# Games for Modal and Temporal Logics

*Martin Lange*

Doctor of Philosophy

Laboratory for Foundations of Computer Science
School of Informatics
University of Edinburgh

2002

# Abstract

Every logic comes with several decision problems. One of them is the *model checking* problem: does a given structure satisfy a given formula? Another is the *satisfiability* problem: for a given formula, is there a structure fulfilling it?

For modal and temporal logics; tableaux, automata and games are commonly accepted as helpful techniques that solve these problems. The fact that these logics possess the tree model property makes tableau structures suitable for these tasks. On the other hand, starting with Büchi's work, intimate connections between these logics and automata have been found. A formula can describe an automaton's behaviour, and automata are constructed to accept exactly the word or tree models of a formula.

In recent years the use of games has become more popular. There, an existential and a universal player play on a formula (and a structure) to decide whether the formula is satisfiable, resp. satisfied. The logical problem at hand is then characterised by the question of whether or not the existential player has a winning strategy for the game.

These three methodologies are closely related. For example the non-emptiness test for an alternating automaton is nothing more than a 2-player game, while winning strategies for games are very similar to tableaux.

Game-theoretic characterisations of logical problems give rise to an interactive semantics for the underlying logics. This is particularly useful in the specification and verification of concurrent systems where games can be used to generate counterexamples to failing properties in a very natural way.

We start by defining simple model checking games for Propositional Dynamic Logic, PDL, in Chapter 4. These allow model checking for PDL in linear running time. In fact, they can be obtained from existing model checking games for the alternating free $\mu$-calculus. However, we include them here because of their usefulness in proving correctness of the satisfiability games for PDL later on. Their winning strategies are history-free.

Chapter 5 contains model checking games for branching time logics. Beginning with the Full Branching Time Logic CTL$^*$ we introduce the notion of a *focus game*. Its key idea is to equip players with a tool that highlights a particular formula in

a set of formulas. The winning conditions for these games consider the players' behaviours regarding the change of the focus. This proves to be useful in capturing the regeneration of least and greatest fixed point constructs in CTL$^*$. Deciding the winner of these games can be done using space which is polynomial in the size of the input. Their winning strategies are history-free, too.

We also show that model checking games for CTL$^+$ arise from those for CTL$^*$ by disregarding the focus. This does not affect the polynomial space complexity. These can be further optimised to obtain model checking games for the Computation Tree Logic CTL which coincide with the model checking games for the alternating free $\mu$-calculus applied to formulas translated from CTL into it. This optimisation improves the games' computational complexity, too. As in the PDL case, deciding the winner of such a game can be done in linear running time. The winning strategies remain history-free.

Focus games are also used to give game-based accounts of the satisfiability problem for Linear Time Temporal Logic LTL, CTL and PDL in Chapter 6. They lead to a polynomial space decision procedure for LTL, and exponential time decision procedures for CTL and PDL. Here, winning strategies are only history-free for the existential player. The universal player's strategies depend on a finite part of the history of a play.

In spite of the strong connections between tableaux, automata and games their differences are more than simply a matter of taste. Complete axiomatisations for LTL, CTL and PDL can be extracted from the satisfiability focus games in an elegant way. This is done in Chapter 7 by formulating the game rules, the winning conditions and the winning strategies in terms of an axiom system. Completeness of this system then follows from the fact that the existential player wins the game on a consistent formula, i.e. it is satisfiable.

We also introduce satisfiability games for CTL$^*$ based on the focus approach. They lead to a double exponential time decision procedure. As in the LTL, CTL and PDL case, only the existential player has history-free winning strategies. Since these strategies witness satisfiability of a formula and stay in close relation to its syntactical structure, it might be possible to derive a complete axiomatisation for CTL$^*$ from these games as well.

Finally, Chapter 9 deals with Fixed Point Logic with Chop, FLC. It extends modal $\mu$-calculus with a sequential composition operator. Satisfiability for FLC is undecidable but its model checking problem remains decidable. In fact it is hard for polynomial space.

We give two different game-based solutions to the model checking problem for FLC. Deciding the winner for both types of games meets this polynomial space lower bound for formulas with fixed alternation (and sequential) depth. In the general case the winner can be determined using exponential time, resp. exponential space. The former result holds for games that give rise to global model checking whereas the latter describes the complexity of local FLC model checking. FLC is interesting for verification purposes since it – unlike all the other logics discussed here – can describe properties which are non-regular.

The thesis concludes with remarks and comments on further research in the area of games for modal and temporal logics.

Acknowledgements

First of all, I wish to thank my supervisor Prof. Colin Stirling for the support and guidance I got from him. His supervision was nothing less than excellent, mainly because he gave me the freedom to choose the topic that I wanted to work on and broadened my horizon by getting me interested in problems related to it. He was always able to supply me with new and good ideas whenever I got stuck on a problem that seemed unsolvable for me at that moment. It only took the first few months of work with him in Edinburgh to make me realise that I need not worry about producing a PhD thesis in reasonable time.

I also would like to thank Prof. Javier Esparza and Prof. Mogens Nielsen for agreeing to examine this thesis. I hope they do not regret it once they have read through all of this.

Further thanks go to LFCS which provided a nice and good research environment for my time at Edinburgh. Although I began to like Edinburgh a lot after surviving a dark and unpleasant winter I am very grateful to LFCS for letting me visit BRICS in Århus, Denmark. Again, they provided a nice and good research environment during my three months stay as a Marie Curie Fellow there as well. I would like to thank Prof. Mogens Nielsen again, this time for the hospitality I received there. The same holds for Uffe Engberg. Claus Brabrand did his best to provide me with a social life there, too. Special thanks go to Jesper Henriksen for initiating my stay, for making me feel welcome there, for not wasting my time on difficult and uninteresting problems and, finally, for giving me special thanks credit in his thesis.

I also wish to thank various people at LFCS who became more than just colleagues. I had a great time with my office mate Marco Kick, with Alex Simpson, Tom Chothia, Daniele Turi, Martin Grohe and Markus Frick.

Prof. Colin Stirling and Prof. Martin Hofmann deserve to be thanked for agreeing on a deal that enabled me to take up a position in Munich before finishing this thesis.

Then, I want to thank my parents who have always supported me in every way. Without their help I would not have had the chance to go to Edinburgh, to study for a PhD or to study at all. Equally, I wish to thank my non-academic friends for not losing touch with me after I went to Scotland.

Finally, I wish to express my utmost gratitude to my wife Becky who has been very understanding and helpful whenever composing this thesis required it. I especially thank her for her long-distance support during my time in Århus and, while I am writing this, my time in Munich. She is without a doubt the best side-effect that my PhD studies in Edinburgh have produced.

# Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification.

Chapter 5 has been published in [LS02b], preliminary versions appeared as [LS00] and [Lan00]. Sections 6.1, 6.2, 7.1 and 7.3 have been published in [LS01]. A slightly different version of Section 9.1 can be found in [LS02a]. Section 9.2 has been published in [Lan02b].

(*Martin Lange*)

*To those who do not dedicate*
*their thesis to themselves.*

# Table of Contents

# List of Figures

# Chapter 1

# Introduction

*What do you need that for, Dude?*

—

THEODORE DONALD KARABOTSOS

## Formal Verification

Computers and electronic devices play an important role in our world today. People constantly rely on the fact that they work correctly. One wants to be sure that a digital alarm clock goes off exactly at the time it is set for. A phone call should be directed only to the number that was dialed. Failure of these features of course is not life threatening. But there are examples where computers perform tasks that simply must not go wrong.

Take an airplane's control for example. Many aspects of steering an airplane are automated, especially those that take effect in a dangerous situation when a machine's precision or speed are preferred over human action. If the actions taken by the computer are wrong it may leave the pilot in a situation without control over the aircraft which can have hazardous effects.

It is therefore necessary to *know* that a computer *works*. It is not within our powers to ensure that a computer physically works. This is left to engineers and the hope that the computer at hand is not hit by a bomb.

Instead, we deal with the question of whether the *specification* of an electronic device or a piece of software functions correctly. Several mechanisms that abstract the behaviour of a computer from the physical device have been developed in computer science. These specification languages can be seen as programming languages whose semantics is a mathematical structure which denotes such a behaviour.

Feasibility is not the only reason for dealing with specifications rather than real applications. Developing costs for any products need to be kept low. Thus, it is desirable to create correct specifications *before* they are turned into a real product. This avoids producing several versions most of which will be thrown away because of faults in their specifications.

Next there is the question of determining whether a specification does what it is supposed to do. It is too vague to say that it should function correctly. In the case of the alarm clock this might be obvious. For the telephone network it is already less clear. If the person whose number is dialed redirects calls then the property mentioned above is not fulfilled. However, this should not be regarded as a failure of the underlying system.

In the example of the airplane it is entirely unclear what it should mean for the control software to function correctly. Therefore, formalisms are needed that allow us to specify correctness properties. Mathematics, as a precise science that does not leave space for interpretations, provides a framework for this: logics.

Logics formalise statements that are made about abstract mathematical structures. This can be used for the formal verification of properties of real systems if their specifications are given as such abstractions. Needed for this are automatic procedures that check for example whether a given structure has a certain property which is given by a logical formula. Such algorithms are called *model checkers*. They are used in *verification tools* like SPIN, [Hol97], SMV, [CGL93], the EDINBURGH CONCURRENCY WORKBENCH, [Mol92], HYTECH, [HHWT97], TRUTH, [LLNT99], and many more.

These programs typically allow a system to be modelled in a certain specification language and automatically generate the mathematical structure from it. The latter is normally a transition system, i.e. a labelled directed graph with nodes being interpreted as *states* that the underlying system can be in and edges as transitions between states in time. This temporal aspect is a natural interpretation of the behaviour of a computer program. Note that the operational semantics of a program is nothing more than such a transition system. For a program that is modelled with such a transition system the states can denote different evaluations for the set of variables that are used in the program. Transitions between these states are then given by the program's control structures like variable assignments.

Consequently, these verification tools typically allow properties to be formalised in a logic which captures temporal aspects of transition systems and to automatically check whether it satisfies the property. Such logics are, not surprisingly, temporal logics like Pnueli's Linear Time Temporal Logic LTL, [Pnu77], Emerson and Halpern's Computation Tree Logic CTL, [EH85], and the Full Branching Time Logic CTL$^*$ by Emerson, Halpern and Sistla, [EH86, ES84]. Typical statements that can be made in these logics concern the question of whether or not something holds on all reachable states or along a path through the transition system.

Modal logics which have their origin in philosophy and which are a superclass of temporal logics are suitable for this task as well. This is because they are interpreted over structures consisting of different *worlds* where something can be true in one world but false in another. Clearly, transition systems as abstractions of programs are examples of such structures since different states need not have the same properties. We will only deal with those modal logics that have gained interest in computer science, namely Fischer and Ladner's Propositional Dynamic Logic PDL, [FL79], Kozen's modal $\mu$-calculus $\mathcal{L}_\mu$, [Koz83], and Müller-Olm's Fixed Point Logic with Chop FLC, [MO99].

Logics can also be used as a specification formalism. Going back to the airplane example, a system may be considered correct if it satisfies several properties. These may interact, for example if a sensor's signal should cause the plane to automatically descend while the autopilot tries to keep it on a certain level.

Suppose each aspect of correctness is given by a logical formula, i.e. the one stating correct behaviour of a single part. Then global correctness is given by the conjunction of all these formulas. It is important to have automatic procedures that test satisfiability of such formulas since some of the properties may exclude each other which causes unsatisfiability of the conjunction. In this case the specification would be considered incorrect.

It is desirable to have verification tools that do more than simply check whether or not a specification satisfies a formula or a logical specification is satisfiable. If the answer is yes then of course the specification and verification task is completed. However, if the answer is no, i.e. the system at hand is incorrect with respect to some property, then the error needs to be repaired. Thus, it is helpful to have verification tools that provide guidance in finding the reasons for incorrectness, i.e. that show the user *where* or *why* a certain property fails.

*Games* provide a natural framework for this feature. This thesis contains two types of games: model checking games and satisfiability checking games. Both are played by two players on a certain game board. One of them has the task to show that a specification is correct with respect to a certain property, resp. that a logical specification does not contain a contradiction. The other player is given the opposite task.

The outcome of a single play against each other provides little information about the correctness of a specification. It carries even less information than a test run. Testing cannot show the absence of errors, at least it can reveal their presence. Generally, a single play cannot do either of these.

However, we define these games in a way such that they characterise the model checking or satisfiability checking problem for a modal or temporal logic in terms of strategies. Thus, a transition system has the property described by a formula if and only if the player whose task it is to show this has a *winning strategy* for the corresponding game. In the satisfiability checking game she has a winning strategy if and only if the underlying formula is satisfiable, i.e. does not contain a contradiction.

Model checking or satisfiability checking is then equivalent to finding a winning strategy for this player. In most cases, certainly for the logics we introduce here and for the class of finite transition systems, this is decidable. Hence, it can be automated.

So far, the game-based method does not reveal any advantage over other methods like tableaux or automata for example. In fact, in computer science automata-theoretic methods are widely believed to be the most efficient for verification purposes and, hence, best.

However, a game-based model checker or satisfiability checking algorithm needs to compute a winning strategy for one of the players in order to determine whether a player has one. Suppose a transition system fails to have a desired property. The corresponding game-based model checker computes a strategy for the player whose task it was to show this. This strategy then witnesses the failure of the property and can be used to prove this failure to the user of a verification tool.

This can be done by letting them play an *interactive play* against the tool which takes its choices according to the winning strategy it has computed. By definition, regardless of the user's choices the tool will win the resulting play. Typically the play follows a path of a transition system and the syntactical structure of the formula representing the desired property. Thus, each play that is won by the tool reveals at which moment in the underlying system's temporal behaviour which part of the property fails.

With game-based satisfiability checking the situation is similar. Here, a play reveals which parts of the formula exactly cause the unsatisfiability, i.e. which parts exclude each other.

## Outline of this Thesis

The goal of this thesis is to give game-based characterisations of the model checking and satisfiability checking problem for the modal and temporal logics mentioned above. It is organised in the following way.

Chapter 2 contains the definition of transition systems and the modal and temporal logics that are studied here. It also recalls basic results about fixed points which are necessary to understand the games of the following chapters since all the logics we deal with feature constructs whose semantics is given as the solution to a certain fixed point equation. 2-player games are formally introduced as well.

Chapter 3 surveys other methods that have been used to tackle the model checking and satisfiability checking problem for modal and temporal logics. Among them, *tableaux* and *automata* have been established as methodologies, i.e. classes of methods, that are useful for these purposes. For almost every logic mentioned here there is a tableau procedure and an automata-theoretic characterisation for the model checking and the satisfiability checking problem. Other techniques like graph-theoretic algorithms or resolution methods only seem to be useful or applicable in special cases. We also sketch areas in computer science that have benefitted from the use of games. This thesis proposes the idea that games are another useful methodology for the logical problems at hand.

The technical part of this thesis starts with Chapter 4 which contains model checking games for PDL. This characterisation in terms of games is straight-forward and not very complicated. In fact, it can easily be derived from Stirling's model checking games for the alternation-free $\mu$-calculus $\mathcal{L}_\mu^0$, [Sti95]. However, there are three reasons for including them here. First, for the sake of completeness since, to the best of our knowledge, they have not been published anywhere else. Second, because of their simplicity they prepare the reader for the following chapters. The third and most important reason is the fact that they serve as a helpful tool for proving correctness of the PDL satisfiability games in Section 6.3 later on.

Chapter 5 contains model checking games for branching time logics. Beginning with CTL$^*$ the notion of a *focus* game is introduced. It is simplified to obtain model checking games for CTL$^*$'s fragments CTL$^+$ and CTL. As with PDL, the CTL model checking games are straight-forward and derivable from the $\mathcal{L}_\mu^0$ games. However, the fact that a simplification of the CTL$^*$ games leads to such natural games can be seen as an argument in favour of the focus game idea which makes them a natural approach to the CTL$^*$ model checking problem.

Focus games are shown to be useful for satisfiability checking in Chapter 6 that contains games for LTL, CTL and PDL. Chapter 7 contains a side-effect of these games. We show how to extract axiom systems from the games that are easily proved to be complete. This chapter can be seen as an argument for the usefulness of satisfiability focus games or as an application of them.

Focus games are used again in Chapter 8 to obtain a game-based characterisation of CTL$^*$'s satisfiability problem. It is presented in a different chapter separated from the other satisfiability games because the games are more complex and, as a consequence, a complete axiomatisation is not easily derived.

Finally, Chapter 9 is concerned with the model checking problem for FLC. Two different game-based approaches to this problem are presented: a global and a local one. These games are not focus games. The local approach is a generalisation of Stirling's $\mathcal{L}_\mu$ model checking games just as FLC is an extension of $\mathcal{L}_\mu$.

Apart from the definitions of the games, all chapters contain their respective correctness proofs, examples and analyses of the complexity of deciding which player has a winning strategy for a given game.

The thesis concludes with remarks on further research in the area of games for modal and temporal logics. In particular, extensions of the logics dealt with here are mentioned for which it might be interesting to have game-theoretic characterisations of their model checking or satisfiability checking problem as well.

# Chapter 2

# Preliminaries

*Mathematics is the art of giving*
*the same name to different things.*

—

HENRI POINCARÉ

## 2.1 Mathematical Logics

A *relational structure* is a tuple $K = (U, R_1, \ldots, R_n)$ where $U$ is a set called the *universe* of $K$ and $R_1, \ldots, R_n$ are relation symbols of arities $a_1, \ldots, a_n$. This means that for every $i = 1, \ldots, n$ we have

$$R_i \subseteq \underbrace{U \times \ldots \times U}_{a_i \text{ times}}$$

A *logic* $\mathcal{L}$ is a set of formulas. These are interpreted over a class of structures $\mathfrak{K}$ by the $\models$ relation. Let $\varphi \in \mathcal{L}$ be a formula with free *first-order variables* $x_1, \ldots, x_n$, i.e. variables for elements of a relational structure's universe. For every structure $K \in \mathfrak{K}$

and every $n$-tuple $k_1, \dots, k_n$ of elements of $K$,

$$K, k_1, \dots, k_n \models \varphi(x_1, \dots, x_n)$$

is written to denote that the structure $K$ has the property described by $\varphi$ where each variable $x_i$ is interpreted by $k_i$, $i \in \{1, \dots, n\}$. In the second-order case, variables ranging over relations are allowed, too.

We will only consider a few special logics, namely *modal* and *temporal logics*. They are also interpreted over certain structures only, called *labelled transition systems*, [Plo81]. These will be defined in Section 2.3.

Most modal and temporal logics can be translated into First-Order or Second-Order Predicate Logic. The resulting formula is not closed but has one free variable. An element $s$ of a structure $K$ has a modal or temporal property $\varphi$ iff $K$ satisfies the translated property $\tilde{\varphi}(x)$ where the free variable $x$ is interpreted by $s$.

$$K, s \models \tilde{\varphi}(x) \tag{2.1}$$

Thus, not only a structure $K$ but $K$ together with an element $s$ of its universe satisfies a modal or temporal formula $\varphi$,

$$K, s \models \varphi$$

Note that the modal or temporal formula $\varphi$ does not have any free variables in the sense of (2.1). Often, we will consider the underlying $K$ to be fixed and omit it, $s \models \varphi$.

The *model checking problem* for a modal or temporal logic $\mathfrak{L}$ and a class of structures $\mathfrak{K}$ is: given $K \in \mathfrak{K}$, an element $s$ of $K$ and $\varphi \in \mathfrak{L}$, does $K, s \models \varphi$ hold?

The *satisfiability checking problem* for a modal or temporal logic $\mathfrak{L}$ and a class of structures $\mathfrak{K}$ is: given a $\varphi \in \mathfrak{L}$, is there a $K \in \mathfrak{K}$ and an $s \in K$, s.t. $K, s \models \varphi$?

The *syntax* of a logic is usually given as a context-free grammar. Hence, formulas are words over a certain alphabet. This enables the easy substitution of formulas into formulas. With $\varphi[\psi/\chi]$ we denote the formula that arises from $\varphi$ by replacing every occurrence of $\chi$ in $\varphi$'s syntax tree by $\psi$.

All the logics defined later subsume propositional boolean logic. Their syntactical definitions will not include negation since games usually require negation to be eliminated. But we will show that negation is implicitly present in most cases. We will also use constructs like $\rightarrow$ from propositional boolean logic appealing to its definition using $\vee$ and negation closure.

The *semantics* of a logic will be given in one of two possible ways. Either directly, i.e. in the style $K, x \models \varphi$ describing when a given structure $K$ with an element $x$ satisfies a given $\varphi$. Or indirectly in the style $[\![\varphi]\!]$ which describes the set of all $x$ of a structure $K$ that satisfy $\varphi$. The satisfaction relation is then easily derived as

$$s \models \varphi \quad \text{iff} \quad s \in [\![\varphi]\!]$$

In both cases the context-freeness of the logic's syntax allows the semantics to be defined inductively.

A fragment of a logic is simply a subset of all its formulas. In many cases this will be a syntactical fragment, i.e. the question of whether or not $\varphi$ belongs to this fragment only depends on the syntactical structure of $\varphi$. These fragments usually impose restrictions on the occurrence of certain constructs of the logic because they permit more efficient decision procedures than the general case.

Each logic also has important semantical fragments. These will of course depend on the class of structures $\mathfrak{K}$ the logic is interpreted over. One such fragment is the set of all *satisfiable* formulas, i.e. those $\varphi$ for which there is a $K \in \mathfrak{K}$ and an $s \in K$ s.t. $K, s \models \varphi$. Another important fragment considers the same question but universally quantified: the set of all formulas that are satisfied by every $K \in \mathfrak{K}$ and every $s \in K$. These formulas are called *validities*. To indicate that $\varphi$ is valid we write $\models \varphi$.

Two formulas $\varphi, \psi$ of $\mathcal{L}$ are *equivalent* over $\mathfrak{K}$, written $\varphi \equiv \psi$, iff $[\![\varphi]\!] = [\![\psi]\!]$ for all $K \in \mathfrak{K}$, i.e. they are satisfied by the same structures and elements. If the semantics is given directly then

$$\varphi \equiv \psi \quad \text{iff} \quad \text{for all } K \in \mathfrak{K} \text{ and } s \in K: \; K, s \models \varphi \text{ iff } K, s \models \psi$$

In other words, $\varphi$ and $\psi$ essentially describe the same property. The semantics of a logic should always be defined such that $\equiv$ is a congruence. This allow a subformula $\psi$ of $\varphi$ for example to be substituted by an equivalent formula without changing the meaning of $\varphi$.

