \alpha \rangle$, with

- Σ a finite input alphabet;
- Q a set of states (not necessarily finite);

- $\delta: Q \times \Sigma \rightarrow B^+(Q \cup \llbracket Q \rrbracket)$ a transition function;
- $\varphi^{\text{in}} \in B^+(Q \cup \llbracket Q \rrbracket)$ an initial condition; and
- $\alpha \subseteq Q$ an acceptance condition. ◆

The terms $\phi \in \llbracket Q \rrbracket$ of a p-automaton are comparable to the q^\square transitions of alternating tree automata, in the sense that e.g. term $\llbracket q \rrbracket_{\bowtie p}$ asserts that q holds *for all* successor locations with an aggregated probability $\bowtie p$. To stress this universal but probabilistic quantification and to stay in line with the notation of alternating tree automata, one could explicitly write $\llbracket q \rrbracket_{\bowtie p}^\square$. Since p-automata do not have an analogue to the \diamond -modality such a superscript \square is superfluous, and thus omitted in this thesis.

Example 50

Let $A = \langle 2^{\{\mathbf{a}, \mathbf{b}\}}, \{q_1, q_2\}, \delta, \llbracket q_1 \rrbracket_{\geq 0.5}, \{q_2\} \rangle$ be a p-automaton where δ is defined by

$$\begin{aligned} \delta(q_1, \{\mathbf{a}, \mathbf{b}\}) &= \delta(q_1, \{\mathbf{a}\}) = q_1 \vee \llbracket q_2 \rrbracket_{\geq 0.5} \\ \delta(q_2, \{\mathbf{b}\}) &= \delta(q_2, \{\mathbf{a}, \mathbf{b}\}) = \llbracket q_2 \rrbracket_{\geq 0.5} \\ \delta(q_1, \{\}) &= \delta(q_1, \{\mathbf{b}\}) = \delta(q_2, \{\}) = \delta(q_2, \{\mathbf{a}\}) = \text{ff} \end{aligned}$$

Term $\llbracket q_2 \rrbracket_{\geq 0.5}$ represents the recursive property ϕ , that atomic proposition \mathbf{b} holds at the location presently read by q_2 , and that ϕ will hold with probability at least 0.5 in the next locations.¹ State q_1 asserts that it is possible to get to a location that satisfies $\llbracket q_2 \rrbracket_{\geq 0.5}$ along a path that satisfies atomic proposition \mathbf{a} .² The initial condition $\llbracket q_1 \rrbracket_{\geq 0.5}$ means the set of paths satisfying $\mathbf{a} \cup \phi$ has probability at least 0.5. ◆

In this thesis we restrict our attention to a variant of p-automata which we call *uniform weak*. This restriction simplifies the presentation and allows

¹Informally, \mathbf{b} holds at the location read by q_2 because the transition of q_2 reading \mathbf{b} is not ff, i.e. $\delta(q_2, \{\mathbf{b}\}) \neq \text{ff}$. We call $\llbracket q_2 \rrbracket_{\geq 0.5}$ a *recursive* property since the transition of q_2 reading \mathbf{b} contains $\llbracket q_2 \rrbracket_{\geq 0.5}$ as condition again.

²Again informally, q_1 makes an assertion about a path, i.e. a sequence of locations in a Markov chain; it asserts that every location on this path satisfies \mathbf{a} , i.e. $\delta(q_1, \{\mathbf{a}, \mathbf{b}\}) = \delta(q_1, \{\mathbf{a}\}) = q_1 \vee \llbracket q_2 \rrbracket_{\geq 0.5} \neq \text{ff}$, and that the successor of such a location satisfies the property asserted by q_1 or satisfies $\llbracket q_2 \rrbracket_{\geq 0.5}$. Eventually a location that satisfies $\llbracket q_2 \rrbracket_{\geq 0.5}$ must be reached as $q_1 \notin \alpha$ is not accepting.

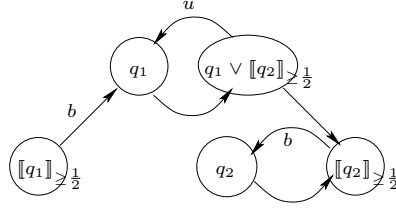


Figure 6.1: Graph G_A of automaton A from Example 50.

us to define acceptance and simulation in terms of existing game notions. The development of appropriate games for general p-automata is beyond the scope of this thesis and will be addressed in future work.

As in the case of alternating tree automata, e.g. [84], acceptance for p-automata is decided by solving an infinite 2-player game, respectively a series of such games. In order to be able to decide acceptance of input for p-automata through the solution of weak stochastic games, we restrict the cycles in the transition graph of p-automata. In doing so, we differentiate states q' appearing within a term in $\llbracket Q \rrbracket$ (bounded transition) from q' appearing “free” in the transition of a state q (unbounded transition). In this way, a p-automaton $A = \langle \Sigma, Q, \delta, \dots \rangle$ determines a labelled, directed graph $G_A = \langle Q', E, E_b, E_u \rangle$:

$$\begin{aligned}
Q' &= Q \cup \text{cl}_p(\delta(Q, \Sigma)) \\
E &= \{(\varphi_1 \wedge \varphi_2, \varphi_i), (\varphi_1 \vee \varphi_2, \varphi_i) \mid \varphi_i \in Q' \setminus Q\} \cup \\
&\quad \{q, \delta(q, \sigma) \mid q \in Q, \sigma \in \Sigma\} \\
E_u &= \{(\varphi \wedge q, q), (q \wedge \varphi, q), (\varphi \vee q, q), (q \vee \varphi, q) \mid \varphi \in Q', q \in Q\} \\
E_b &= \{(\varphi, q) \mid \varphi \in \llbracket Q \rrbracket \text{ and } q \in \text{gs}(\varphi)\}
\end{aligned}$$

where $\text{gs}(\varphi)$ is the set of *guarded* states of φ : all $q' \in Q$ occurring in some term in φ . Elements $(\varphi, q) \in E_u$ are *unbounded* transitions. Elements $(\varphi, q') \in E_b$ are *bounded* transitions. Elements of E are called *simple* transitions. We mark $(\varphi, q) \in E_b$ with $*$ (and respectively with a \checkmark) if there is some $p \in [0, 1]$ for which $\llbracket q' \rrbracket_{\triangleright p}$ occurs in φ within the scope of a $*$ (respectively \checkmark) operator. Note that E , E_u , and E_b are pairwise disjoint.

Let $\varphi \preceq_A \tilde{\varphi}$ iff there is a finite path from φ to $\tilde{\varphi}$ in $E \cup E_b \cup E_u$. For $\varphi \in Q \cup \text{cl}_p(\delta(Q, \Sigma))$, let (φ) denote the equivalence class of φ according to

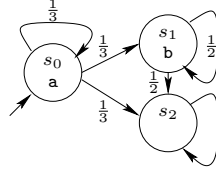


Figure 6.2: A Markov chain M .

pre-order \preceq_A . We denote this equivalence relation $\preceq_A \cap \preceq_A^{-1}$ as \equiv . Each equivalence class $((\varphi))$ of \equiv is a SCC in the directed graph G_A .

Definition 68

A p-automaton A is called *uniform* if:

- For each cycle in G_A , its set of transitions is either in $E \cup E_b$ or in $E \cup E_u$.
- For each cycle in $\langle Q, E \cup E_b \rangle$, its set of markings is either $\{\}$, $\{*\}$ or $\{\checkmark\}$, and so cannot be $\{*, \checkmark\}$.
- There are only finitely many equivalence classes $((\varphi))$ with $\varphi \in Q \cup \text{cl}_p(\delta(Q, \Sigma))$.

A (not necessarily uniform) p-automaton A is called *weak* if for all $q \in Q$, either $((q)) \cap Q \subseteq \alpha$ or $((q)) \cap \alpha = \{\}$. \blacklozenge

Then, A is uniform, if the full subgraph of every equivalence class of \equiv contains only one type of non-simple transition and at most one kind of marking $*$ or \checkmark . Also, all states $q' \in Q$ or formulae φ occurring in $\delta(q, \sigma)$ for some $q \in Q$ and $\sigma \in \Sigma$ can be classified as unbounded, bounded with $*$, bounded with \checkmark , or simple – according to SCC $((q))$.

Example 51

Figure 6.1 depicts G_A for A of Example 50. p-automaton A is uniform:

- $((q_1)) = \{q_1, q_1 \vee \llbracket q_2 \rrbracket \geq 0.5\}$ and $((q_2)) = \{q_2, \llbracket q_2 \rrbracket \geq 0.5\}$;
- in $((q_1))$ there are no bounded edges, in $((q_2))$ there are no unbounded edges; and

- G_A has no markings for $*$ or \heartsuit .
- The SCC $(\llbracket q_1 \rrbracket_{\geq 0.5}) = \{\llbracket q_1 \rrbracket_{\geq 0.5}\}$ is *trivial*, i.e. it contains only a single state.

In addition, A is weak as $\alpha = \{q_2\}$. ◆

Intuitively, the cycles in the structure of a uniform p-automaton A take either no bounded edges or no unbounded edges, and cycles that take bounded edges do not have both markings $*$ and \heartsuit . Subsequently, all p-automata are uniform weak unless mentioned otherwise. Uniformity allows to define acceptance of an input for a p-automaton A through the solution of weak stochastic games. But it is a more relaxed notion of uniformity that really drives the proof of well-definedness: any chain in the partial order on SCCs on the graph of a p-automaton has only finitely many alternations between bounded and unbounded SCCs.

The requirement of weakness is made merely to simplify the presentation. Using a parity condition instead would still allow us to decide acceptance of an input for uniform p-automaton A , by solving stochastic parity games.

6.2 Acceptance games

Let MC_{AP} be the set of all Markov chains over AP . For AP , p-automata $A = \langle 2^{\text{AP}}, Q, \delta, \varphi^{\text{in}}, \alpha \rangle$ have MC_{AP} as set of inputs. For $M = (S, s^{\text{in}}) \in \text{MC}_{\text{AP}}$, we exploit the uniform structure of A to reduce the decision of whether A accepts M to solving a sequence of weak games and stochastic weak games. Intuitively, unbounded cycles in G_A correspond to weak stochastic games and bounded cycles to weak games. The weak acceptance of A implies that these games are weak. Then the language of A is

$$\mathcal{L}(A) = \{M \in \text{MC}_{\text{AP}} \mid A \text{ accepts } M\} .$$

Just as in the case of acceptance games for alternating tree automata, all states of A and all sub-formulae appearing in its transitions form part of the acceptance games.

For A as above, let

$$T = Q \cup \text{cl}_p(\delta(Q, 2^{\text{AP}})) .$$

Finite partial order $(T/\equiv, \leq_A)$ has set $\{((t)) \mid t \in T\}$ ordered by

$$((t)) \leq_A ((\tilde{t})) \text{ iff } \tilde{t} \preceq_A t .$$

For M as above, each non-trivial $((t))$ (i.e., there is a cycle in the subgraph of $((t))$ in G_A) determines a game

$$G_{M,((t))} = ((V, E), (V_0, V_1, V_p), \kappa, \tilde{\alpha})$$

The construction is such that $(s^{\text{in}}, \varphi^{\text{in}})$ occurs as configuration in exactly one of these games $G_{M,((t))}$, and $\text{val}(s^{\text{in}}, \varphi^{\text{in}}) \in [0, 1]$. Then

$$A \text{ accepts } M \text{ iff } \text{val}(s^{\text{in}}, \varphi^{\text{in}}) = 1 .$$

We define these games as follows. Since A is uniform weak, each $((t))$ is of one of three types and each type determines (at least) one weak game or weak stochastic game as detailed in the four cases below. All game values already computed for games $G_{M,((\tilde{t}))}$ of SCCs $((\tilde{t}))$ higher up with respect to \leq_A (i.e. by induction) are used as pre-seeded values in $G_{M,((t))}$. Below, we write $\text{val}(s, \varphi) = \perp$ for configurations (s, φ) in $G_{M,((t))}$ whose game value has not been computed yet.

Case 1: For an SCC $((t))$ such that none of its transitions are in E_b , game $G_{M,((t))}$ is a stochastic weak game with

$$\begin{aligned}
V &= \{(s, \tilde{t}) \mid s \in S \text{ and } \tilde{t} \preceq_A t\} \\
V_0 &= \{(s, \varphi_1 \vee \varphi_2) \in V\} \\
V_1 &= \{(s, \varphi_1 \wedge \varphi_2) \in V\} \\
V_p &= (S \times Q) \cap V \\
\kappa(s, q)(s', \delta(q, L(s))) &= P(s, s') \\
\tilde{\alpha} &= V \text{ iff some state (equally, all states)} \\
&\quad q \text{ in } ((t)) \text{ is in } \alpha \\
\tilde{\alpha} &= \{\} \text{ otherwise}
\end{aligned}$$

and

$$\begin{aligned}
E &= \{((s, \varphi_1 \wedge \varphi_2), (s, \varphi_i)) \in V \times V \mid i \in \{1, 2\}\} \cup \\
&\quad \{((s, \varphi_1 \vee \varphi_2), (s, \varphi_i)) \in V \times V \mid i \in \{1, 2\}\} \cup \\
&\quad \{((s, q), (s', \delta(q, L(s)))) \in V \times V \mid P(s, s') > 0\} .
\end{aligned}$$

By Theorem 1, for every configuration $c \in V$ we have $\text{val}_0(c) \in [0, 1]$. We set $\text{val}(c) = \text{val}_0(c)$.

Case 2: Let $((t))$ be a non-trivial SCC none of whose transitions are in E_u and none have \checkmark markings. For each formula $\varphi \in ((t)) \cap \llbracket Q \rrbracket^*$ of the form $\ast(\llbracket q_1 \rrbracket_{\bowtie_1 p_1}, \dots, \llbracket q_n \rrbracket_{\bowtie_n p_n})$ we define, for each $s \in S$, sets $V_0^{s, \varphi}$, $V_1^{s, \varphi}$, and $E^{s, \varphi}$ further below. Then

$$\begin{aligned}
V_i &= \bigcup_{s, \varphi} V_i^{s, \varphi} \\
E &= \bigcup_{s, \varphi} E^{s, \varphi} \\
V_p &= \{\} \\
\tilde{\alpha} &= V \text{ iff some state (equally, all states)} \\
&\quad q \text{ in } ((t)) \text{ is in } \alpha \\
\tilde{\alpha} &= \{\} \text{ otherwise}
\end{aligned}$$

defines the weak game $G_{M,((t))}$. It remains to define $V_0^{s,\varphi}$, $V_1^{s,\varphi}$, and $E^{s,\varphi}$, for which we make use of values $\text{val}(s, \tilde{t})$ already defined for all $s \in S$ and all $\tilde{t} \notin ((t))$ with $((t)) \leq_A ((\tilde{t}))$.

As $\text{Succ}(s)$ and $\delta(q_i, L(s))$ are finite, so are

$$R_{s,\varphi} = \bigcup_{i=1}^n \{(s', \varphi') \mid s' \in \text{Succ}(s), \varphi' \in \text{cl}_{\mathbf{p}}(\delta(q_i, L(s)))\}$$

$$\text{val}_{s,\varphi} = \{0, 1, \text{val}(s', \varphi') \mid (s', \varphi') \in R_{s,\varphi}, \text{val}(s', \varphi') \neq \perp\}$$

Intuitively, $R_{s,\varphi}$ is the set of configurations reachable from (s, φ) using *one* transition of a state in φ . Thus, s' are the successors of s and φ' are subformulae of $\delta(q_i, L(s))$. Set $\text{val}_{s,\varphi}$ includes 0, 1, and values of configurations in $R_{s,\varphi}$. In game $G_{M,((t))}$, a play proceeding from (s, φ) reaches either a configuration whose value is in $\text{val}_{s,\varphi}$ or a configuration (s, ψ) for $\psi \in ((t))$. Before we formally define sets $V_0^{s,\varphi}$, $V_1^{s,\varphi}$ and $E^{s,\varphi}$, and functions $\mathcal{F}_{s,\varphi}^*$, we first give an intuitive description of the game.

The intuition behind this weak game is as follows: Configuration (s, φ) means that the transition of each q_i holds with probability $\bowtie_i p_i$ where the sets X_i measured by these probabilities are pairwise disjoint. In order to check that, given configuration (s, φ) , Player 0 chooses a function $f \in \mathcal{F}_{s,\varphi}^*$ (formally defined below) that associates with location $s' \in \text{Succ}(s)$ and state q_i the value Player 0 can achieve playing from $(s', \delta(q_i, L(s)))$. The play continues with Player 1 choosing a successor s' of s and a state q_i , and the play then reaches configuration $(s', \delta(q_i, L(s)), f(i, s'))$. From such value-annotated configurations, Player 0 and Player 1 choose successors according to the usual resolution of \vee and \wedge :

- In a configuration for which the value v was already determined, either $f(i, s') \bowtie_i v$, i.e. Player 0 achieved the promised value and wins immediately; or Player 0 failed to achieve the promised value and loses immediately.
- Otherwise, the play ends up in another configuration of the form (s', φ') for $\varphi' \in \llbracket Q \rrbracket^*$ and the play continues and ignores the value $f(i, s')$ (as obviously $f(i, s') \leq 1$). If the play continues ad infinitum, the winner is determined according to acceptance condition $\tilde{\alpha}$.

It remains to define sets $V_0^{s,\varphi}$, $V_1^{s,\varphi}$, and $E^{s,\varphi}$ below. For $n \in \mathbb{N}$, let

$$[n] = \{1, \dots, n\}.$$

$$\begin{aligned} V_0^{s,\varphi} &= \{(s, \varphi)\} \cup \\ &\{(s', \varphi', v) \mid s' \in \text{Succ}(s), \varphi' \in R_{s,\varphi}, \text{val}(s', \varphi') \neq \perp, \text{val}(s', \varphi') < v\} \cup \\ &\{(s', \varphi_1 \vee \varphi_2, v) \mid s' \in \text{Succ}(s), \varphi_1 \vee \varphi_2 \in R_{s,\varphi}, \text{val}(s', \varphi_1 \vee \varphi_2) = \perp\} \end{aligned}$$

$$\begin{aligned} V_1^{s,\varphi} &= \{(s, \varphi, f) \mid f \in \mathcal{F}_{s,\varphi}^*\} \cup \\ &\{(s', \varphi', v) \mid s' \in \text{Succ}(s), \varphi' \in R_{s,\varphi}, \text{val}s'\varphi' \neq \perp, \text{val}(s', \varphi') \geq v\} \cup \\ &\{(s', \varphi_1 \wedge \varphi_2, v) \mid s' \in \text{Succ}(s), \varphi_1 \wedge \varphi_2 \in R_{s,\varphi}, \text{val}(s', \varphi_1 \wedge \varphi_2) = \perp\} \end{aligned}$$

$$\begin{aligned} E^{s,\varphi} &= \{((s, \varphi), (s, \varphi, f)) \mid f \in \mathcal{F}_{s,\varphi}^*\} \cup \\ &\{(s', \varphi', v), (s', \varphi') \mid s' \in \text{Succ}(s), \varphi' \in \llbracket Q \rrbracket \text{val}(s', \varphi') = \perp\} \cup \\ &\{((s, \varphi, f), (s', \delta(q_i, L(s)), f(i, s'))) \mid s' \in \text{Succ}(s), i \in [n], f(i, s') > 0\} \cup \\ &\{(s', \varphi_1 \vee \varphi_2, v), (s', \varphi_i, v) \mid s' \in \text{Succ}(s), \varphi_1 \vee \varphi_2 \in R_{s,\varphi}, i \in \{1, 2\}, \\ &\quad \text{val}(s', \varphi_1 \vee \varphi_2) = \perp\} \cup \\ &\{(s', \varphi_1 \wedge \varphi_2, v), (s', \varphi_i, v) \mid s' \in \text{Succ}(s), \varphi_1 \wedge \varphi_2 \in R_{s,\varphi}, i \in \{1, 2\}, \\ &\quad \text{val}(s', \varphi_1 \wedge \varphi_2) = \perp\} \end{aligned}$$

We now define the function space $\mathcal{F}_{s,\varphi}^*$ that captures terms built from the separation operator $*$. Throughout, let $X \rightarrow Y$ be the set of total functions from set X to set Y . Let $\mathcal{F}_{s,\varphi}$ be $[n] \times \text{Succ}(s) \rightarrow \text{val}_{s,\varphi}$, the set of functions from pairs consisting of ‘sub-stars’ of φ and successors of s to values in $\text{val}_{s,\varphi}$.