**Definition 1** We say that a logic $\mathcal{L}$ is *negation closed* if for every $\varphi \in \mathcal{L}$ there is a $\overline{\varphi} \in \mathcal{L}$ s.t. for every $K \in \mathfrak{K}$ and every $s \in K$:

$$K, s \models \varphi \quad \text{iff} \quad K, s \not\models \overline{\varphi}$$

Note that a formula $\varphi$ is satisfiable iff its negation $\overline{\varphi}$ is not valid.

A logic itself is a mathematical construct and, hence, has or lacks certain properties. Important properties for modal and temporal logics are

- the *tree model property*: if $\varphi$ is satisfiable then it has a model which is a tree.
- the *finite model property*: if $\varphi$ is satisfiable then it has a model of finite size.
- the *small model property*: there is a function $f : \mathbb{N} \to \mathbb{N}$, s.t. if $\varphi$ is satisfiable then it has a model of size $f(|\varphi|)$, where $|\varphi|$ denotes the syntactical length of $\varphi$.

Note that, if a logic has the tree model property and the finite model property, it does not necessarily mean that every satisfiable formula is satisfied by a finite tree.

Another important aspect of a logic is its *expressive power*. $\mathcal{L}$ subsumes $\mathcal{L}'$ in expressive power over $\mathfrak{K}$ if for every $\varphi \in \mathcal{L}'$ there is a $\psi \in \mathcal{L}$ s.t. $\varphi \equiv \psi$ over $\mathfrak{K}$.

One of the most important modal logics is the *modal μ-calculus* $\mathcal{L}_\mu$, defined in [Koz83]. Its importance is based on the fact that it subsumes semantically most other propositional modal and temporal logics. In fact, it does so for all logics defined in Sections 2.4 and 2.5 apart from FLC which is itself an extension of $\mathcal{L}_\mu$. The relations between all the logics used here and $\mathcal{L}_\mu$ are depicted in Figure 3.4 at the end of Chapter 3.

[EFT94] contains a good introduction to the theory of mathematical logics. For an overview of temporal and modal logics in particular consider [Eme90], [Sti92], [Sti96b] and [BS01].

## 2.2 Fixed Points

It is well known that adding *quantifiers* to a logic usually increases its expressive power. The degree of this increase is of course dependant on the kind of quantification.

First-order quantifiers that speak about the existence or non-existence of elements of the underlying domain are weaker than second-order quantifiers that speak about relations between elements.

The increased expressive power goes hand in hand with an increase in the complexity of decision problems associated with these logics and might even result in these problems becoming undecidable. Therefore, compromises have been sought and found which allow *restricted quantification*. One example is *guarded first-order logic*, [AvBN98], which features existential and universal first-order quantifiers over certain elements only.

Another way of restricting the power of general quantification is by using *fixed points*. Mathematically, a fixed point of a function $f$ satisfies the equation

$$f(X) = X$$

[Tar55] has shown that this concept is particularly useful if the function $f$ is monotone and applied to members of a complete lattice with bottom element $\bot$ and top element $\top$. In this case there are two distinguished fixed points with nice algorithmic properties.

**Definition 2** Let $(M, \leq)$ be a set which is partially ordered by $\leq$ s.t.

1. for all $x \in M$: $x \leq x$ (reflexivity)
2. for all $x,y,z \in M$: if $x \leq y$ and $y \leq z$ then $x \leq z$ (transitivity)
3. for all $x,y \in M$: if $x \leq y$ and $y \leq x$ then $x = y$ (anti–symmetry)

The element $z$ is a *maximum* of $x$ and $y$ if $x \leq z$ and $y \leq z$. If $z \leq x$ and $z \leq y$ then $z$ is a *minimum* of $x$ and $y$. The *supremum* is the least maximum of two elements and is denoted $x \sqcup y$ while the greatest minimum $x \sqcap y$ is called *infimum*.

A partially ordered set $(M, \leq)$ is called a *lattice* if $x \sqcup y$ and $x \sqcap y$ exist in $M$ for all $x,y \in M$. It is called *complete*, if $\bigsqcup X$ and $\bigsqcap X$ exist for all $X \subseteq M$. In this case there are two distinguished elements $\top := \bigsqcap \emptyset$ and $\bot := \bigsqcup \emptyset$ s.t. for all $x \in M$: $x \leq \top$ and $\bot \leq x$.

The *height* of a lattice $(M, \leq)$ is the maximal number of elements of $M$ in a chain

$$x_1 \;\leq\; x_2 \;\leq\; \ldots \;\leq\; x_{n-1} \;\leq\; x_n$$

Note that, if $M$ is not finite, it is possible to have such chains whose lengths can only be measured using ordinals, $\mathbb{O}$rd, beyond the natural numbers.

A function $f : M \to M$ is called *monotone* iff

$$\text{for all } x, y \in M : \quad x \leq y \quad \text{implies} \quad f(x) \leq f(y)$$

$x$ is a *pre-fixed point* of $f$ iff $f(x) \leq x$ and a *post-fixed point* of $f$ iff $x \leq f(x)$.

**Theorem 3** ***(Knaster–Tarski)*** [Tar55] *Let $(M, \leq)$ be a complete lattice, and $f : M \to M$ a monotone function. The least fixed point of $f$, denoted $\mu f$, exists uniquely and is the infimum of all pre-fixed points.*

$$\mu f \;:=\; \bigsqcap \{\, x \in M \mid f(x) \leq x \,\}$$

*Dually, the greatest fixed point is the supremum of all post-fixed points.*

$$\nu f \;:=\; \bigsqcup \{\, x \in M \mid x \leq f(x) \,\}$$

For a proof see [Win93] or [Sti01] for example. However, there is a more efficient way to evaluate fixed points other than to calculate the infimum of all pre-fixed points for example.

Suppose $f$ is monotone. Then, $f$ can be applied iteratively starting with $\bot$ to obtain a sequence $\bot, f(\bot), f(f(\bot)), \ldots$ of elements of $M$. By monotonicity

$$\bot \;\leq\; f(\bot) \;\leq\; f(f(\bot)) \;\leq\; \ldots \;\leq\; f^i(\bot) \;\leq\; \ldots \tag{2.2}$$

It is easy to show that

$$f^i(\bot) = f^{i+1}(\bot) \;\text{ implies }\; f^i(\bot) = f^j(\bot) \text{ for all } j \geq i$$

Thus, if the underlying lattice has finite height $h \in \mathbb{N}$ the sequence will eventually become stationary with the value $f^h(\bot)$.

Dually, one obtains a monotonically decreasing sequence of elements of the lattice if this iteration is started with $\top$.

$$\top \;\geq\; f(\top) \;\geq\; f(f(\top)) \;\geq\; \dots \;\geq\; f^i(\top) \;\geq\; \dots$$

Again, the sequence becomes stationary with $f^h(\top)$ or even earlier.

For general lattices with heights given by an ordinal $\alpha$ we define *approximants* of $f$'s least fixed point for every ordinal $\beta \leq \alpha$.

$$f^0(\bot) \;:=\; \bot \;, \quad f^{\beta+1}(\bot) \;:=\; f(f^\beta(\bot)) \;, \quad f^\lambda(\bot) \;:=\; \bigsqcup_{\beta<\lambda} f^\beta(\bot)$$

with $\beta, \lambda \in \mathbb{O}\mathrm{rd}$ and $\lambda$ being a limit ordinal. Dually, approximants of the greatest fixed point of $f$ are given by

$$f^0(\top) \;:=\; \top \;, \quad f^{\beta+1}(\top) \;:=\; f(f^\beta(\top)) \;, \quad f^\lambda(\top) \;:=\; \bigsqcap_{\beta<\lambda} f^\beta(\top)$$

**Lemma 4** *Let $(M, \leq)$ be a complete lattice with height $\alpha \in \mathbb{O}\mathrm{rd}$, and $f : M \to M$ a monotone function. Then*

$$\mu f = f^\alpha(\bot) \qquad \textit{and} \qquad \nu f = f^\alpha(\top)$$

PROOF $f^0(\bot) = \bot \leq \mu f$ by the definition of $\bot$. Then $f^1(\bot) = f(\bot) \leq f(\mu f) \leq \mu f$ by monotonicity and the fact that $\mu f$ is a pre-fixed point of $f$. Iterating this yields $f^n(\bot) \leq \mu f$ for all $n \in \mathbb{N}$. The claim holds for ordinals in general by transfinite induction. Suppose $f^\beta(\bot) \leq \mu f$ for all $\beta < \lambda$, i.e. $\mu f$ is a maximum for all $f^\beta(\bot)$. Then $f^\lambda(\bot) \leq \mu f$ because $f^\lambda(\bot)$ is the least maximum of them all. The case for $\nu f$ is dual. ∎

This means that in case the height of the underlying lattice is finite, least and greatest fixed points of $f$ can be found iteratively. This iterative nature has led to the idea of using fixed point operators as quantifiers. All the logics introduced in the following sections feature fixed point constructs. Most of them do this in an implicit way: they have constructs which can be regarded as solutions to an equation in the above sense. One of the logics allows explicit fixed point quantification, i.e. formulas with *free variables* are interpreted as functions on elements of a certain lattice while fixed point operators quantify exactly over those elements that are fixed points of these functions.

For further reading on the use of fixed points in mathematical logics consult [EF95]. [GW99] shows properties of the guarded fragment with fixed points which can be seen as a generalization of modal and temporal logics with extremal fixed points. Finally, [BS01] provides an introduction into fixed points for modal logics.

## 2.3 Labelled Transition Systems

**Definition 5** Let $\mathcal{P} = \{\mathtt{tt}, \mathtt{ff}, q, \overline{q}, \ldots\}$ be a set of propositional constants, i.e. unary relation symbols, that is closed under complementation: for every $q \in \mathcal{P}$ there is a $\overline{q} \in \mathcal{P}$. Moreover, $\overline{\overline{q}} = q$ and $\overline{\mathtt{tt}} = \mathtt{ff}$. Let $\mathcal{A} = \{a, b, \ldots\}$ be a set of action names. A *labelled transition system*, LTS, is a triple

$$\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$$

where

- $\mathcal{S} = \{s, t, \ldots\}$ is a set of states,
- $\xrightarrow{a}$ for each $a \in \mathcal{A}$ is a binary relation on states, and
- $L : \mathcal{S} \to 2^{\mathcal{P}}$ labels the states in a maximally consistent manner. This means for every $s \in \mathcal{S}$ and every $q \in \mathcal{P}$ either $q \in L(s)$ or $\overline{q} \in L(s)$. Furthermore, $\mathtt{tt} \in L(s)$ for every $s \in \mathcal{S}$.

If we mention a labelling of a certain state explicitly we will often omit $\mathtt{tt}$ since it is included by default.

We will use infix notation $s \xrightarrow{a} t$ instead of $(s,t) \in \xrightarrow{a}$. To indicate that there is no $t$ s.t. $s \xrightarrow{a} t$ we will write $s \stackrel{a}{\not\to}$, and $s \not\to$ if $s$ has no successor at all.

If the set of action names is a singleton, $\mathcal{A} = \{a\}$, we omit the explicit mentioning of the action and write $s \to t$ instead of $s \xrightarrow{a} t$. In this case a transition system is denoted $\mathcal{T} = (\mathcal{S}, \to, L)$.

A *path* of a transition system $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ is a maximal sequence of states $\pi = s_0 s_1 \ldots$ s.t. for all $i$ there is an $a_i \in \mathcal{A}$ with $s_i \xrightarrow{a_i} s_{i+1}$ if $s_i$ is not the last state of this sequence. Maximality means the path cannot be prolonged. This is the case if it is infinite or a finite sequence $s_0 \dots s_n$ and $s_n \not\rightarrow$.

Let $\pi^k$ denote the suffix of $\pi$ beginning with the $k$-th state, i.e. $\pi^k = s_k s_{k+1} \dots$. The $k$-th state $s_k$ of $\pi$ is denoted by $\pi^{(k)}$.

A transition system $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ is *total* if for every $s \in \mathcal{S}$ there is at least one $t \in \mathcal{S}$ s.t. $s \rightarrow t$. Note that paths of total transition systems are necessarily infinite.

**Definition 6** Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with $s_0 \in \mathcal{S}$. The *unravelling* of $\mathcal{T}$ with respect to $s_0$ is an LTS $\mathcal{R}_{s_0}(\mathcal{T}) = (\mathcal{S}', \{\xrightarrow{a}{}' \mid a \in \mathcal{A}\}, L')$ with state set

$$\mathcal{S}' \;:=\; \{\, s_0 \dots s_n \mid \text{ for all } i < n : s_i \xrightarrow{a} s_{i+1} \text{ for some } a \in \mathcal{A} \,\}$$

Transitions in $\mathcal{R}_{s_0}(\mathcal{T})$ are defined as

$$s_0 \dots s_n \xrightarrow{a}{}' s_0 \dots s_n s_{n+1} \quad \text{iff} \quad s_n \xrightarrow{a} s_{n+1}$$

Finally, the labelling of the states is given by

$$L'(s_0 \dots s_n) \;:=\; L(s_n)$$

## Symbolic Representations

It is useful to distinguish finite and infinite transition systems. The first reason for this is decidability. The model checking problems for the logics introduced in the next section are undecidable for arbitrary infinite transition systems because they can express properties like reachability of a certain state for example. However, for finite transition systems they are decidable.

The second reason for this distinction is the question of representing a transition system. In the finite case it can be written down as a directed graph with labellings. Arbitrary infinite transition systems obviously cannot be represented in this way. However, there are classes of infinite transition systems that have finite representations. Depending on the expressive power of a logic regarded over these classes the model checking problem might still be decidable.

Representations of infinite transition systems can be process algebraic ones like Basic Process Algebra BPA, Basic Parallel Processes BPP, Pushdown Automata PDA, etc. For an overview of these classes and their decidability results see [HM96] for example. [May00] is about Process Rewrite Systems which subsume all these process algebras.

Other examples of process algebras are the Calculus of Communicating Systems CCS, [Mil80], Communicating Sequential Processes CSP, [Hoa78a, Hoa78b], the $\pi$-calculus, [MPW92] and Petri-Nets, [Pet62, Rei85]. However, in general all of these give rise to arbitrary transition systems and not finite ones only. But the advantage of using such process algebras is the fact that they allow model checking algorithms to be local, see Section 2.8 for explanations.

The idea of using process algebras to represent infinite transition systems is also beneficial for finite ones. In verification tasks the underlying transition systems can be very large and a process algebraic specification can be a much more succinct representation of a transition system than the adjacency matrix of a graph for example. Moreover, if transition systems specify a hardware circuit or a software module then it is often easier to find a process algebraic term that abstracts its behaviour.

The state-of-the-art formalism to represent finite transition systems are *Ordered Binary Decision Diagrams*, [Bry86]. They are compact acyclic graph representations of boolean functions. The reason why they can be used to encode transition system is the fact that an LTS $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ is nothing more than a collection of binary relations $\{\xrightarrow{a} \mid a \in \mathcal{A}\}$ each of which can be stored as an OBDD.

OBDDs are particularly useful for model checking modal and temporal logics since it is relatively easy to evaluate boolean operators and to calculate fixed points on OBDDs, [McM93]. Using OBBDs for model checking resulted in a major breakthrough concerning the size of transition systems up to which model checking is practically feasible. In fact, these *symbolic* techniques enable model checking for transition systems with more than 100 boolean variables, [BCM$^+$92].

We will not be concerned with the question of how a given transition system is represented. Generally, we will assume it to be present and represented in some way. If it is known to be finite we will assume it can be represented in some process-algebraic or other way that allows a construction to proceed state-by-state.

For finite transition systems we will measure the complexity of deciding the winner of a model checking game as a function of the formula size and the number of states a transition system has.

## Equivalences

There are a number of ways in which two states $s$ and $t$ of a transition system can be regarded as equivalent. One criterion is graph isomorphism of the subgraphs of reachable states from $s$ and $t$. This is far too strong if one uses transition systems to describe program behaviour. A much weaker version considers $s$ and $t$ to be equivalent if the transition system regarded as a Büchi-automaton accepts the same language regardless of whether $s$ or $t$ is the starting state. In order to do so, every state of such an automaton is considered to be final. Hence, every run of the automaton is accepting.

A useful equivalence between graph isomorphism and language equivalence is bisimilarity, [Mil89, vB96]. We mention this explicitly because Section 2.5 introduces the logic FLC and proves that, like all other logics appearing in the next two sections, it does not distinguish bisimilar states of a transition system.

**Definition 7** Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$. A *bisimulation* is a symmetric binary relation $R \subseteq \mathcal{S} \times \mathcal{S}$ fulfilling the following.

- If $(s,t) \in R$ and $s \xrightarrow{a} s'$ for some $a \in \mathcal{A}$ then there is a $t' \in \mathcal{S}$, s.t. $(s',t') \in R$.
- If $(s,t) \in R$ and $q \in L(s)$ then $q \in L(t)$.

$s$ and $t$ are called *bisimilar*, $s \sim t$, if there is a bisimulation $R$ s.t. $(s,t) \in R$.

A *simulation* is a relation with the same requirements as above but which is not necessarily symmetric. $t$ simulates $s$ iff there is a simulation relating $s$ and $t$.

We say that a logic $L$ respects bisimulation if for all $\varphi \in L$, all transition systems $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ and all $s,t \in \mathcal{S}$: $s \sim t$ implies $s \models \varphi$ iff $t \models \varphi$.

## 2.4 Temporal Logics

The temporal logics defined here do not make use of different action labels, i.e. they are interpreted over transition systems of the form $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$. Furthermore, we assume these transition systems to be total. This is a common approach but also avoids a lot of technical detail.

### Linear Time Temporal Logic

Temporal logics over linear structures have been studied for a long time. The most important result regarding these logics is from [Kam68] where it is shown that a temporal logic with an *until* operator and its dual for the past, *since*, is equi-expressive to first-order formulas with one free variable interpreted over linear orders. Because of this, *Linear Time Temporal Logic* LTL is believed to be a natural specification formalism for temporal properties. [Pnu77] introduced LTL to computer science and showed that it can be used for program verification purposes. For a detailed introduction to LTL see [MP92]. Here we regard LTL with future operators only. Its *syntax* is given by the following grammar.

$$\varphi \quad ::= \quad q \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathtt{X}\varphi \mid \varphi \mathtt{U} \varphi \mid \varphi \mathtt{R} \varphi$$

where $q$ ranges over $\mathcal{P}$. $\mathtt{X}$ is the *next* operator, $\mathtt{U}$ the *until*, and $\mathtt{R}$ its dual, called *release*. The traditional *eventually* and *generally* operators are abbreviated as

$$\mathtt{F}\varphi := \mathtt{tt}\mathtt{U}\varphi \qquad \text{and} \qquad \mathtt{G}\varphi := \mathtt{ff}\mathtt{R}\varphi$$

LTL is interpreted over paths $\pi = s_0 s_1 \ldots$ of a total LTS. We usually assume an LTS to be fixed and write $\pi \models \varphi$ instead of $\mathcal{T}, \pi \models \varphi$. The *semantics* of an LTL formula is inductively defined as

$$\begin{array}{lll} \pi \models q & \text{iff} & q \in L(\pi^{(0)}) \\ \pi \models \varphi \vee \psi & \text{iff} & \pi \models \varphi \text{ or } \pi \models \psi \\ \pi \models \varphi \wedge \psi & \text{iff} & \pi \models \varphi \text{ and } \pi \models \psi \end{array}$$

$$
\begin{array}{lll}
\pi \models \mathtt{X}\varphi & \text{iff} & \pi^1 \models \varphi \\
\pi \models \varphi\mathtt{U}\psi & \text{iff} & \text{there is a } k \in \mathbb{N}, \text{ s.t. } \pi^k \models \psi \text{ and} \\
 & & \text{for all } j \in \mathbb{N}: \text{ if } 0 \leq j < k \text{ then } \pi^j \models \varphi \\
\pi \models \varphi\mathtt{R}\psi & \text{iff} & \text{for all } k \in \mathbb{N}: \pi^k \models \psi \text{ or} \\
 & & \text{there is a } j \in \mathbb{N} \text{ s.t. } 0 \leq j < k \text{ and } \pi^j \models \varphi
\end{array}
$$

The temporal operators $\mathtt{U}$ and $\mathtt{R}$ can also be characterised by the recursive equations

$$
\begin{array}{lll}
\varphi\mathtt{U}\psi & \equiv & \psi \vee (\varphi \wedge \mathtt{X}(\varphi\mathtt{U}\psi)) \\
\varphi\mathtt{R}\psi & \equiv & \psi \wedge (\varphi \vee \mathtt{X}(\varphi\mathtt{R}\psi))
\end{array}
$$

where $\varphi\mathtt{U}\psi$ is the least solution and $\varphi\mathtt{R}\psi$ the greatest solution to the corresponding equivalence. The right sides of these equations are called the *unfoldings* of an $\mathtt{U}$, resp. a $\mathtt{R}$.

As *subformulas* of a $\varphi \in \mathrm{LTL}$ we do not just consider formulas that occur in the syntax tree of $\varphi$. Instead, the unfoldings have to be taken care of as well.

$$
\begin{array}{lll}
Sub(q) & = & \{q\} \\
Sub(\varphi \vee \psi) & = & \{\varphi \vee \psi\} \cup Sub(\varphi) \cup Sub(\psi) \\
Sub(\varphi \wedge \psi) & = & \{\varphi \wedge \psi\} \cup Sub(\varphi) \cup Sub(\psi) \\
Sub(\mathtt{X}\varphi) & = & \{\mathtt{X}\varphi\} \cup Sub(\varphi) \\
Sub(\varphi\mathtt{U}\psi) & = & \{\varphi\mathtt{U}\psi, \mathtt{X}(\varphi\mathtt{U}\psi), \varphi \wedge \mathtt{X}(\varphi\mathtt{U}\psi), \psi \vee (\varphi \wedge \mathtt{X}(\varphi\mathtt{U}\psi))\} \\
 & & \cup\ Sub(\varphi) \cup Sub(\psi) \\
Sub(\varphi\mathtt{R}\psi) & = & \{\varphi\mathtt{R}\psi, \mathtt{X}(\varphi\mathtt{R}\psi), \varphi \vee \mathtt{X}(\varphi\mathtt{R}\psi), \psi \wedge (\varphi \vee \mathtt{X}(\varphi\mathtt{R}\psi))\} \\
 & & \cup\ Sub(\varphi) \cup Sub(\psi)
\end{array}
$$

**Lemma 8 (Negation closure)** *LTL is closed under negation.*

PROOF For every $\varphi \in \mathrm{LTL}$ we define $\overline{\varphi}$ in the following way.

$$
\begin{array}{lllllll}
\overline{\varphi \wedge \psi} & := & \overline{\varphi} \vee \overline{\psi} & \quad & \overline{\varphi\mathtt{U}\psi} & := & \overline{\varphi}\mathtt{R}\overline{\psi} \\
\overline{\varphi \vee \psi} & := & \overline{\varphi} \wedge \overline{\psi} & \quad & \overline{\varphi\mathtt{R}\psi} & := & \overline{\varphi}\mathtt{U}\overline{\psi} \\
\overline{\mathtt{X}\varphi} & := & \mathtt{X}\overline{\varphi} & & & &
\end{array}
$$

Then,

$$\pi \models \overline{\varphi} \quad \text{iff} \quad \pi \not\models \varphi$$

for all LTL formulas $\varphi$ and all paths $\pi$ of all total transition systems $\mathcal{T}$. Note that the equivalence $\overline{\mathtt{X}\varphi} \equiv \mathtt{X}\overline{\varphi}$ in general does not hold on finite paths. ∎

For correctness proofs in later chapters we will need approximants of U and R formulas.

**Definition 9** Let $k \in \mathbb{N}$. *Approximants* of $\varphi\mathtt{U}\psi$ are defined as

$$\begin{aligned} \varphi\mathtt{U}^0\psi &:= \mathtt{ff} \\ \varphi\mathtt{U}^{k+1}\psi &:= \psi \vee (\varphi \wedge \mathtt{X}(\varphi\mathtt{U}^k\psi)) \end{aligned}$$

Dually, approximants of $\varphi\mathtt{R}\psi$ are defined as

$$\begin{aligned} \varphi\mathtt{R}^0\psi &:= \mathtt{tt} \\ \varphi\mathtt{R}^{k+1}\psi &:= \psi \wedge (\varphi \vee \mathtt{X}(\varphi\mathtt{R}^k\psi)) \end{aligned}$$

**Lemma 10 (Approximants)** *Let $\pi$ be a path of a total transition system $\mathcal{T}$ and $\varphi, \psi \in$ LTL.*
*a) $\pi \models \varphi\mathtt{U}\psi$ iff there is a $k \in \mathbb{N}$ s.t. $\pi \models \varphi\mathtt{U}^k\psi$,*
*a) $\pi \models \varphi\mathtt{R}\psi$ iff for all $k \in \mathbb{N}$: $\pi \models \varphi\mathtt{R}^k\psi$.*

PROOF a) Suppose $\pi \models \varphi\mathtt{U}\psi$. Then there is a $k \in \mathbb{N}$ s.t. $\pi^k \models \psi$ and for all $j < k$: $\pi^j \models \varphi$. Thus,

$$\pi \models \underbrace{\varphi \wedge \mathtt{X}(\varphi \wedge \mathtt{X}(\ldots \varphi}_{k-1 \text{ times}} \wedge \mathtt{X}\psi))$$

Then $\pi \models \varphi\mathtt{U}^k\psi$ because

$$\varphi\mathtt{U}^k\psi \equiv \psi \vee (\underbrace{\varphi \wedge \mathtt{X}(\psi \vee (\varphi \wedge \mathtt{X}(\ldots \varphi}_{k-1 \text{ times}} \wedge \mathtt{X}\psi)))) \tag{2.3}$$

Suppose now $\pi \models \varphi\mathtt{U}^k\psi$ for some $k \in \mathbb{N}$. Take the least such $k$. Again, by (2.3), $\pi \models \varphi\mathtt{U}\psi$ since every disjunction must be fulfilled by the disjunct containing $\varphi$. Otherwise, $k$ would not be least.

b) First we show by induction on $k$ that $\varphi \mathtt{R}^k \psi \equiv \overline{(\overline{\varphi}\mathtt{U}^k\overline{\psi})}$. This is true for $k = 0$. Suppose it is true for an arbitrary $k$.

$$\begin{aligned}
\varphi \mathtt{R}^{k+1} \psi &\equiv \psi \wedge (\varphi \vee \mathtt{X}(\varphi \mathtt{R}^k \psi)) \\
&\equiv \psi \wedge (\varphi \vee \mathtt{X}(\overline{\overline{\varphi}\mathtt{U}^k\overline{\psi}})) \\
&\equiv \overline{\overline{\psi} \vee (\overline{\varphi} \wedge \overline{\mathtt{X}(\overline{\overline{\varphi}\mathtt{U}^k\overline{\psi}})})} \\
&\equiv \overline{\overline{\psi} \vee (\overline{\varphi} \wedge \mathtt{X}(\overline{\varphi}\mathtt{U}^k\overline{\psi}))} \\
&\equiv \overline{\overline{\varphi}\mathtt{U}^{k+1}\overline{\psi}}
\end{aligned}$$

Now, $\pi \models \varphi \mathtt{R} \psi$ iff $\pi \not\models \overline{\varphi}\mathtt{U}\overline{\psi}$ iff for all $k \in \mathbb{N}$: $\pi \not\models \overline{\varphi}\mathtt{U}^k\overline{\psi}$ iff for all $k \in \mathbb{N}$: $\pi \models \varphi \mathtt{R}^k \psi$. ■

### Branching Time Logics

As in the case of LTL, branching time logics existed well before they found their way into computer science. In this framework, the future of a moment is not unique, instead there can be several possible future moments. I.e. states of models for branching time logics have several successors in general. The question of which of these views on time is preferable or more useful has been discussed by many people, see [EH86] and [Sti89] for example. [Var01] is meant to be the final say in this controversial matter.

One of the first branching time temporal logics to be used in computer science is the Computation Tree Logic CTL, introduced in [EH85] together with CTL$^+$. Similar logics have been proposed in [BAPM83], [EC80] and [Lam80]. Shorty afterwards, [EH86] defined the Full Branching Time Logic CTL$^*$ which was meant to unify CTL and LTL and allow them to be compared with one another.

Here, we build branching time logics from a set of operators similar to the ones of linear time logic. In addition to that, they are able to quantify over paths and therefore are interpreted over transition systems directly. These are assumed to be total, too. The *syntax* of CTL$^*$ is given by

$$\varphi \quad ::= \quad q \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathtt{X}\varphi \mid \varphi \mathtt{U} \varphi \mid \varphi \mathtt{R} \varphi \mid \mathtt{A}\varphi \mid \mathtt{E}\varphi$$

where $q$ ranges over $\mathcal{P}$.

For a CTL* formula $\varphi$ the set of *subformulas* $Sub(\varphi)$ is defined in the same way as it is for an LTL formula. Additionally,

$$\begin{aligned} Sub(\mathtt{A}\varphi) &= \{\mathtt{A}\varphi\} \cup Sub(\varphi) \\ Sub(\mathtt{E}\varphi) &= \{\mathtt{E}\varphi\} \cup Sub(\varphi) \end{aligned}$$

The *semantics* is defined inductively using paths $\pi$ of a total transition system $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$.

$$\begin{array}{lll}
\pi \models q & \text{iff} & q \in L(\pi^{(0)}) \\
\pi \models \varphi \vee \psi & \text{iff} & \pi \models \varphi \text{ or } \pi \models \psi \\
\pi \models \varphi \wedge \psi & \text{iff} & \pi \models \varphi \text{ and } \pi \models \psi \\
\pi \models \mathtt{X}\varphi & \text{iff} & \pi^1 \models \varphi \\
\pi \models \varphi\mathtt{U}\psi & \text{iff} & \text{there is a } k \in \mathbb{N}, \text{ s.t. } \pi^k \models \psi \text{ and} \\
 & & \text{for all } j \in \mathbb{N}: \text{ if } 0 \leq j < k \text{ then } \pi^j \models \varphi \\
\pi \models \varphi\mathtt{R}\psi & \text{iff} & \text{for all } k \in \mathbb{N}: \pi^k \models \psi \text{ or} \\
 & & \text{there is a } j \in \mathbb{N} \text{ s.t. } 0 \leq j < k \text{ and } \pi^j \models \varphi \\
\pi \models \mathtt{A}\varphi & \text{iff} & \text{for all paths } \pi' : \text{ if } \pi^{(0)} = \pi'^{(0)} \text{ then } \pi' \models \varphi \\
\pi \models \mathtt{E}\varphi & \text{iff} & \text{there is a path } \pi', \text{ s.t. } \pi^{(0)} = \pi'^{(0)} \text{ and } \pi' \models \varphi
\end{array}$$

E and A are called *path quantifiers*.

A CTL* formula $\varphi$ is called a *state formula* iff $\varphi \equiv \mathtt{A}\varphi$, and *path formula* otherwise. We will consider state formulas only. Therefore, one can assume every CTL* state formula to begin with an A. Note that

$$Q_2 Q_1 \varphi \equiv Q_1 \varphi \quad \text{for } Q_1, Q_2 \in \{\mathtt{A}, \mathtt{E}\}$$

The truth value of state formulas only depends on a single state. Is is therefore possible to write $s_0 \models \varphi$ if $\pi \models \varphi$ for all $\pi = s_0 s_1 \ldots$.

The *pure branching time logic* CTL is obtained as a fragment of CTL* by requiring the path operators X, U and R to be preceded immediately by a path quantifier.

$$\begin{array}{lll}
\varphi & ::= & q \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid Q\mathtt{X}\varphi \mid Q(\varphi\mathtt{U}\varphi) \mid Q(\varphi\mathtt{R}\varphi) \\
Q & ::= & \mathtt{A} \mid \mathtt{E}
\end{array}$$

Although the set of *subformulas* of a CTL formula can be defined by regarding it as a CTL* formula it is helpful to use a more specialised definition.

$$\begin{array}{lcl}
Sub(q) & = & \{q\} \\
Sub(\varphi \vee \psi) & = & \{\varphi \vee \psi\} \cup Sub(\varphi) \cup Sub(\psi) \\
Sub(\varphi \wedge \psi) & = & \{\varphi \wedge \psi\} \cup Sub(\varphi) \cup Sub(\psi) \\
Sub(Q\mathtt{X}\varphi) & = & \{Q\mathtt{X}\varphi\} \cup Sub(\varphi) \\
Sub(Q(\varphi\mathtt{U}\psi)) & = & \{Q(\varphi\mathtt{U}\psi), Q\mathtt{X}Q(\varphi\mathtt{U}\psi), \varphi \wedge Q\mathtt{X}Q(\varphi\mathtt{U}\psi), \psi \vee (\varphi \wedge Q\mathtt{X}Q(\varphi\mathtt{U}\psi))\} \\
 & & \cup\ Sub(\varphi) \cup Sub(\psi) \\
Sub(Q(\varphi\mathtt{R}\psi)) & = & \{Q(\varphi\mathtt{R}\psi), Q\mathtt{X}Q(\varphi\mathtt{R}\psi), \varphi \vee Q\mathtt{X}Q(\varphi\mathtt{R}\psi), \psi \wedge (\varphi \vee Q\mathtt{X}Q(\varphi\mathtt{R}\psi))\} \\
 & & \cup\ Sub(\varphi) \cup Sub(\psi)
\end{array}$$

In CTL the following equivalences hold:

$$\begin{array}{lcl}
Q(\varphi\mathtt{U}\psi) & \equiv & \psi \vee (\varphi \wedge Q\mathtt{X}Q(\varphi\mathtt{U}\psi)) \\
Q(\varphi\mathtt{R}\psi) & \equiv & \psi \wedge (\varphi \vee Q\mathtt{X}Q(\varphi\mathtt{R}\psi))
\end{array}$$

CTL$^+$ is the fragment of CTL* that allows boolean combinations of path formulas in the immediate scope of a path quantifier but forbids nesting of them.

$$\begin{array}{lcl}
\varphi & ::= & q \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid Q\psi \\
\psi & ::= & \psi \vee \psi \mid \psi \wedge \psi \mid \mathtt{X}\varphi \mid \varphi\mathtt{U}\varphi \mid \varphi\mathtt{R}\varphi \\
Q & ::= & \mathtt{A} \mid \mathtt{E}
\end{array}$$

Again, $q \in \mathcal{P}$. The set of *subformulas* for a CTL$^+$ formula is given by the subformula definition for CTL*. However, the CTL$^+$ unfolding of a $\varphi\mathtt{U}\psi$ or $\varphi\mathtt{R}\psi$ is the same as the one for CTL.

**Lemma 11 (Negation closure)** *CTL*,CTL and CTL$^+$ are closed under negation.*

PROOF We define the complement of a branching time formula in the following way.

$$\begin{array}{lcl@{\qquad\qquad}lcl}
\overline{\mathtt{A}\varphi} & := & \mathtt{E}\overline{\varphi} & \overline{\varphi\mathtt{U}\psi} & := & \overline{\varphi}\mathtt{R}\overline{\psi} \\
\overline{\mathtt{E}\varphi} & := & \mathtt{A}\overline{\varphi} & \overline{\varphi\mathtt{R}\psi} & := & \overline{\varphi}\mathtt{U}\overline{\psi} \\
\overline{\varphi \wedge \psi} & := & \overline{\varphi} \vee \overline{\psi} & \overline{\mathtt{X}\varphi} & := & \mathtt{X}\overline{\varphi} \\
\overline{\varphi \vee \psi} & := & \overline{\varphi} \wedge \overline{\psi} & & &
\end{array}$$

This construction preserves the special structure of CTL and CTL$^+$ formulas. ■

LTL, as defined in this section, can be interpreted directly over total transition systems, too. This is done by regarding *all* paths that begin with a designated state $s$. This is the same as preceding an LTL formula $\varphi$ with the $\mathtt{A}$ path quantifier and regarding the result as the CTL$^*$ state formula $\mathtt{A}\varphi$ interpreted in $s$.

However, not every CTL$^*$ formula can be represented in this way. Therefore, it is useful to consider this fragment of CTL$^*$ as a logic on its own. To avoid confusion with the real linear time LTL, we call this logic the *branching time version of* LTL, BLTL. Its *syntax* is given by

$$\begin{array}{lcl} \varphi & ::= & \mathtt{A}\psi \\ \psi & ::= & q \mid \psi \vee \psi \mid \psi \wedge \psi \mid \mathtt{X}\psi \mid \psi \mathtt{U} \psi \mid \psi \mathtt{R} \psi \end{array}$$

where $q \in \mathcal{P}$. The set of *subformulas* of an BLTL formula is given by regarding it as a CTL$^*$ formula.

BLTL is not closed under negation according to Definition 1. By Lemma 11, the negation of a BLTL formula is of the form $\mathtt{E}\psi$ where $\psi \equiv \overline{\varphi}$ for some $\varphi \in$ LTL. Since LTL is negation closed (Lemma 8) negation closure of BLTL would imply the fact that every universally path quantified property can also be expressed as an existentially quantified one. This is not the case.

Since the pure linear time part of BLTL, namely LTL, is negation closed one could define negation in BLTL as $\overline{\mathtt{A}\varphi} := \mathtt{A}\overline{\varphi}$. But this results in the fact that it is possible for a transition system to neither satisfy a formula nor its negation. At least it is impossible for a transition system to satisfy both.

If negation closure is defined as

$$\mathtt{A}\varphi \text{ is satisfiable} \quad \text{iff} \quad \mathtt{A}\overline{\varphi} \text{ is not valid}$$

for an LTL formula $\varphi$ then BLTL is negation closed.

**Example 12** A simple CTL formula is $\mathtt{AGEXtt}$ which says that no reachable state is a deadlock, i.e. does not have any successor states. This is in fact a validity since CTL is interpreted over total transition system, i.e. this property is always trivially fulfilled.

An example of a CTL$^*$ formula is $\varphi := \mathtt{E}(\overline{q}\mathtt{U}(\mathtt{G}q))$. It postulates the existence of an infinite path with a finite prefix, s.t. no state on the prefix satisfies $q$ whereas all other states do.

Another example is $\varphi := \mathtt{A}(\mathtt{X}q \vee \mathtt{X}\overline{q})$. $\varphi$ simply says that every path's next state is either labelled with $q$ or $\overline{q}$. This is not the most interesting property but a simple and good example to illustrate the CTL$^*$ model checking games in Chapter 5.2. In fact, $\varphi$ is already a CTL$^+$ and a BLTL formula.

As a last example we consider $\varphi := \mathtt{E}(\mathtt{F}q \wedge \mathtt{GF}q)$. This is a genuine CTL$^*$ formula that postulates the existence of a path on which $q$ holds infinitely often. It is not the shortest formula that expresses this property but, again, will be useful to illustrate the CTL$^*$ model checking games in Chapter 5.

## 2.5 Modal Logics

Unlike temporal logics, modal logics distinguish transitions of an LTS with different labels. Thus, they are interpreted over transition systems $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$.

### Propositional Dynamic Logic

PDL, as introduced in [FL79], augments basic modal logic with an infinite but regular set of action names. They are usually called *programs*. Formulas $\varphi$ and programs $\alpha$ are mutually recursively defined as

$$\begin{array}{lcl} \varphi & ::= & q \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle\alpha\rangle\varphi \mid [\alpha]\varphi \\ \alpha & ::= & a \mid \alpha \cup \alpha \mid \alpha;\alpha \mid \alpha^* \mid \varphi? \end{array}$$

where $q$ ranges over $\mathcal{P}$ and $a$ over $\mathcal{A}$.

The transition relations $\xrightarrow{a}$ of an LTS can be extended to programs $\alpha$ in the following way. In the case of the *test operator* $\varphi?$ it refers to the semantics of $\varphi$. Since the formula sizes get reduced this mutual recursion is well-founded.

$$
\begin{array}{lll}
s \xrightarrow{\alpha\cup\beta} t & \text{iff} & s \xrightarrow{\alpha} t \ \text{ or } \ s \xrightarrow{\beta} t \\
s \xrightarrow{\alpha;\beta} t & \text{iff} & \text{there is an } u \in \mathcal{S}, \ \text{s.t. } \ s \xrightarrow{\alpha} u \ \text{ and } \ u \xrightarrow{\beta} t \\
s \xrightarrow{\alpha^*} t & \text{iff} & \text{there is an } n \in \mathbb{N}, \ \text{s.t. } \ s \xrightarrow{\alpha^n} t \ \text{ where} \\
 & & \text{for all } s,t \in \mathcal{S} : s \xrightarrow{\alpha^0} s \ \text{ and } \ s \xrightarrow{\alpha^{k+1}} t \ \text{ iff } \ s \xrightarrow{\alpha;\alpha^k} t \\
s \xrightarrow{\varphi?} s & \text{iff} & s \models \varphi
\end{array}
$$

The *subformulas* of a PDL formula φ depend on both φ's formula structure and the programs contained in it.

$$
\begin{array}{lcl}
Sub(q) & = & \{q\} \\
Sub(\varphi \vee \psi) & = & \{\varphi \vee \psi\} \ \cup \ Sub(\varphi) \ \cup \ Sub(\psi) \\
Sub(\varphi \wedge \psi) & = & \{\varphi \wedge \psi\} \ \cup \ Sub(\varphi) \ \cup \ Sub(\psi) \\
Sub(\langle a \rangle \varphi) & = & \{\langle a \rangle \varphi\} \ \cup \ Sub(\varphi) \\
Sub([a]\varphi) & = & \{[a]\varphi\} \ \cup \ Sub(\varphi) \\
Sub(\langle \alpha \cup \beta \rangle \varphi) & = & \{\langle \alpha \cup \beta \rangle \varphi\} \ \cup \ Sub(\langle \alpha \rangle \varphi \vee \langle \beta \rangle \varphi) \\
Sub([\alpha \cup \beta]\varphi) & = & \{[\alpha \cup \beta]\varphi\} \ \cup \ Sub([\alpha]\varphi \wedge [\beta]\varphi) \\
Sub(\langle \alpha;\beta \rangle \varphi) & = & \{\langle \alpha;\beta \rangle \varphi\} \ \cup \ Sub(\langle \alpha \rangle \langle \beta \rangle \varphi) \\
Sub([\alpha;\beta]\varphi) & = & \{[\alpha;\beta]\varphi\} \ \cup \ Sub([\alpha][\beta]\varphi) \\
Sub(\langle \alpha^* \rangle \varphi) & = & \{\langle \alpha^* \rangle \varphi, \langle \alpha \rangle \langle \alpha^* \rangle \varphi, \varphi \vee \langle \alpha \rangle \langle \alpha^* \rangle \varphi\} \ \cup \ Sub(\varphi) \\
Sub([\alpha^*]\varphi) & = & \{[\alpha^*]\varphi, [\alpha][\alpha^*]\varphi, \varphi \wedge [\alpha][\alpha^*]\varphi\} \ \cup \ Sub(\varphi) \\
Sub(\langle \varphi? \rangle \psi) & = & \{\langle \varphi? \rangle \psi\} \ \cup \ Sub(\varphi) \ \cup \ Sub(\psi) \\
Sub([\varphi?]\psi) & = & \{[\varphi?]\psi\} \ \cup \ Sub(\overline{\varphi}) \ \cup \ Sub(\psi)
\end{array}
$$

This is also called the *Fischer-Ladner closure*. To maintain consistency with the other logics we prefer the term $Sub(\varphi)$. The negation $\overline{\varphi}$ of φ, needed in the subformula definition of $[\varphi?]\psi$ will be defined in Lemma 13 later on.

Formulas of PDL are interpreted over states of an LTS $\mathcal{T}$ which need not be total. Thus, paths of $\mathcal{T}$ can be finite or infinite. Again, we assume the LTS to be fixed and write $s \models \varphi$ instead of $\mathcal{T}, s \models \varphi$ for $s \in \mathcal{S}$.

$$
\begin{array}{lll}
s \models q & \text{iff} & q \in L(s) \\
s \models \varphi \vee \psi & \text{iff} & s \models \varphi \ \text{ or } \ s \models \psi
\end{array}
$$

$$\begin{array}{lll} s \models \varphi \wedge \psi & \text{iff} & s \models \varphi \text{ and } s \models \psi \\ s \models \langle\alpha\rangle\varphi & \text{iff} & \text{there is a } t \in \mathcal{S} \text{ s.t. } s \xrightarrow{\alpha} t \text{ and } t \models \varphi \\ s \models [\alpha]\varphi & \text{iff} & \text{for all } t \in \mathcal{S} : \text{ if } s \xrightarrow{\alpha} t \text{ then } t \models \varphi \end{array}$$

The implicit fixed point constructs of PDL are $\langle\alpha^*\rangle\varphi$ and $[\alpha^*]\varphi$. They can be characterised by the following equivalences.

$$\begin{array}{rcl} \langle\alpha^*\rangle\varphi & \equiv & \alpha \vee \langle\alpha\rangle\langle\alpha^*\rangle\varphi \\ [\alpha^*]\varphi & \equiv & \alpha \wedge [\alpha][\alpha^*]\varphi \end{array}$$

As with the temporal logics, the first equivalence is to be taken as the least solution and the second as the greatest.