Definition 69 (Disjoint function)

A function $f \in \mathcal{F}_{s,\varphi}$ is *disjoint* if there are $\{a_{i,s'} \in [0, 1] \mid i \in [n] \text{ and } s' \in \text{Succ}(s)\}$ such that

- (i) $\sum_{s' \in \text{Succ}(s)} a_{i,s'} f(i, s') P(s, s') \bowtie_i p_i$ for all $i \in [n]$; and
- (ii) $\sum_{i \in [n]} a_{i,s'} = 1$ for all $s' \in \text{Succ}(s)$.

We denote by $\mathcal{F}_{s,\varphi}^*$ the set of disjoint functions. ◆

Intuitively, a function $f \in \mathcal{F}_{s,\varphi}$ associates with q_1, \dots, q_n and s' the value that Player 0 can achieve from configuration $(s', \delta(q_i, L(s)))$. We call f “disjoint”, as all the requirements from the different q_i can be achieved using a partition of the probability of all successors.

By Theorem 1, V partitions into winning regions W_0 and W_1 of configurations for Player 0 and Player 1, respectively. We set $\text{val}(c) = 1$ for $c \in W_0$ and $\text{val}(c) = 0$ for $c \in W_1$.

Case 3: Finally, let $((t))$ be an SCC such that none of its transitions is in E_u and none has $*$ markings. For formulae $\varphi \in ((t)) \cap \llbracket Q \rrbracket^\forall$ of form $\forall (\llbracket q_1 \rrbracket_{\bowtie_1 p_1}, \dots, \llbracket q_n \rrbracket_{\bowtie_n p_n})$ we reuse the definitions of $R_{s,\varphi}$, $\text{val}_{s,\varphi}$, and $\mathcal{F}_{s,\varphi}$. Weak game $G_{M,((t))}$ is defined as in Case 2. Sets $V_0^{s,\varphi}$, $V_1^{s,\varphi}$, and $E^{s,\varphi}$ are as before except that functions f do not range over $\mathcal{F}_{s,\varphi}^*$ but now range over the set of intersecting functions (the dual of $\mathcal{F}_{s,\varphi}^*$ of Case 2).

Definition 70 (Intersecting function)

A function $f \in \mathcal{F}_{s,\varphi}$ is *intersecting* if for all sets $\{a_{i,s'} \in [0, 1] \mid i \in [n] \text{ and } s' \in \text{Succ}(s)\}$ either

- (i) there is $i \in [n]$ with $\sum_{s' \in \text{Succ}(s)} a_{i,s'} f(i, s') P(s, s') \bowtie_i p_i$; or
- (ii) there is $s' \in \text{Succ}(s)$ with $\sum_{i \in [n]} a_{i,s'} \neq 1$.

We denote $\mathcal{F}_{s,\varphi}^\forall$ the set of intersecting functions. ◆

As in Case 2, we say that wins for Player 0 have value 1, and wins for Player 1 have value 0.

The intuition for this weak game is the same as that of the weak game in Case 2, except that Player 0 chooses a function f that is in $\mathcal{F}_{s,\varphi}^\forall$ instead of in $\mathcal{F}_{s,\varphi}^*$.

We point out that when n above is 1, i.e. in handling $\varphi = \llbracket q_1 \rrbracket_{\bowtie_1 p_1}$, the definition of $*$ and \forall coincide as then there is no difference between a universal and an existential choice. Indeed, there is then exactly one option for choosing $\{a_{1,s'} \mid s' \in \text{Succ}(s)\}$: the value $a_{1,s'}$ has to be 1 for all $s' \in \text{Succ}(s)$. This justifies dropping the $*$ or \forall when applied to one operand.

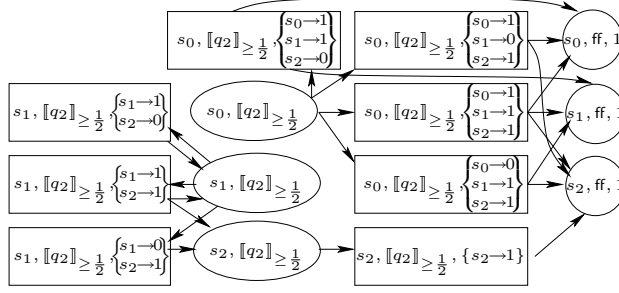


Figure 6.3: Case 3 of acceptance game.

Case 4: Trivial SCCs $((t))$, for which $((t)), E \cup E_b \cup E_u$ is cycle-free, may satisfy more than one of the three cases above. This ambiguity is unproblematic as game values in $G_{M,((t))}$ are then determined via propagation of pre-seeded game values.

Example 52

We verify $M \in \mathcal{L}(A)$ for A from Example 50 and M from Figure 6.2 on page 164, where propositions that hold at location s are written within that location – e.g. $L(s_0) = \{\mathbf{a}\}$. The weak game of SCC $((q_2))$, shown in Figure 6.3, has only accepting configurations. So Player 0 wins only $(s_1, [q_2] \geq 0.5)$ and $(s_1, [q_2] \geq 0.5, \{s_1 \rightarrow 1, s_2 \rightarrow 0\})$ and loses all other configurations.

The stochastic weak game $G_{M,((q_1))}$ for the SCC $((q_1))$, shown in Figure 6.4, depicts stochastic configurations with a diamond and configurations from other SCCs are put into hexagons (with the hexagon labelled $(s_1, [q_2] \geq 0.5)$ having value 1 and all others having value 0). As none of the configurations are accepting, Player 0 can only win by reaching optimal hexagons. Hexagon $(s_1, [q_2] \geq 0.5)$ has value 1 and is the optimal choice for Player 0 from configuration $(s_1, q_1 \vee [q_2] \geq 0.5)$. Player 0 configuration $(s_2, q_1 \vee [q_2] \geq 0.5)$ has value 0. So the value for Player 0 of diamond configuration (s_0, q_1) is 0.5. Initial configuration $(s_0, [q_1] \geq 0.5)$ makes up a trivial bounded SCC (e.g. Case 2), so its value equals 1 as $\frac{1}{3}\text{val}(s_0, q_1 \vee [q_2] \geq 0.5) + \frac{1}{3}\text{val}(s_1, q_1 \vee [q_2] \geq 0.5) + \frac{1}{3}\text{val}(s_2, q_1 \vee [q_2] \geq 0.5)$ is 0.5. Therefore, $M \in \mathcal{L}(A)$. ♦

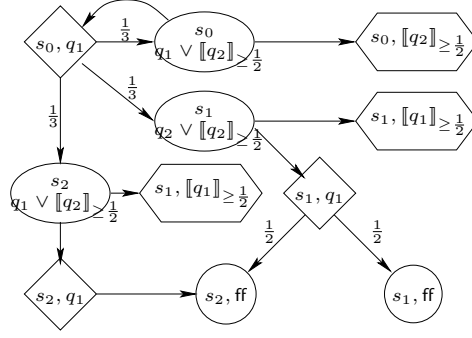


Figure 6.4: Case 1 of acceptance game.

Theorem 6

Given a p -automaton A with alphabet 2^{AP} , its language $\mathcal{L}(A)$ is well defined. If A and $M \in \text{MC}_{\text{AP}}$ are finite with size $|A|$ and $|M|$ respectively, $M \in \mathcal{L}(A)$ can be decided in EXPTIME with regard to $|A| + |M|$. ●

Proof

Well definedness of acceptance follows directly from Theorem 1. For finite Markov chain M and finite p -automaton A we observe the following:

- The stochastic weak game arising from the combination of a Markov chain M and an unbounded SCC can be solved in $\text{NP} \cap \text{co-NP}$.
- The weak game arising from the combination of Markov chain M and a bounded SCC can be solved in linear time but may have exponential size due to the number of possible *disjoint* and *intersecting* functions f .

Thus, membership $M \in \mathcal{L}(A)$ is decidable by a sequence of weak games and stochastic weak games, which can be solved in EXPTIME with regard to $|A| + |M|$. ■

For finite Markov chains M and p -automaton A , checking acceptance $M \in \mathcal{L}(A)$ is exponential in the branching degree of M and $*$ operators of A , but not in the number of states or locations. If A has only trivial bounded SCCs, checking $M \in \mathcal{L}(A)$ reduces to solving a linear number of linear sized stochastic weak games.

The EXPTIME complexity of the acceptance game is certainly a drawback for the practical application of p-automata for probabilistic verification. The complexity of PCTL model checking for example is polynomial. But as we show in Theorem 9 below, at least for a certain class of p-automata, i.e. the ones which correspond to PCTL formulae, the complexity of p-automata acceptance matches the complexity of PCTL model checking. There might be other interesting classes of p-automata for which the EXPTIME complexity of acceptance can be improved. In fact, we do not yet know whether or not the EXPTIME bound is optimal even for the most general case of p-automata.

6.3 Expressiveness of p-automata

To address the main research question of this thesis, i.e. the PCTL completeness problem, p-automata need to fulfill certain expressiveness requirements. In particular we would like to be able to phrase the completeness question fully within the p-automata formalism. In this section we therefore show the following properties of p-automata:

- (i) the languages of p-automata are closed under Boolean operations;
- (ii) the languages of p-automata are closed under bisimulation;
- (iii) emptiness and containment of languages are inter-reducible;
- (iv) every Markov chain determines a p-automaton whose language is the bisimulation class of that Markov chain; and
- (v) every PCTL formula determines a p-automaton whose language consists of all Markov chains satisfying that formula.

6.3.1 Closure of languages

It is routine to see that p-automata are closed under union and intersection. They are also closed under complementation: Given a p-automaton $A = \langle \Sigma, Q, \delta, \varphi^{\text{in}}, \alpha \rangle$, its dual is

$$\text{dual}(A) = \langle \Sigma, \bar{Q}, \bar{\delta}, \text{dual}(\varphi^{\text{in}}), Q \setminus \alpha \rangle$$

with $\bar{Q} = \{\bar{q} \mid q \in Q\}$ and $\bar{\delta}(\bar{q}, \sigma) = \text{dual}(\delta(q, \sigma))$, where $\text{dual}(\varphi)$ is defined as follows:

$$\text{dual}(\forall(\varphi_1, \dots, \varphi_n)) = *(\text{dual}(\varphi_1), \dots, \text{dual}(\varphi_n))$$

$$\text{dual}(*(\varphi_1, \dots, \varphi_n)) = \forall(\text{dual}(\varphi_1), \dots, \text{dual}(\varphi_n))$$

$$\text{dual}(\varphi_1 \wedge \varphi_2) = \text{dual}(\varphi_1) \vee \text{dual}(\varphi_2)$$

$$\text{dual}(\varphi_1 \vee \varphi_2) = \text{dual}(\varphi_1) \wedge \text{dual}(\varphi_2)$$

$$\text{dual}(q) = \bar{q}$$

$$\text{dual}(\bar{q}) = q$$

$$\text{dual}(\llbracket q \rrbracket_{\geq p}) = \llbracket \bar{q} \rrbracket_{\text{dual}(\geq p)}$$

$$\text{dual}(\geq p) = >1 - p$$

$$\text{dual}(>p) = \geq 1 - p$$

The structure of uniform weak p-automata ensures that $\text{dual}(A)$ is also uniform weak. We identify $\bar{\bar{q}}$ with q and so $\text{dual}(\text{dual}(A)) = A$ follows. The class of uniform weak p-automata is thus closed under dualisation, which is important for this thesis, as we restrict ourselves to uniform weak p-automata.

We now show that A and $\text{dual}(A)$ are complements.

Theorem 7

Let A be a p-automaton A over 2^{AP} . Then $\mathcal{L}(A) = \text{MC}_{\text{AP}} \setminus \mathcal{L}(\text{dual}(A))$. ●

The key part of the proof for that theorem is to show that, for every state q of A and every location s of M , we have $\text{val}(s, q) = 1 - \text{val}(s, \bar{q})$ for the acceptance games.

Proof

We prove a stronger claim, namely that for every $s \in S$ and $\varphi \in \text{cl}_p(\delta(Q), \Sigma)$ we have

$$\text{val}(s, \varphi) = 1 - \text{val}(s, \text{dual}(\varphi))$$

The proof is by induction on the structure of the automaton. Consider an equivalence class $((t))$ in A . Assume by induction that the claim holds for all the SCCs in A that are greater than $((t))$.

- If $((t))$ is a trivial SCC, the lemma follows from the dualisation and the duality of min and max.
- Suppose that $((t))$ is a nontrivial SCC and that all transitions in $((t))$ are unbounded. Then, the lemma follows from the dualisation and the determinacy of stochastic weak games.
- Suppose that $((t))$ is a nontrivial SCC and that no transition in $((t))$ is in the scope of \forall . It follows that $((\text{dual}(t)))$ is also a nontrivial SCC and that no transition in $((\text{dual}(t)))$ is in the scope of $*$.

Given a strategy for Player 0 in $G_{M,((t))}$, we show how to construct a strategy for Player 1 in $G_{M,((\text{dual}(t)))}$. The two strategies produce plays that are always in the same locations of the Markov chain M and same states of the automaton A (modulo dualisation $t \mapsto \text{dual}(t)$). For simplicity we denote $G_{M,((t))}$ by G and $G_{M,((\text{dual}(t)))}$ by \overline{G} .

Consider two matching configurations (s, φ) and $(s, \text{dual}(\varphi))$ in G and \overline{G} . Let $\varphi = *(\llbracket q_1 \rrbracket_{\bowtie_1 p_1}, \dots, \llbracket q_n \rrbracket_{\bowtie_n p_n})$, where $n > 1$. Consider the configuration $(s, \text{dual}(\varphi))$. By playing for Player 1 in G we make Player 0 ‘reveal’ her strategy in G and using her strategy we react to the moves of Player 0 in \overline{G} by constructing a strategy for Player 1 in \overline{G} .

Consider two plays ending in (s, φ) and $(s, \text{dual}(\varphi))$. Let $f: [n] \times \text{Succ}(s) \rightarrow \text{val}_{s, \varphi}$ be the function chosen by Player 0 in G and let $f': [n] \times \text{Succ}(s) \rightarrow \text{val}_{s, \text{dual}(\varphi)}$ be the function chosen by Player 0 in \overline{G} . By definition there are $\{a_{i, s'}\}$ that witness the disjointness of f and for every i we have

$$\sum_{s' \in \text{Succ}(s)} a_{i, s'} \cdot P(s, s') \cdot f(i, s') \bowtie_i p_i.$$

By using the same $\{a_{i, s'}\}$ stemming from the fact that f' is intersecting, we get that there is some i such that

$$\sum_{s' \in \text{Succ}(s)} a_{i, s'} \cdot P(s, s') \cdot f'(i, s') \text{dual}(\bowtie_i) 1 - p_i.$$

It follows that there is an $s' \in \text{Succ}(s)$ such that $f(i, s') + f'(i, s') > 1$. It is now Player 1’s turn to move in both G and \overline{G} . In G we make Player 1 choose $(s', \delta(q_i, L(s)), f(i, s'))$ and the strategy for Player 1

in \overline{G} is extended by $(s', \text{dual}(\delta(q_i, L(s))), f'(i, s'))$. We now proceed by utilising the duality between \vee and \wedge to use Player 0's choices in \overline{G} to suggest moves for Player 1 in G and use Player 0's strategy in G to suggest how to extend the strategy for Player 1 in \overline{G} .

Suppose that we reach configurations

$$(s', \varphi', f(i, s')) \text{ and } (s', \text{dual}(\varphi'), f'(i, s'))$$

such that

$$\text{val}(s', \varphi') \neq \perp \text{ and } \text{val}(s', \text{dual}(\varphi')) \neq \perp .$$

Then, by assumption $\text{val}(s', \varphi') = 1 - \text{val}(s', \text{dual}(\varphi'))$ and thus if $\text{val}(s', \varphi') \geq f(s')$ it must be the case that $\text{val}(s', \text{dual}(\varphi')) < f'(s')$.

If we do not reach such configurations, the game proceeds to a new configuration in $S \times \llbracket Q \rrbracket$. If the two plays are infinite, then by the duality of α and $Q \setminus \alpha$ if Player 0 wins the play in G then Player 1 wins the play in \overline{G} .

Showing that a win of Player 1 in G is translated to a win of Player 0 in \overline{G} is similar, and so omitted.

- The case that $((t))$ is a nontrivial SCC and that some transitions in $((q))$ are in scope of \forall is similar, and so omitted. ■

Corollary 4

1. The set of languages accepted by p-automata over 2^{AP} is closed under Boolean operations.
2. Language containment of p-automata over 2^{AP} reduces to language emptiness of such p-automata, and vice versa. ●

Proof

1. Showing that these languages are closed under intersections and unions is trivial because the transition function includes conjunction and disjunction. By Theorem 7, these languages are closed under complements.

2. Given two p-automata A_1 and A_2 , we have

$$\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2) \text{ iff } \mathcal{L}(A_1) \cap \mathcal{L}(\text{dual}(A_2)) = \{\} .$$

Therefore, checking language containment reduces to checking language emptiness, as p-automata are closed under intersection. Conversely, we can construct a p-automaton E such that $\mathcal{L}(E) = \{\}$. The language of A is empty iff $\mathcal{L}(A) \subseteq \mathcal{L}(E)$. ■

Next we show that languages of p-automata are closed under bisimulation.

Lemma 14

For p-automaton $A = \langle 2^{\text{AP}}, Q, \delta, \varphi^{\text{in}}, \alpha \rangle$ and $M_1, M_2 \in \text{MC}_{\text{AP}}$ with $M_1 \sim M_2$:
 $M_1 \in \mathcal{L}(A)$ iff $M_2 \in \mathcal{L}(A)$. ●

To prove this, we use induction on the partial order on the SCCs in A to show that for all $t \in Q \cup \llbracket Q \rrbracket$ and for all locations s_1 in M_1 and locations s_2 in M_2 with $s_1 \sim s_2$ we have $\text{val}(s_1, t) = \text{val}(s_2, t)$.

Proof

Let $M_i = (S_i, \mathbf{P}_i, L_i, s_i^{\text{in}})$, for $i \in \{1, 2\}$, with the same set of labels AP . Let $A = \langle \Sigma, Q, \delta, \llbracket q_0 \rrbracket_{\times p}, \alpha \rangle$, where $\Sigma = 2^{\text{AP}}$. Let $\sim \subseteq S_1 \times S_2$ be the maximal bisimulation between M_1 and M_2 .

We show that for every state $q \in Q$ and locations $s_1 \in S_1$, and $s_2 \in S_2$ such that $s_1 \sim s_2$, we have $\text{val}(s_1, q) = \text{val}(s_2, q)$. We prove this claim by induction on the partial order on the SCCs in A . Suppose that the claim holds for all SCCs greater than $((q))$ in the partial order. Consider the games $G_{M_1, ((q))}$ and $G_{M_2, ((q))}$. Consider a winning strategy σ for Player 0 in $G_{M_1, ((q))}$. We show how this is also a winning strategy for Player 0 in $G_{M_2, ((q))}$.