**Lemma 13 (Negation closure)** *PDL is closed under negation.*

PROOF We define the complement of a PDL formula in the following way.

$$\begin{array}{rclcrcl} \overline{\varphi \vee \psi} & := & \overline{\varphi} \wedge \overline{\psi} & \qquad & \overline{\langle\alpha\rangle\varphi} & := & [\alpha]\overline{\varphi} \\ \overline{\varphi \wedge \psi} & := & \overline{\varphi} \vee \overline{\psi} & \qquad & \overline{[\alpha]\varphi} & := & \langle\alpha\rangle\overline{\varphi} \end{array}$$

■

Important equivalences of PDL formulas are

$$\begin{array}{rclcrcl} \langle\alpha \cup \beta\rangle\varphi & \equiv & \langle\alpha\rangle\varphi \vee \langle\beta\rangle\varphi & \qquad & \langle\alpha;\beta\rangle\varphi & \equiv & \langle\alpha\rangle\langle\beta\rangle\varphi \\ [\alpha \cup \beta]\varphi & \equiv & [\alpha]\varphi \wedge [\beta]\varphi & \qquad & [\alpha;\beta]\varphi & \equiv & [\alpha][\beta]\varphi \\ \langle\varphi?\rangle\psi & \equiv & \varphi \wedge \psi & \qquad & [\varphi?]\psi & \equiv & \overline{\varphi} \vee \psi \end{array}$$

Again, for the correctness proofs of the PDL games in Chapter 4 and Section 6.3 we need approximants of the implicit fixed point constructs in PDL.

**Definition 14** Let $\alpha$ be a program, $\varphi \in$ PDL, and $k \in \mathbb{N}$. Approximants of $\langle\alpha^*\rangle\varphi$ are defined as

$$\begin{array}{rcl} \langle\alpha^0\rangle\varphi & := & \mathtt{ff} \\ \langle\alpha^{k+1}\rangle\varphi & := & \varphi \vee \langle\alpha\rangle\langle\alpha^k\rangle\varphi \end{array}$$

Dually,

$$\begin{array}{lcl} [\alpha^0]\varphi & := & \texttt{tt} \\ [\alpha^{k+1}]\varphi & := & \varphi \wedge [\alpha][\alpha^k]\varphi \end{array}$$

are approximants of $[\alpha^*]\varphi$.

**Lemma 15 (Approximants)** *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with $s_0 \in \mathcal{S}$ and $\varphi \in PDL$.*
*a) $s_0 \models \langle\alpha^*\rangle\varphi$ iff there is a $k \in \mathbb{N}$, s.t. $s_0 \models \langle\alpha^k\rangle\varphi$.*
*b) $s_0 \models [\alpha^*]\varphi$ iff for all $k \in \mathbb{N}$: $s_0 \models [\alpha^k]\varphi$.*

PROOF a) Suppose $s \models \langle\alpha^*\rangle\varphi$. Then there is a path $\pi = s_0 s_1 \dots s_{k-1} \dots$ in $\mathcal{T}$ s.t. $s_{i-1} \xrightarrow{\alpha} s_i$ for all $1 \leq i < k$ and $s_{k-1} \models \varphi$. Thus, $s_0 \models \langle\alpha^k\rangle\varphi$. Note that $k = 0$ is impossible.

Suppose now $s_0 \models \langle\alpha^k\rangle\varphi$ for some $k \in \mathbb{N}$. Take the least such $k$. By definition $s_0 \models \varphi$ or $s_0 \models \langle\alpha\rangle\langle\alpha^{k-1}\rangle\varphi$. If the former is true then $s_0 \models \langle\alpha^*\rangle\varphi$. Suppose the latter holds. Then there is a $s_1 \in \mathcal{S}$ s.t. $s_0 \xrightarrow{\alpha} s_1$ and $s_1 \models \langle\alpha^{k-1}\rangle\varphi$. This argument can be iterated until a $s_{k-1} \in \mathcal{S}$ is reached s.t. $s_{k-1} \models \varphi$ or $s_{k-1} \models \langle\alpha\rangle\langle\alpha^0\rangle\varphi$. But $\langle\alpha\rangle\langle\alpha^0\rangle\varphi \equiv \texttt{ff}$, hence, the former must hold. But then the sequence $s_0 s_1 \dots s_{k-1}$ witnesses that $s_0 \models \langle\alpha^*\rangle\varphi$.

b)

$$\begin{array}{rll} s_0 \models [\alpha^*]\varphi & \text{iff} & s_0 \not\models \overline{[\alpha^*]\varphi} \\ & \text{iff} & s_0 \not\models \langle\alpha^*\rangle\overline{\varphi} \\ & \text{iff} & \text{for all } \beta \in \mathbb{O}\text{rd} : s_0 \not\models \langle\alpha^\beta\rangle\overline{\varphi} \\ & \text{iff} & \text{for all } \beta \in \mathbb{O}\text{rd} : s_0 \models [\alpha^\beta]\varphi \end{array}$$

using part a), Definition 14 and Lemma 13. ■

**Example 16** An example of a PDL formula is

$$\varphi := \langle(\overline{q}?; a)^*\rangle q$$

It expresses an existentially path quantified *until* property: "there is a path labelled with $a$s on which $q$ does not hold until it holds". The $\langle\alpha^*\rangle q$ makes sure that $q$ holds eventually on some path. The program $\overline{q}?; a$ forces every state before that on this path not to satisfy $q$ and to have an $a$-transition to the next state.

Several extensions of PDL have been considered in the literature, for instance in [Str81] or [Str85]. One example is the use of *converse operators* which allow formulas to speak about the backwards-execution of a program. Formally, the set of programs is defined as

$$\alpha \quad ::= \quad a \mid \alpha \cup \alpha \mid \alpha;\alpha \mid \alpha^* \mid \varphi? \mid \overline{\alpha}$$

where $a$ ranges over $\mathcal{A}$.

The *semantics* of converse transitions is given by

$$s \xrightarrow{\overline{\alpha}} t \text{ iff } t \xrightarrow{\alpha} s$$

It is sufficient to allow the converse operator to be applied to atomic programs only since the following equivalences hold.

$$\begin{array}{lll} s \xrightarrow{\overline{\alpha \cup \beta}} t & \text{iff} & s \xrightarrow{\overline{\alpha} \cup \overline{\beta}} t \\ s \xrightarrow{\overline{\alpha;\beta}} t & \text{iff} & s \xrightarrow{\overline{\beta};\overline{\alpha}} t \\ s \xrightarrow{\overline{\alpha^*}} t & \text{iff} & s \xrightarrow{\overline{\alpha}^*} t \end{array}$$

Another extension is PDL-$\Delta$ which features a new formula construct $\mathit{repeat}(\alpha)$ where $\alpha$ is a program. Its *semantics* is given by

$$s_0 \models \mathit{repeat}(\alpha) \text{ iff there is an infinite path } \pi = s_0 s_1 \ldots \text{ s.t. } s_i \xrightarrow{\alpha} s_{i+1} \text{ for all } i \in \mathbb{N}$$

## Fixed Point Logic with Chop

*Fixed point logic with chop*, FLC, was defined in [MO99]. It extends $\mathcal{L}_\mu$ and thus features explicit fixed point constructs. Its *syntax* assumes the existence of a set $\mathcal{V} = \{Z, Y, \ldots\}$ of *propositional variables*, and is given by

$$\varphi \quad ::= \quad q \mid Z \mid \tau \mid \langle a \rangle \mid [a] \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mu Z.\varphi \mid \nu Z.\varphi \mid \varphi;\varphi$$

where $q \in \mathcal{P}$, $Z \in \mathcal{V}$, and $a \in \mathcal{A}$. We let $\sigma Z.\varphi$ stand for either $\mu Z.\varphi$ or $\nu Z.\varphi$. Furthermore, we assume formulas to be *well-named*, i.e. different fixed point formulas do not use the same variable to do quantification. In this case there is a function *fp* that maps every variable $Z$ to its defining fixed point formula, i.e. $fp(Z) = \sigma Z.\psi$ for some $\psi$. The *fixed point type* of a variable $Z$ is $\mu$ if $fp(Z) = \mu Z.\psi$ for some $\psi$ and $\nu$ otherwise.

The set of *subformulas* of an FLC formula is defined as follows.

$$
\begin{array}{lcl}
Sub(q) & = & \{q\} \\
Sub(Z) & = & \{Z\} \\
Sub(\tau) & = & \{\tau\} \\
Sub(\langle a \rangle) & = & \{\langle a \rangle\} \\
Sub([a]) & = & \{[a]\} \\
Sub(\varphi \vee \psi) & = & \{\varphi \vee \psi\} \ \cup \ Sub(\varphi) \ \cup \ Sub(\psi) \\
Sub(\varphi \wedge \psi) & = & \{\varphi \wedge \psi\} \ \cup \ Sub(\varphi) \ \cup \ Sub(\psi) \\
Sub(\sigma Z.\varphi) & = & \{\sigma Z.\varphi\} \ \cup \ Sub(\varphi) \\
Sub(\varphi;\psi) & = & \{\varphi;\psi\} \ \cup \ Sub(\varphi) \ \cup \ Sub(\psi)
\end{array}
$$

The set of *free variables* of an FLC formula is given by

$$
\begin{array}{lcl}
free(q) & := & \emptyset \\
free(Z) & := & \{Z\} \\
free(\tau) & := & \emptyset \\
free(\langle a \rangle) & := & \emptyset \\
free([a]) & := & \emptyset \\
free(\varphi \vee \psi) & := & free(\varphi) \ \cup \ free(\psi) \\
free(\varphi \wedge \psi) & := & free(\varphi) \ \cup \ free(\psi) \\
free(\sigma Z.\varphi) & := & free(\varphi) - \{Z\} \\
free(\varphi;\psi) & := & free(\varphi) \ \cup \ free(\psi)
\end{array}
$$

A formula $\varphi$ is *closed* if it contains no free variables, i.e. $free(\varphi) = \emptyset$.

In the following we will define syntactical fragments of FLC. Like modal $\mu$-calculus, the alternation depth of a formula for example is an important factor for the efficiency of a model checking algorithm. But it is not the only one. The number of times variables occur sequentially composed to themselves is equally important.

We say that $Z$ depends on $Y$ in $\varphi$, written $Z \prec_\varphi Y$, if $Y \in free(fp(Z))$. We write $Z <_\varphi Y$ iff $(Z,Y)$ is in the transitive closure of $\prec_\varphi$. The *alternation depth* of $\varphi$, $ad(\varphi)$, is the maximal number $k$ in a chain of variables $Z_0 <_\varphi Z_1 <_\varphi \ldots <_\varphi Z_k$ s.t. $Z_{i-1}$ and $Z_i$ are of different fixed point types for $0 < i \leq k$.

Informally the *sequential depth* of a formula measures the number of times a variable occurs in a sequence of formulas that are sequentially composed.

**Definition 17** The sequential depth of $\varphi$ is defined as

$$sd(\varphi) \;:=\; max\ \{\ sd_Z(fp(Z)) \mid Z \in Sub(\varphi)\ \}$$

where

$$\begin{array}{lcl}
sd_Z(\psi) & := & 0 \quad \text{if } \psi \in \mathcal{P} \cup \{\langle a \rangle, [a], \tau\} \\
sd_Z(\varphi \vee \psi) & := & max\ \{\ sd_Z(\varphi), sd_Z(\psi)\ \} \\
sd_Z(\varphi \wedge \psi) & := & max\ \{\ sd_Z(\varphi), sd_Z(\psi)\ \} \\
sd_Z(\varphi; \psi) & := & sd_Z(\varphi) + sd_Z(\psi) \\
sd_Z(\sigma Y.\varphi) & := & sd_Z(\varphi) \\
sd_Z(Y) & := & \begin{cases} 1 & \text{if } Y = Z \\ 0 & \text{o.w.} \end{cases}
\end{array}$$

Important syntactical fragments of FLC are those with fixed alternation and sequential depth because they allow model checking algorithms to be more efficient than those for the general FLC. However, this usually comes at the expense of a reduced expressive power. The question of whether or not the hierarchy of levels with fixed alternation, resp. sequential depth is strict, is open.

$$\begin{array}{lcl}
\text{FLC}^{k,n} & := & \{\ \varphi \in \text{FLC} \mid ad(\varphi) \leq k, sd(\varphi) \leq n\ \} \\
\text{FLC}^{k} & := & \bigcup_{n \in \mathbb{N}} \text{FLC}^{k,n} \\
\text{FLC}^{\omega,n} & := & \bigcup_{k \in \mathbb{N}} \text{FLC}^{k,n}
\end{array}$$

**Definition 18** The *tail* of a variable $Z$ in a formula $\varphi$, $tl_Z$, is a set consisting of those formulas that occur "behind" $Z$ in $fp(Z)$. In order to define it technically we use sequential composition for sets of formulas in a straightforward way:

$$\{\varphi_0, \ldots, \varphi_n\}; \psi \;:=\; \{\varphi_0; \psi, \ldots, \varphi_n; \psi\}$$

We also use the eponymous function $tl_Z : Sub(\varphi) \to 2^{Sub(\varphi)}$ where

$$
\begin{array}{lcl}
tl_Z(q) & := & \{q\} \\
tl_Z(\tau) & := & \{\tau\} \\
tl_Z(\langle a \rangle) & := & \{\langle a \rangle\} \\
tl_Z([a]) & := & \{[a]\} \\
tl_Z(\varphi \vee \psi) & := & tl_Z(\varphi) \cup tl_Z(\psi) \\
tl_Z(\varphi \wedge \psi) & := & tl_Z(\varphi) \cup tl_Z(\psi) \\
tl_Z(\sigma Y.\psi) & := & tl_Z(\psi) \\
tl_Z(Y) & := & \begin{cases} \{Y\} & \text{if } Y \neq Z \\ \{\tau\} & \text{o.w.} \end{cases} \\
tl_Z(\varphi;\psi) & := & T_1 \cup T_2
\end{array}
$$

with

$$
T_1 \;:=\; \begin{cases} tl_Z(\varphi);\psi & \text{if } Z \in Sub(\varphi) \\ \{\tau\} & \text{o.w.} \end{cases}
$$

$$
T_2 \;:=\; \begin{cases} tl_Z(\psi) & \text{if } Z \in Sub(\psi) \\ \{\tau\} & \text{o.w.} \end{cases}
$$

The tail of $Z$ in $\varphi$ is simply calculated as $tl_Z := tl_Z(fp(Z))$.

Another important syntactical fragment of FLC is the one containing those formulas whose variables have trivial tails only.

$$
\mathrm{FLC}^- \quad := \quad \{ \varphi \in \mathrm{FLC} \mid tl_Z = \{\tau\} \text{ for all } Z \in Sub(\varphi) \}
$$

Note that $\mathrm{FLC}^-$ subsumes $\mathcal{L}_\mu$, [MO99]. It is the fragment of FLC considered here which permits the most efficient model checking procedure. This is basically because it does not bear an essential difference to modal $\mu$-calculus.

FLC is interpreted over transition systems $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ which need not be total. The *semantics* of an FLC formula is a monotone *state transformer* $f : 2^{\mathcal{S}} \to 2^{\mathcal{S}}$. To allow an inductive definition one needs to handle *open* formulas. This is done using an *environment* which is a function $\rho : \mathcal{V} \to (2^{\mathcal{S}} \to 2^{\mathcal{S}})$ that maps variables to state transformers. $\rho[Z \mapsto f]$ is the function that maps $Z$ to $f$ and agrees with $\rho$ on all other arguments. The semantics $[\![\cdot]\!]_\rho^{\mathcal{T}} : 2^{\mathcal{S}} \to 2^{\mathcal{S}}$ of an FLC formula, relative to $\mathcal{T}$ and $\rho$, is a monotone function on subsets of states with respect to the inclusion ordering on $2^{\mathcal{S}}$. These functions together with the partial order given by

$$f \sqsubseteq g \quad \text{iff} \quad \text{for all } X \subseteq \mathcal{S} : \quad f(X) \subseteq g(X)$$

form a complete lattice with suprema $\sqcup$ and infima $\sqcap$. By Theorem 3, the least and greatest fixed points of functionals $F : (2^{\mathcal{S}} \to 2^{\mathcal{S}}) \to (2^{\mathcal{S}} \to 2^{\mathcal{S}})$ exist. They are used to interpret fixed point formulas of FLC. We use $\lambda$-notation for functions.

$$\begin{array}{lcl}
[\![q]\!]_\rho & = & \lambda X.\{s \in \mathcal{S} \mid q \in L(s)\} \\
[\![Z]\!]_\rho & = & \rho(Z) \\
[\![\tau]\!]_\rho & = & \lambda X.X \\
[\![\langle a \rangle]\!]_\rho & = & \lambda X.\{s \in \mathcal{S} \mid \exists t \in X,\ \text{s.t. } s \xrightarrow{a} t\} \\
[\![[a]]\!]_\rho & = & \lambda X.\{s \in \mathcal{S} \mid \forall t \in \mathcal{S},\ \text{if } s \xrightarrow{a} t \text{ then } t \in X\} \\
[\![\varphi \vee \psi]\!]_\rho & = & \lambda X.[\![\varphi]\!]_\rho(X) \cup [\![\psi]\!]_\rho(X) \\
[\![\varphi \wedge \psi]\!]_\rho & = & \lambda X.[\![\varphi]\!]_\rho(X) \cap [\![\psi]\!]_\rho(X) \\
[\![\mu Z.\varphi]\!]_\rho & = & \bigsqcap \{f : 2^{\mathcal{S}} \to 2^{\mathcal{S}} \mid f \text{ monotone, } [\![\varphi]\!]_{\rho[Z \mapsto f]} \sqsubseteq f\} \\
[\![\nu Z.\varphi]\!]_\rho & = & \bigsqcup \{f : 2^{\mathcal{S}} \to 2^{\mathcal{S}} \mid f \text{ monotone, } f \sqsubseteq [\![\varphi]\!]_{\rho[Z \mapsto f]}\} \\
[\![\varphi;\psi]\!]_\rho & = & [\![\varphi]\!]_\rho \circ [\![\psi]\!]_\rho
\end{array}$$

Given a $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$, a state $s \in \mathcal{S}$ satisfies a formula, $s \models_\rho \varphi$, if $s \in [\![\varphi]\!]_\rho^{\mathcal{T}}(\mathcal{S})$. Note that an environment is not needed if $\varphi$ is closed.

**Lemma 19** [MO99] *Let $\varphi \in$ FLC be closed. Then $[\![\varphi]\!]$ is monotone, i.e. for all $S, T \subseteq \mathcal{S}$ of an LTS $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$: if $S \subseteq T$ then $[\![\varphi]\!](S) \subseteq [\![\varphi]\!](T)$.*

Two formulas $\varphi$ and $\psi$ are *equivalent*, $\varphi \equiv \psi$, iff their semantics are the same, i.e. for every $\mathcal{T}$ and every $\rho$: $[\![\varphi]\!]_\rho^{\mathcal{T}} = [\![\psi]\!]_\rho^{\mathcal{T}}$. This equivalence is a congruence and thus permits substituitivity. There is no FLC formula $\varphi$ that does not contain $\tau$ as a subformula, s.t. $\varphi \equiv \tau$.

For model checking purposes it is useful to consider a weaker equivalence. $\varphi$ and $\psi$ are called *weakly equivalent*, written $\varphi \approx \psi$, iff they are satisfied by the same states, i.e. $s \models_\rho \varphi$ iff $s \models_\rho \psi$ for any state $s$ of any transition system $\mathcal{T}$ and every $\rho$. Note that weak equivalence is not a congruence as the next example shows.

**Example 20** Consider the two FLC formulas $\langle a \rangle$ and $\langle a \rangle; \mathtt{tt}$. They are weakly equivalent because both say that a state has an $a$-transition to any other state. Now take the context $\_; \langle a \rangle$.

$$\langle a \rangle; \langle a \rangle; \mathtt{tt} \approx \langle a \rangle; \langle a \rangle \not\approx \langle a \rangle; \mathtt{tt}; \langle a \rangle \equiv \langle a \rangle; \mathtt{tt}$$

In this context the second formula still requires a state to have one $a$-transition whereas the first now says that two successive $a$-transitions must be possible.

In [MO99] it is shown how to embed $\mathcal{L}_\mu$ into FLC by using sequential composition. For instance, $\langle a \rangle \varphi$ becomes $\langle a \rangle; \varphi$. Therefore, we will sometimes omit the semicolon to maintain a strong resemblance to the syntax of $\mathcal{L}_\mu$. For example, $\langle a \rangle Z \langle a \rangle$ abbreviates $\langle a \rangle; Z; \langle a \rangle$.

Again, for correctness proofs we need to introduce approximants of FLC fixed point formulas. However, unlike the LTL and PDL cases, $\mathbb{N}$ does not suffice. Instead, one has to use ordinals.

**Definition 21** ***(Approximants)*** Let $fp(Z) = \mu Z.\varphi$ for some $\varphi$ and let $\alpha, \lambda \in \mathbb{O}\mathrm{rd}$ where $\lambda$ is a limit ordinal. Then

$$Z^0 := \mathtt{ff}, \qquad Z^{\alpha+1} = \varphi[Z^\alpha/Z], \qquad Z^\lambda = \bigvee_{\alpha<\lambda} Z^\alpha$$

For greatest fixed points the approximants are defined dually. Let $fp(Z) = \nu Z.\varphi$ for some $\varphi$ and, again, $\alpha, \lambda \in \mathbb{O}\mathrm{rd}$ with $\lambda$ being a limit ordinal. Then

$$Z^0 := \mathtt{tt}, \qquad Z^{\alpha+1} = \varphi[Z^\alpha/Z], \qquad Z^\lambda = \bigwedge_{\alpha<\lambda} Z^\alpha$$

Note that $\mu Z.\varphi \equiv \bigvee_{\alpha \in \mathbb{O}\mathrm{rd}} Z^\alpha$ and $\nu Z.\varphi \equiv \bigwedge_{\alpha \in \mathbb{O}\mathrm{rd}} Z^\alpha$. If only finite transition systems are considered $\mathbb{O}\mathrm{rd}$ can be replaced by $\mathbb{N}$. If the underlying transition system is fixed then its size is an upper bound on the number of approximants needed to calculate fixed point formulas.

**Example 22** Let $\mathcal{A} = \{a,b\}$ and

$$\varphi \;:=\; \nu Y.[b]\mathtt{ff} \wedge [a](\nu Z.[b] \wedge [a](Z;Z));(([a]\mathtt{ff} \wedge [b]\mathtt{ff}) \vee Y)$$

$\varphi$ expresses "on every path at every moment the number of $b$s so far never exceeds the number of $a$s so far". This property is non-regular and, hence, is not expressible in $\mathcal{L}_\mu$. This is an interesting property of protocols when $a$ and $b$ are the actions *send* and *receive*.

The subformula $\psi = \nu Z.[b] \wedge [a](Z;Z)$ expresses "there can be at most one $b$ more than there are $a$s". This is understood best by unfolding the fixed point formula and thus obtaining sequences of modalities and variables. Replacing a $Z$ with a $[b]$ decreases the number of $Z$s whereas replacing it with the other conjunct adds a new $Z$ to the sequence. The games of Chapter 9 will provide a better way of explaining what property is expressed by a given FLC formula.

Then, $[b]\mathtt{ff} \wedge [a]\psi$ postulates that at the beginning no $b$ is possible and for every $n$ $a$s there can be at most $n$ $b$s. Finally, the $Y$ in $\varphi$ allows such sequences to be composed or finished in a deadlock state.

Among the logics presented in this and the previous section, FLC is without a doubt the least known. Therefore, we present a few basic results pertaining to FLC in order to get the reader acquainted with it and to show that FLC is indeed a modal logic in the sense that it has the properties a modal logic is expected to have.

**Theorem 23** [MO99] *Satisfiability of FLC formulas is undecidable.*

This is proved by reduction from the simulation equivalence problem for BPA processes. For a BPA process Q one can construct FLC formulas $\phi_Q, \phi_Q^-, \phi_Q^+$ s.t.

$$\begin{array}{lcllll} P & \models & \phi_Q & \text{iff} & P \sim Q \\ P & \models & \phi_Q^- & \text{iff} & P \text{ simulates } Q \\ P & \models & \phi_Q^+ & \text{iff} & P \text{ is simulated by } Q \end{array}$$

A consequence of this is the following.

**Theorem 24** [MO99] *FLC does not have the finite model property.*

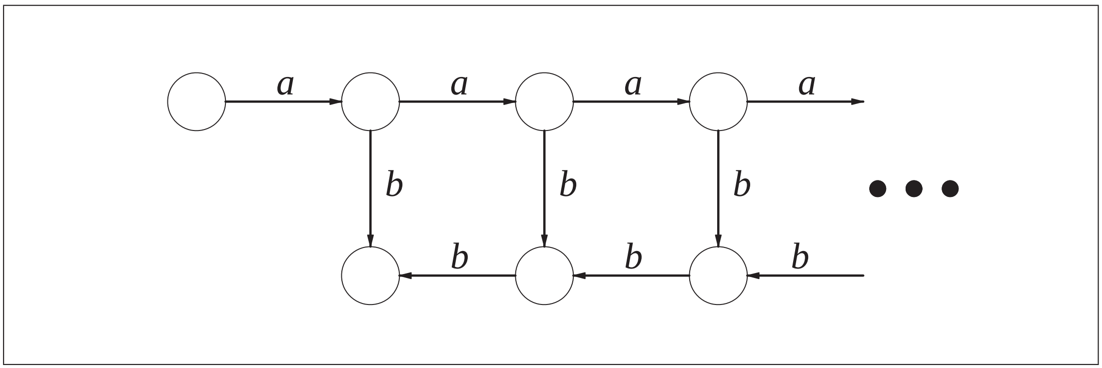


Figure 2.1: The transition system for Example 25.

The finite model property would imply that every BPA process is bisimilar to a finite state transition system.

Next, we give an example of an FLC formula that is satisfiable but does not have a finite model.

**Example 25** Let $\mathcal{A} = \{a, b\}$ and

$$\varphi \;:=\; (\nu Z.\langle a\rangle(Z \wedge \tau);([b] \wedge \langle b\rangle));([a]\mathtt{ff} \wedge [b]\mathtt{ff})$$

The formula postulates the existence of an infinite $a$-path, s.t. after every prefix of $n$ $a$s exactly $n$ $b$s are possible. The body of the fixed point formula can be rewritten as

$$\langle a\rangle(([b] \wedge \langle b\rangle) \wedge Z;([b] \wedge \langle b\rangle))$$

This expresses that there must be a path with transition labels $ab$ at the beginning and all such $b$s lead to states that have similar properties. Moreover, after the $a$ there is another path of the same style with one more $b$ at the end.

$\varphi$ has an infinite model like the one depicted in Figure 2.1. Suppose $\varphi$ has a finite model, too. This could be regarded as a finite automaton $\mathfrak{A}$ with final states being the deadlock states. But $\mathfrak{A}$ would accept the context-free and non-regular language $L = \{a^n b^n \mid n \in \mathbb{N}\}$.

For model checking purposes *converse modalities* that allow formulas to speak about predecessors of states can be integrated without causing any difficulties. Note that in general this does not hold for satisfiability checking problems.

The syntax of FLC can be extended with primitives $\langle\overline{a}\rangle$ and $[\overline{a}]$ where $a$ ranges over $\mathcal{A}$. Their semantics is

$$\begin{aligned} [\![\langle\overline{a}\rangle]\!] &= \lambda X.\{s \in \mathcal{S} \mid \exists t \in X,\ \text{s.t. } t \xrightarrow{a} s\} \\ [\![[\overline{a}]]\!] &= \lambda X.\{s \in \mathcal{S} \mid \forall t \in \mathcal{S}, t \xrightarrow{a} s \Rightarrow t \in X\} \end{aligned}$$

But note that in general $S \neq [\![\langle a\rangle;\langle\overline{a}\rangle]\!](S)$, i.e. $[\![\langle\overline{a}\rangle]\!]$ is not the inverse function of $[\![\langle a\rangle]\!]$. As a counterexample take the transition system with states $\{1,2,3\}$ and transitions $1 \xrightarrow{a} 3$ and $2 \xrightarrow{a} 3$. Then $\{3\} = [\![\langle\overline{a}\rangle]\!](\{1\})$ but $\{1,2\} = [\![\langle a\rangle]\!](\{3\})$.

This extension of FLC is capable of defining *uniform inevitability*, which means property $\psi$ holds on all paths of a transition system at the same moment. In [Eme87] it is shown that $\mathcal{L}_\mu$ cannot do this.

**Example 26** Let $\mathcal{A} = \{a\}$ and

$$\varphi \ := \ \mu Y.\langle a\rangle Y \vee (\psi \wedge (\nu Z.[\overline{a}];(Z \wedge \tau);[a]);\psi)$$

$\varphi$ is an instance of an *eventually* formula of $\mathcal{L}_\mu$, i.e. $\mu Y.\langle a\rangle Y \vee \psi'$ says that there is a path on which $\psi'$ eventually holds. $(\nu Z.[\overline{a}];(Z \wedge \tau);[a]);\psi$ says that at every state that can be reached by a sequence of $n$ $a$s backwards and then $n$ $a$s forwards $\psi$ holds. Composing these two formulas achieves uniform inevitability.

**Lemma 27 (Equivalences)**
*a) If* $\varphi \equiv \psi$ *then* $\varphi \approx \psi$.
*b) If* $\varphi \approx \psi$ *then* $\varphi;\mathtt{tt} \equiv \psi;\mathtt{tt}$.
*c)* $\varphi \approx \varphi;\mathtt{tt}$.
*d) Let* $\mathcal{I} \subseteq \mathbb{O}\mathrm{rd}$. $(\bigvee_{i\in\mathcal{I}} \varphi_i);\psi \equiv \bigvee_{i\in\mathcal{I}}(\varphi_i;\psi)$ *and* $(\bigwedge_{i\in\mathcal{I}} \varphi_i);\psi \equiv \bigwedge_{i\in\mathcal{I}}(\varphi_i;\psi)$
*e)* $\tau;\varphi \equiv \varphi \equiv \varphi;\tau$.
*f)* $q;\varphi \equiv q$ *for* $q \in \mathcal{P}$.

PROOF a) If $\varphi \equiv \psi$ then $[\![\varphi]\!]_\rho(\mathcal{S}) = [\![\psi]\!]_\rho(\mathcal{S})$ for every $\rho$ and every set of states $S \subseteq \mathcal{S}$, in particular $S = \mathcal{S}$. Therefore $\varphi \approx \psi$.
b) $[\![\varphi;\mathtt{tt}]\!]_\rho = [\![\varphi]\!]_\rho \circ [\![\mathtt{tt}]\!]_\rho = \lambda X.[\![\varphi]\!]_\rho(X) \circ \lambda X.\mathcal{S} = [\![\varphi]\!]_\rho(\mathcal{S})$ for any transition system with state set $\mathcal{S}$ and any $\rho$. But $\varphi \approx \psi$ means $[\![\varphi]\!]_\rho(\mathcal{S}) = [\![\psi]\!]_\rho(\mathcal{S})$ and therefore

$\varphi;\mathtt{tt} \equiv \psi;\mathtt{tt}$.

c) Trivial.

d) $[\![(\bigvee_{i\in\mathcal{I}}\varphi_i);\psi]\!]_\rho = (\bigsqcup_{i\in\mathcal{I}}[\![\varphi_i]\!]_\rho)\circ[\![\psi]\!]_\rho = (\lambda X.\bigcup_{i\in\mathcal{I}}[\![\varphi_i]\!]_\rho(X))\circ[\![\psi]\!]_\rho = (\lambda X.\bigcup_{i\in\mathcal{I}}[\![\varphi_i]\!]_\rho([\![\psi]\!]_\rho(X))) = \bigsqcup_{i\in\mathcal{I}}([\![\varphi_i]\!]_\rho\circ[\![\psi]\!]_\rho) = [\![\bigvee_{i\in\mathcal{I}}(\varphi_i;\psi)]\!]_\rho$. The case of distributivity for conjunctions is similar.

e)–f) Trivial. ∎

**Theorem 28 (Bisimulation invariance)** *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ and $s,t \in \mathcal{S}$. If s and t are bisimilar, $s \sim t$, then for all closed $\varphi \in FLC$: $s \models \varphi$ iff $t \models \varphi$.*

Proof Let $\varphi \in$ FLC be closed. $\varphi$ is equivalent to a $\varphi'$ of infinitary FLC without fixed point operators and variables, using $\mu Z.\psi \equiv \bigvee_{\alpha\in\mathbb{O}\mathrm{rd}} Z^\alpha$ and $\nu Z.\psi \equiv \bigwedge_{\alpha\in\mathbb{O}\mathrm{rd}} Z^\alpha$. Note that each approximant has a finite representation. Lemma 27 c) says that $\varphi'$ is weakly equivalent to $\varphi';\mathtt{tt}$. Using parts d)–f) of Lemma 27, one can transform $\varphi';\mathtt{tt}$ into a formula $\alpha$ that does not contain $\tau$ and which is a (possibly infinitary) boolean combination of sequences of the form $q$ or $\langle a\rangle;\psi$ or $[a];\psi$ where $\psi$ again is of the described form. Every $\alpha$, obtained in such a way, is equivalent to an infinitary modal formula $q$ or $\langle a\rangle\psi$ or $[a]\psi$, where equivalence means being satisfied by the same states. But a formula of infinitary modal logic cannot distinguish between bisimilar states and weak equivalence preserves this property. ∎

An immediate consequence of Theorem 28 is the tree model property.

**Corollary 29 (Tree model property)** *FLC has the tree model property.*

**Theorem 30 (Approximants)** *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ be finite with $s \in \mathcal{S}, S \subseteq \mathcal{S}$.*
*a) $s \in [\![\mu Z.\psi]\!]^{\mathcal{T}}_\rho(S)$ iff $\exists k \le |\mathcal{S}|$, s.t. $s \in [\![Z^k]\!]^{\mathcal{T}}_\rho(S)$.*
*b) $s \in [\![\nu Z.\psi]\!]^{\mathcal{T}}_\rho(S)$ iff $\forall k \le |\mathcal{S}|$: $s \in [\![Z^k]\!]^{\mathcal{T}}_\rho(S)$.*

Proof a) The “if” part is trivial. For the “only if” part consider the general approximant characterisation of fixed point formulas. It implies the existence of a $\alpha \in \mathbb{O}\mathrm{rd}$ that makes $s \in [\![Z^\alpha]\!]_\rho(S)$ true. To show that it is bounded we introduce a new proposition $q_S$ s.t. $[\![q_S]\!]_\rho = \lambda X.S$. Then $s \in [\![\mu Z.\psi]\!]_\rho(S)$ iff $s \models (\mu Z.\psi);q_S$. According to Theorem 28, $(\mu Z.\varphi);q_S$ can be translated into a set $\{\varphi'_\alpha \mid \alpha \in \mathbb{O}\mathrm{rd}\}$ of formulas of infinitary modal logic. We show by induction on the fixed point depth of the formula at hand that finitary modal logic suffices.

Suppose $\varphi$ does not contain any $\sigma Y.\psi$. In this case every $\alpha_i$ is a formula of finitary modal logic. Consider now the function $f : \varphi'_\alpha \mapsto \varphi'_{\alpha+1}$ for every $\alpha \in \mathbb{O}\text{rd}$. $f$ is monotone since $\varphi'_{\alpha+1}$ arises from $\varphi'_\alpha$ by variable substitution and transformations that preserve equivalence. Then,

$$\emptyset \;=\; [\![\varphi'_0]\!] \;\subseteq\; [\![\varphi'_1]\!] \;\subseteq\; \ldots$$

Thus, if $s \in [\![\varphi'_k]\!]$ for some $k$ then $s \in [\![\varphi'_j]\!]$ for all $j \geq k$. Therefore $[\![\varphi'_{|\mathbb{S}|}]\!] = [\![\varphi'_j]\!]$ for all $j \geq |\mathbb{S}|$. This means that $s \models (\mu Z.\psi); q_S$ implies the existence of a $k \leq |\mathbb{S}|$ s.t. $s \models \varphi'_k$. But then $s \in [\![Z^k]\!](S)$.

Suppose now that $\varphi$ has fixed point depth $n+1$ and every $\sigma Y.\psi \in Sub(\varphi)$ has fixed point depth at most $n$ and can therefore be translated into a formula of finitary modal logic. Replacing every such $\mu Y.\psi$ in $\varphi$ by $\bigvee_{k=0}^{|\mathbb{S}|} Z^k$, and every $\nu Y.\psi$ with $\bigwedge_{k=0}^{|\mathbb{S}|} Z^k$ yields a formula $\varphi'$ of fixed point depth 1 that is equivalent to $\varphi$. The latter substitution uses part b) of the lemma on a smaller formula. The same argument as above holds now for translating $\mu Z.\varphi'$ into a sequence $\{\varphi''_k \mid k \leq |\mathbb{S}|\}$.

b) Here, the "only if" part is trivial. The "if" part is dual to the "only if" of part a). ■

In [MO99], Müller-Olm has shown that FLC model checking is undecidable for BPA processes already. We improve this result slightly.

**Theorem 31** *FLC model checking is undecidable for normed deterministic BPA.*

PROOF Based on an early result from language theory in [Fri76] it is shown in [GH94] that the simulation problem for deterministic normed BPA is undecidable. Given a BPA process $Q$ one can construct an FLC formula $\phi_Q^-$, s.t. $P \models \phi_Q^-$ iff $P$ simulates $Q$. The construction for arbitrary BPA processes is shown in [MO99] and works in particular for normed deterministic BPA. ■

## Modal $\mu$-Calculus

With FLC being defined it is possible to introduce Kozen's modal $\mu$-calculus $\mathcal{L}_\mu$, [Koz83], as a fragment of FLC. In $\mathcal{L}_\mu$ formulas the left argument of a sequential composition operator is always a modality. Conversely, modalities can only occur at these positions. The *syntax* of $\mathcal{L}_\mu$ is given by the following grammar.

$$\varphi \quad ::= \quad q \mid Z \mid \langle a \rangle;\varphi \mid [a];\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mu Z.\varphi \mid \nu Z.\varphi$$

where $q$ ranges over $\mathcal{P}$, $Z$ over $\mathcal{V}$ and $a$ over $\mathcal{A}$. Note that $\tau$ is not an $\mathcal{L}_\mu$ formula.

Since sequential composition in $\mathcal{L}_\mu$ formulas is only used in this restricted form we omit the semicolon and write $\langle a \rangle\varphi$ and $[a]\varphi$ instead.

**Theorem 32** [MO99] *FLC is strictly more expressive than* $\mathcal{L}_\mu$.

In fact, the FLC formulas of Examples 22 and 25 are not expressible in $\mathcal{L}_\mu$. This is because $\mathcal{L}_\mu$ can only express properties that are definable in the bisimulation invariant fragment of Monadic Second-Order Logic, [JW96]. However, these properties are "regular" in the sense that the language of strings formed by the actions along paths of a model for a $\mathcal{L}_\mu$ formula is regular. However, the formulas of Example 22 and 25 express context-free properties. An attempt to measure FLC's exact expressive power has been made in [Lan02a] showing that on linear models FLC can express exactly those properties that are definable by alternating context-free $\omega$-grammars with a parity generation condition. There are context-sensitive properties which cannot be defined in FLC unless PTIME=PSPACE.

The definition of subformulas and alternation depth for a $\mathcal{L}_\mu$ formula can easily be derived from the definitions for FLC formulas. We do not include them here since the following chapters do not contain games for $\mathcal{L}_\mu$. As for FLC, $\mathcal{L}_\mu^k$ denotes the fragment of $\mathcal{L}_\mu$ which contains formulas of alternation depth at most $k$ only. We only use $\mathcal{L}_\mu$ to link together the modal and temporal logics that we introduce games for.

For an introduction to $\mathcal{L}_\mu$ see [Koz83]. Model checking games for $\mathcal{L}_\mu$ have been defined in [Sti95] already. $\mathcal{L}_\mu$'s satisfiability problem was addressed in [NW97] in a game-based way.

CTL and PDL can easily be embedded into $\mathcal{L}_\mu$. Thus, games for these logics could be obtained from the corresponding games for $\mathcal{L}_\mu$ applied to translated formulas. The PDL and CTL model checking games of Chapter 4 and Section 5.3 are essentially the same as the $\mathcal{L}_\mu$ model checking games from [Sti95] with the following translations.

The translation $t$ : PDL$\rightarrow \mathcal{L}_\mu^0$ is defined inductively as follows.

$$
\begin{array}{lcl}
t(q) & = & q \\
t(Y) & = & Y \\
t(\varphi \vee \psi) & = & t(\varphi) \vee t(\psi) \\
t(\varphi \wedge \psi) & = & t(\varphi) \wedge t(\psi) \\
t(\langle a \rangle \varphi) & = & \langle a \rangle t(\varphi) \\
t([a]\varphi) & = & [a]t(\varphi) \\
t(\langle \alpha \cup \beta \rangle \varphi) & = & t(\langle \alpha \rangle \varphi) \vee t(\langle \beta \rangle \varphi) \\
t([\alpha \cup \beta]\varphi) & = & t([\alpha]\varphi) \wedge t([\beta]\varphi) \\
t(\langle \alpha ; \beta \rangle \varphi) & = & t(\langle \alpha \rangle \langle \beta \rangle \varphi) \\
t([\alpha ; \beta]\varphi) & = & t([\alpha][\beta]\varphi) \\
t(\langle \psi ? \rangle \varphi) & = & t(\psi) \wedge t(\varphi) \\
t([\psi ?]\varphi) & = & t(\overline{\psi}) \vee t(\varphi) \\
t(\langle \alpha^* \rangle \varphi) & = & \mu Y. t(\varphi) \vee t(\langle \alpha \rangle Y) \\
t([\alpha^*]\varphi) & = & \nu Y. t(\varphi) \wedge t([\alpha] Y)
\end{array}
$$

where $a$ is an atomic program and $q \in \mathcal{P}$. Note that PDL's syntax does not contain variables. But since the translation introduces variables in the scope of a modality the translation function must be defined for them as well.

CTL does not distinguish different action names. Therefore we use the abbreviations

$$\langle - \rangle \varphi \;:=\; \bigvee_{a \in \mathcal{A}} \langle a \rangle \varphi \qquad \text{and} \qquad [-]\varphi \;:=\; \bigwedge_{a \in \mathcal{A}} [a]\varphi$$

The translation $t$ : CTL $\rightarrow \mathcal{L}_\mu^0$ is given by

$$
\begin{array}{lcl}
t(q) & = & q \\
t(\varphi \vee \psi) & = & t(\varphi) \vee t(\psi) \\
t(\varphi \wedge \psi) & = & t(\varphi) \wedge t(\psi) \\
t(\mathtt{AX}\varphi) & = & [-]t(\varphi) \\
t(\mathtt{EX}\varphi) & = & \langle - \rangle t(\varphi) \\
t(\mathtt{A}(\varphi \mathtt{U} \psi)) & = & \mu Z. t(\psi) \vee (t(\varphi) \wedge \langle - \rangle \mathtt{tt} \wedge [-]X) \\
t(\mathtt{E}(\varphi \mathtt{U} \psi)) & = & \mu Z. t(\psi) \vee (t(\varphi) \wedge \langle - \rangle X)
\end{array}
$$

$$t(\mathtt{A}(\varphi\mathtt{R}\psi)) = \nu Z.t(\psi) \wedge (t(\varphi) \vee (\langle - \rangle \mathtt{tt} \wedge [-]X))$$
$$t(\mathtt{E}(\varphi\mathtt{R}\psi)) = \nu Z.t(\psi) \wedge (t(\varphi) \vee \langle - \rangle X)$$

The $\langle - \rangle \mathtt{tt}$ formulas are needed to take into account the fact that CTL unlike $\mathcal{L}_\mu$ is interpreted over total transition systems only. They require each state in which the formula is examined to have a successor state.