Consider a play in an unbounded SCC $((q))$. We build by induction a play in $G_{M_1, ((q))}$ and a play in $G_{M_2, ((q))}$ with the invariant that the plays end in configurations of the form (s_1, t) and (s_2, t) such that $s_1 \sim s_2$. Clearly, the initial configuration in both games satisfies this invariant. We show how to extend the play to maintain this invariant. If t is of the form $\varphi_1 \wedge \varphi_2$ and Player 1 chooses φ_i in $G_{M_2, ((q))}$, then we emulate the same choice in

$G_{M_1,((q))}$. If t is of the form $\varphi_1 \vee \varphi_2$, then σ instructs Player 0 to choose φ_i in $G_{M_1,((q))}$ and we emulate the same choice in $G_{M_2,((q))}$. If t is of the form q' for some state $q' \in Q$ then choices in (s_1, q') and (s_2, q') are resolved by the stochastic player. As $s_1 \sim s_2$ the successors of s_1 and s_2 can be partitioned to equivalence classes such that for each equivalence class C_1 in M_1 and C_2 in M_2 we have $P_1(s_1, C_1) = P_2(s_2, C_2)$. Consider now the probability measure of plays in $G_{M_1,((q))}$ and $G_{M_2,((q))}$ that are winning according to this composed strategy. The plays can be partitioned according to bisimulation equivalence classes and every choice has the same weight. It follows that the probability measure of winning plays is identical in both games.

Consider a play in a bounded SCC $((q))$ where no transition uses \forall . Disjunctions and conjunctions are handled as above. Consider a pair of configurations

$$(s_1, t) \text{ and } (s_2, t)$$

where $s_1 \sim s_2$ and t is of the form

$$*([\![q_1]\!]_{\times_1 p_1}, \dots, [\![q_n]\!]_{\times_n p_n}) .$$

Let f_1 be the function chosen by Player 0 in $G_{M_1,((q))}$. As $s_1 \sim s_2$, we can find a function f_2 such that for every s'_2 we have $f_2(i, s'_2) = f_1(i, s'_1)$ for some $s'_1 \sim s'_2$ that satisfies the requirement of the game. Next, Player 1 chooses a state $s' \in \text{Succ}(s_2)$ and a state q_i . The same choice can be mimicked in $G_{M_1,((q))}$. As $s_1 \sim s_2$, it follows that $L(s_1) = L(s_2)$ and the automaton component in both configurations remains the same.

The treatment of a play in a bounded SCC $((q))$ where some transitions use \forall is similar, and so omitted. ■

6.3.2 Embedding of Markov chains

A Markov chain $M = (S, \mathbf{P}, L, s^{\text{in}}) \in \text{MC}_{\text{AP}}$ can be converted into a p-automaton

$$A_M = \langle 2^{\text{AP}}, Q, \delta, \varphi^{\text{in}}, \alpha \rangle$$

whose language $\mathcal{L}(A_M)$ is the set of Markov chains bisimilar to M . The definition of A_M implicitly appeals in $*$ expressions to an enumeration of

each set $\text{Succ}(s')$:

$$\begin{aligned}
Q &= \{(s, s') \in S \times S \mid P(s, s') > 0\} \\
\delta((s, s'), L(s)) &= *(\llbracket (s', s'') \rrbracket_{\geq P(s', s'')} \mid s'' \in \text{Succ}(s')) \\
\delta((s, s'), \sigma) &= \text{ff} \quad \text{if } \sigma \neq L(s) \\
\varphi^{\text{in}} &= *(\llbracket (s^{\text{in}}, s') \rrbracket_{\geq P(s^{\text{in}}, s')} \mid P(s^{\text{in}}, s) > 0) \\
\alpha &= Q
\end{aligned}$$

State (s, s') represents the transition from s to s' . Labels are compared for location s . Location s' is used to require that there be successors of probability at least $P(s, s')$. This p-automaton A_M has only bounded transitions and uses only the $*$ operator. In particular, it is uniform weak.

Theorem 8

For any Markov chain $M \in \text{MC}_{\text{AP}}$, the language $\mathcal{L}(A_M)$ is the bisimulation equivalence class of M . ●

By Lemma 14, one half of Theorem 8 follows from a proof that A_M accepts M . To show this, it suffices to demonstrate that Player 0 can infinitely often reach configurations of the form $(s, *(\llbracket (s, s') \rrbracket_{\geq P(s, s')}))$ with $s' \in \text{Succ}(s)$ for all locations s in M . For the other half, we use proof by contradiction: given M' with initial state t^{in} such that $M' \not\sim M$, we appeal to the partition refinement algorithm to get a coarsest partition that witnesses $s^{\text{in}} \not\sim t^{\text{in}}$. That witnessing information can then be transformed into a winning strategy for Player 1 in the acceptance game for deciding $M' \in \mathcal{L}(A_M)$, and so $M' \notin \mathcal{L}(A_M)$ follows.

Proof

1. By Lemma 14, we know that $M' \sim M$ implies $M' \in \mathcal{L}(A_M)$ as soon as we have that $M \in \mathcal{L}(A_M)$. To simplify the proof of $M \in \mathcal{L}(A_M)$, we assume that all locations of M are in one SCC. (The general case follows by induction on the reachability order of SCCs.)

Consider a location $s \in S$ and $(s, s') \in Q$. Let $\varphi_s = *(\llbracket (s, s') \rrbracket_{\geq P(s, s')} \mid s' \in \text{Succ}(s))$. We show that from a configuration of the form (s, φ_s) , Player 0 has a strategy that keeps returning to configurations of this

form. As $\alpha = Q$, Player 0 can continue playing forever and wins. We start from the configuration (s, φ_s) . Let

$$\varphi_s = *([\![s, s_1]\!]_{\geq P(s, s_1)}, \dots, [\![s, s_n]\!]_{\geq P(s, s_n)}) .$$

Then Player 0 chooses the function $f: [n] \times \text{Succ}(s) \rightarrow \{0, 1\}$ such that $f(i, s') = 1$ iff $s_i = s'$. The trivial assignment $a_{i, s'} = 1$ iff $s_i = s'$ shows that f is disjoint. Then, Player 1 chooses a successor $(s_i, \delta((s, s_i), L(s)), 1)$. As $\delta((s, s_i), L(s)) = \varphi_{s_i}$ the claim follows and Player 0 has a strategy to continue the play forever.

The initial configuration in the game is

$$*([\![s^{\text{in}}, s']\!]_{\geq P(s^{\text{in}}, s')} \mid s' \in \text{Succ}(s^{\text{in}})) .$$

The same intuition as above shows that this is winning for Player 0 as well.

2. Conversely, if $M' \not\sim M$ we show that $M' \notin \mathcal{L}(A_M)$. Let $M = (S, \mathbf{P}, L, s^{\text{in}})$ and $M' = (T, \mathbf{P}, L, t^{\text{in}})$. To simplify notations we assume that $S \cap T = \{\}$ and use \mathbf{P} and L for the probability matrix and labelling function of both Markov chains. We use the partition refinement algorithm that computes the bisimulation equivalence sets for a Markov chain. Let

$$\Xi_0 = \{S' \subseteq S \cup T \mid \forall s, s' \in S': L(s) = L(s') \text{ and } S' \text{ is maximal}\} .$$

Clearly, Ξ_0 is a partition of $S \cup T$. Let Ξ_{i+1} be the coarsest partition of $S \cup T$ that refines Ξ_i and in addition for every $G \in \Xi_{i+1}$, for every $s, s' \in G$, and for every $G' \in \Xi_i$ we have $P(s, G') = P(s', G')$. It is well known that if $s \not\sim s'$ there is some $i_{s, s'}$ such that s and s' belong to different sets in $\Xi_{i_{s, s'}}$ [60].³

By assumption, $s^{\text{in}} \not\sim t^{\text{in}}$. Let i_0 be minimal such that s^{in} and t^{in} are in different sets in Ξ_{i_0} . Denote $s_{i_0} = s^{\text{in}}$, $t_{i_0} = t^{\text{in}}$, $\varphi_{i_0} = \varphi^{\text{in}}$, and

³As our Markov chains have only finite branching, it is enough to consider $i_{s, s'} \in \mathbb{N}$. Otherwise, we would need to use transfinite induction.

$c_{i_0} = (t_{i_0}, \varphi_{i_0})$. Consider the configuration $c_{i_j} = (t_{i_j}, \varphi_{i_j})$, where

$$\varphi_{i_j} = \underset{s' \in \text{Succ}(s_{i_j})}{*} \llbracket (s_{i_j}, s') \rrbracket_{\geq P(s_{i_j}, s')}$$

and s_{i_j} and t_{i_j} are in different sets in Ξ_{i_j} . We show that from configuration c_{i_j} Player 1 either wins immediately or finds a similar configuration for $i_{j+1} < i_j$.

If $i_j = 0$, then $L(t_{i_j}) \neq L(s_{i_j})$. Regardless of the immediate choices of *Player* 0, we have $\delta((s_{i_j}, s'), L(t_{i_j})) = \text{ff}$ and *Player* 1 wins.

Otherwise, $i_j > 0$. By assumption, there is some $i_{j+1} < i_j$ and $G \in \Xi_{i_{j+1}}$ such that $P(s_{i_j}, G) \neq P(t_{i_0}, G)$. Without loss of generality we assume that $P(s_{i_j}, G) > P(t_{i_j}, G)$. Indeed, if $P(s_{i_j}, G) < P(t_{i_j}, G)$, then as $P(s_{i_j}, S) = 1$ there must be a different set $G' \in \Xi_{i_{j+1}}$ such that $P(s_{i_j}, G') > P(t_{i_j}, G')$.

Let $S_{i_{j+1}} = G \cap S$. Let $(t_{i_j}, \varphi_{i_j}, f)$ be the configuration chosen by *Player* 0. By disjointness of f , and as $P(t_{i_j}, G) < P(s_{i_j}, G)$, there must be a location $s_{i_{j+1}} \in G$ and a location $t_{i_{j+1}} \notin G$ such that $f(t_{i_{j+1}}, s_{i_{j+1}}) > 0$. *Player* 1 chooses $c_{i_{j+1}} = (t_{i_{j+1}}, \varphi_{i_{j+1}}, v)$, where $\varphi_{i_{j+1}} = \delta((s_{i_j}, s_{i_{j+1}}), L(t_{i_j}))$. As $t_{i_{j+1}} \notin G$, *Player* 1 has forced the game to a similar configuration with $i_{j+1} < i_j$ and eventually wins by reaching Ξ_0 . ■

The construction of A_M is the only reason why we allow p-automata with infinite state sets. Finite sets Q suffice for finite Markov chains. The conjunctive operator $*$ used in the construction of A_M effectively hides an exponential blowup. If a Markov chain is label-deterministic (all successors s' of location s disagree on their labellings), we can eliminate the use of $*$ in A_M and still secure Theorem 8. But this embedding without $*$ does break Theorem 8 for Markov chains which are not label-deterministic (as is shown in Section 6.5 below).

6.3.3 Embedding of PCTL formulae

Each PCTL formula ϕ over AP yields a p-automaton A_ϕ , without $*$ markings,

$$A_\phi = \langle 2^{\text{AP}}, \text{cl}_t(\phi) \cup \text{AP}, \rho_X, \rho_\epsilon(\phi), F \rangle$$

that accepts exactly those Markov chains which satisfy ϕ . The construction resembles the translation from CTL to alternating tree automata:

- $\text{cl}_t(\phi)$ denotes the set of temporal subformulae of ϕ ;
- F consists of AP and all ψ of $\text{cl}_t(\phi)$ *not* of the form $\psi_1 \text{ U } \psi_2$;
- transition function ρ_X unfolds fixpoints and replaces the threshold context $[\cdot]_{\triangleright p}$ with $\llbracket \cdot \rrbracket_{\triangleright p}$. That replacement is also done by function ρ_ϵ for the initial condition. The effect of these functions is similar to that achieved by using ϵ transitions to translate CTL formulae into two-way tree automata. Function ρ_X and auxiliary function ρ_ϵ are defined at follows:

$$\begin{aligned} \rho_X(\mathbf{a}, \sigma) &= \text{tt if } \mathbf{a} \in \sigma \\ \rho_X(\mathbf{a}, \sigma) &= \text{ff if } \mathbf{a} \notin \sigma \\ \rho_X(\neg \mathbf{a}, \sigma) &= \text{tt if } \mathbf{a} \notin \sigma \\ \rho_X(\neg \mathbf{a}, \sigma) &= \text{ff if } \mathbf{a} \in \sigma \\ \\ \rho_X(\mathbf{X} \varphi_1, \sigma) &= \rho_\epsilon(\varphi_1) \\ \rho_X(\varphi_1 \text{ U } \varphi_2, \sigma) &= (\rho_\epsilon(\varphi_1) \wedge \varphi_1 \text{ U } \varphi_2) \vee \rho_\epsilon(\varphi_2) \\ \rho_X(\varphi_1 \text{ W } \varphi_2, \sigma) &= (\rho_\epsilon(\varphi_1) \wedge \varphi_1 \text{ W } \varphi_2) \vee \rho_\epsilon(\varphi_2) \\ \\ \rho_\epsilon(\mathbf{a}) &= \mathbf{a} \\ \rho_\epsilon(\neg \mathbf{a}) &= \neg \mathbf{a} \\ \\ \rho_\epsilon(\varphi_1 \vee \varphi_2) &= \rho_\epsilon(\varphi_1) \vee \rho_\epsilon(\varphi_2) \\ \rho_\epsilon(\varphi_1 \wedge \varphi_2) &= \rho_\epsilon(\varphi_1) \wedge \rho_\epsilon(\varphi_2) \\ \\ \rho_\epsilon([\mathbf{X} \varphi_1]_{\triangleright p}) &= \llbracket \mathbf{X} \varphi_1 \rrbracket_{\triangleright p} \\ \rho_\epsilon([\varphi_1 \text{ U } \varphi_2]_{\triangleright p}) &= (\rho_\epsilon(\varphi_1) \wedge \llbracket \varphi_1 \text{ U } \varphi_2 \rrbracket_{\triangleright p}) \vee \rho_\epsilon(\varphi_2) \\ \rho_\epsilon([\varphi_1 \text{ W } \varphi_2]_{\triangleright p}) &= (\rho_\epsilon(\varphi_1) \wedge \llbracket \varphi_1 \text{ W } \varphi_2 \rrbracket_{\triangleright p}) \vee \rho_\epsilon(\varphi_2) \end{aligned}$$

We have $\psi \in \text{cl}_t(\phi)$ for subformulae $[\psi]_{\triangleright p}$ of ϕ . Also, $[\psi_1 \text{ U } \psi_2]_{\triangleright p}$ may be an element in $\text{cl}_t(\phi)$ whereas $\llbracket \psi_1 \text{ U } \psi_2 \rrbracket_{\triangleright p}$ can only be an element of

$\llbracket \text{cl}_t(\phi) \rrbracket_{>} = \{ \llbracket \psi \rrbracket_{\bowtie p} \mid \psi \in \text{cl}_t(\phi), \bowtie \in \{\geq, >\}, p \in [0, 1] \}$; it wraps $\psi_1 \text{ U } \psi_2$ in the probabilistic quantification $\llbracket \cdot \rrbracket_{\bowtie p}$ of A_ϕ .

Theorem 9

For every PCTL formula ϕ over AP and all $M \in \text{MC}_{\text{AP}}$, we have $M \models \phi$ iff $M \in \mathcal{L}(A_\phi)$. Deciding $M \in \mathcal{L}(A_\phi)$ is polynomial in the size of M and linear in the size of ϕ . ●

The above result is optimal as it matches the lower complexity bound of PCTL model checking [88].

The proof of Theorem 9 uses structural induction on PCTL formulae (i.e., state formulae) to show that for all locations s in M and all PCTL sub-formulae φ' of PCTL formula φ we have $s \in \llbracket \varphi' \rrbracket$ iff $\text{val}(s, \rho_\epsilon(\varphi')) = 1$ for configuration $(s, \rho_\epsilon(\varphi'))$ in the acceptance game for deciding $M \in \mathcal{L}(A_\varphi)$. As for the complexity, the membership game for $M \in \mathcal{L}(A_\phi)$ collapses to solving a sequence (linear in the size of ϕ) of weak stochastic games with solely probabilistic configurations. Such games are solvable in polynomial time, see e.g. [50].

Proof

We prove

For every location s of M and subformula φ' of φ we have $s \models_M \varphi'$ iff the configuration $(s, \rho_\epsilon(\varphi'))$ has value 1 for Player 0 in the acceptance game of A_φ on M .

by induction on the structure of the formula. For a proposition \mathbf{a} , notice that the value of (s, \mathbf{a}) depends on the values of $(s', \rho_X(\mathbf{a}, L(s)))$ for successors s' of s . By definition, $\rho_X(\mathbf{a}, L(s)) = \text{tt}$ if $\mathbf{a} \in L(s)$ and ff otherwise. The claim holds similarly for negated propositions, and by induction on Boolean combinations of formulae.

Consider a subformula of the form $\varphi' = [X\psi]_{\bowtie p}$. By induction $s' \models_M \psi$ iff the configuration $(s', \rho_\epsilon(\psi))$ is winning for Player 0. By definition $\rho_\epsilon([X\psi]_{\bowtie p}) = \llbracket [X\psi]_{\bowtie p} \rrbracket$. Consider the function $f: [1] \times \text{Succ}(s) \rightarrow [0, 1]$ such that $f(1, s') = 1$ iff $\text{val}(s', \rho_\epsilon(\psi)) = 1$. By assumption,

$$\sum_{s' \in \text{Succ}(s)} f(1, s') \text{val}(s', \rho_\epsilon(\psi)) \bowtie p .$$

The claim follows.

Consider a formula of the form $\varphi' = [\psi_1 \mathbf{U} \psi_2]_{\bowtie p}$. By induction $s \models_M \psi_i$ iff the configuration $(s, \rho_\epsilon(\psi_i))$ is winning for Player 0, for $i \in \{1, 2\}$. Consider the stochastic weak game induced by the SCC of $\psi_1 \mathbf{U} \psi_2$ in the structure of A_φ . The optimal strategy for both players is memoryless and pure. Restricting our attention to these memoryless pure strategies we can think about the game as restricted to configurations of the form $(s', \rho_\epsilon(\psi_1))$, where all configurations are probabilistic. A play that is winning for Player 0 is exactly a play that remains in states s' such that $s' \models_M \psi_1$ until reaching states s'' such that $s'' \models_M \psi_2$ (as $\psi_1 \mathbf{U} \psi_2$ is “unfair”). It follows that the value of $(s, \psi_1 \mathbf{U} \psi_2)$ in the stochastic game is exactly $\text{Prob}(s, \psi_1 \mathbf{U} \psi_2)$. Finally,

$$\rho_\epsilon([\psi_1 \mathbf{U} \psi_2]_{\bowtie p}) = \rho_\epsilon(\psi_1) \wedge \llbracket [\psi_1 \mathbf{U} \psi_2]_{\bowtie p} \rrbracket \vee \rho_\epsilon(\psi_2) .$$

Consider a location s and the configuration $(s, \rho_\epsilon([\psi_1 \mathbf{U} \psi_2]_{\bowtie p}))$. If configuration $(s, \rho_\epsilon(\psi_2))$ is winning for Player 0, then clearly $(s, \rho_\epsilon([\psi_1 \mathbf{U} \psi_2]_{\bowtie p}))$ is winning as well. Otherwise, by assumption $s \models [\psi_1 \mathbf{U} \psi_2]_{\bowtie p}$, so it must be the case that $s \models \psi_1$. It follows that Player 0 can choose the disjunct $\rho_\epsilon(\psi_1) \wedge \llbracket [\psi_1 \mathbf{U} \psi_2]_{\bowtie p} \rrbracket$. Furthermore, the function $f: [1] \times \text{Succ}(s) \rightarrow [0, 1]$ that associates $\text{val}(s', \psi_1 \mathbf{U} \psi_2)$ with s' is disjoint. The claim follows.

The treatment of a formula of the form $\varphi' = [\psi_1 \mathbf{W} \psi_2]_{\bowtie p}$ is similar, and so omitted.