CTL$^*$ can be translated into $\mathcal{L}_\mu^1$, [Dam94]. However, this translation is not as simple as the two above since it does not map subformulas to subformulas. Therefore, we will not try to compare the CTL$^*$ model checking games that will be presented in Chapter 5 to the $\mathcal{L}_\mu$ model checking games applied to translated formulas. Consequently, we will not present this translation here.

## 2.6 Games

The games of the following chapters are played by two *players*, called $\forall$ and $\exists$. Other usual names for them are I and II, *Abelard* and *Eloise*, *refuter* and *verifier*, *opponent* and *player*, *pessimist* and *optimist*, etc. We will write $p$ to denote either of them, and $\overline{p}$ to denote $p$'s opponent. Furthermore, we will use personal pronouns according to the genders of *Abelard* and *Eloise*, i.e. player $\forall$ will be male and player $\exists$ will be female.

**Definition 33** A *game* $\mathcal{G}$ is a quintuple $(\mathcal{C}, \lambda, C_0, \mathcal{P}, W)$ where

- $\mathcal{C}$ is a set of *configurations*, also known as the *game board*,
- $\lambda : \mathcal{C} \to \{\forall, \exists\}$ assigns to each configuration a player, namely the one who makes the next move,
- $C_0$ is the starting configuration,
- $\mathcal{P}$, the set of *plays*, is a prefixed closed set of finite and infinite sequences of configurations starting with $C_0$. A play $P$ is called *finished* if it is maximal in $\mathcal{P}$, i.e. there is no $P' \in \mathcal{P}$ s.t. $P$ is a genuine prefix of $P'$.

- $W$ assigns to each finished play a winner, i.e.

$$W : \mathcal{P} \to \{\forall, \exists\}, \quad W(P) = \texttt{undef} \text{ iff } P \text{ is not finished}$$

Even though according to this definition one of the players formally has a choice in the last configuration of a finished play this choice can be ignored since there is nothing to be chosen.

Prefix closure makes it possible to regard a game as a tree, with a play being a branch in this tree.

It is more convenient to use a slightly specialised definition for the games in this thesis. For example, it is possible to finitely represent the plays and winning assignments.

We will usually introduce a game as a quadruple $(\mathcal{C}, C_0, \mathcal{R}, W)$ where $\mathcal{C}$ is the set of configurations as above with $C_0$ being the starting configuration. $\mathcal{R}$ is a finite set of *rules* which determines $\lambda$ and $\mathcal{P}$ from above. $W$ is a finite set of *winning conditions* which replaces the winning assignment above.

In this notation, a play is a maximal finite or infinite sequence of configurations $C_0, C_1, \ldots$ iff

- $C_0$ is the starting configuration, and
- every pair $(C_i, C_{i+1})$ is an instance of a rule $r \in \mathcal{R}$.

The winner of a play is determined by the finite set $W$ of winning conditions. Each condition is a scheme of a play, i.e. a play either fulfills a winning condition or not. It is part of the correctness proof of the games to show that every play fulfils at least one condition and that there is no play which fulfils two conditions that assign different winners to the play.

**Definition 34** The *game graph* of $\mathcal{G} = (\mathcal{C}, C_0, \mathcal{R}, W)$ is a directed graph $(V, E)$ whose set of nodes is the set of configurations of $\mathcal{G}$, i.e. $V = \mathcal{C}$. Edges in the game graph are given by

$$(C, C') \in E \quad \text{iff} \quad (C, C') \text{ is an instance of a rule } r \in \mathcal{R}$$

The *game tree* is the tree of all plays, and is also obtained as the unravelling of the game graph.

**Definition 35** A game $\mathcal{G} = (\mathcal{C}, C_0, \mathcal{R}, W)$ is called *finite* if the underlying set of configurations $\mathcal{C}$ is finite, $|\mathcal{C}| < \infty$.

It is called of *perfect information* if at every moment of the game both players have full knowledge about the actual configuration and the history of the play. This means their strategies can make use of the entire history.

Note that our definition of games does not allow hiding of information. We do not formalise this since all the games in this thesis are of perfect information. All of them are finite provided that underlying transition systems are finite apart from the ones of Section 9.2.

The definition of the winner of a play gives rise to the winner of a game: player $p$ is said to win a particular game $\mathcal{G}$ if $p$ can enforce a play that is winning for herself. In other terms, winning a game is short-hand for having a winning strategy for that game. Note the crucial difference between winning a play and winning a game.

**Definition 36** A *winning strategy* for $p$ in a game $\mathcal{G} = (\mathcal{C}, \lambda, C_0, \mathcal{P}, W)$ is a partial function $\eta : \mathcal{C}^+ \to \mathcal{C}$ satisfying

- if $(C_0, \ldots, C_n) \in \mathcal{P}$ and $\lambda(C_n) = p$ then $\eta(C_0, \ldots, C_n)$ is defined, and
- if $p$ always chooses $\eta(C_0, \ldots, C_n)$ at this moment then he or she wins every possible resulting play regardless of their opponent's moves.

If $p$ has a winning strategy $\eta$ for $\mathcal{G}$ then the *game tree for player* $p$ is derived from the full game tree in the following way.

- For every finite prefix $C_0, \ldots, C_n$ of a play s.t. $\lambda(C_n) = p$ discard all subtrees except the one starting with $\eta(C_0, \ldots, C_n)$.
- Retain all other nodes.

The game tree for player $p$ can be seen as either a determinisation of $p$'s role in the game, or a representation of the winning strategy $\eta$.

A class of games has the property of *determinacy* if for every possible game of this class one of the players has a winning strategy. Note that by definition at most one can have a winning strategy. *Zermelo's Theorem*, an important general theorem that proves determinacy for most games of the following chapters is one of the earliest results in game theory. Actually, Zermelo was concerned with the question of whether or not there is a winning strategy for a chess player.

**Theorem 37** [Zer13] *Let $\mathcal{G}$ be a 2-player game of perfect information, s.t. every play is of finite length and has a unique winner. Then one of the players has a winning strategy for $\mathcal{G}$.*

Much stronger results have been found since, mostly relaxing the requirement that plays can only be of finite length. See the Gale-Stewart Theorem, [GS53], and Martin's Theorem, [Mar75], for example.

We introduce two different types of games: *model checking games* and *satisfiability games*. A model checking game $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ is played on the set of states of an LTS $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with $s \in \mathcal{S}$, and the set of subformulas of a formula $\varphi$ of one of the logics introduced in Sections 2.4 and 2.5. It is player $\forall$'s task to show that $\mathcal{T}, s \not\models \varphi$ whereas player $\exists$ tries to show that $\mathcal{T}, s \models \varphi$.

A satisfiability game $\mathcal{G}(\varphi)$ is played on the set of subformulas of $\varphi$. Player $\forall$ attempts to show that $\varphi$ is not satisfiable whereas player $\exists$'s task is to show that $\varphi$ is satisfiable.

The goal of the following chapters is to characterise model checking and satisfiability checking problems for the logics of Sections 2.4 and 2.5 in a game-theoretic way. This means the rules and winning conditions of the games need to be defined such that a player has a winning strategy for a particular game iff the semantical property he or she intends to show is true.

In general the correctness proofs split up into two parts: soundness and completeness. A class of games for a logic and possibly a class of structures is sound if, whenever player $\exists$ wins a game then the corresponding semantical property holds. This is equivalent to saying that player $\forall$ wins if the semantical property fails. It is complete if the converse holds: player $\exists$ wins if the semantical property is true.

**Definition 38** A class of model checking games for a logic $\mathcal{L}$ and a class of structures $\mathcal{K}$ is closed under *dual games* if the logic is closed under negation and

- for every rule that requires player $p$ to make a choice on a formula $\varphi$ there is a *dual rule* in which player $\overline{p}$ makes a choice on the negated formula $\overline{\varphi}$, and
- for every winning condition for player $p$ there is a *dual winning condition* for player $\overline{p}$ s.t. every occurrence of a formula $\varphi$ is replaced by its negation $\overline{\varphi}$.

Then for every $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ of this class of games there is the dual game $\mathcal{G}_{\mathcal{T}}(s,\overline{\varphi})$ for the negated formula.

**Theorem 39** ***(Duality principle)*** *Let $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ be a model checking game of a class of games which is closed under dual games. Player $p$ wins $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ iff player $\overline{p}$ wins the dual game $\mathcal{G}_{\mathcal{T}}(s,\overline{\varphi})$.*

PROOF Suppose player $p$ wins $\mathcal{G}_{\mathcal{T}}(s,\varphi)$, i.e. there is a strategy for $p$ which enforces a winning play for $p$ regardless of $\overline{p}$'s choices. Then $\overline{p}$ can use this strategy in the game $\mathcal{G}_{\mathcal{T}}(s,\overline{\varphi})$ because whenever he has to make a choice then by duality there is a corresponding rule which requires $p$ to make a choice in $\mathcal{G}_{\mathcal{T}}(s,\varphi)$. This way, regardless of player $p$'s choices, he is able to enforce a winning play for himself, namely one that is dual to a winning play for $p$ in $\mathcal{G}_{\mathcal{T}}(s,\varphi)$. Thus, he or she wins $\mathcal{G}_{\mathcal{T}}(s,\overline{\varphi})$. ■

## Fixed Point Constructs and Unfolding

The way fixed point operators are handled in the games of the following chapters is called *unfolding*. Whenever a fixed point construct occurs it is simply replaced by its defining equation. This is *locally* correct because a fixed point can by definition be replaced with its defining equation without changing its semantics. However, *global correctness* must also be obeyed which distinguishes least from greatest fixed points.

If a fixed point construct $X$ gets replaced by a formula $f(X)$ then at some point later in a play $X$ can occur again since game rules follow the syntactical structure of formulas. In this case we call $X$ *regenerating* if its second occurrence stems from the unfolding. Note that sometimes configurations are sets and $X$ could get unfolded but then disappear since the play might follow another path in the syntax tree of $f(X)$. On the other hand, $X$ might appear in a later configuration again if it occurs as a subformula of another formula in there. In this case $X$ is not regenerating.

A least fixed point is only found if the corresponding construct is not unfolded infinitely often. Suppose it is, i.e. there is an infinite play in which a certain configuration $C$ occurs infinitely often. Moreover, suppose $C$ features a least fixed point construct. Since the game rules follow the syntactical structure of formulas and fixed point constructs are unfolded the situation at hand can be interpreted in the following way.

The truth value of the least fixed point construct in a certain context depends on itself. Note that the context is given by everything else in $C$ which can be other formulas, a state of a transition system, etc. In other words, the truth value of the $j$-th unfolding of the construct is actually determined by the $i$-th unfolding where $i < j$. This argument can be iterated down the sequence (2.2). At the end of this sequence there is the bottom element $\bot$ of the underlying lattice. For model checking and satisfiability checking, the lattices can be regarded as boolean in some way, i.e. the $\bot$ is usually the boolean *false*. Thus, an infinite unfolding of a least fixed point construct indicates that, regardless of where one starts in the sequence (2.2), it is always $\bot$ that determines the truth value of the fixed point construct. Hence, it is not fulfilled.

The same argument applied to greatest fixed points shows that an infinite unfolding corresponds to the construct being true in the actual context since the top element $\top$ will be the boolean *true* in some way.

Greatest fixed points are in every way dual to least fixed point. Thus, in order to refute a property described in terms of an explicit or implicit greatest fixed point constructor one must eventually leave the unfolding.

This has consequences for model checking and satisfiability checking games. Depending on the nature of configurations of a game, one of the players will have the task to explicitly show the regeneration of a least or greatest fixed point construct. For instance, if configurations are sets of formulas that are interpreted conjunctively then player $\forall$ will win if he is able to show the regeneration of a least fixed point in this set. If there is a regenerating one then it will be false according to the argumentation above. By the nature of these configurations they will be false which is what player $\forall$ wants to show. On the other hand, regenerating greatest fixed points are uninteresting in such a situation since they are fulfilled which does not determine the truth value of a conjunction.

## 2.7 Winning Strategies

The *history* of a prefix $C_0,\dots,C_n$ of a play which is in the actual configuration $C_n$ is the sequence $C_0,\dots,C_{n-1}$. Remember that in general a *winning strategy* is a partial function $\eta : \mathcal{C}^+ \to \mathcal{C}$. A winning strategy $\eta$ is called *history-free* iff for all sequences $C_0,C_1,\dots,C_n$ and $C_0,C_1',\dots,C_m'$ and configurations $C$: $\eta(C_0,C_1\dots,C_n,C) = \eta(C_0,C_1',\dots,C_m',C)$. Thus, it can be seen as a function of the type $\eta : \mathcal{C} \to \mathcal{C}$ since the player's choices only depend on the actual configuration.

If winning strategies for a game are history-free, then game trees can be represented as a graph. The graph representation simply results from the tree representation by identifying nodes in the tree that represent syntactically equal configurations. Since the winning strategies for the underlying game are assumed to be history-free, the winning player's choices only depend on the actual configuration. Thus, the choices are always the same regardless of the position in the tree. The choices made by the loser of the game are all preserved anyway.

This has an important consequence for finite games. In this case the graph representation of a winning strategy is always finite even though a tree representation of the same winning strategy might be infinite. If this is the case then winning conditions can be simplified. A play of the form $C_0,\dots,C_n,\dots,C_m,\dots$ with $C_n = C_m$ can be terminated after $C_m$ since the winner of this play is already determined at this moment.

When considering a game as a tree, namely its game tree, the notion of a *subgame* comes for free. It is given by a subtree in the whole game tree. As well as a game can be composed of several subgames, a strategy for a game can be composed of strategies for subgames in a natural way if they are history-free.

Suppose the set of configurations $\mathcal{C}$ of a game is partitioned into $\mathcal{C}_1, \mathcal{C}_2, \ldots$ s.t. each $\mathcal{C}_i$ represents a subgame. Moreover, suppose there are strategies $\eta_1, \eta_2, \ldots, \eta_i : \mathcal{C}_i \to \mathcal{C}$, i.e. a strategy can require a player to move into another subgame. Then they extend to a strategy $\eta : \mathcal{C} \to \mathcal{C}$ by

$$\eta(C) = C' \quad \text{iff} \quad C \in \mathcal{C}_i \text{ for some } i \text{ and } \eta_i(C) = C'$$

**Fact 40** *The union of history-free strategies is a history-free strategy.*

This thesis features history-free as well as history-dependent games. However, in the latter case the contained games are not fully history-dependent in the sense that one of the player's choices depends on more than the actual configuration but not on the entire history of the play so far. They depend on a finite amount of information about the history of a play.

In fact, the history-dependence is even more restricted. The player's choices only depend on the order in which a finite number of configurations has been visited, but not on the number of times a certain configuration occurred in the play. This idea is captured by the definition of a *latest visitation record* LVR, [McN93, GH82]. For a set $I \subseteq \mathcal{C}$ of "interesting configurations", at any moment of the play, it contains the order in which the elements of $I$ have appeared in the history of the play.

**Definition 41** Let $\mathcal{C}$ be the set of configurations of a game with $I \subseteq \mathcal{C}$. A *LVR* over $I$ is a sequence $l = C_1, \ldots, C_n$ of configurations with

- $C_i \in I$ for all $i = 1, \ldots, n$, and
- $C_i \neq C_j$ for all $1 \leq i < j \leq n$, and
- $n \leq |I|$.

Let $\mathfrak{I}$ denote the set of all LVRs over $I$. An *LVR winning strategy* is a strategy $\eta : \mathcal{C} \times \mathfrak{I} \to \mathcal{C}$ that is winning in the above sense.

## 2.8 Algorithms

The games introduced in the following chapters characterise the model checking and satisfiability checking problem for various logics (and classes of transition systems). This means they provide results like "$\varphi$ is satisfiable iff player $\exists$ wins the game associated with $\varphi$". The games alone do not yield an automatic procedure to check satisfiability of a formula for example. However, the soundness and completeness results of the next chapters can be used to construct algorithms which decide the winner of a game and thus solve the logical problem.

We assume the reader to be familiar with the notion of a deterministic and a nondeterministic Turing Machine, and thus the basic complexity classes $\mathrm{DTIME}(t(n))$, $\mathrm{DSPACE}(s(n))$, $\mathrm{NTIME}(t(n))$ and $\mathrm{NSPACE}(s(n))$. For technical definitions and an introduction see [Pap94]. The complexity classes that will be mentioned here are defined using these basic classes in the following way.

$$
\begin{array}{lcl}
\mathrm{LINTIME} & := & \bigcup_{k\in\mathbb{N}} \mathrm{DTIME}(k\cdot n) \\
\mathrm{PTIME} & := & \bigcup_{k\in\mathbb{N}} \mathrm{DTIME}(n^k) \\
\mathrm{NP} & := & \bigcup_{k\in\mathbb{N}} \mathrm{NTIME}(n^k) \\
\mathrm{PSPACE} & := & \bigcup_{k\in\mathbb{N}} \mathrm{DSPACE}(n^k) \\
\mathrm{NPSPACE} & := & \bigcup_{k\in\mathbb{N}} \mathrm{NSPACE}(n^k) \\
\mathrm{EXPTIME} & := & \bigcup_{k\in\mathbb{N}} \mathrm{DTIME}(2^{k\cdot n}) \\
\mathrm{EXPSPACE} & := & \bigcup_{k\in\mathbb{N}} \mathrm{DSPACE}(2^{k\cdot n}) \\
\text{2-EXPTIME} & := & \bigcup_{k\in\mathbb{N}} \mathrm{DTIME}(2^{(2^{k\cdot n})})
\end{array}
$$

Furthermore, the class $\Delta_2$ of the so-called polynomial-time hierarchy consists of all problems that can be solved in polynomial time by a deterministic Turing Machine with an NP oracle. See [Sto76] for further details on the polynomial-time hierarchy.

Note that PSPACE = NPSPACE according to [Sav69].

Another class that will be mentioned in Chapter 9 is co-NP. In general, the co-class of a complexity class $\mathcal{C}$ contains all the complements of languages in $\mathcal{C}$.

$$
\text{co-}\mathcal{C} \quad := \quad \{\, L \mid \overline{L} \in \mathcal{C} \,\}
$$

### Alternating Algorithms

The theory of *alternating algorithms* proves to be helpful for analysing the complexities of game-based algorithms. Remember that nondeterministic algorithms are allowed to guess the next right step in a computation, i.e. the one that leads to an accepting configuration. Co-nondeterministic algorithms, also called *universal*, have the ability to guess the next wrong step, i.e. the one that leads to a rejecting configuration. An alternating algorithm can do both. It is nothing more than a game played by two players of which one tries to reach an accepting configuration in a Turing Machine's computation by choosing successor configurations of existential ones. The other player tries to reach a refuting configuration by choosing successors of universal configurations. This gives rise to the basic complexity classes ATIME($t(n)$) and ASPACE($s(n)$). Classes like APTIME or APSPACE are constructed just like their deterministic counterparts.

Alternating algorithms and the corresponding complexity classes have been studied in [CKS81]. The results concerning us are the relationships between alternating and deterministic complexity classes. If a problem can be decided by an alternating algorithm using time $f(n)$, then it can be decided by a deterministic algorithm using space $(f(n))^2$. On the other hand, alternating space $f(n)$ can be embedded into deterministic time $2^{O(f(n))}$. Similar results hold for the converse inclusions. This yields the following useful equalities of complexity classes.

$$\begin{array}{lcl} \text{ALOGSPACE} & = & \text{PTIME} \\ \text{APTIME} & = & \text{PSPACE} \\ \text{APSPACE} & = & \text{EXPTIME} \\ \text{AEXPTIME} & = & \text{EXPSPACE} \\ \text{AEXPSPACE} & = & \text{2-EXPTIME} \end{array}$$

We will make use of these results to give upper bounds on the complexity of deciding the winner of the games in the next chapters. The size of the input for a model checking algorithm will always be the number of states of the underlying transition system $\mathcal{T}$ and the size of the formula $\varphi$ where

$$|\varphi| \;:=\; |Sub(\varphi)|$$

Note that the number of subformulas of $\varphi$ is linear in its syntactical length.

### Local Algorithms

Regarding the model checking problem we distinguish between two different kinds of algorithms: *global* and *local* ones. A model checking algorithm is local if it fulfills two conditions:

1. It must be local with respect to the formula. This means it does not necessarily exploit the entire game graph of a game because the evaluation of disjunctions and conjunctions is *non-strict*: if a disjunct evaluates to true then the other can be ignored. The condition is dual for conjuncts.

2. It must be local with respect to the transition system. This means it avoids necessarily exploiting the entire game graph by constructing the transition system *on demand*. If the evaluation of a subformula on a successor state determines the truth value of a superformula on the predecessor state then other successor states are not examined anymore.

The second condition implies that the algorithm does not "jump" to arbitrary states in the transition system at hand. Any model checking algorithm not satisfying these two conditions will be called global.

This is just one definition of locality and by no means the only possibility.

For verification purposes local algorithms are desirable, [CVWY91]. Since the transition systems used there tend to be very large it is helpful to use algorithms that do not need to allocate space for the entire game graph at the beginning of their execution.

Note that an algorithm can be local and still construct a game graph completely. This is for example the case with universal properties for which there is no counterexample.

### The Subformula Property

Another requirement for model checking and satisfiability checking algorithms is the *subformula property*. In order to make these algorithms useful for verification purposes they should only work on subformulas of the input formula. Suppose player ∀'s winning strategy in a model checking game is used to illustrate that a transition system fails to have a certain property given by the input formula. The subformula property guarantees that the overall failure is linked to player ∀'s moves in the game. This is because the syntax and the semantics of formulas are defined inductively.

One way of defining games for the temporal logics introduced in Section 2.4 is to translate them into the modal $\mu$-calculus $\mathcal{L}_\mu$ as was shown in [Dam94] for CTL$^*$. Then, the $\mathcal{L}_\mu$ model checking games from [Sti95] can be applied to the translated formulas. This violates the subformula requirement and makes it hard for the user of a verification tool to understand why a certain property fails if the failure is demonstrated to the user by letting him play against the tool's winning strategy.

# Chapter 3

# Background

*I can't believe it! Reading*
*and writing actually paid off!*

—

HOMER J. SIMPSON

## 3.1 Tableaux

A *tableau* is simply a tree whose nodes are labelled in some way. The name suggests that originally they were table-like structures. When using tableaux to decide the model checking or satisfiability checking problem for modal and temporal logics the node labellings are usually formulas or sets of formulas or one of these plus states of a transition system. A branch of a tableau tree comes with the notion of being successful, and a tableau is successful if all its branches are.

Branches are sequences of configurations which are built from a set of rules in a very similar way to game rules. Usually, existential constructs in the underlying logic are reflected by choices in the tableau rules while universal constructs cause a branching in the tree. Fixed point constructs are unfolded and can potentially lead to infinite branches.

A tableau-based model checker or satisfiability checker attempts to build a successful tableau for a formula, resp. a formula and a state of a transition system.

The logical problem at hand is characterised by the question of whether there exists a successful tableau for a formula (and a state of a transition system). While a successful tableau witnesses a positive instance of the problem at hand, there is usually no witness for a negative answer. For example, a formula is satisfiable if there is a successful tableau for it. Hence, it is unsatisfiable if all possible tableaux fail to be successful. I.e. unsatisfiability is a universal property which in this way cannot easily be illustrated to hold.

Tableaux, as they are used nowadays, have two different roots, a syntactic and a semantic one. The history of syntactic tableaux dates back to the work of Gentzen who used tableaux for syntax-directed proofs in classical logics, [Gen35]. His work has been extended to modal logics, for example in [Cur52, Kan57].

Tableaux systems with a semantic flavour are rooted in the work of Beth who was also studying classical logics, [Bet55]. With the introduction of Kripke semantics, these tableau systems became interesting for modal logics as well, [Kri59]. Moreover, Hintikka structures, [Hin69], which are based on Smullyan's semantical tableaux, [Smu95], have been used to decide modal and temporal logics as well, [EH85].

These two routes merged later on when it was realised that they are essentially the same, [Zem73, Rau79, Fit83]. Nowadays, syntactic tableaux have their applications in proof theory while semantic tableaux are mostly used for automated reasoning, i.e. to decide whether a given formula is valid for instance.

Tableaux have been used to solve model checking and satisfiability checking problems for modal and temporal logics, [Gor99]. One of the reasons for the usefulness of tableaux for these logics is the tree model property which all the logics discussed in this thesis possess. [Pra80] for example used tableaux to decide satisfiability of PDL. A successful tableau basically incorporates a model for the formula at hand.

Recently, tableaux have also been used to decide satisfiability of LTL formulas, for example in [LP00] and [SGL97]. The latter construction uses intervals of points in a possible model for the formula. In contrast to this, the tableaux of [LP00] work on subformulas of the input formula only. They also are used to obtain a complete axiomatisation.

The advantage that tableau-based satisfiability checking offers is the close connection between the syntactical structure of the input formula and the tableau which witnesses a semantical property. Tableaux could be used to construct complete axiomatisations for various logics. This is because completeness of an axiom system is a connection between syntax and semantics: every consistent formula must be satisfiable, see Chapter 7 for details.

[EH85] gives a tableau-like decision procedure for satisfiability of CTL formulas and uses the tableaux to prove completeness of a certain axiomatisation. This usually involves a processing of the tableaux in order to construct a model for the formula at hand. Other tableau approaches to decide the satisfiability problem for branching time logics can be found in [EC82, BAPM83]. [Eme85] states that they are essentially the same together with the maximal model constructions of [VW86b, SVW83].

The completeness of PDL was shown in a similar way, based on the satisfiability tableaux from [Pra80], see [KT90] for details.

A tableau model checker for BLTL was given in [LP85]. Its running time is exponential in the size of the formula but linear in the size of the underlying transition system. This was the reason to believe that despite the relatively high complexity LTL model checking can efficiently be done since formulas tend to be small while a transition system usually forms the biggest part of the input to a model checker when the number of states is taken as the size of a transition system.

A local tableau model checker for CTL$^*$ was given in [BCG95]. In fact it is a model checker for BLTL which is not surprising since it has been observed that model checking for LTL and CTL$^*$ is basically the same. This means both problems are easily reducible to each other. A CTL$^*$ formula can be seen as a collection of BLTL formulas.

In general, a CTL$^*$ model checker has to determine whether a path quantified $Q\psi$ holds in a certain state $s$ of a transition system. This only depends on $s$ and $\psi$. Doing this inductively one can assume $\psi$ to be free of path quantifiers. Thus, in case of $Q = \mathtt{A}$ the input is a transition system and a BLTL formula. If $Q = \mathtt{E}$ then one can model check the BLTL formula $\mathtt{A}\overline{\psi}$ and negate the result to establish whether the state satisfies $\mathtt{E}\psi$. We will also make use of this observation in Section 5.2.

## 3.2 Automata

An *automaton* is a simple technical device that takes an input, runs on it and outputs either yes or no. According to Church's Thesis, Turing Machines are the most general automata. Several downgraded versions of them – usually called automata for short – have been defined since, mainly to capture levels of the Chomsky hierarchy algorithmically.

In the setting of linear time temporal logic one regards automata over strings or words. A string is a finite or infinite sequence of symbols over an alphabet $\Sigma$. Let $\Sigma^*$ denote the set of all finite strings over $\Sigma$ and $\Sigma^\omega$ the set of all infinite strings over $\Sigma$.

In general, an automaton consists of a finite set of states with a distinguished starting state, an acceptance condition, a memory and a transition function. Its behaviour is determined by the transition function which, applied to a current state, a position in the input string and the state of the memory, yields the next actual state, a possible change of the memory and a new position in the input word.

A *run* of an automaton is a sequence of configurations consisting of the actual state, the content of the memory and the position in the input word. An automaton accepts a word if, beginning in the starting state, the run induced by the transition function meets the acceptance condition. The set of all $w \in \Sigma^*$, s.t. automaton $\mathcal{A}$ accepts $w$ is called the *language accepted by* $\mathcal{A}$ and denoted $L(\mathcal{A})$. The same holds of course for $\Sigma^\omega$.

Several different acceptance conditions for automata on infinite structures that lead to different types of automata have been used. The most important ones are the following.

- *Büchi automata*. A run must visit at least one state of a given set infinitely often.

- *Rabin automata*. In a given set of pairs $(F_i, G_i)$, $i = 1, \ldots, n$, there is a pair of sets of states $(F_k, G_k)$ s.t. at least one state in $G_k$ is visited infinitely often while states in $F_k$ are only visited a finite number of times.

- *Streett automata*. In a given set of pairs $(F_i, G_i)$, $i = 1, \ldots, n$, every pair $(F_i, G_i)$ satisfies the following. A state in $G_i$ is visited infinitely often or all states in $F_i$ are only visited a finite number of times.

- *Muller automate*. For a given set of sets of states $F_1, \ldots, F_n$ there is an $i$ s.t. $F_i$ contains exactly those states which are visited infinitely often in a run.

- *Parity automata*. Each state is assigned a natural number called the parity index. The least index which is seen infinitely often in a run must be even.

Büchi used finite automata to obtain a decision procedure for *Second-Order Logic of One Successor Function* S1S, a generalisation of Presburger Arithmetic, [Pre29]. The acceptance condition of these memoryless automata is, as it is stated above, the existence of a certain state that is visited infinitely often in a run. In particular, he showed that for every formula $\phi$ of *Monadic Second-Order Logic with a Successor Relation over Infinite Strings* MSO$[<]$, there is a finite-state automaton $\mathcal{A}_\phi$ s.t.

$$L(\mathcal{A}_\phi) \;=\; \{\, w \in \Sigma^\omega \mid w \models \phi \,\}$$

An infinite string $w$ can be regarded as a mathematical structure whose universe is the set of natural numbers representing the positions of the string. Monadic predicates are interpreted as labels on the positions, resp. letters of the string.

Furthermore, the converse inclusion also holds. For every automaton there is a formula whose models are exactly the words accepted by the automaton. Thus, MSO$[<]$ defines exactly the regular languages.

Star-free languages, a genuine subclass of regular languages, were shown to coincide exactly with the class of languages definable in *First-Order Logic* with a successor relation over strings FO, [Tho79]. In [Kam68] it was shown that LTL with past operators is expressive complete, i.e. that it defines exactly those properties that are expressible in FO. This is based on the observation that an infinite path $\pi = s_0 s_1 \ldots$ of an LTS where the $s_i$ are labelled with elements of $\mathcal{P}$ is nothing more than an infinite string over the alphabet $2^{\mathcal{P}}$.

## Automata for Linear Time Temporal Logic

Since all these results relating to automata and temporal formulas are constructive then finite-state automata can be used to decide the model checking and satisfiability checking problem for LTL. Given a $\varphi \in$ LTL one can build the corresponding automaton $\mathcal{A}_\varphi$ that accepts exactly the models of $\varphi$. Model checking for a word $w$ and $\varphi$ is done by testing whether the run induced by $w$ is accepting. Satisfiability checking is done by testing whether the language accepted by $\mathcal{A}_\varphi$ is non-empty.

In order to decide LTL the nondeterministic version of these *Büchi-automata* proved to be helpful, [VW86a, SVW83]. These guess truth values of subformulas at every position of the path, and their transition function is used to check whether the guesses are correct. The non-emptiness problem for these automata can be decided using polynomial space, [Var96].

To do model checking for BLTL, i.e. to test whether all paths of a given total transition system satisfy a linear time formula, the transition system is interpreted as a Büchi-automaton $\mathcal{T}$ as well. This is done by regarding every state as final. Hence, every run of the transition system is an accepting one since it necessarily visits a final state infinitely often.

$\mathcal{T}$ is then paired with the automaton for the negation of the input formula. The product automaton $\mathcal{T} \times \mathcal{A}_{\overline{\varphi}}$ simulates runs of $\mathcal{T}$ and $\mathcal{A}_{\overline{\varphi}}$ in parallel. Model checking BLTL is then reduced to checking for language inclusion between these automata which is nothing more than an emptiness test on the product automaton.

$$\text{for all paths } \pi \text{ of } \mathcal{T}: \; \pi \models \varphi \quad \text{iff} \quad L(\mathcal{T}) \subseteq L(\mathcal{A}_\varphi) \quad \text{iff} \quad L(\mathcal{T} \times \mathcal{A}_{\overline{\varphi}}) = \emptyset$$

Again, this is possible using polynomial space.

The translation from LTL formulas into nondeterministic Büchi-automata can yield automata that are exponentially larger than the formula at hand. However, the non-emptiness problem for Büchi-automata is just NLOGSPACE-complete, [VW94].

On the other hand, [Var96] gives a linear translation from LTL formulas into *alternating Büchi-automata*.

A nondeterministic automaton allows existential choices in its transitions. Technically, the transition function is in fact a relation. Thus, an input word generally induces several runs. The acceptance condition then quantifies over these runs existentially, i.e. a word is accepted by the automaton if there is an accepting run.

It is easy to imagine universal quantification which results in co-nondeterministic automata. Alternating automata on the other hand allow both choices on the level of a transition. This means there are some configurations which are accepting iff there is a successor configuration which is accepting, and others which are accepting iff all possible transitions lead to an accepting configuration. The run of an alternating automaton is a tree of configurations which is nothing more than a system of boolean equations. It is accepting iff the corresponding system has a solution. For this correspondence configurations are regarded as boolean variables, nondeterministic choices are translated into disjunctions while co-nondeterminism is modelled by conjunctions.

This approach is more natural when translating formulas into automata since they usually feature existential and universal constructs. Thus, it is not surprising that the translation from LTL into alternating Büchi-automata given in [Var96] basically follows the syntactical structure of the formula. Disjunctions are translated into nondeterministic choices, conjunctions into co-nondeterministic ones, fixed point constructs are unfolded, etc.

Since these automata are more succinct, their non-emptiness problem is expected to be harder than the one for nondeterministic Büchi-automata. [Var96] showed that it is PSPACE-complete.

The route via alternating automata promises better efficiency than the one using nondeterministic automata because the costly operation is applied after the cheap one: emptiness test after translation. For nondeterministic automata the former is easy while the latter is hard. Generally, the composition of an exponential function with a polynomial yields a function which is asymptotically worse than a polynomial composed with an exponential function. Consider for example the two functions

$f(n) = 2^n$ and $g(n) = n^2$. Then $g \circ f = o(f \circ g)$ because $(g \circ f)(n) = (2^n)^2 = 2^{2 \cdot n}$ and $(f \circ g)(n) = 2^{n^2}$.

### Automata for Branching Time Temporal and Modal Logics

For branching time logics as well as modal logics, automata over strings are not applicable. This is simply because models of branching time formulas are not strings. However, the general idea of using automata to decide the model checking and satisfiability checking problem for these logics is still admissible. The right machinery in this case are automata over trees.

Rabin noticed that Büchi's work on Monadic Second-Order Logic with One Successor Function can be extended to MSO with several successor relations. This is the natural logical framework for trees instead of strings where sons of a node are considered to be ordered different successors of the node.

The technicalities for automata over strings carry over to automata over trees. $\Sigma^*$, resp. $\Sigma^\omega$, gets replaced by the set of all finite, resp. infinite, trees with nodes labelled by $\Sigma$. The run of an automaton is necessarily a tree of configurations and is accepting if some condition on its branches is met.

[Rab69] showed that finite state automata over trees accept exactly those languages of trees that can be defined by MSO with several successor relations. This defines a notion of a *regular* tree languages. Similar to LTL's expressive completeness result a fragment of MSO with several successors could be identified that coincides exactly with CTL$^*$. [HT87] showed that CTL$^*$ can be translated into *Monadic Path Logic over infinite binary trees* MPL and vice-versa. The name "Path Logic" indicates that this fragment of MSO allows second-order quantification over paths only. Later, it could be shown that the requirement of regarding binary trees only can be relaxed, but then CTL$^*$ only corresponds to the bisimulation-invariant fragment of MPL, [MR99].

Using the observation that a CTL$^*$ formula is simply a collection of existentially and universally path quantified linear time formulas one can extend the approach taken for LTL to CTL$^*$. The first attempts to use tree automata for testing satisfiability of a CTL$^*$ formula yielded a decision procedure whose running time is quadruple exponential in the size of the formula. The reason for the high complexity is the need to determinise automata in an inductive process and the testing for non-emptiness. Determinisation is necessary because of the following.

Consider two paths with a finite common prefix and a CTL$^*$ formula $\mathtt{A}\varphi$. Even if $\varphi$ holds on both paths the automaton in general has to guess which path it is going to follow while it is still processing the common prefix.

One of the four nested exponents in this approach results from the translation of LTL formulas into nondeterministic automata over strings with an acceptance condition using pairs. Then using McNaughton's construction for the determinisation of these automata causes a double exponential blow-up of the automaton's size, [Büc62, McN66]. Finally, checking for non-emptiness of these automata needs exponential time.

In several attempts the complexity could be pushed down to deterministic triple exponential time, [ES84], nondeterministic double exponential time, [Eme85], and finally deterministic double exponential time, [EJ00]. These optimisations exploit the fact that automata resulting as a translation from CTL$^*$ have a very special structure such that complementation and non-emptiness test can be done more efficiently than it is possible in the general case, [Saf88, MS95, Tho99]. In [VS85] it was shown that the last result is optimal.

The downside of this automata-theoretic approach is the fact that determinisation only preserves the semantical connection between a formula and an automaton. The syntactical relationship, however, is destroyed. This is the reason why automata for example are believed to be of no use in constructing a complete axiom system for CTL$^*$.

There is one thing that distinguishes the branching time from the linear time framework conceptually. For linear time logics the model checking problem as well as the satisfiability checking problem can be reduced to the inclusion problem for languages of infinite words. Remember that automata-based model checking for BLTL is done by checking language inclusion between two Büchi-automata.

In the branching time setting, the model checking problem cannot easily be reduced to a language inclusion problem. Instead, formulas are translated into automata that accept trees, i.e. to check whether a given tree satisfies a formula $\varphi$, one has to test for membership of the tree in the language accepted by the automaton corresponding to $\varphi$. This holds of course for general transition systems as well which can be unravelled to a tree.

$$\mathcal{T}, s \models \varphi \quad \text{iff} \quad \mathcal{R}_s(\mathcal{T}) \in L(\mathcal{A}_\varphi)$$

where $\mathcal{R}_s(\mathcal{T})$ is the unravelling of $\mathcal{T}$ with respect to the state $s$.

This conceptual difference has consequences regarding the efficiency of the automata-theoretic approach to branching time model checking.

The model checking problem for CTL for example is PTIME-complete. [CES83] gives a decision procedure that runs in linear time in both the size of the transition system and the formula. The satisfiability checking problem on the other hand is EXPTIME-complete.

The gap for CTL$^*$ is even wider: model checking is PSPACE-complete, [EL87], while satisfiability checking is 2-EXPTIME-complete. For CTL$^+$ the model checking problem was shown to be $\Delta_2$-complete, [LMS01]. The satisfiability problem is in 2-EXPTIME and EXPTIME-hard. There are known exponential lower bounds for the translation of CTL$^+$ formulas into CTL, [Wil99, AI01].

These results make the approach taken for linear time formulas seem unfeasible for branching time logics. Solving the model checking problem by building automata used for satisfiability testing cannot lead to optimal decision procedures unless the translation yields suitably small automata.

Another technical problem dates back to Rabin's work on tree automata. Remember that they were originally used to decide MSO with $n$ successor relations, $n \in \mathbb{N}$. Therefore, those tree automata work on trees in which every node has exactly $n$ sons. For satisfiability checking this is no impediment since all the logics discussed here preserve bisimulation. Thus, if a formula has a tree model in which every node has at most $n$ sons then it is also satisfied by the tree which results from the original one by duplicating subtrees such that every node has exactly $n$ sons. This is, however, not possible in model checking where the underlying tree is derived from a transition system in which states can have arbitrary and different out-degrees.

[KG96] uses automata with flexible transition relations that adapt to various out-degrees of a node of a transition system. It also uses *simultaneous trees* which allow different nodes of a path to be visited simultaneously. The CTL model checking problem is then reduced in linear time to the non-emptiness problem for these automata and trees which can be checked in quadratic time.

Later, [KVW00] used alternating automata over trees to decide the model checking problem for CTL. It is observed that the linear translation from CTL formulas into alternating automata yields automata of a special structure, so-called *weak alternating automata*, already identified in [MSS88]. The product of these with a transition system is an even more special structure, a so-called *hesitant alternating automaton*. Model checking CTL is then reduced to the 1-letter non-emptiness problem for these automata which can be decided in space linear in the size of the formula and polylogarithmic in the size of the transition system.

The same approach works for CTL$^*$ as well. Not surprisingly, the automata resulting from this translation admit a less efficient non-emptiness check. [VB00] describes how non-emptiness checking for these automata can be seen as a game which can be implemented efficiently.

[VW86b] showed that Büchi automata on infinite trees can be used to decide satisfiability of PDL formulas as well.

## 3.3 Games

In computer science games have often been used to provide an understandable account of a certain problem. It is maybe because games are part of everyday life that game-based solutions are considered accessible. In fact, many situations besides obvious games can be defined in terms of two players and the notion of a play which is won by one of them. An exam is nothing more than a game between the examiner and a candidate in which the candidate wins iff he or she is able to produce correct answers to at least half of the examiners questions regardless of what exactly they are.

Besides the area of logics in computer science, games have also proved useful in other fields like combinatorics, [Dem01], or programming languages semantics for example, [Abr97].

Probably the most commonly known example of a logical game is that of an Ehrenfeucht-Fraïssé game which is played on two mathematical structures in order to establish whether a formula of a certain logic can distinguish them from each other, [Ehr61, Fra54].

In Ehrenfeucht-Fraïssé games two players take turns in colouring or picking elements of one of the structures such that player $\exists$ always has to reply to player $\forall$'s moves in the other structure. The moves are designed for a certain logic $\mathcal{L}$ to obtain a result of the following form. Player $\exists$ wins the game of length $n$ on $\mathcal{K}_1$ and $\mathcal{K}_2$ iff for all $n$-tuples $\overline{k}_1$ of $\mathcal{K}_1$ and $\overline{k}_2$ of $\mathcal{K}_2$ and all formulas $\varphi(x_1,\ldots,x_n) \in \mathcal{L}$ with $n$ free variables

$$\mathcal{K}_1,\overline{k}_1 \models \varphi(x_1,\ldots,x_n) \quad \text{iff} \quad \mathcal{K}_2,\overline{k}_2 \models \varphi(x_1,\ldots,x_n)$$

Remember that a model checking game is played on a structure and a formula in order to establish whether the structure satisfies the formula. In this respect, an Ehrenfeucht-Fraïssé game can be seen as two model checking games that are synchronised on the formula component. If one of the model checking players makes a move in one structure then this is guided by the underlying formula. Thus, for the other structure to also (not) satisfy the formula at hand, the same move must be possible in the other structure.

Ehrenfeucht-Fraïssé games have mostly been used for classical logics like First- or Second-Order Logics and fixed point extensions of them. This is because they have become the main tool to separate logics from each other in terms of their expressive power. This is done by finding two structures such that player $\forall$ has a winning strategy for the game corresponding to one logic while player $\exists$ wins all the games corresponding to the other logic.

Separation results for these logics are important in Finite Model Theory since many complexity classes have logical characterisation. NP for example corresponds to the existential fragment of Second-Order Logic $\Sigma^1_1$, [Fag74], and on ordered structures PSPACE is characterised by First-Order Logic with Partial Fixed Points FO+PFP, [Imm82, AVV97], while First-Order Logic with Least Fixed Points FO+LFP captures PTIME, [Imm86, Var82]. For surveys see [Imm89] and [EF95].

Ehrenfeucht-Fraïssé games for modal and temporal logics have not been studied with such intensity. This might be because it is easier to obtain separation results for these logics in a direct way, see the section on their expressive powers at the end of this chapter.

On the other hand, Ehrenfeucht-Fraïssé games for logics with extremal fixed point constructs are interesting because they provide insight into the question about the differences between fixed point and general quantifiers. Ehrenfeucht-Fraïssé games for $\mathcal{L}_\mu$ can be found in [Sti96a]. For basic modal logics, i.e. modal logics without any recursion mechanism like fixed points in FLC or $\mathcal{L}_\mu$ or the Kleene-Star in PDL programs, Ehrenfeucht-Fraïssé games coincide with simple bisimulation games, [Sti96a].

In computer science, modal and temporal logics are widely used for program specification and verification purposes. Not surprisingly, games for these logics deal with problems arising in this area as well. Besides the model checking and satisfiability checking problem there is program synthesis for example, [Tho95].