The treatment of bounded strong Until and of bounded weak Until is a variant of the above cases, and so omitted. ■

Example 53

For $\varphi = [\mathbf{a} \mathbf{U} [\mathbf{X} \mathbf{b}]_{>0.5}]_{\geq 0.3}$ we have

$$A_\varphi = \langle 2^{\{\mathbf{a}, \mathbf{b}\}}, \text{cl}_t(\varphi) \cup \{\mathbf{a}, \mathbf{b}\}, \rho_X, \rho_\epsilon(\varphi), F \rangle$$

where

$$\begin{aligned}
\text{cl}_t(\varphi) &= \{\mathbf{a} \mathbf{U}[\mathbf{X} \mathbf{b}]_{>0.5}, \mathbf{X} \mathbf{b}\} \\
\rho_\epsilon(\varphi) &= (\mathbf{a} \wedge \llbracket \mathbf{a} \mathbf{U}[\mathbf{X} \mathbf{b}]_{>0.5} \rrbracket_{\geq 0.3}) \vee \llbracket \mathbf{X} \mathbf{b} \rrbracket_{>0.5} \\
F &= \{\mathbf{X} \mathbf{b}, \mathbf{a}, \mathbf{b}\} \\
\rho_{\mathbf{X}}(\mathbf{X} \mathbf{b}) &= \mathbf{b} \\
\rho_{\mathbf{X}}(\mathbf{a} \mathbf{U}[\mathbf{X} \mathbf{b}]_{>0.5}) &= (\mathbf{a} \wedge \mathbf{a} \mathbf{U}[\mathbf{X} \mathbf{b}]_{>0.5}) \vee \llbracket \mathbf{X} \mathbf{b} \rrbracket_{>0.5} . \quad \blacklozenge
\end{aligned}$$

Corollary 4 and Theorem 9 imply that any algorithm for solving language emptiness or containment of p-automata would prove that satisfiability of PCTL is decidable. Decidability of PCTL satisfiability is a well-known open problem [90, 30]. Therefore we can not expect that e.g. language emptiness of p-automata will be solved easily. On the other hand, for solving the language emptiness problem for p-automata one might be able to take advantage of the powerful techniques that were developed in automata theory. Thus the above results could open a new line of attack to the long-standing problem of PCTL satisfiability.

For p-automata which correspond to qualitative PCTL, i.e. a PCTL fragment for which satisfiability is known to be decidable [30], language emptiness should be decidable, too. A proof for this conjecture is part of the future work suggested in Section 7.3.

In comparing automata and temporal logic, automata usually can count but temporal logics cannot. Thus, just as alternating tree automata are more expressive than CTL and CTL*, p-automata are more expressive than PCTL. We show that p-automata are more expressive than PCTL, using an adaptation of the known result by Pierre Wolper showing that LTL cannot count [181].

Let $A_W = \langle 2^{\{\mathbf{a}, \mathbf{b}\}}, \{q_0, q_1\}, \delta, \llbracket q_0 \rrbracket_{>0}, \{\} \rangle$ be a p-automaton with transition function δ as follows:

$$\begin{aligned}
\delta(q_0, \{\mathbf{b}\}) &= \delta(q_1, \{\mathbf{b}\}) = \delta(q_0, \{\mathbf{a}, \mathbf{b}\}) = \delta(q_1, \{\mathbf{a}, \mathbf{b}\}) = \text{tt} \\
\delta(q_0, \{\mathbf{a}\}) &= \llbracket q_1 \rrbracket_{>0} \\
\delta(q_0, \{\}) &= \text{ff} \\
\delta(q_1, \{\}) &= \delta(q_1, \{\mathbf{a}\}) = \llbracket q_0 \rrbracket_{>0}
\end{aligned}$$

Lemma 15

Every Markov chain $M \in \mathcal{L}(A_W)$ has a finite path $s_0 s_1 \dots s_n$ with $n > 1$ such that $b \in L(s_n)$ and for all $0 \leq i < n$, either i is odd or $a \in L(s_i)$. ●

We do not prove formally that $\mathcal{L}(A_W) \neq \mathcal{L}(A_\phi)$ for all PCTL formulae ϕ over $\{\mathbf{a}, \mathbf{b}\}$. However, as the path from Lemma 15 is finite, its existence is equivalent to the probability of such a path being greater than 0. Thus, if it were possible to express this property in PCTL it would be possible to express it in CTL as well.

First we note that as both q_0 and q_1 are not fair, a winning play for Player 0 has to be finite and end with a transition that reads \mathbf{b} . Before reaching \mathbf{b} a winning play includes moves of the following form:

- Going from configurations of the form $(s_{2i}, \llbracket q_0 \rrbracket_{>0})$ to configurations $(s_{2i+1}, \llbracket q_1 \rrbracket_{>0})$ such that $P(s_{2i}, s_{2i+1}) > 0$ and $\mathbf{a} \in L(s_{2i})$.
- Going from configurations of the form $(s_{2i+1}, \llbracket q_1 \rrbracket_{>0})$ to configurations $(s_{2i+2}, \llbracket q_0 \rrbracket_{>0})$ such that $P(s_{2i+1}, s_{2i+2}) > 0$.

This implies the existence of a path as required.

Additionally, p-automata can encode *recursive, probabilistic properties* that we believe to be not expressible in PCTL. For example,

$$A_R = \langle 2^{\{\mathbf{a}\}}, \{q_1\}, \delta, \llbracket q_1 \rrbracket_{>0}, \{q_1\} \rangle$$

with $\delta(q_1, \{\mathbf{a}\}) = \llbracket q_1 \rrbracket_{\geq 0.5}$ and $\delta(q_1, \{\}) = \text{ff}$, asserts the recursive, probabilistic property that a location is labelled \mathbf{a} , and that the probability of its successors with the same property is at least 0.5. A naive attempt of expressing this in PCTL would be $\eta = \mathbf{a} \wedge [(\neg \mathbf{a} \vee [\mathbf{X} \mathbf{a}]_{\geq 0.5}) \mathbf{W} \neg \mathbf{a}]_{\geq 1}$. Then $\mathcal{L}(A_\eta) \subset \mathcal{L}(A_R)$ but this inclusion is strict.

6.4 Simulation of p-automata

We now define simulation of p-automata as a combination of fair simulation [92], simulation for alternating word automata [79], probabilistic bisimulation [131], and the games defined in Section 6.1. This simulation takes into

account the structure of p-automata, their acceptance condition and local probabilistic constraints. It under-approximates language containment: if p-automaton B simulates p-automaton A (denoted $A \leq B$), then $\mathcal{L}(A)$ is contained in $\mathcal{L}(B)$, under qualifications detailed in the formal theorem below. We furthermore show that whether B simulates A can be decided in EXPTIME.

We define simulation through a series of games G_{\leq} on the product of states and transitions of A and B : state u of B simulates state r of A iff Player 0 wins from configuration (r, u) in the corresponding game. More general configurations (α, β) are such that α is part of a transition of A and β is part of a transition of B . The classification of α and β as unbounded, bounded with $*$, bounded with \checkmark , or simple classifies (α, β) as one of 9 types. Here, we restrict our attention to the case that A and B do not use the \checkmark operator. Furthermore, a state that is part of a bounded SCC in B cannot simulate a state that is part of an unbounded SCC in A . These restrictions lead to the consideration of five cases, and are sufficient for handling simulation of automata that result from embedding PCTL formulae or Markov chains.

For sake of simplicity, p-automata

$$A = \langle \Sigma, Q, \delta, \varphi_a^{\text{in}}, F \rangle \text{ and } B = \langle \Sigma, U, \delta, \psi_b^{\text{in}}, F \rangle$$

satisfy $Q \cap U = \{\}$ and we use δ for the transition function of both automata and F for both acceptance conditions. We determine whether B simulates A by a sequence of weak and stochastic weak games.

The strict versions of the partial orders on equivalence classes of A and B are well-founded and so their lexicographical ordering is a well-founded ordering \prec on the sets of configurations of the game. Namely, $((\varphi), (\psi)) \prec ((\tilde{\varphi}), (\tilde{\psi}))$ if either $((\varphi)) \prec_A ((\tilde{\varphi}))$ or $((\varphi)) = ((\tilde{\varphi}))$ and $((\psi)) \prec_B ((\tilde{\psi}))$.

Consider a pair of equivalence classes $((\varphi), (\psi))$, where φ is in A and ψ is in B . As before, all pairs larger than $((\varphi), (\psi))$ with respect to \prec have already been handled: for every φ' and ψ' with $((\varphi), (\psi)) \prec ((\varphi'), (\psi'))$ value $\text{val}(\varphi', \psi') \neq \perp$ is pre-seeded.

Case 1: Let $((\varphi))$ and $((\psi))$ be SCCs where $((\varphi))$ has no transitions in E_b , and $((\psi))$ no transitions in E_u and no \checkmark markings. We set $\text{val}(\varphi, \psi) = 0$; bounded-with- $*$ states cannot simulate unbounded states.

Case 2: Let $((\varphi))$ and $((\psi))$ be SCCs such that both $((\varphi))$ and $((\psi))$ have no transitions in E_b . Then $G_{\leq}((\varphi), (\psi))$ is a stochastic weak game with

$$V = \{(\tilde{\varphi}, \tilde{\psi}) \mid \tilde{\varphi} \preceq_A \varphi \text{ and } \tilde{\psi} \preceq_B \psi\} \quad V_p = \{\}$$

where V_0 , V_1 , and E are defined as follows:

$$V_0 = \{c \in V \mid \exists \varphi_i, \psi_i: c = (\varphi_1 \wedge \varphi_2, \psi_1 \vee \psi_2)\} \cup \\ \{c \in V \mid \exists q': c = (q', \psi_1 \vee \psi_2), \text{ or } \exists u': c = (\varphi_1 \wedge \varphi_2, u')\}$$

$$V_1 = \{c \in V \mid \exists q', u': c = (q', u')\} \cup \\ \{c \in V \mid \exists \varphi_i, \psi: c = (\varphi_1 \vee \varphi_2, \psi), \text{ or } \exists \varphi, \psi_i: c = (\varphi, \psi_1 \wedge \psi_2)\}$$

$$E = \{((q', u'), (\delta(q', \sigma), \delta(u', \sigma))) \in V \times V \mid \sigma \in \Sigma\} \cup \\ \{((\varphi_1 \vee \varphi_2, \psi), (\varphi_i, \psi)), ((\varphi, \psi_1 \wedge \psi_2), (\varphi, \psi_i)) \in V \times V \mid i \in \{1, 2\}\} \cup \\ \{((\varphi_1 \wedge \varphi_2, \psi_2 \vee \psi_2), (\varphi_i, \psi_j)) \in V \times V \mid i, j \in \{1, 2\}\} \cup \\ \{((\varphi_1 \wedge \varphi_2, u'), (\varphi_i, u')), ((q', \psi_1 \vee \psi_2), (q', \psi_i)) \in V \times V \mid i \in \{1, 2\}\}$$

As pre-seeded values $\text{val}(\tilde{\varphi}, \tilde{\psi})$ for configurations $(\tilde{\varphi}, \tilde{\psi})$ with $((\varphi), (\psi)) \prec ((\tilde{\varphi}), (\tilde{\psi}))$ may be in the open interval $(0, 1)$, we treat $G_{\leq}((\varphi), (\psi))$ as a stochastic weak game.

Intuitively, Player 1 resolves disjunctions on the left and conjunctions on the right and does this before Player 0 needs to move. Player 0 resolves conjunctions on the left and disjunctions on the right when Player 1 cannot move. From configurations of the form (q', u') , where q' is a state of A and u' is a state of B , Player 1 chooses a letter $\sigma \in \Sigma$ and applies the transitions of q' and u' reading σ .

Finally, an infinite play in $G_{\leq}((q), (u))$ is winning for Player 0 if $((\varphi)) \cap Q \subseteq F$ implies $((\psi)) \cap U \subseteq F$.

By Theorem 1 every configuration c has a value for Player 0. We set $\text{val}(c)$ to that value.

Case 3: Let $((\varphi))$ and $((\psi))$ be SCCs such that both have neither transitions in E_u nor \checkmark markings. Then $G_{\leq}(((\varphi)), ((\psi)))$ is a weak game. Let

$$\begin{aligned}\tilde{\varphi} &= *(\llbracket q_1 \rrbracket_{\bowtie_1 p_1}, \dots, \llbracket q_n \rrbracket_{\bowtie_n p_n}) \\ \tilde{\psi} &= *(\llbracket u_1 \rrbracket_{\bowtie'_1 p'_1}, \dots, \llbracket u_m \rrbracket_{\bowtie'_m p'_m}) \\ \mathcal{F}_{\tilde{\varphi}, \tilde{\psi}} &= [n] \times [m] \rightarrow [0, 1]\end{aligned}$$

Also, $f \in \mathcal{F}_{\tilde{\varphi}, \tilde{\psi}}$ is *disjoint* if there is $\{a_{i,j} \in [0, 1] \mid i \in [n] \text{ and } j \in [m]\}$ with (a) $\sum_{j \in [m]} a_{i,j} = 1$ for all $i \in [n]$ and (b) $\sum_{i \in [n]} a_{i,j} \cdot p_i \cdot f(i, j) > p'_j$ for all $j \in [m]$, or $\sum_{i \in [n]} a_{i,j} \cdot p_i \cdot f(i, j) = p'_j$ and either \bowtie'_j is \geq or there is i' with $a_{i',j} > 0$ and $\bowtie_{i'}$ is $>$. Let $\mathcal{F}_{\tilde{\varphi}, \tilde{\psi}}^*$ be the set of disjoint functions. The configurations of $G_{\leq}(((\varphi)), ((\psi)))$ are

$$\begin{aligned}V &= \{(\tilde{\varphi}, \tilde{\psi}, f) \mid \tilde{\varphi} \in ((\varphi)), \tilde{\psi} \in ((\psi)), \text{ and } f \in \mathcal{F}_{\tilde{\varphi}, \tilde{\psi}}^*\} \cup \\ &\quad \{(\tilde{\varphi}, \tilde{\psi}), (\tilde{\varphi}, \tilde{\psi}, v) \mid \tilde{\varphi} \preceq_A \varphi, \tilde{\psi} \preceq_B \psi, \text{ and } v \in [0, 1]\}\end{aligned}$$

where V_0 , V_1 , and E are as follows:

$$\begin{aligned}V_0 &= \{(\alpha_1 \wedge \alpha_2, \beta_1 \vee \beta_2, v), (\alpha_1 \wedge \alpha_2, \epsilon, v), (\gamma, \beta_1 \vee \beta_2, v), (\gamma, \epsilon)\} \cup \\ &\quad \{(\alpha, \beta, v) \mid \text{val}(\alpha, \beta) \neq \perp \text{ and } v > \text{val}(\alpha, \beta)\}\end{aligned}$$

$$\begin{aligned}V_1 &= \{(\gamma, \epsilon, f), (\alpha_1 \vee \alpha_2, \beta, v), (\alpha, \beta_1 \wedge \beta_2, v)\} \cup \\ &\quad \{(\alpha, \beta, v) \mid \text{val}(\alpha, \beta) = \perp \text{ or } v \leq \text{val}(\alpha, \beta)\}\end{aligned}$$

$$\begin{aligned}E &= \{((\alpha_1 \vee \alpha_2, \beta, v), (\alpha_i, \beta, v)), ((\alpha, \beta_1 \wedge \beta_2, v), (\alpha, \beta_i, v)) \mid i \in \{1, 2\}\} \cup \\ &\quad \{((\gamma, \epsilon), (\gamma, \epsilon, f))\} \cup \\ &\quad \{((\alpha_1 \wedge \alpha_2, \epsilon, v), (\alpha_i, \epsilon, v)), ((\gamma, \beta_1 \vee \beta_2, v), (\gamma, \beta_i, v)) \mid i \in \{1, 2\}\} \cup \\ &\quad \{((\gamma, \epsilon, f), (\delta(q_i, \sigma), \delta(u_j, \sigma), f(i, j))) \mid f(i, j) > 0 \text{ and } \sigma \in \Sigma\} \cup \\ &\quad \{((\alpha_1 \wedge \alpha_2, \beta_2 \vee \beta_2, v), (\alpha_i, \beta_j, v)) \mid i, j \in \{1, 2\}\}\end{aligned}$$

In these definitions α and β range over formulae in transitions of A and B , respectively, γ and ϵ range over formulae in $\llbracket Q \rrbracket^*$ and $\llbracket U \rrbracket^*$, respectively.

The set V above is uncountable and infinitely branching, as branching

includes a choice of a function $f: [n] \times [m] \rightarrow [0, 1]$. Standard techniques can be used to make these games finite branching. If both A and B are finite, the games will be finite, too.

For $(\gamma, \epsilon) \in \llbracket Q \rrbracket^* \times \llbracket U \rrbracket^*$ with

$$\begin{aligned}\gamma &= *(\llbracket q_1 \rrbracket_{\bowtie_1 p_1}, \dots, \llbracket q_n \rrbracket_{\bowtie_n p_n}) \\ \epsilon &= *(\llbracket u_1 \rrbracket_{\bowtie'_1 p'_1}, \dots, \llbracket u_m \rrbracket_{\bowtie'_m p'_m})\end{aligned}$$

in order to show that ϵ simulates γ , Player 0 needs to show that the probability of ϵ (and its partition) can be supported by γ . Accordingly, from (γ, ϵ) Player 0 chooses $f: [n] \times [m] \rightarrow [0, 1]$ and moves to configuration (γ, ϵ, f) . Such a configuration relates to the claim that q_i is related to u_j with proportion $f(i, j)$ and that f can be partitioned (using the $\{a_{i,j}\}$ to support the different u_j 's). Then, Player 1 chooses i and j such that $f(i, j) > 0$ and an alphabet letter $\sigma \in \Sigma$, leading to a configuration of the form

$$(\delta(q_i, \sigma), \delta(u_j, \sigma), f(i, j)) .$$

Conjunctions and disjunctions are resolved in the usual way until the game either reaches another configuration in $\llbracket Q \rrbracket^* \times \llbracket U \rrbracket^* \times [0, 1]$, in which case the value $f(i, j)$ is ignored (as $f(i, j) \leq 1$), or until the play reaches a configuration with a pre-seeded value v . Then, if $f(i) \leq v$ Player 0 has fulfilled her obligation and she wins. If $f(i) > v$, Player 0 failed and she loses. An infinite play in $G_{\leq}(\llbracket \varphi \rrbracket, \llbracket \psi \rrbracket)$ is winning for Player 0 if $\llbracket \varphi \rrbracket \cap Q \subseteq F$ implies $\llbracket \psi \rrbracket \cap U \subseteq F$.

By Theorem 1, every $c \in V$ has a value in $\{0, 1\}$ for Player 0. We set $\text{val}(c)$ to that value.

Case 4: Let $\llbracket \varphi \rrbracket$ and $\llbracket \psi \rrbracket$ be SCCs where $\llbracket \varphi \rrbracket$ has transitions in E_b without \star markings, and $\llbracket \psi \rrbracket$ has transitions in E_u . Then $G_{\leq}(\llbracket \varphi \rrbracket, \llbracket \psi \rrbracket)$

is a stochastic weak game with

$$\begin{aligned}
V &= \{(\tilde{\varphi}, \tilde{\psi}) \mid \tilde{\varphi} \preceq_A \varphi \text{ and } \tilde{\psi} \preceq_B \psi\} \cup \llbracket Q \rrbracket \times U \times \Sigma \\
V_0 &= \{(\alpha_1 \wedge \alpha_2, \beta_1 \vee \beta_2), (\alpha_1 \wedge \alpha_2, u), (\gamma, \beta_1 \vee \beta_2)\} \\
V_1 &= \{(\alpha_1 \vee \alpha_2, \beta), (\alpha, \beta_1 \wedge \beta_2), (\gamma, u)\} \\
V_p &= \llbracket Q \rrbracket^* \times U \times \Sigma \\
E &= \{((\alpha_1 \vee \alpha_2, \beta), (\alpha_i, \beta)) \mid i \in \{1, 2\}\} \cup \\
&\quad \{((\alpha, \beta_1 \wedge \beta_2), (\alpha, \beta_i)) \mid i \in \{1, 2\}\} \cup \\
&\quad \{((\alpha_1 \wedge \alpha_2, \beta_1 \vee \beta_2), (\alpha_i, \beta_j)) \mid i, j \in \{1, 2\}\} \cup \\
&\quad \{((\gamma, u), (\gamma, u, \sigma)), ((\gamma, u, \sigma), (\delta(q_i, \sigma), \delta(u, \sigma)))\}
\end{aligned}$$

and

$$\kappa((\gamma, u, \sigma))((\delta(q_i, \sigma), \delta(u, \sigma))) = p_i$$

where α, α_i and β, β_i range over formulae in transitions of A and B , respectively, while γ and u range over $\llbracket Q \rrbracket^*$ and U , respectively. For probabilities p_i that do not sum up to 1, we add a sink state (losing for Player 0) that fills that gap.

An infinite play in $G_{\leq}((\varphi), (\psi))$ is winning for Player 0 if $((\varphi)) \cap Q \subseteq F$ implies $((\psi)) \cap U \subseteq F$.

By Theorem 1 every configuration c has a value. We set $\text{val}(c)$ to the value of configuration c for Player 0.

Intuitively, a state u measures the probability of some regular set of paths, and a state $\llbracket q \rrbracket_{\bowtie p}$ can restrict the immediate steps taken by a Markov chain as well as enforce some regular structure on paths. Thus, this stochastic weak game establishes the conditions under which a Markov chain accepted from $\llbracket q \rrbracket_{\bowtie p}$ can be also accepted from u .

Case 5: The case when $((\varphi))$ or $((\psi))$ is a trivial SCC is subsumed by at least one of the four preceding cases. As for the acceptance game, this ambiguity is unproblematic as the game values in $G_{\leq}((\varphi), (\psi))$ are then determined by the propagation of pre-seeded game values.

We are now in a position to state the formal definition for simulation

between p-automata.

Definition 71 (Simulation)

We say that B simulates A , denoted $A \leq B$, if the value of configuration $(\varphi_a^{\text{in}}, \varphi_b^{\text{in}})$, computed in the previous sequence of games, is 1. \blacklozenge

Theorem 10

Let A and B be p-automata over 2^{AP} that contain no occurrence of \forall^* . If A and B are finite, then

$$A \leq B \text{ implies } \mathcal{L}(A) \subseteq \mathcal{L}(B) \tag{6.2}$$

and $A \leq B$ can be decided in EXPTIME.

For p-automata which stem from Markov chains this implication becomes an equivalence: If A is A_M for a Markov chain $M \in \text{MC}_{\text{AP}}$, then $A \leq B$ iff $\mathcal{L}(A) \subseteq \mathcal{L}(B)$ for all $B \in \text{MC}_{\text{AP}}$. \bullet

The second part of the theorem, i.e. the coincidence of simulation and language inclusion for p-automata of Markov chains, is crucial for the completeness of the p-automata framework for full PCTL. In fact, the full completeness result follows as direct Corollary 5 from the above theorem.

Another direct consequence of the theorem is that $N \sim M$ iff $A_M \leq A_N$.

Before we prove the theorem we give a proof sketch first: The first claim of Theorem 10 is proved as follows. Assuming $M \in \mathcal{L}(A)$ and $A \leq B$ we consider configurations (s, φ) and (φ, ψ) in the corresponding games, respectively. This determines a configuration (s, ψ) in the acceptance game for $M \in \mathcal{L}(B)$. We show an invariant, that

$$\text{val}(s, \varphi) \cdot \text{val}(\varphi, \psi) \leq \text{val}(s, \psi)$$

for all such ‘‘synchronised’’ configurations. In particular, we get $\text{val}(s^{\text{in}}, \varphi^{\text{in}}) \cdot \text{val}(\varphi^{\text{in}}, \psi^{\text{in}}) = 1 \cdot 1 \leq \text{val}(s^{\text{in}}, \psi^{\text{in}})$ which proves $M \in \mathcal{L}(B)$. Extending this result to infinite-state automata seems to require the treatment of infinite converging products of real numbers, which we do not pursue in this thesis.

The second claim of Theorem 10 follows since the simulation game collapses to an acceptance game when the automaton of the left in (6.2) is

derived from a Markov chain.

Proof

We note that when A equals A_M for some $M \in \text{MC}_{\text{AP}}$, the simulation game for $A_M \leq B$ and the acceptance game for $M \in \mathcal{L}(B)$ collapse to the same game. Thus, regardless of whether A_M or B is infinite-state we have $A_M \leq B$ iff $M \in \mathcal{L}(B)$. The latter is equivalent to $\mathcal{L}(A_M) \subseteq \mathcal{L}(B)$ by Lemma 14 and Theorem 8.

In order to prove (6.2) for finite-state A and B , consider a Markov chain $M = (S, \mathbf{P}, L, s^{\text{in}})$. Consider two formulae φ and ψ such that φ appears in the transition of A and ψ appears in the transition of B .

We construct a strategy for Player 0 in G_B and plays in G_A , G_B , and G_{\leq} such that the plays start from (s, φ) , (s, ψ) , and (φ, ψ) , respectively and such that the values of these plays satisfy

$$\text{val}(s, \varphi) \cdot \text{val}(\varphi, \psi) \leq \text{val}(s, \psi) .$$

Thus, we prove that $M \in \mathcal{L}(A)$ implies $M \in \mathcal{L}(B)$.

Suppose that the claim holds by induction for plays starting in configurations $((\tilde{\varphi}), (\tilde{\psi}))$, where $((\tilde{\varphi}), (\tilde{\psi})) \prec ((\varphi), (\psi))$.

- In case that $\varphi \in Q$ and $\psi \in \llbracket U \rrbracket^*$ we have $\text{val}(\varphi, \psi) = 0$ and the claim holds trivially.
- Suppose that both φ and ψ are in unbounded SCCs. The game G_A is a stochastic weak game and Player 0 secures $\text{val}(s, \varphi)$ in configuration (s, φ) .

Consider the configurations (s, φ) , (s, ψ) , and (φ, ψ) in the games G_A , G_B , and G_{\leq} , respectively.

If φ is a disjunction, then the strategy of Player 0 in G_A instructs her to choose a disjunct φ_1 of φ . Then (φ, ψ) is a Player 1 configuration in G_{\leq} and we instruct Player 1 to choose the successor (φ_1, ψ) . If ψ is a conjunction, then Player 1 chooses a successor (s, ψ_1) of (s, ψ) in G_B . We update the game G_{\leq} by mimicking the same choice of Player 1 from (φ, ψ) . If φ is a conjunction and ψ is not a conjunction, then the strategy of Player 0 in G_{\leq} instructs Player 0 to choose a conjunct φ_1 of φ . This choice can be mimicked in G_A in which Player 1 needs

to move. If φ is not a disjunction and ψ is a disjunction, then the strategy of Player 0 in G_{\leq} instructs Player 0 to choose a disjunct ψ_1 of φ . This choice resolves Player 0's choice in G_B .

Consider three plays produced in this way. If all plays are infinite, the claim follows from the winning condition in G_{\leq} and the values of the plays in G_A and G_B . If one of the plays is finite then the claim follows from the induction assumption, as the play passes in G_{\leq} to a different SCC.

- Suppose that φ and ψ are in bounded SCCs. The game G_A is a weak game and Player 0 secures $\text{val}(s, \varphi)$ in configuration (s, φ) . By definition $\text{val}(s, \varphi) \in \{0, 1\}$.

There is nothing to show if $\text{val}(s, \varphi) = 0$.

Suppose that $\text{val}(s, \varphi) = 1$, i.e., Player 0 wins from configuration (s, φ) in G_A . In G_{\leq} we have $\text{val}(\varphi, \psi) \in \{0, 1\}$. Again there is nothing to show if $\text{val}(\varphi, \psi) = 0$, so let's suppose that $\text{val}(\varphi, \psi) = 1$. We now have to give a strategy for Player 0 in G_B such that $\text{val}(s, \psi) = 1$.

Let $\varphi = *(\llbracket q_1 \rrbracket_{\triangleright_{1p_1}}, \dots, \llbracket q_n \rrbracket_{\triangleright_{np_n}})$ and $\psi = *(\llbracket u_1 \rrbracket_{\triangleright_{1p'_1}}, \dots, \llbracket u_m \rrbracket_{\triangleright_{mp'_m}})$. Let $f: [n] \times \text{Succ}(s) \rightarrow [0, 1]$ be the function chosen by Player 0's strategy in G_A and let $f': [n] \times [m] \rightarrow [0, 1]$ be the function chosen by Player 0's strategy in G_{\leq} . We set Player 0's strategy in G_B to choose the function $f'': [m] \times \text{Succ}(s) \rightarrow [0, 1]$ where $f''(j, s')$ is the minimal value in $\text{val}_{s, \psi}$ that is at least $\max_{i \in [n]} f(i, s') \cdot f'(i, j)$. We have to show that f'' is disjoint.

Claim 1

f'' is disjoint. ●

Proof

Let $a_{j, s'} = \sum_{i \in [n]} a_{i, s'} \cdot a_{i, j}$. First, one can see that for every $s' \in \text{Succ}(s)$ we have

$$\sum_{j \in [m]} a_{j, s'} = \sum_{j \in [m]} \sum_{i \in [n]} a_{i, s'} \cdot a_{i, j} = \sum_{i \in [n]} a_{i, s'} \sum_{j \in [m]} a_{i, j} = \sum_{i \in [n]} a_{i, s'} = 1$$

Second, consider some $j \in [m]$. Then,

$$\begin{aligned}
& \sum_{s' \in \text{Succ}(s)} a_{j,s'} \cdot f''(j, s') \cdot P(s, s') \\
&= \sum_{s' \in \text{Succ}(s)} \left(\sum_{i \in [n]} a_{i,s'} \cdot a_{i,j} \right) \cdot f''(j, s') \cdot P(s, s') \\
&\geq \sum_{s' \in \text{Succ}(s)} \left(\sum_{i \in [n]} a_{i,s'} \cdot a_{i,j} \right) \cdot \max_{i \in [n]} (f(i, s') \cdot f'(i, j)) \cdot P(s, s') \\
&\geq \sum_{s' \in \text{Succ}(s)} \sum_{i \in [n]} a_{i,s'} \cdot a_{i,j} \cdot f(i, s') \cdot f'(i, j) \cdot P(s, s') \\
&= \sum_{i \in [n]} \sum_{s' \in \text{Succ}(s)} a_{i,s'} \cdot a_{i,j} \cdot f(i, s') \cdot f'(i, j) \cdot P(s, s') \\
&= \sum_{i \in [n]} a_{i,j} \cdot f'(i, j) \cdot \sum_{s' \in \text{Succ}(s)} a_{i,s'} \cdot f(i, s') \cdot P(s, s') \\
&\bowtie \sum_{i \in [n]} a_{i,j} \cdot f'(i, j) \cdot p_i \\
&\bowtie' p_j
\end{aligned}$$

and \bowtie is $>$ if for some $i \in [n]$ we have \bowtie_i equals $>$ and then \bowtie' is \geq , otherwise either \bowtie' is $>$ or \bowtie'_j is \geq and the claim is proved. \blacksquare

With f'' established as disjoint, we get back to the games. In G_B Player 1 chooses j and $s' \in \text{Succ}(s)$ and moves to

$$(s', \delta(u_j, L(s)), f''(s', j)) .$$

We mimic this choice in G_A by making Player 1 choose the state q_i such that $f(i, s') \cdot f'(i, j)$ is maximal. The game thus moves to

$$(s', \delta(q_i, L(s)), f(i, s')) .$$

We mimic this choice in G_{\leq} by making Player 1 choose the states q_i ,

u_j , and the letter $L(s)$ leading to configuration

$$(\delta(q_i, L(s)), \delta(u_j, L(s)), f'(i, j)) .$$

If the plays continue indefinitely inside the same SCC in G_{\leq} the claim follows from the winning condition in G_{\leq} and the winning conditions of G_A and G_B .

If the plays exits the SCC in G_{\leq} , the triplet of configurations is

$$(s'', \varphi'', v_1), (s'', \psi'', v_2), (\varphi'', \psi'', v) .$$

By induction assumption $\text{val}(s'', \varphi'') \cdot \text{val}(\varphi'', \psi'') \leq \text{val}(s'', \psi'')$ holds. Furthermore, we have to show that $\text{val}(s'', \psi'') \geq v$. Let (s, φ) , (s, ψ) and (φ, ψ) be the last configurations that are part of the SCC before reaching the above triplet of configurations. It follows that

$$\text{val}(s'', \psi'') \in \text{val}_{s, \psi} .$$

By the choices of f , f' and f'' we know that v is the minimal value in $\text{val}_{s, \psi}$ that is at least $\max_{i \in [n]} f(i, s') \cdot f'(i, j)$. In addition, the last choice in G_A was exactly the state q_i such that i is maximal. We know that $\text{val}(s'', \varphi'') \geq v_1$, and that $\text{val}(\varphi'', \psi'') \geq v$. It follows that $\text{val}(s'', \varphi'') \cdot \text{val}(\varphi'', \psi'') \geq v \cdot v_1$. But, v_2 is exactly $v \cdot v_1$ leading to the desired result.

- Suppose that φ is in a bounded SCC and ψ is in an unbounded SCC. The game G_A is a weak game while the games G_B and G_{\leq} are stochastic weak games. The interesting cases are only those where

$$\text{val}(s, \varphi) = 1 \text{ and } \text{val}(\varphi, \psi) > 0 .$$

Given a strategy of Player 1 in G_B , we show how to use the winning strategies of Player 0 in G_A and G_{\leq} to produce a winning strategy for Player 0 in G_B . We also resolve all the choices for Player 1 in G_A and G_{\leq} leading to both G_{\leq} and G_B being reduced to Markov decision processes. These Markov decision processes capture all the possible

evolutions of the games in G_B and G_{\leq} according to the possible choices in probabilistic configurations in G_A . We then show how to use these Markov decision processes to prove that the claim holds.

Consider three configurations (s, φ') in G_A , (φ', ψ') in G_{\leq} , and (s, ψ') in G_B . If ψ' is a conjunction, then, in G_B , Player 1 chooses a conjunct of ψ' . The same choice is mimicked in G_{\leq} by making Player 1 choose the same conjunct. If ψ' is a disjunction, then Player 0's strategy in G_{\leq} instructs her to choose one disjunct. The same choice is mimicked in G_B . If φ' is a conjunction, then Player 0's strategy in G_{\leq} chooses a conjunct of φ' . We make Player 1 in G_A choose the same conjunct. If φ' is a disjunction, then Player 0's strategy in G_A chooses a disjunct of φ' . We make Player 1 in G_{\leq} choose the same disjunct. The remaining cases are where $\psi' = u$ is a state of B and $\varphi' = *([q_1]_{\bowtie_1 p_1}, \dots, [q_n]_{\bowtie_n p_n})$. The configuration (s, u) in G_B is probabilistic. The configuration (φ', u) in G_{\leq} is a Player 1 configuration. We make Player 1 choose $L(s)$ in G_{\leq} leading to configuration $(\varphi', u, L(s))$, which is probabilistic. The configuration (s, φ') is a Player 0 configuration in G_A . The strategy of Player 0 on G_A instructs her to choose a disjoint function $f : [n] \times \text{Succ}(s) \rightarrow [0, 1]$. Let $\{a_{i, s'}\}$ be witnesses to the disjointness of f . Consider a location s' that is chosen with probability $P(s, s')$ in G_B . Here, we make multiple possible choices of continuing in the games, giving rise to Markov decision processes (with a matching between the choices in them). Consider all indices i such that $a_{i, s'} > 0$. It follows that for every such index there is a way to continue unraveling the plays by making Player 1 in G_A choose the successor $(s', \delta(q_i, L(s)), f(i, s'))$ and continuing to configurations $(\delta(q_i, L(s)), \delta(u, L(s)))$ in G_{\leq} and $(s', \delta(u, L(s)))$ in G_B . By using these strategies, this effectively creates from G_B and G_{\leq} a Markov decision processes where the choices are angelic in G_B and demonic in G_{\leq} . That is, the actual value of G_B is the best possible value in the Markov decision process arising from G_B and the value in G_{\leq} is the worst possible value in G_{\leq} . Hence, it is enough to show one choice such that the value in the Markov decision process arising from G_B satisfies the requirement of the claim. Indeed, the actual value in G_B could only be higher while the actual value in G_{\leq} could only be

lower.

Consider now three configurations (s, φ') , (φ', ψ') , and (s, ψ') , and the resulting Markov decision processes from (φ', ψ') and (s, ψ') . By the construction of the strategy, every play starting in (s, ψ') is associated with plays that start in (s, φ') and (φ', ψ') such that at every stage the three configurations use the same state of the Markov chain and formulae in the transitions of A and B . We consider four cases:

(1) Consider a triplet of configurations $(s', \tilde{\varphi})$, $(\tilde{\varphi}, \tilde{\psi}')$, and $(s', \tilde{\psi})$ such that $(\tilde{\varphi}, \tilde{\psi})$ is not in the equivalence class of $(((\varphi)), ((\psi)))$. By induction $\text{val}(s', \psi'') \geq \text{val}(s', \varphi'') \cdot \text{val}(\varphi'', \psi'')$.

(2) Consider a triplet of configurations (s, φ') , (φ', ψ') , and (s, ψ') such that there is some choice in the Markov decision process that arises from G_B such that all plays which start in (φ', ψ') , remain in $(((\varphi)), ((\psi)))$ and are winning for Player 0 in G_{\leq} . The matching choice of plays starting from (s, ψ') are winning for Player 0 in G_B . Indeed, if this were not the case, there would be a play in G_B that is losing. It follows that the corresponding play in G_{\leq} does not satisfy the acceptance of A and that the play in G_A is losing. However G_A is a weak game and this is impossible.

(3) Consider a triplet of configurations (s, φ') , (φ', ψ') , and (s, ψ') such that for all choices in the Markov decision process that arises from G_{\leq} we have all plays starting in (φ', ψ') remain in $(((\varphi)), ((\psi)))$ and are losing for Player 0 in G_{\leq} . One can see that $\text{val}(s, \psi') \geq 0$.

(4) Consider now a triplet (s, φ') , (φ', ψ') , and (s, ψ') such that $(\varphi', \psi') \in ((\varphi)), ((\psi))$ and there is no choice in the Markov decision process arising from G_{\leq} such that (i) all paths are winning for Player 0 and (ii) for all choices the probability for Player 0 to win is positive. As the automata and the Markov chain are finite, so are the resulting Markov decision processes. It follows that the probability of winning in G_{\leq} equals the probability of getting to one of the previous three types of configurations. Then we show that the probability to reach

one of the three previous types of configurations in n steps satisfies the requirements of the theorem, for every n . The requirement of the claim will follow.

For every triplet (s', φ') , (φ', ψ') , and (s', ψ') let $P_0(\varphi', \psi')$ be $\text{val}(\varphi', \psi')$ and $P_0(s', \psi')$ be $\text{val}(s', \psi')$ if (φ', ψ') is one of the three types of configurations mentioned above. Let $P_0(\varphi', \psi')$ and $P_0(s', \psi')$ be 0, otherwise.

Consider a triplet (s, φ') , (φ', ψ') , (s, ψ') such that

$$\varphi' = *([\![q_1]\!]_{\times_1 p_1}, \dots, [\![q_n]\!]_{\times_n p_2}) \text{ and } \psi' = u$$

such that $P_0(\varphi', \psi') = P_0(s, \psi') = 0$ but for some successor s' of s there is a choice of i such that $P_0(\delta(q_i, L(s')), \delta(u, L(s))) > 0$ and $P_0(s'', \delta(u, L(s'))) > 0$. Let I denote the set of such indices i . Then P_1 satisfies the following requirement:

$$P_1(s, u) = \sum_{s' \in \text{Succ}(s) | \exists i \in I. a_{i, s'} > 0} P(s, s') P_0(s', \delta(u, L(s)))$$

From this equation we now derive $P_1(s, u) \geq P_1(\varphi', u)$ as follows, where each step of the derivation is explained:

$$P_1(s, u) = \sum_{s' \in \text{Succ}(s) | \exists i \in I. a_{i, s'} > 0} P(s, s') P_0(s', \delta(u, L(s)))$$

For all such s' , we have $P_0(s', \delta(u, L(s))) = \text{val}(s', \delta(u, L(s)))$.

$$\dots = \sum_{s' \in \text{Succ}(s) | \exists i \in I. a_{i, s'} > 0} P(s, s') \cdot \text{val}(s', \delta(u, L(s)))$$

We have already proven the requirement of the theorem for these configurations.

$$\dots \geq \sum_{s' \in \text{Succ}(s) | \exists i \in I. a_{i, s'} > 0} P(s, s') \cdot \text{val}(\delta(q_i, L(s)), \delta(u, L(s))) \cdot \text{val}(s', \delta(q_i, L(s)))$$

By $\text{val}(s', \delta(q_i, L(s))) \geq \sum_{i \in I} f(i, s') \cdot a_{i, s'}$.

$$\dots \geq \sum_{s' \in \text{Succ}(s) | \exists i \in I. a_{i, s'} > 0} P(s, s') \cdot \sum_{i \in I} f(i, s') \cdot a_{i, s'} \cdot \text{val}(\delta(q_i, L(s)), \delta(u, L(s)))$$

Changing the order of summation.

$$\dots = \sum_{i \in I} \text{val}(\delta(q_i, L(s)), \delta(u, L(s))) \cdot \sum_{s' \in \text{Succ}(s) | \exists i \in I. a_{i, s'} > 0} f(i, s') \cdot a_{i, s'} \cdot P(s, s')$$

By f being disjoint, $\sum_{s' \in \text{Succ}(s) | \exists i \in I. a_{i, s'} > 0} f(i, s') \cdot a_{i, s'} \cdot P(s, s') \geq p_i$.

$$\dots \geq \sum_{i \in I} \text{val}(\delta(q_i, L(s)), \delta(u, L(s))) \cdot p_i$$

Finally:

$$\dots = P_1(\varphi', u)$$

Consider a triplet (s, φ') , (φ', ψ') , and (s, ψ') . The strategy defined above fixes most such configurations as deterministic in their respective Markov decision processes. The only interesting case is when $\varphi' \in \llbracket Q \rrbracket$ and $\psi' \in U$. In this case (φ', ψ') and (s, ψ') are probabilistic configurations and the strategy includes some choice in the matching between successors of (φ', ψ') and (s, ψ') . Let $\varphi' = *(\llbracket q_1 \rrbracket \bowtie_{p_1}, \dots, \llbracket q_n \rrbracket \bowtie_{p_n})$ and $\psi' = u$. Then,

$$P_{n+1}(s, u) = \sum_{s' \in \text{Succ}(s)} P(s, s') \cdot P_n(s', \delta(u, L(s)))$$

Recall that the way to extend the game from configuration (φ', u) (matching a move to $\delta(q_i, L(s))$ with the move to $(s', \delta(u, L(s)))$) depends on which $a_{i, s'}$ are positive in a disjoint function f .

$$P_{n+1}(\varphi', u) = \sum_{i \in [n]} \max_{i: a_{i, s'} > 0} \text{val}(s', \delta(q_i, L(s))) \cdot P_n(\delta(q_i, L(s)), \delta(u, L(s)))$$

We now assume by induction that for possible matching triplets (s, φ') ,

(φ', ψ') , and (s, ψ') we have:

$$P_n(s, \psi') \geq \text{val}(s, \varphi') \cdot P_n(\varphi', \psi')$$

and prove the same for P_{n+1} . We concentrate on the only interesting case, where $\varphi' = *(\llbracket q_1 \rrbracket_{\triangleright_1 p_1}, \dots, \llbracket q_n \rrbracket_{\triangleright_n p_n})$ and $\psi' = u$. Again we write explanations in between the steps of the derivation:

$$P_{n+1}(s, u) = \sum_{s' \in \text{Succ}(s)} P(s, s') \cdot P_n(s', \delta(u, L(s)))$$

By induction, where $i_{s'}$ is such that $\text{val}(s', \delta(q_{i_{s'}}, L(s))) \cdot P_n(\delta(q_{i_{s'}}, L(s)), \delta(u, L(s)))$ is maximal among all $i \in [n]$.

$$\dots \geq \sum_{s' \in \text{Succ}(s)} P(s, s') \cdot \text{val}(s', \delta(q_{i_{s'}}, L(s))) \cdot P_n(\delta(q_{i_{s'}}, L(s)), \delta(u, L(s)))$$

By $\sum_{i \in [n]} a_{i, s'} = 1$, for all s' .

$$\dots = \sum_{s' \in \text{Succ}(s)} P(s, s') \cdot \left(\sum_{i \in [n]} a_{i, s'} \cdot \text{val}(s', \delta(q_{i_{s'}}, L(s))) \cdot P_n(\delta(q_{i_{s'}}, L(s)), \delta(u, L(s))) \right)$$

By choice of $i_{s'}$ such that $\text{val}(s', \delta(q_{i_{s'}}, L(s))) \cdot P_n(\delta(q_{i_{s'}}, L(s)), \delta(u, L(s)))$ is maximal.

$$\dots \geq \sum_{s' \in \text{Succ}(s)} P(s, s') \cdot \sum_{i \in [n]} a_{i, s'} \cdot \text{val}(s', \delta(q_i, L(s))) \cdot P_n(\delta(q_i, L(s)), \delta(u, L(s)))$$

By choice of f and win in G_A , we have $\text{val}(s', \delta(q_{i_{s'}}, L(s))) \geq f(i_{s'}, s')$.

$$\dots \geq \sum_{s' \in \text{Succ}(s)} P(s, s') \cdot \sum_{i \in [n]} a_{i, s'} \cdot f(i_{s'}, s') \cdot P_n(\delta(q_{i_{s'}}, L(s)), \delta(u, L(s)))$$

Change order of summation.

$$\dots = \sum_{i \in [n]} P_n(\delta(q_i, L(s)), \delta(u, L(s))) \cdot \sum_{s' \in \text{Succ}(s)} a_{i,s'} \cdot f(i, s') \cdot P(s, s')$$

By choice of f and $a_{i,s'}$

$$\text{we have } \sum_{s' \in \text{Succ}(s)} a_{i,s'} \cdot f(i, s') \cdot P(s, s') \geq p_i.$$

$$\dots \geq \sum_{i \in [n]} P_n(\delta(q_i, L(s)), \delta(u, L(s))) \cdot p_i$$

Finally:

$$\dots = P_{n+1}(\varphi', u) \quad \blacksquare$$

Deciding simulation is EXPTIME complete, which is potentially a severe restriction for practical applications as the models in real applications tend to be large. Nevertheless the decidability of simulation is an important result – even from a practical perspective – as simulation soundly under-approximates language inclusion for which we do not yet know whether or not it is decidable.

We now get sound and complete verification of $M \models \phi$ through simulations, in the sense of Dams and Namjoshi [58], using simulation as abstraction.

Corollary 5

For every infinite $M \in \text{MC}_{\text{AP}}$ and PCTL formula ϕ over AP we have $M \models \phi$ iff there is a finite p-automaton A with $A_M \leq A$ and $A \leq A_\phi$. \bullet

Proof

To see this, any such A implies $\mathcal{L}(A_M) \subseteq \mathcal{L}(A)$ and $\mathcal{L}(A) \subseteq \mathcal{L}(A_\phi)$ by both parts of Theorem 10 – noting that neither A_M nor A_ϕ have any occurrence of \forall . Thus, $M \models \phi$ holds by Theorems 8 and 9.

Conversely, if there is no such A , then A_ϕ can also not be such an A . Since $A_\phi \leq A_\phi$ this implies $A_M \not\leq A_\phi$ and so $\mathcal{L}(A_M) \not\subseteq \mathcal{L}(A_\phi)$ from the converse

of (6.2). So there is some $M' \sim M$ with $M' \not\models \phi$. Since $M' \sim M$, we get $M \not\models \phi$ as well by Lemma 14. ■

This method for verifying $M \models \phi$ via simulation is thus complete in the sense of [58]. To our knowledge, this is the first such completeness result for PCTL and Markov chains.

6.5 Wrong embedding of Markov chains

To shed some light on the motivation and development of the $*$ operator (and its dual \forall), we show that a simpler conversion of Markov chains to automata without $*$ operator produces automata that accept Markov chains which are not necessarily bisimilar to the original one.

Consider Markov chain $M = (S, \mathbf{P}, L, s^{\text{in}})$. We suggest the following *very-weak* embedding of a Markov chain in a p-automaton. We define the following p-automaton $A_M^w = \langle \Sigma, Q, \delta, \varphi^{\text{in}}, \alpha \rangle$, where

$$\begin{aligned} Q &= \{(s, s') \mid P(s, s') > 0\} \\ \varphi^{\text{in}} &= \bigwedge_{s' \in \text{Succ}(s^{\text{in}})} \llbracket (s^{\text{in}}, s') \rrbracket_{\geq P(s^{\text{in}}, s')} \\ \alpha &= Q \\ \delta((s, s'), \sigma) &= \bigwedge_{\{s'' \mid P(s', s'') > 0\}} \llbracket (s', s'') \rrbracket_{\geq P(s', s'')} \quad \text{if } \sigma = L(s) \\ \delta((s, s'), \sigma) &= \text{ff} \quad \text{if } \sigma \neq L(s) \end{aligned}$$

A state (s, s') represents the transition from s to s' . The embedding is the same as the one defined in Section 6.3.2, except that this automaton uses \wedge instead of $*$.

Consider the Markov chain M in Figure 6.5. No distinct locations of M are bisimilar. Let M_1 be M with s_1 as initial location, and let M_2 be M with s_2 as initial location. We show that $A_{M_1}^w$ accepts M_2 although M_1 and M_2 are not bisimilar.

The Markov chains M_1 and M_2 are not bisimilar, as the transitions from s_1 to locations whose label is **b** have probability $\frac{2}{3}$ and the transitions from s_2 to locations whose label is **b** have probability $\frac{1}{3}$.

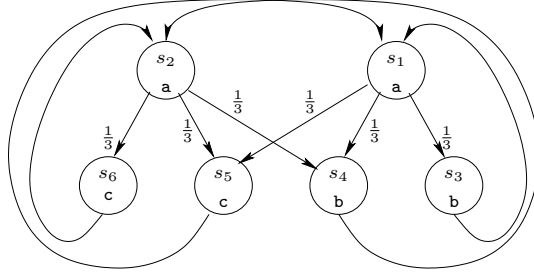


Figure 6.5: A Markov chain M whose very weak embedding accepts non-bisimilar Markov chains.

Nevertheless, $A_{M_1}^w$ accepts M_2 and is therefore not a “correct” embedding of M_1 .

Lemma 16

The automaton $A_{M_1}^w$ accepts M_2 . ●

Proof

The initial configuration of the acceptance game for $M_2 \in \mathcal{L}(A_{M_1}^w)$ is configuration $(s_2, \varphi^{\text{in}})$. As φ^{in} is a conjunction, Player 1 can choose one of three successor configurations:

$$\begin{aligned} & (s_2, \llbracket (s_1, s_3) \rrbracket_{\geq \frac{1}{3}}) \\ & (s_2, \llbracket (s_1, s_4) \rrbracket_{\geq \frac{1}{3}}) \\ & (s_2, \llbracket (s_1, s_5) \rrbracket_{\geq \frac{1}{3}}) \end{aligned}$$

One can see that Player 0 wins from the latter two.

Suppose that Player 1 chooses the configuration $(s_2, \llbracket (s_1, s_3) \rrbracket_{\geq \frac{1}{3}})$. Player 0 chooses the configuration $(s_2, \llbracket (s_1, s_3) \rrbracket_{\geq \frac{1}{3}}, f)$ where f is the function that sets $f(1, s_4) = 1$ and $f(1, s_5) = f(1, s_6) = 0$. The next configuration is $(s_4, \llbracket (s_3, s_1) \rrbracket_{\geq 1}, 1)$. We complete a cycle by going back to configuration $(s_2, \varphi^{\text{in}})$.

This completes a winning strategy for Player 0. ■

6.6 Summary of chapter

We presented a novel kind of automata, called p-automata, that read an entire Markov chain and either accept or reject that input. We demonstrated how this acceptance can be decided through a series of stochastic weak games and weak games, at worst case exponential in the size of the automaton and in the size of the Markov chain.

We proved that p-automata are closed under Boolean operations; that language containment and emptiness are inter-reducible; and that the language of a p-automaton is closed under bisimulation. The bisimulation equivalence class of a Markov chain, and the set of models of a PCTL formula were shown to be expressible as such languages. In particular, the complexity of the acceptance game matches that of probabilistic model checking for automata which stem from such formulae.

These results suggest that emptiness, universality, and containment of p-automata are all tightly related to the open problem of decidability of PCTL satisfiability.

We developed a (fair) simulation between p-automata that stem from Markov chains or PCTL formulae. We proved this simulation to be decidable in EXPTIME and to under-approximate language containment. For p-automata A_M that stem from Markov chains, simulation $A_M \leq B$ is even equivalent to language inclusion $\mathcal{L}(A_M) \subseteq \mathcal{L}(B)$, which is essential for proving that p-automata are complete for full PCTL.

Finally, we proved that p-automata are a complete abstraction framework for full PCTL. If an infinite Markov chain satisfies a PCTL formula, there is a finite p-automaton that abstracts this Markov chain and whose language is contained in that of the p-automaton stemming from that PCTL formula.

7 Evaluation

In this chapter we set the results from this thesis in the context of relevant related work; we reflect on the achievement of our initial research aims; and we suggest directions for future work which could continue from the main results in this thesis.

As relevant related work we discuss work on completeness of abstractions for linear-time, branching-time and probabilistic logics; and work on automata and games for probabilistic logics.

To evaluate our achievements, we re-visit the research aims as they are set out in Section 1.7.

The suggested future work builds on our results from Chapter 5 and Chapter 6, with a particular focus on the p-automata framework which opens a whole range of promising future research questions.

7.1 Related work on completeness of abstractions

Historically the completeness of abstractions came up in the contexts of *optimality* and *precision* of (sound) abstractions [48, 57, 61, 116, 44]. First completeness results were developed for safety properties, e.g. the ACTL* fragment of CTL* which also contains the safety properties of LTL, [44]. For safety properties conservative approximations are already sufficient for completeness. Liveness, i.e. existential quantification, and the related notions of fairness were more difficult [48]. Completeness for full LTL was achieved by Yonit Kesten and Amir Pnueli [117]. They overcame the problems with liveness properties by *augmenting* the concrete model with so-called *progress monitors* prior to abstraction. These progress monitors explicitly encode fairness constraints into the model. A conservative approximation of the augmented model then yields sound and complete abstraction for full LTL.

For branching time logics, e.g. CTL and CTL*, abstraction techniques were developed and evaluated with respect to their precision [48, 57, 44].

To find a complete abstraction framework for branching time turned out to be much more difficult than for LTL. Abstract models with 3-valued labels or “may- and must-transitions” were important developments, but these techniques were not sufficient to achieve completeness for full CTL. Thus a new class of more expressive abstract model had to be developed. Already in 1994 Edmund Clarke suggested that such abstractions would require “more complex finite-state model[s] such as AND/OR graphs” to preserve enough information about the branching behaviour of the concrete model [44]. Completeness for branching-time logics was finally achieved by Dennis Dams and Kedar Namjoshi [58], and several complete abstraction frameworks were consecutively developed.

Dams and Namjoshi defined *focused transition systems* (FTS) as the first complete abstraction framework for branching-time logics [58]. Focused transition systems are “a class of transition systems that closely corresponds to [alternating tree] automata” [58]. The models use focusing and defocusing operations which “are the transition system analogues of the OR and AND operators in alternating tree automata” [58].

Some weaknesses of focus transition systems were pointed out by Harald Fecher and Michael Huth in [73]. Other abstract models, that were developed for complete abstraction frameworks, include *alternating transition systems* [146], *disjunctive modal transition systems (DMTS)* [73] and *hypermixed Kripke structures* [74]. All these abstract models use some kind of “OR transition to sets of successors” to abstract the concrete branching behaviour. Dams and Namjoshi conjecture that “any abstraction framework that is complete is likely to correspond closely to the [alternating tree] automaton framework” [58].

The completeness question for branching time was brought to a conclusion through the correspondence between the modal μ -calculus and alternating tree automata. By phrasing the completeness problem purely in the automata framework, with automata both as models and as specification, Dams and Namjoshi provide a unifying view on the other proposed complete abstractions [59]. The automata framework also provides a very elegant solution for the completeness problem as the finite automaton which corresponds to the logical specification, by definition, constitutes a finite state abstraction of all automata which correspond to models of that specification [59].

There is only little work on the completeness problem for probabilistic logics, in particular over discrete-time Markov chains which are the main focus of this thesis. Abstraction for continuous-time stochastic processes which respects certain fragments of the widely used Continuous Stochastic Logic (CSL) [12, 13, 19] over continuous-time Markov chains are considered in [18] and [21]. Bisimulations which preserve CSL over continuous-time Markov decision processes are discussed in [147]. In particular [21] discusses abstractions which preserve liveness and safety properties respectively. Micheal Smith presents compositional abstractions for continuous-time Markov chains based on aggregation of states and on stochastic bounds [162, 163]. These abstraction techniques resemble state lumping [64] and interval Markov chains [39] respectively, and yield useful bounds for CSL reachability properties. Finite-state abstractions for PCTL over infinite-state discrete-time Markov chains are constructed via predicate abstraction in [100]. The method is shown to be – in a certain sense – *optimal*, but is not complete. In the context of PCTL satisfiability, the synthesis of finitary descriptions of infinite models for formulae of qualitative PCTL is addressed in [30]. A 3-valued abstraction for continuous-time Markov chains is discussed in [109]. The non-determinism of Markov decision processes allows for similar abstraction hierarchies as for non-probabilistic models. Abstraction refinement techniques for Markov decision processes have been studied in [65, 126, 63]. The game-based abstraction of Markov decision processes in [126, 113] was shown to be not complete by Kattenbelt and Huth in [111]. None of these works address the completeness question for PCTL over (discrete-time) Markov chains.

7.1.1 Related Work on Automata and Games

Hintikka games as a means to define an operational semantics have been used for various non-probabilistic logics. Colin Stirling developed semantic games [167] and model-checking games [166, 129] for CTL^* and the μ -calculus. Model-checking games for the modal μ -calculus [72, 146] also played an important role in achieving completeness for branching time [146, 58, 59].

Quantitative parity games are developed and shown to correspond to model checks for formulae of a quantitative μ -calculus in [78]. However, winning strategies are no longer memoryless in general as they may have

to “make up” for discount factors encountered en-route in a play – even in games with finite set of configurations.

In [142], a quantitative μ -calculus ($\text{qM}\mu$) is defined over models that contain both non-deterministic and probabilistic choice but no discounting. A denotational semantics generalising Kozen’s familiar one [121] is given. For any finite-state model and formula of $\text{qM}\mu$ a probabilistic analogue of parity games is given, and the determinacy of this game is shown. It is also proved that its game value equals that of the denotational semantics for the model and formula in question and that there exist memoryless winning strategies.

We are not aware of any previous effort to develop a game-based semantics for PCTL over Markov chains. Nevertheless games are an established method to reason about PCTL over Markov decision processes where they allow for “angelic” respectively “demonic” interpretation of non-deterministic choices. The stochastic games of [126, 113] abstract Markov decision processes as 2-person games where two sources of non-determinism, stemming from the Markov decision process and the state space partition respectively, are controlled by different players. This separation allows for more precise abstractions but is not complete in the sense of [58], as shown in [111].

Rabin’s probabilistic automata [157] give only rise to probabilistic languages of *non*-probabilistic models.

In [176], automata also accept Markov chains – when interpreted as co-algebras of a certain functor. These automata are shown to be equivalent to a fixed point logic that enjoys the finite model property. Since PCTL does not have that property, these co-algebraic automata cannot express PCTL – notably its path modalities. Also, this work does not feature simulations or other abstraction mechanisms.

Probabilistic processes [108] use automata-theoretic techniques for refinement checking only. Probabilistic verification of specifications written in linear-time temporal logic (LTL) [173, 175] uses automata-theoretic machinery but cannot reason about combinations of LTL operators and probability thresholds as found in PCTL.

7.2 Evaluation of achievements

The aims for our research are stated in Section 1.7. We now re-visit these aims and assess how our achievements match our aims.

The overall aim of this thesis was to find a complete abstraction framework for PCTL over discrete-time Markov chains. This goal was achieved in full through our p-automata framework in Chapter 6.

Further we set out to develop a better understanding of PCTL and probabilistic verification in general. Our operational semantics for PCTL in Chapter 5 contributes to this aim, as does the identification of a “benign” PCTL fragment in Chapter 4 for which we secured completeness via 3-valued unfoldings. Our p-automata enhance the understanding of PCTL further, as they open a new line of attack on the open problem of PCTL satisfiability.

Our last aim was to develop the complete abstraction framework in such a way, that it will be useful above and beyond answering our completeness question. This aim was met, as our p-automata constitute a novel framework for Markov chains and PCTL which provides the foundation for an automata-based approach to discrete-time probabilistic verification.

7.3 Future work

The most promising direction for future work based on the achievements in this thesis, is clearly the further development of our p-automata. The other two main results, i.e. the Hintikka game and completeness for a PCTL fragment via unfoldings, offer less potential for future work. Nevertheless, all of our results should scale reasonably well to standard extensions of the class of discrete-time Markov chains considered in this thesis, such as Markov chains with multiple initial states or with a distribution over a set of initial states.

The fragment of PCTL for which completeness can be achieved with 3-valued unfoldings is fully defined and its maximality is proved. The work on this result is therefore finished and we do not intend to pursue further research on this fragment. For practical applications it could be of interest to explore the usability of specifications which are restricted to that fragment. For Markov decision processes not even our fragment $\text{PCTL}_{>}$ is complete as shown in [112].

Our Hintikka game could possibly be extended from PCTL to PCTL^* , but Colin Stirling’s games for CTL^* [167] indicate that such an extension would need to be rather complex. An extension to PCTL with a suitable semantics over Markov decision processes should be reasonable straightforward, as the

non-determinism of Markov decision processes can be captured by adding another non-probabilistic, non-deterministic choice for one of the two players in the Hintikka game.

The Hintikka game for PCTL over Markov chains provided valuable insights into PCTL and enabled us to understand the limitations of 3-valued abstractions for achieving completeness for full PCTL. Based on these insights we developed our p-automata, which solve the completeness problem for full PCTL. Our p-automata are more general than PCTL*, i.e. PCTL* can be embedded into p-automata just as PCTL can. The automata-based framework for PCTL and Markov chains uses games for acceptance and refinement. These games resemble, respectively generalise, our Hintikka game. Thus we do not expect much added value from extending the Hintikka game to, e.g., PCTL*. More interesting would be to relate future developments on probabilistic μ -calculi and the corresponding games [142] directly to our p-automata.

Our p-automata achieve completeness for full PCTL, and therefore fully answer the main research question of this thesis. Beyond the completeness question, our p-automata form a novel framework for automata based probabilistic verification. This opens a whole range of new research questions and suggests some promising directions for future work. In particular we would like:

1. To develop a notion of p-automata that embed Markov decision processes and stochastic games.
2. To understand the difference between alternating and non-deterministic p-automata, where the latter notion still needs to be defined.
3. To prove or refute Theorem 10, i.e. that simulation implies language inclusion, for infinite-state p-automata.
4. To develop a more general notion of game such that acceptance of input for *non-uniform* p-automata can be decided by solving a *single* such game.
5. To prove matching lower bounds for acceptance by p-automata.

In Section 6.3.3 we define an embedding of PCTL into p-automata which establishes some highly interesting connections and potential for future research. In particular acceptance of p-automata is directly related to PCTL

model checking, and language emptiness for p-automata is related to PCTL satisfiability. As we mention in Section 6.3.3, we believe that language emptiness is decidable for p-automata which correspond to qualitative PCTL, i.e. a PCTL fragment for which satisfiability is known to be decidable. Future work in this direction could attempt to prove this conjecture. Furthermore, it could investigate the relations between p-automata and other formalisms for which PCTL model checking or satisfiability is known to be decidable [90, 30].

Another interesting direction for future research is to leave the setting of PCTL over discrete-time systems which we fixed for this thesis. It could be interesting to transfer the results from this thesis to other settings, e.g. to other probabilistic logic and probabilistic models in particular in continuous-time settings. Many of our results can be lifted to continuous-time Markov chains by uniformising continuous-time Markov chain M and considering its embedded discrete-time Markov chain [19]. An approach via uniformisation and embedded discrete-time Markov chains is appealing and has been used in the context of model checking before [110, 19]. Nevertheless, our results clearly live in a discrete-time world. It is not clear whether and how our step-based game semantics and our p-automata framework could be transferred to a continuous-time logic such as CSL.

Some of these questions for future work are currently addressed in joint work with Michael Huth and Nir Piterman. This research is work in progress and beyond the scope of this thesis.

Bibliography

- [1] PRISM model checker. <http://www.prismmodelchecker.org/>.
- [2] Conference on verified software: Theories, tools, experiments. <http://vstte.inf.ethz.ch/>, 2005.
- [3] Jean-Raymond Abrial. A formal approach to large software construction. In *MPC 1989*, Mathematics of Program Construction, volume 375 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, 1989.
- [4] Jean-Raymond Abrial. *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.
- [5] Jean-Raymond Abrial, Matthew K. O. Lee, David Neilson, P. N. Scharbach, and Ib Holm Sørensen. The B-method. In *VDM Europe (2)*, 4th International Symposium of VDM Europe, volume 552 of *Lecture Notes in Computer Science*, pages 398–405. Springer-Verlag, 1991.
- [6] Husain Aljazzar and Stefan Leue. Counterexamples for model checking of Markov decision processes. Technical Report soft-08-01, Department of Computer and Information Science, University of Konstanz, Germany, 2008.
- [7] Husain Aljazzar and Stefan Leue. Debugging of dependability models using interactive visualization of counterexamples. In *QEST 2008*, 5th International Conference on the Quantitative Evaluation of Systems, pages 189–198. IEEE Computer Society, 2008.
- [8] Bowen Alpern and Fred B. Schneider. Defining liveness. *Inf. Process. Lett.*, 21(4):181–185, 1985.
- [9] Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.
- [10] Henrik Reif Andersen and Glynn Winskel. Compositional checking of satisfaction. *Formal Methods in System Design*, 1(4):323–354, 1992.
- [11] Miguel E. Andrés, Pedro R. D’Argenio, and Peter van Rossum. Significant diagnostic counterexamples in probabilistic model

- checking. *The Computing Research Repository (CoRR)*, abs/0806.1139:1–17, 2008.
- [12] Adnan Aziz, Kumud Sanwal, Vigyan Singhal and Robert K. Brayton. Verifying Continuous Time Markov Chains. In *CAV 1996*, 8th International Conference on Computer Aided Verification, volume 1102 of *Lecture Notes in Computer Science*, pages 269–276. Springer-Verlag, 1996.
 - [13] Adnan Aziz, Kumud Sanwal, Vigyan Singhal and Robert K. Brayton. Model-checking continuous-time Markov chains. *ACM Trans. Comput. Log.*, 1(1):162–170, 2000.
 - [14] Adnan Aziz, Vigyan Singhal, and Felice Balarin. It usually works: The temporal logic of stochastic systems. In *CAV 1995*, 7th International Conference on Computer Aided Verification, volume 939 of *Lecture Notes in Computer Science*, pages 155–165. Springer-Verlag, 1995.
 - [15] Charles Babbage. On the mathematical powers of the calculating engine. In *The Origins of Digital Computers: Selected Papers*, pages 19–54. Springer-Verlag, 1975.
 - [16] Christel Baier, Nathalie Bertrand, and Marcus Größer. Probabilistic acceptors for languages over infinite words. In *SOFSEM 2009*, volume 5404 of *Lecture Notes in Computer Science*, pages 19–33. Springer-Verlag, 2009.
 - [17] Christel Baier, Edmund M. Clarke, Vassili Hartonas-Garmhausen, Marta Z. Kwiatkowska, and Mark Ryan. Symbolic model checking for probabilistic processes. In *ICALP 1997*, 24th International Colloquium on Automata, Languages and Programming, volume 1256 of *Lecture Notes in Computer Science*, pages 430–440. Springer-Verlag, 1997.
 - [18] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns and Joost-Pieter Katoen. Model Checking Continuous-Time Markov Chains by Transient Analysis. In *CAV 2000*, 12th International Conference on Computer Aided Verification, volume 1855 of *Lecture Notes in Computer Science*, pages 358–372. Springer-Verlag, 2000.
 - [19] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns and Joost-Pieter Katoen. Model-Checking Algorithms for Continuous-Time Markov Chains. In *IEEE Trans. Software Eng.*, 29(6):524–541, 2003.

- [20] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.
- [21] Christel Baier, Joost-Pieter Katoen, Holger Hermanns and Boudewijn R. Haverkort. Simulation for Continuous-Time Markov Chains. In *CONCUR 2002*, 13th International Conference on Concurrency Theory, volume 2421 of *Lecture Notes in Computer Science*, pages 338–354. Springer-Verlag, 2002.
- [22] Christel Baier, Joost-Pieter Katoen, Holger Hermanns, and Verena Wolf. Comparative branching-time semantics for Markov chains. *Inf. Comput.*, 200(2):149–214, 2005.
- [23] Thomas Ball, Rupak Majumdar, Todd D. Millstein, and Sriram K. Rajamani. Automatic predicate abstraction of C programs. In *PLDI*, pages 203–213. ACM, 2001.
- [24] Danièle Beauquier, Alexander Moshe Rabinovich, and Anatol Slissenko. A logic of probability with decidable model-checking. In *CSL 2002*, 16th International Workshop on Computer Science Logic, volume 2471 of *Lecture Notes in Computer Science*, pages 306–321. Springer-Verlag, 2002.
- [25] Orna Bernholtz, Moshe Y. Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking (extended abstract). In *CAV 1994*, 6th International Conference on Computer Aided Verification, volume 818 of *Lecture Notes in Computer Science*, pages 142–155. Springer-Verlag, 1994.
- [26] Andrea Bianco and Luca de Alfaro. Model checking of probabalistic and nondeterministic systems. In *FSTTCS 1995*, 15th Conference on Foundations of Software Technology and Theoretical Computer Science, volume 1026 of *Lecture Notes in Computer Science*, pages 499–513. Springer-Verlag, 1995.
- [27] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In *TACAS 1999*, 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer-Verlag, 1999.
- [28] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. Cambridge University Press, 2001.
- [29] Julian C. Bradfield. On the expressivity of the modal mu-calculus. In *STACS*, 13th Annual Symposium on Theoretical Aspects of

Computer Science, volume 1046 of *Lecture Notes in Computer Science*, pages 479–490. Springer-Verlag, 1996.

- [30] Tomáš Brázdil, Vojtech Forejt, Jan Kretínský, and Antonín Kucera. The satisfiability problem for probabilistic CTL. In *LICS 2008*, 23rd Annual IEEE Symposium on Logic in Computer Science, pages 391–402. IEEE Computer Society, 2008.
- [31] Glenn Bruns and Patrice Godefroid. Model checking partial state spaces with 3-valued temporal logics. In *CAV 2000*, 12th International Conference on Computer Aided Verification, volume 1633 of *Lecture Notes in Computer Science*, pages 274–287. Springer-Verlag, 1999.
- [32] Glenn Bruns and Patrice Godefroid. Generalized model checking: Reasoning about partial state spaces. In *CONCUR 2000*, 11th International Conference on Concurrency Theory, volume 1877 of *Lecture Notes in Computer Science*, pages 168–182. Springer-Verlag, 2000.
- [33] Richard J. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6(1-6):66–92, 1960.
- [34] M. Calder and A. Miller. Five ways to use induction and symmetry in the verification of networks of processes by model-checking. In *AVoCS 2002*, 2nd Workshop on Automated Verification of Critical Systems, pages 29–42, 2002.
- [35] Sérgio Vale Aguiar Campos, Edmund M. Clarke, Wilfredo R. Marrero, and Marius Minea. Verus: A tool for quantitative analysis of finite-state real-time systems. In *Workshop on Languages, Compilers, & Tools for Real-Time Systems*, pages 70–78. ACM, 1995.
- [36] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.
- [37] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Generalized parity games. In *FoSSaCS 2007*, 10th International Conference on Foundations of Software Science and Computational Structures, volume 4423 of *Lecture Notes in Computer Science*, pages 153–167. Springer-Verlag, 2007.
- [38] Krishnendu Chatterjee, Marcin Jurdzinski, and Thomas A. Henzinger. Quantitative stochastic parity games. In *SODA 2004*, 15th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 121–130. ACM, 2004.

- [39] Krishnendu Chatterjee, Koushik Sen and Thomas A. Henzinger. Model-Checking omega-Regular Properties of Interval Markov Chains. In *FoSSaCS 2008*, 11th International Conference on Foundations of Software Science and Computational Structures, volume 4962 of *Lecture Notes in Computer Science*, pages 302–317. Springer-Verlag, 2008.
- [40] Marsha Chechik, Steve M. Easterbrook, and Victor Petrovykh. Model-checking over multi-valued logics. In *FME 2001: Formal Methods for Increasing Software Productivity*, International Symposium of Formal Methods Europe, volume 2021 of *Lecture Notes in Computer Science*, pages 72–98. Springer-Verlag, 2001.
- [41] Edmund M. Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.
- [42] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer-Verlag, 1981.
- [43] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In *CAV 2000*, 12th International Conference on Computer Aided Verification, volume 1855 of *Lecture Notes in Computer Science*, pages 154–169. Springer-Verlag, 2000.
- [44] Edmund M. Clarke, Orna Grumberg, and David E. Long. Model checking and abstraction. *ACM Trans. Program. Lang. Syst.*, 16(5):1512–1542, 1994.
- [45] Edmund M. Clarke and Somesh Jha. Symmetry and induction in model checking. In *Computer Science Today*, volume 1000 of *Lecture Notes in Computer Science*, pages 455–470. Springer-Verlag, 1995.
- [46] Edmund M. Clarke, Somesh Jha, Yuan Lu, and Helmut Veith. Tree-like counterexamples in model checking. In *LICS 2002*, 17th IEEE Symposium on Logic in Computer Science, pages 19–29. IEEE Computer Society, 2002.
- [47] Edmund M. Clarke and Orna Grumberg Doron A. Peled. *Model checking*. MIT Press, 1999.
- [48] Rance Cleaveland, S. Purushothaman Iyer, and Daniel Yankelevich. Optimality in abstractions of model checking. In *SAS 1995*, 2nd

- International Symposium on Static Analysis, volume 983 of *Lecture Notes in Computer Science*, pages 51–63. Springer-Verlag, 1995.
- [49] Hubert Comon, Max Dauchet, Remi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications. <http://www.grappa.univ-lille3.fr/tata>, 2007.
- [50] Anne Condon. The complexity of stochastic games. *Inf. Comput.*, 96(2):203–224, 1992.
- [51] Costas Courcoubetis, Moshe Y. Vardi, Pierre Wolper, and Mihalis Yannakakis. Memory-efficient algorithms for the verification of temporal properties. *Formal Methods in System Design*, 1(2/3):275–288, 1992.
- [52] Costas Courcoubetis and Mihalis Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In *FOCS 1988*, 29th Annual Symposium on Foundations of Computer Science, pages 338–345. IEEE Computer Society, 1988.
- [53] Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.
- [54] Mads Dam. CTL* and ECTL* as fragments of the modal μ -calculus. In *CAAP 1992*, 17th Colloquium on Trees in Algebra and Programming, volume 581 of *Lecture Notes in Computer Science*, pages 145–164. Springer-Verlag, 1992.
- [55] Berteun Damman, Tingting Han, and Joost-Pieter Katoen. Regular expressions for PCTL counterexamples. In *QEST 2008*, 5th International Conference on the Quantitative Evaluation of Systems, pages 179–188. IEEE Computer Society, 2008.
- [56] Dennis Dams. Comparing abstraction refinement algorithms. *Electr. Notes Theor. Comput. Sci.*, 89(3):405–416, 2003.
- [57] Dennis Dams, Rob Gerth, and Orna Grumberg. Abstract interpretation of reactive systems. *ACM Trans. Program. Lang. Syst.*, 19(2):253–291, 1997.
- [58] Dennis Dams and Kedar S. Namjoshi. The existence of finite abstractions for branching time model checking. In *LICS 2004*, 19th IEEE Symposium on Logic in Computer Science, pages 335–344. IEEE Computer Society, 2004.

- [59] Dennis Dams and Kedar S. Namjoshi. Automata as abstractions. In *VMCAI 2005*, 6th International Conference on Verification, Model Checking, and Abstract Interpretation, volume 3385 of *Lecture Notes in Computer Science*, pages 216–232. Springer-Verlag, 2005.
- [60] Pedro R. D’Argenio, Bertrand Jeannet, Henrik Ejersbo Jensen, and Kim Guldstrand Larsen. Reachability analysis of probabilistic systems by successive refinements. In *PAPM-PROBMIV 2001*, Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification, volume 2165 of *Lecture Notes in Computer Science*, pages 39–56. Springer-Verlag, 2001.
- [61] Luca de Alfaro, Patrice Godefroid, and Radha Jagadeesan. Three-valued abstractions of games: Uncertainty, but with precision. In *LICS 2004*, 19th IEEE Symposium on Logic in Computer Science, pages 170–179. IEEE Computer Society, 2004.
- [62] Luca de Alfaro, Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Roberto Segala. Symbolic model checking of probabilistic processes using MTBDDs and the Kronecker representation. In *TACAS 2000*, 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems, volume 1785 of *Lecture Notes in Computer Science*, pages 395–410. Springer-Verlag, 2000.
- [63] Luca de Alfaro and Pritam Roy. Magnifying-lens abstraction for Markov decision processes. In *CAV 2007*, 19th International Conference on Computer Aided Verification, volume 4590 of *Lecture Notes in Computer Science*, pages 325–338. Springer-Verlag, 2007.
- [64] Salem Derisavi, Holger Hermanns and William H. Sanders. Optimal state-space lumping in Markov chains. *Inf. Process. Lett.*, 87(6):309–315, 2003.
- [65] Josee Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Approximating labeled Markov processes. In *LICS 2000*, 15th IEEE Symposium on Logic in Computer Science, pages 95–106. IEEE Computer Society, 2000.
- [66] Josee Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Weak bisimulation is sound and complete for PCTL*. In *CONCUR 2002*, 13th International Conference on Concurrency Theory, volume 2421 of *Lecture Notes in Computer Science*, pages 355–370. Springer-Verlag, 2002.

- [67] Andrzej Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fundamenta Mathematicae*, 49:129–141, 1961.
- [68] E. Allen Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 995–1072. Elsevier and MIT Press, 1990.
- [69] E. Allen Emerson and Edmund M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *ICALP 1980, 7th Colloquium on Automata, Languages and Programming*, volume 85 of *Lecture Notes in Computer Science*, pages 169–181. Springer-Verlag, 1980.
- [70] E. Allen Emerson and Joseph Y. Halpern. “Sometimes” and “Not Never” revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.
- [71] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FOCS 1991, 32nd Annual Symposium on Foundations of Computer Science*, pages 368–377. IEEE Computer Society, 1991.
- [72] E. Allen Emerson, Charanjit S. Jutla, and A. Prasad Sistla. On model-checking for fragments of μ -calculus. In *CAV 1993, 5th International Conference on Computer Aided Verification*, volume 697 of *Lecture Notes in Computer Science*, pages 385–396. Springer-Verlag, 1993.
- [73] Harald Fecher and Michael Huth. Complete abstractions through extensions of disjunctive modal transition systems. Technical Report 0604, Christian-Albrechts-Universität zu Kiel, 2006.
- [74] Harald Fecher and Michael Huth. Ranked predicate abstraction for branching time: Complete, incremental, and precise. In *ATVA 2006, 4th International Symposium on Automated Technology for Verification and Analysis*, volume 4218 of *Lecture Notes in Computer Science*, pages 322–336. Springer-Verlag, 2006.
- [75] Harald Fecher, Michael Huth, Nir Piterman, and Daniel Wagner. Hintikka games for PCTL on labeled Markov chains. In *QEST 2008, 5th International Conference on the Quantitative Evaluation of Systems*, pages 169–178. IEEE Computer Society, 2008.
- [76] Harald Fecher, Michael Huth, Nir Piterman, and Daniel Wagner. PCTL model checking of Markov chains: Truth and falsity as winning strategies in games. *Performance Evaluation*, 67:858–872, 2010.

- [77] Harald Fecher, Michael Huth, Heiko Schmidt, and Jens Schönborn. Refinement sensitive formal semantics of state machines with persistent choice. *Electr. Notes Theor. Comput. Sci.*, 250(1):71–86, 2009.
- [78] Diana Fischer, Erich Grädel, and Lukasz Kaiser. Model checking games for the quantitative mu-calculus. In *STACS*, volume 1 of *LIPICs*, pages 301–312. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2008.
- [79] Carsten Fritz and Thomas Wilke. State space reductions for alternating Büchi automata. In *FSTTCS 2002*, 22nd Conference on Foundations of Software Technology and Theoretical Computer Science, volume 2556 of *Lecture Notes in Computer Science*, pages 157–168. Springer-Verlag, 2002.
- [80] Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *PSTV 1995*, 15th International Symposium on Protocol Specification, Testing and Verification, volume 38 of *IFIP Conference Proceedings*, pages 3–18. Chapman & Hall, 1995.
- [81] Patrice Godefroid, Michael Huth, and Radha Jagadeesan. Abstraction-based model checking using modal transition systems. In *CONCUR 2001*, 12th International Conference on Concurrency Theory, volume 2154 of *Lecture Notes in Computer Science*, pages 426–440. Springer-Verlag, 2001.
- [82] Patrice Godefroid and Radha Jagadeesan. On the expressiveness of 3-valued models. In *VMCAI 2003*, 4th International Conference on Verification, Model Checking, and Abstract Interpretation, volume 2575 of *Lecture Notes in Computer Science*, pages 206–222. Springer-Verlag, 2003.
- [83] Herman Goldstine and John von Neumann. *Planning and Coding of Problems for an Electronic Computing Instrument*. Institute for Advanced Study, Princeton, 1948. Reprinted in *von Neumann's Collected Works*, volume 5, pages 152-214. Pergamon, London, 1963.
- [84] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research (Outcome of a Dagstuhl seminar, February 2001)*, volume 2500 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.
- [85] Charles M. Grinstead and Laurie Snell. *Introduction to Probability*. AMS, 1997.

- [86] Orna Grumberg and David E. Long. Model checking and modular verification. *ACM Trans. Program. Lang. Syst.*, 16(3):843–871, 1994.
- [87] Tingting Han and Joost-Pieter Katoen. Counterexamples in probabilistic model checking. In *TACAS 2007*, 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, volume 4424 of *Lecture Notes in Computer Science*, pages 72–86. Springer-Verlag, 2007.
- [88] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994.
- [89] Sergiu Hart and Micha Sharir. Probabilistic temporal logics for finite and bounded models. In *STOC 1984*, 16th Annual ACM Symposium on Theory of Computing, pages 1–13. ACM, 1984.
- [90] Sergiu Hart and Micha Sharir. Probabilistic propositional temporal logics. *Information and Control*, 70(2/3):97–155, 1986.
- [91] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.
- [92] Thomas A. Henzinger, Orna Kupferman, and Sriram K. Rajamani. Fair simulation. *Information and Computation*, 173(1):64–81, 2002.
- [93] Holger Hermanns, Björn Wachter, and Lijun Zhang. Probabilistic CEGAR. In *CAV 2008*, 20th International Conference on Computer Aided Verification, volume 5123 of *Lecture Notes in Computer Science*, pages 162–175. Springer-Verlag, 2008.
- [94] Jaakko Hintikka. *Logic, Language-Games and Information: Kantian Themes in the Philosophy of Logic*. Oxford University Press, 1973.
- [95] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.
- [96] C. A. R. Hoare. The verifying compiler: A grand challenge for computing research. *J. ACM*, 50(1):63–69, 2003.
- [97] Michael Huth. A unifying framework for model checking labeled Kripke structures, modal transition systems and interval transition systems. In *FSTTCS 1999*, 19th Conference on Foundations of Software Technology and Theoretical Computer Science, volume 1738 of *Lecture Notes in Computer Science*, pages 369–380. Springer-Verlag, 1999.
- [98] Michael Huth. Domains of view: A foundation for specification and analysis. In *Domains and Processes*, pages 183–218. Kluwer Academic Publishers, 2001.

- [99] Michael Huth. An abstraction framework for mixed non-deterministic and probabilistic systems. In *Validation of Stochastic Systems – A Guide to Current Research*, volume 2925 of *Lecture Notes in Computer Science*, pages 419–444. Springer-Verlag, 2004.
- [100] Michael Huth. On finite-state approximants for probabilistic computation tree logic. *Theor. Comput. Sci.*, 346(1):113–134, 2005.
- [101] Michael Huth. Some current topics in model checking. *Int J Softw Tools Technol Transfer*, 9:25–36, 2007.
- [102] Michael Huth, Radha Jagadeesan, and David A. Schmidt. Modal transition systems: A foundation for three-valued program analysis. In *ESOP 2001*, 10th European Symposium on Programming Languages and Systems, volume 2028 of *Lecture Notes in Computer Science*, pages 155–169. Springer-Verlag, 2001.
- [103] Michael Huth, Nir Piterman, and Daniel Wagner. Three-valued abstractions of Markov chains: Completeness for a sizeable fragment of PCTL. In *FCT 2009*, 17th International Symposium on Fundamentals of Computation Theory, volume 5699 of *Lecture Notes in Computer Science*, pages 205–216. Springer-Verlag, 2009.
- [104] Michael Huth, Nir Piterman, and Daniel Wagner. p-Automata: New foundations for discrete-time probabilistic verification. In *QEST 2010*, 7th International Conference on Quantitative Evaluation of Systems, pages 161–170. IEEE Computer Society, 2010.
- [105] Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2004.
- [106] Somesh Jha. *Symmetry and Induction in Model Checking*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1996.
- [107] Cliff B. Jones. The early search for tractable ways of reasoning about programs. *IEEE Annals of the History of Computing*, 25(2):26–49, 2003.
- [108] Bengt Jonsson and Kim Guldstrand Larsen. Specification and refinement of probabilistic processes. In *LICS 1991*, 6th Annual IEEE Symposium on Logic in Computer Science, pages 266–277. IEEE Computer Society, 1991.
- [109] Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf. Three-valued abstraction for continuous-time Markov chains. In *CAV*, volume 4590 of *Lecture Notes in Computer Science*, pages 311–324. Springer-Verlag, 2007.

- [110] Joost-Pieter Katoen, Marta Z. Kwiatkowska, Gethin Norman and David Parker. Faster and Symbolic CTMC Model Checking. In *PAPM-PROBMIV 2001*, Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification, volume 2165 of *Lecture Notes in Computer Science*, pages 23–38. Springer-Verlag, 2001.
- [111] Mark Kattenbelt and Michael Huth. Abstraction framework for Markov decision processes and PCTL via game. Technical Report RR-09-01, Oxford University Computing Laboratory, 2009.
- [112] Mark Kattenbelt and Michael Huth. Verification and refutation of probabilistic specifications via games. In *FSTTCS 2009*, 29th Conference on Foundations of Software Technology and Theoretical Computer Science, volume 4 of *LIPICs*, pages 251–262. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2009.
- [113] Mark Kattenbelt, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Game-based probabilistic predicate abstraction in PRISM. *Electr. Notes Theor. Comput. Sci.*, 220(3):5–21, 2008.
- [114] Mark Kattenbelt, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Abstraction refinement for probabilistic software. In *VMCAI 2009*, 10th International Conference Verification, Model Checking, and Abstract Interpretation, volume 5403 of *Lecture Notes in Computer Science*, pages 182–197. Springer-Verlag, 2009.
- [115] J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains*. Springer, second edition edition, 1976.
- [116] Yonit Kesten and Amir Pnueli. Modularization and abstraction: The keys to practical formal verification. In *Mathematical Foundations of Computer Science*, volume 1450 of *Lecture Notes in Computer Science*, pages 54–71. Springer-Verlag, 1998.
- [117] Yonit Kesten and Amir Pnueli. Verification by augmented finitary abstraction. *Inf. Comput.*, 163(1):203–243, 2000.
- [118] Stephen Kleene. *Introduction to Metamathematics*. Van Nostrand, 1952.
- [119] S. Rao Kosaraju. Analysis of structured programs. In *STOC 1973*, 5th Annual ACM Symposium on Theory of Computing, pages 240–252. ACM, 1973.
- [120] Dexter Kozen. Semantics of probabilistic programs. In *FOCS 1979*, 20th Annual Symposium on Foundations of Computer Science, pages 101–114. IEEE Computer Society, 1979.

- [121] Dexter Kozen. Semantics of probabilistic programs. *J. Comput. Syst. Sci.*, 22(3):328–350, 1981.
- [122] Dexter Kozen. Results on the propositional μ -calculus. In *ICALP 1982*, 9th Colloquium on Automata, Languages and Programming, volume 140 of *Lecture Notes in Computer Science*, pages 348–359. Springer-Verlag, 1982.
- [123] Orna Kupferman and Moshe Y. Vardi. Modular model checking. In *COMPOS 1997*, International Symposium on Compositionality: The Significant Difference, volume 1536 of *Lecture Notes in Computer Science*, pages 381–401. Springer-Verlag, 1997.
- [124] Orna Kupferman, Moshe Y. Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. *J. ACM*, 47(2):312–360, 2000.
- [125] Marta Z. Kwiatkowska. Model checking for probability and time: from theory to practice. In *LICS 2003*, 18th IEEE Symposium on Logic in Computer Science, pages 351–360. IEEE Computer Society, 2003.
- [126] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Game-based abstraction for Markov decision processes. In *QEST 2006*, 3rd International Conference on the Quantitative Evaluation of Systems, pages 157–166. IEEE Computer Society, 2006.
- [127] Leslie Lamport. “Sometime” is sometimes “Not Never” - on the temporal logic of programs. In *POPL 1980*, Principles of Programming Languages, pages 174–185. ACM, 1980.
- [128] Leslie Lamport. What good is temporal logic? In *IFIP Congress*, pages 657–668. Chapman & Hall, 1983.
- [129] Martin Lange and Colin Stirling. Model checking games for branching time logics. *J. Log. Comput.*, 12(4):623–639, 2002.
- [130] François Laroussinie. About the expressive power of CTL combinators. *Inf. Process. Lett.*, 54(6):343–345, 1995.
- [131] Kim Guldstrand Larsen and Arne Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991.
- [132] Kim Guldstrand Larsen and Arne Skou. Compositional verification of probabilistic processes. In *CONCUR 1992*, 3rd International Conference on Concurrency Theory, volume 630 of *Lecture Notes in Computer Science*, pages 456–471. Springer-Verlag, 1992.

- [133] Kim Guldstrand Larsen and Bent Thomsen. A modal process logic. In *LICS 1988*, 3rd IEEE Symposium on Logic in Computer Science, pages 203–210. IEEE Computer Society, 1988.
- [134] Orna Lichtenstein and Amir Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *POPL 1985*, Principles of Programming Languages, pages 97–107. ACM, 1985.
- [135] Paul Lorenzen. Logik und Agon. In *Arti del XII Congresso Internazionale de Filosofia*, pages 187–194, 1958. Reprinted in Kuno Lorenz and Paul Lorenzen, *Dialogische Logik*, pages 1–8, Wissenschaftliche Buchgesellschaft Darmstadt, 1978.
- [136] Paul Lorenzen. *Infinitistic Methods (Proc. Symp. Foundations of Mathematics, Warsaw, 1959)*, chapter Ein dialogisches Konstruktivitätskriterium, pages 193–200. Pergamon Press, 1961.
- [137] Zohar Manna. Properties of programs and the first-order predicate calculus. *J. ACM*, 16(2):244–255, 1969.
- [138] Panagiotis Manolios and Richard J. Treffer. Safety and liveness in branching time. In *LICS 2001*, 16th IEEE Symposium on Logic in Computer Science, pages 366–374. IEEE Computer Society, 2001.
- [139] Alan Martin. Adequate sets of temporal connectives in CTL. *Electr. Notes Theor. Comput. Sci.*, 52(1):21–31, 2001.
- [140] Donald A. Martin. Borel determinacy. *Annals of Mathematics*, 102:363–371, 1975.
- [141] Annabelle McIver and Carroll Morgan. Games, probability and the quantitative μ -calculus $qM\mu$. In *LPAR 2002*, 9th International Conference Logic for Programming, Artificial Intelligence, and Reasoning, volume 2514 of *Lecture Notes in Computer Science*, pages 292–310. Springer-Verlag, 2002.
- [142] Annabelle McIver and Carroll Morgan. Results on the quantitative μ -calculus $qM\mu$. *ACM Trans. Comput. Log.*, 8(1):1–45, 2007.
- [143] Kenneth L. McMillan. *Symbolic Model Checking*. Kluwer Academic, 1993.
- [144] David Michael and Ritchie Park. Concurrency and automata on infinite sequences. In *TCS 1981*, 5th GI-Conference on Theoretical Computer Science, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

- [145] Robin Milner. An algebraic definition of simulation between programs. Technical Report CS-TR-71-205, Stanford University, 1971.
- [146] Kedar S. Namjoshi. Abstraction for branching time properties. In *CAV 2003*, 15th International Conference on Computer Aided Verification, volume 2725 of *Lecture Notes in Computer Science*, pages 288–300. Springer-Verlag, 2003.
- [147] Martin R. Neuhäufser and Joost-Pieter Katoen. Bisimulation and Logical Preservation for Continuous-Time Markov Decision Processes. In *CONCUR 2007*, 18th International Conference on Concurrency Theory, volume 4703 of *Lecture Notes in Computer Science*, pages 412–427. Springer-Verlag, 2007.
- [148] Damian Niwinski. Fixed point characterization of infinite behavior of finite-state systems. *Theor. Comput. Sci.*, 189(1-2):1–69, 1997.
- [149] James R. Norris. *Markov Chains*. Cambridge Series on Statistical & Probabilistic Mathematics. Cambridge University Press, 1998.
- [150] Benjamin C. Pierce. *Basic Category Theory for Computer Scientists*. MIT Press, 1993.
- [151] Amir Pnueli. The temporal logic of programs. In *FOCS 1977*, 18th Annual Symposium on Foundations of Computer Science, pages 46–57. IEEE Computer Society, 1977.
- [152] Amir Pnueli. The temporal semantics of concurrent programs. In *SCC 1979*, International Symposium on Semantics of Concurrent Computation, volume 70 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, 1979.
- [153] Amir Pnueli. The temporal semantics of concurrent programs. *Theor. Comput. Sci.*, 13:45–60, 1981.
- [154] Jean-Pierre Queille. *Le Systeme CESAR: description, specification et analyse des applications reparties*. PhD thesis, Grenoble, 1982.
- [155] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In *ISOP 1982*, International Symposium on Programming, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer-Verlag, 1982.
- [156] Michael Rabin and Dana Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, 1959.

- [157] Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.
- [158] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Bull. Am. Math. Soc.*, 74:1025–1029, 1968.
- [159] Shahid Rahman and Laurent Keiff. *Logic, Thought and Action*, volume 2, chapter On How to Be a Dialogician: A Short Overview on Recent Developments on Dialogues and Games, pages 359–408. Springer-Verlag, 2005.
- [160] Peter J.G. Ramadge and W. Murray Wonham. The control of discrete event systems. *IEEE Transactions on Control Theory*, 77:81–98, 1989.
- [161] David A. Schmidt. Binary relations for abstraction and refinement. Technical Report 2000-3, Kansas State University, 2000.
- [162] Michael J. A. Smith. Compositional Abstraction of PEPA Models for Transient Analysis. In *Computer Performance Engineering*, volume 6342 of *Lecture Notes in Computer Science*, pages 252-267. pringer-Verlag, 2010.
- [163] Michael J. A. Smith. *Stochastic Abstraction of Programs: Towards Performance-Driven Development*. PhD thesis, School of Informatics, University of Edinburgh, 2010.
- [164] Philippe Schnoebelen. The complexity of temporal logic model checking. In *Advances in Modal Logic*, pages 393–436. King’s College Publications, 2002.
- [165] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985.
- [166] Colin Stirling. Local model checking games. In *CONCUR 1995*, 6th International Conference on Concurrency Theory, volume 962 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 1995.
- [167] Colin Stirling. Games and modal mu-calculus. In *TACAS 1996*, 2nd International Workshop on Tools and Algorithms for Construction and Analysis of Systems, volume 1055 of *Lecture Notes in Computer Science*, pages 298–312. Springer-Verlag, 1996.
- [168] Wolfgang Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 133–192. Elsevier and MIT Press, 1990.

- [169] Wolfgang Thomas. On the Ehrenfeucht-Fraïssé game in theoretical computer science. In *TAPSOFT 1993*, International Joint Conference on Theory and Practice of Software Development, volume 668 of *Lecture Notes in Computer Science*, pages 559–568. Springer-Verlag, 1993.
- [170] Wolfgang Thomas. Languages, automata, and logic. Technical Report 9607, Christian-Albrechts-Universität zu Kiel, 1996.
- [171] Alan Turing. Checking a large routine. In *The early British computer conferences*, pages 70–72. MIT Press, 1989.
- [172] Antti Valmari. A stubborn attack on state explosion. *Formal Methods in System Design*, 1(4):297–322, 1992.
- [173] Moshe Vardi. Automatic verification of probabilistic concurrent finite state programs. In *FOCS 1985*, 26th Annual Symposium on Foundations of Computer Science, pages 327–338. IEEE Computer Society, 1985.
- [174] Moshe Vardi. Branching vs. linear time: Final showdown. In *TACAS 2001*, 7th International Conference on Tools and Algorithms for Construction and Analysis of Systems, volume 2031 of *Lecture Notes in Computer Science*, pages 1–22. Springer-Verlag, 2001.
- [175] Moshe Y. Vardi. Probabilistic linear-time model checking: An overview of the automata-theoretic approach. In *ARTS 1999*, 5th International AMAST Workshop on Formal Methods for Real-Time and Probabilistic Systems, volume 1601 of *Lecture Notes in Computer Science*, pages 265–276. Springer-Verlag, 1999.
- [176] Yde Venema. Automata and fixed point logic: A coalgebraic perspective. *Inf. Comput.*, 204(4):637–678, 2006.
- [177] Steven Vickers. *Topology via logic*. Cambridge University Press, 1989.
- [178] Willem Visser and Howard Barringer. Practical CTL* model checking: Should SPIN be extended? *STTT*, 2(4):350–365, 2000.
- [179] Thomas Wilke. Alternating tree automata, parity games, and modal μ -calculus. *Bulletin of the Belgian Mathematical Society*, 8(2):359–391, 2001.
- [180] Ralf Wimmer, Bettina Braitling, and Bernd Becker. Counterexample generation for discrete-time Markov chains using bounded model checking. In *VMCAI 2009*, 10th International Conference on Verification, Model Checking, and Abstract Interpretation, volume

5403 of *Lecture Notes in Computer Science*, pages 366–380. Springer-Verlag, 2009.

- [181] Pierre Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1/2):72–99, 1983.
- [182] Jim Woodcock. GC6 dependable systems evolution. In *Grand Challenges in Computing - Research*. British Computer Society, 2004.
- [183] Jim Woodcock, Cliff Jones, and Peter O’Hearn. GC6: Dependable systems evolution - BCS website.
<http://www.bcs.org/server.php?show=ConWebDoc.4721>.