The basis for most of the games in this thesis is [Sti95] where a game-based approach to $\mathcal{L}_\mu$'s model checking problem is presented. In these games the players essentially move one pebble through the underlying transition system and one through the syntax tree of the formula at hand. Moves are guided by the formula, and players do not necessarily take turns to move. The winner of a play is determined by an atomic formula, or a situation in which one of the players cannot move anymore, or a condition on the visited formulas in an infinite play.

This condition concerns the occurrence of fixed point constructs. In fact, the winner is decided by the fixed point type of the outermost variable occurring infinitely often. This is where the strength of games for modal and temporal logics can be seen. The winning condition is very natural to the underlying logic and not too hard to understand for those who are reasonably familiar with least and greatest fixed point in general.

Computationally, $\mathcal{L}_\mu$'s model checking problem is relatively hard since no polynomial time algorithm has been found for it so far. However, the problem's complexity is entirely captured by the task of finding a winning strategy for one of the players. Checking which player wins a particular play is easy. But it is the definition of a play rather than a game which provides understanding of the property expressed by a formula.

Games have also been used in a less direct way for two other problems concerning the logics dealt with here. [VB00] defines games to determine whether the language accepted by a CTL$^*$ model checking automaton is empty. [NW97] builds tableaux for $\mathcal{L}_\mu$ formulas and uses games to test whether particular branches of these tableaux are successful.

### Comparisons

Tableaux, automata and games are not entirely different techniques. Often, it is possible to turn one of them into another.

The easiest transition is made from games to tableaux. Given a model checking or satisfiability game as the ones in the following chapters, the game tree for player $\exists$ is nothing more than a tableau for a formula (and a state of a transition system). The duality property of the games automatically yields refuting tableaux. These are player $\forall$'s winning strategies. To define a tableaux system formally from a given game one would usually replace the notion of a play with a tree in a way that player $\exists$'s choices remain as they are while player $\forall$'s choices correspond to a branching in the tree. Thus, a play would be a branch of the resulting tree. A branch of this tree is successful iff it fulfils one of player $\exists$'s winning conditions.

The transition from tableaux to games is not much harder. Given a tableau system for a logic it can be turned into a game in which player $\exists$ chooses the form of successor configurations to the actual one while player $\forall$ selects the path to follow through the tableau. The notion of a successful branch must be translated into a winning condition for player $\exists$ while player $\forall$'s winning conditions need to be made complementary to them such that a branch is not successful iff it corresponds to a winning play for player $\forall$.

Often, alternating automata are seen as games. In fact, it is the non-emptiness test of the language accepted by an alternating automaton which is a 2-player game. The configurations are the automaton's states while player $\exists$'s winning conditions are derived from the automaton's acceptance condition. An accepting run can then be seen as a game tree for player $\exists$.

Conversely, player $\forall$'s game trees, i.e. witnesses for the failure of a property, are accepting runs of the dual automaton in which nondeterministic and universal choices are swapped, and the acceptance conditions are dualised. This is particularly easy if the automaton is of a type whose acceptance conditions are closed under complementation, for example Rabin or parity automata. This is why games correspond more closely to these kinds of automata rather than Büchi automata, [Eme96, EJS01].

Finally, there is a close connection between automata and tableaux as well. The transition table of an alternating automaton is in fact a tableau. Note that from an existential point of view an automaton chooses the next state nondeterministically if the actual one is existential, but spawns off copies that run simultaneously in a universal state. This corresponds exactly to the idea underlying tableau rules described above. See [Eme85] for details.

## 3.4 Overviews

### Model Checking

Figure 3.1 lists the most important publications in the area of model checking for the logics used in this thesis. $\mathcal{L}_\mu$ and $\mathcal{L}_\mu^0$ are included for the sake of completeness. Moreover, because of embeddings some of the results for $\mathcal{L}_\mu$ carry over to PDL for example. To the best of our knowledge automata-based model checking for PDL has not explicity been published but it can easily be obtained from automata for $\mathcal{L}_\mu^0$. The complexity remains the same.

The empty fields in the $\mathcal{L}_\mu^0$ row are due to the fact that tableaux or games for $\mathcal{L}_\mu$ can easily be simplified to yield tableaux and games for $\mathcal{L}_\mu^0$. Again, as far as we know

| logic | tableaux | automata | games | others |
|---|---|---|---|---|
| BLTL | [LP85]<br>[BCG95]<br>[GPVW95] | [SVW83]<br>[VW86a] | Section 5.5 | |
| CTL | [BCG95] | [CES83]<br>[BVW94] | Section 5.3 | [QS82] |
| $CTL^*$ | [BCG95] | [BVW94]<br>[VB00] | Section 5.2 | |
| $CTL^+$ | | | Section 5.4 | [LMS01] |
| PDL | | | Chapter 4 | [FL77]<br>[AI00] |
| FLC | [LS02a] | | Chapter 9 | |
| $\mathcal{L}_\mu$ | [SW91]<br>[Cle90] | [BVW94] | [Sti95] | [Eme97] |
| $\mathcal{L}_\mu^0$ | | [MSS92]<br>[BVW94] | | [And94]<br>[CS92]<br>[BC96] |

Figure 3.1: The history of model checking.

the model checking problem for $CTL^+$ has only been addressed in [LMS01] using a reduction technique.

We do not include LTL in this table since model checking for linear time temporal logic is only really interesting if it is interpreted over all paths of a total transition system, i.e. if BLTL is considered in fact.

Remember that for branching time logics the detour via satisfiability checking automata is not feasible for model checking. In the FLC case it is not even possible since satisfiability checking is undecidable. Consequently, the right automaton model for FLC formulas is one whose membership problem is decidable although the emptiness checking problem is undecidable. Section 9.2 will suggest that alternating tree pushdown automata could serve as the right choice for automata-based FLC model checking. This is also hinted in [Lan02a] where a translation from FLC interpreted solely over linear models into alternating pushdown automata over finite and infinite words is given.

## Satisfiability Checking

Figure 3.2 lists the most important publications concerning the satisfiability checking problem for these logics. Here, there is no distinction between LTL and BLTL since a model for an LTL formula is also a model for the corresponding BLTL formula. Conversely, every path of a model for a BLTL formula is also a model for the corresponding LTL formula. Thus, a BLTL formula is satisfiable iff its LTL pendant is satisfiable.

The empty tableaux for $CTL^*$ field is due to a conjecture stated by Emerson that determinisation is essential for checking satisfiability of $CTL^*$ formulas. Therefore there would be no tableau-based decision procedure for $CTL^*$. This is refuted in Chapter 8. The winning strategies for the games introduced there can easily be seen as tableaux for $CTL^*$.

The empty fields in the games column will be addressed at the end of this thesis regarding further work. It is not entirely clear whether [NW97] should be listed under tableaux or games or both. In fact, the method proposed there builds tableaux for $\mathcal{L}_\mu$

| logic | tableaux | automata | games | others |
| --- | --- | --- | --- | --- |
| LTL | [LP00] [SGL97] | [SVW83] [VW86a] | Section 6.1 | [Fis91] |
| CTL | [CE81] [BAPM83] | [VW86a] | Section 6.2 | [EH85] |
| CTL$^*$ | | [ES84] [Eme85] [EJ00] | Chapter 8 | |
| CTL$^+$ | | | | |
| PDL | [Pra80] | [VW86a] | Section 6.3 | [FL77] [Pra79] |
| $\mathcal{L}_\mu$ | [NW97] | [SE84] [EJ91] [EJ00] | | |
| $\mathcal{L}_\mu^0$ | | [BVW94] | | |

Figure 3.2: The history of satisfiability checking.

| logic | model checking | satisfiability checking |
|---|---|---|
| PDL | PTIME-complete | EXPTIME-complete |
| LTL | PTIME-complete | PSPACE-complete |
| BLTL | PSPACE-complete | PSPACE-complete |
| CTL | PTIME-complete | EXPTIME-complete |
| $\text{CTL}^{+}$ | $\Pi_2$-complete | EXPTIME-hard, $\in$ 2-EXPTIME |
| $\text{CTL}^{*}$ | PSPACE-complete | 2-EXPTIME-complete |
| FLC | PSPACE-hard, $\in$ EXPTIME | undecidable |
| $\text{FLC}^{k}$ | PSPACE-complete | undecidable |
| $\mathcal{L}_\mu$ | PTIME-hard, $\in$ NP$\cap$co-NP | EXPTIME-complete |
| $\mathcal{L}_\mu^k$ | PTIME-complete | EXPTIME-complete |

Figure 3.3: The model checking and satisfiability checking complexities.

formulas which are only pre-witnesses for the satisfiability of a formula. To obtain witnesses a game is played on these tableaux. This can be simplified to obtain a decision procedure for $\mathcal{L}_\mu^0$ on the same basis.

Again, to the best of our knowledge, the only known decision procedures for $\text{CTL}^{+}$ are based on regarding the input as a $\text{CTL}^{*}$ formula or translating it into CTL.

## Complexities

Figure 3.3 shows known lower and upper bounds for the computational complexities of these logics. Again, we include $\mathcal{L}_\mu$ and $\mathcal{L}_\mu^k$ to allow comparisons. Note that $\text{FLC}^k$ and $\mathcal{L}_\mu^k$ denote all fragments of arbitrary but fixed alternation depth.

PTIME-hardness of the model checking problems follows trivially from the

PTIME-hardness of the evaluation problem for boolean formulas. Note that all the logics featured here subsume propositional boolean logic.

PSPACE-hardness of BLTL's model checking and LTL's satisfiability checking problem was shown in [SC85]. The former also proves that $\mathrm{CTL}^*$ model checking is PSPACE-hard. PSPACE-hardness of FLC's and $\mathrm{FLC}^k$'s model checking problem was shown in [LS02a]. A different but unpublished proof was found by Müller-Olm earlier on.

$\mathrm{CTL}^+$'s lower bound for model checking was found in [LMS01] together with its upper bound. All the other upper bounds result from complexity analyses of the work summarised in Figure 3.1.

EXPTIME-hardness of PDL's satisfiability problem was proved in [FL77]. $\mathcal{L}_\mu^0$'s and $\mathcal{L}_\mu$'s EXPTIME-hardness is a consequence of this. The proof of CTL's EXPTIME-hardness is not a consequence of this but proceeds along the same lines. This makes it a lower bound for the complexity of $\mathrm{CTL}^+$'s satisfiability problem, too.

2-EXPTIME-hardness of $\mathrm{CTL}^*$'s satisfiability problem was shown in [VS85]. [MO99] proved that $\mathrm{FLC}^0$ and with it FLC and $\mathrm{FLC}^k$ are undecidable for all $k \in \mathbb{N}$.

Membership in EXPTIME of PDL's satisfiability problem was shown in [Pra79]. Again, the other results providing upper bounds can be found in Figure 3.2.

## Expressiveness

Figure 3.4 shows how the logics discussed here relate to each other in terms of their expressive powers.

PDL is easily seen to be embeddable into $\mathcal{L}_\mu^0$, [KT90]. The translation is uniform and, hence, even preserves the subformula property to some extend. The same holds for the translation of CTL into $\mathcal{L}_\mu^0$, as well as the translation from $\mathcal{L}_\mu$ into FLC, [MO99].

A translation from $\mathrm{CTL}^*$ into $\mathcal{L}_\mu^1$ is given in [Dam94]. It does not preserve the subformula property. This is not surprising if one considers the complexities for these logics. Clearly, an embedding that preserves the syntactical structure of a formula gives rise to a polynomial reduction from one logic's satisfiability checking problem

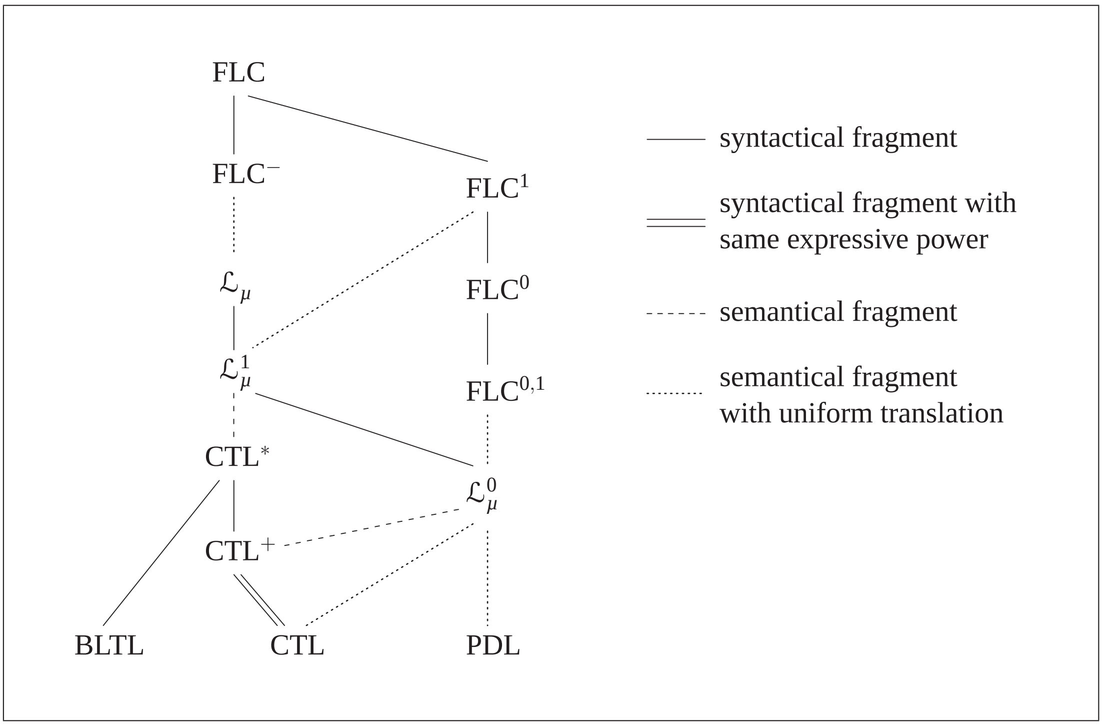


Figure 3.4: Expressiveness in the family of modal and temporal logics.

to the other's. But the fact that there is a double exponential lower bound for deciding CTL$^*$ and the membership of $\mathcal{L}_\mu$'s satisfiability problem in EXPTIME show that every translation from CTL$^*$ to $\mathcal{L}_\mu$ has to produce certain formulas of exponential length.

# Chapter 4

# Model Checking Games for Propositional Dynamic Logic

*Though this be madness,*
*yet there is method in it.*
—
POLONIUS

Model checking games for PDL are played on an LTS $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with starting state $s \in \mathcal{S}$ and a PDL formula $\varphi$. Player $\exists$ wants to show that $s \models \varphi$ whereas player $\forall$ tries to show $s \not\models \varphi$. The set of configurations of the game $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ is

$$\mathcal{C} \quad = \quad \mathcal{S} \times Sub(\varphi)$$

A configuration is written $t \vdash \psi$ where $t \in \mathcal{S}$ and $\psi \in Sub(\varphi)$.

The game rules are given in Figure 4.1. They are usually written

$$(r) \ \frac{C}{C'} \ p \ c$$

and to be read as: If the actual configuration $C_i$ in a play is of the form $C$ then player $p$ performs a choice $c$ and the next configuration $C_{i+1}$ is $C'$ with the same instantiations as those for $C$. $(r)$ is the name of the rule. A player/choice combination like $\forall i$ means that player $\forall$ chooses an $i$ from a domain which should become clear by inspecting $C$ and $C'$.

One of the reasons for calling the players $\forall$ and $\exists$ becomes apparent in this moment. A notation like $\exists i$ can be read as "player $\exists$ chooses an $i$" but also as "if the upper configuration is true then there exists an $i$ that makes the lower configuration true". The same holds for a player/choice combination like $\forall i$ for example.

An empty $p\ c$ means the rule is deterministic. In this case it does not matter which player makes the next move since the outcome would be the same. Therefore, we omit player names in deterministic rules.

Another possible game rule pattern is

$$(r)\ \frac{C}{C' \quad | \quad C''}\ p$$

Here, if the actual configuration $C_i$ is an instance of $C$ then player $p$ has the choice whether the next configuration $C_{i+1}$ will be an instance of $C'$ or $C''$.

A disjunction is easy to prove, therefore it is player $\exists$'s task to choose a disjunct with rule $(\vee)$. A conjunction is easy to refute. This is done by player $\forall$ in rule $(\wedge)$. Rules $(\langle\cup\rangle)$, $([\cup])$, $(\langle;\rangle)$, $([;])$, $(\langle*\rangle)$, $([*])$, $(\langle?\rangle)$ and $([?])$ simply apply the equivalences for PDL formulas with modalities given in Section 2.5 to obtain formulas or programs of smaller size. Some of these equivalences yield boolean combinations. In these cases the following choice using rule $(\vee)$ or $(\wedge)$ has been built into the modality rule already.

Finally, if the actual configuration contains a modality with an atomic program one of the players has to choose a successor state that is reachable along a transition labelled with the program at hand. This is reflected in rules $(\langle a\rangle)$ and $([a])$.

There are three different types of plays. An atomic formula can be reached in which case no rule applies. One of the players can get stuck by being unable to choose a successor state. Or the play can proceed ad infinitum.

$$(\vee)\ \frac{s \vdash \varphi_0 \vee \varphi_1}{s \vdash \varphi_i}\ \exists i \qquad (\wedge)\ \frac{s \vdash \varphi_0 \wedge \varphi_1}{s \vdash \varphi_i}\ \forall i$$

$$(\langle\cup\rangle)\ \frac{s \vdash \langle\alpha_0 \cup \alpha_1\rangle\varphi}{s \vdash \langle\alpha_i\rangle\varphi}\ \exists i \qquad ([\cup])\ \frac{s \vdash [\alpha_0 \cup \alpha_1]\varphi}{s \vdash [\alpha_i]\varphi}\ \forall i$$

$$(\langle;\rangle)\ \frac{s \vdash \langle\alpha_0;\alpha_1\rangle\varphi}{s \vdash \langle\alpha_0\rangle\langle\alpha_1\rangle\varphi} \qquad ([;])\ \frac{s \vdash [\alpha_0;\alpha_1]\varphi}{s \vdash [\alpha_0][\alpha_1]\varphi}$$

$$(\langle*\rangle)\ \frac{s \vdash \langle\alpha^*\rangle\varphi}{s \vdash \varphi \quad | \quad s \vdash \langle\alpha\rangle\langle\alpha^*\rangle\varphi}\ \exists \qquad ([*])\ \frac{s \vdash [\alpha^*]\varphi}{s \vdash \varphi \quad | \quad s \vdash [\alpha][\alpha^*]\varphi}\ \forall$$

$$(\langle?\rangle)\ \frac{s \vdash \langle\varphi_0?\rangle\varphi_1}{s \vdash \varphi_i}\ \forall i \qquad ([?])\ \frac{s \vdash [\psi?]\varphi}{s \vdash \overline{\psi} \quad | \quad s \vdash \varphi}\ \exists$$

$$(\langle a\rangle)\ \frac{s \vdash \langle a\rangle\varphi}{t \vdash \varphi}\ \exists\ s \xrightarrow{a} t \qquad ([a])\ \frac{s \vdash [a]\varphi}{t \vdash \varphi}\ \forall\ s \xrightarrow{a} t$$

Figure 4.1: The rules for the PDL model checking games.

Player $\forall$ wins the play $C_0, C_1, \ldots$ iff

1. there is an $n \in \mathbb{N}$ s.t. $C_n = t \vdash q$ and $q \notin L(t)$, or
2. there is an $n \in \mathbb{N}$ s.t. $C_n = t \vdash \langle a\rangle\psi$ and $t \stackrel{a}{\not\to}$, or
3. there are infinitely many $i \in \mathbb{N}$ s.t. $C_i = t_i \vdash \langle\alpha^*\rangle\psi$ for some $t_i \in \mathcal{S}$.

Player $\exists$ wins the play $C_0, C_1, \ldots$ iff

4. there is an $n \in \mathbb{N}$ s.t. $C_n = t \vdash q$ and $q \in L(t)$, or
5. there is an $n \in \mathbb{N}$ s.t. $C_n = t \vdash [a]\psi$ and $t \stackrel{a}{\not\to}$, or

6. there are infinitely many $i \in \mathbb{N}$ s.t. $C_i \ = \ t_i \vdash [\alpha^*]\psi$ for some $t_i \in \mathcal{S}$.

**Example 42** Let $\mathcal{T}$ be the transition system consisting of states $\{s,t\}$ with transitions $s \xrightarrow{a} t$ and $t \xrightarrow{a} t$. The labelling of the states is $L(s) = \{\overline{q}\}$ and $L(t) = \{q\}$. The formula to be checked is

$$\varphi \ := \ \langle(\overline{q}?;a)^*\rangle q$$

$\varphi$ says "there is a path labelled with $a$s on which $q$ does not hold until it holds", see also Example 16. $\mathcal{T}$ with starting state $s$ satisfies $\varphi$. The full game tree is given in Figure 4.2. The players' choices are annotated at the right side of the rules.

Player $\forall$ wins the plays ending with $s \vdash q$ and $t \vdash \overline{q}$ because of condition 1. The rightmost path results in an infinite play that visits the configuration

$$t \vdash \langle(\overline{q}?;a)^*\rangle q$$

infinitely often. Thus, it is won by player $\forall$, too. Player $\exists$ wins the other plays with winning condition 4. She has a winning strategy since she can force the game into the position $t \vdash q$ unless player $\forall$ has forced it into a defeat for himself beforehand.

We remark that applying the model checking games for $\mathcal{L}_\mu$ from [Sti95] to translations of PDL formulas into $\mathcal{L}_\mu^0$ results in basically the same games as the PDL model checking games of this chapter.

## Correctness

**Fact 43** *Rules* $(\vee)$, $(\wedge)$, $(\langle\cup\rangle)$, $([\cup])$, $(\langle?\rangle)$, $([?])$, $(\langle a\rangle)$ *and* $([a])$ *reduce the size of the actual configuration. Rules* $(\langle*\rangle)$ *and* $([*])$ *can both decrease or increase it. Rules* $(\langle;\rangle)$ *and* $([;])$ *reduce the size of a program occurring in the actual configuration.*

**Lemma 44** *Every play of* $\mathcal{G}_\mathcal{T}(s,\varphi)$ *has a uniquely determined winner.*

PROOF A play can either be of finite or infinite length. Suppose it is of finite length. Note that there is a rule for each type of formula except atomic propositions $q$. Furthermore, all rules apart from $(\langle a\rangle)$ and $([a])$ are always applicable in a

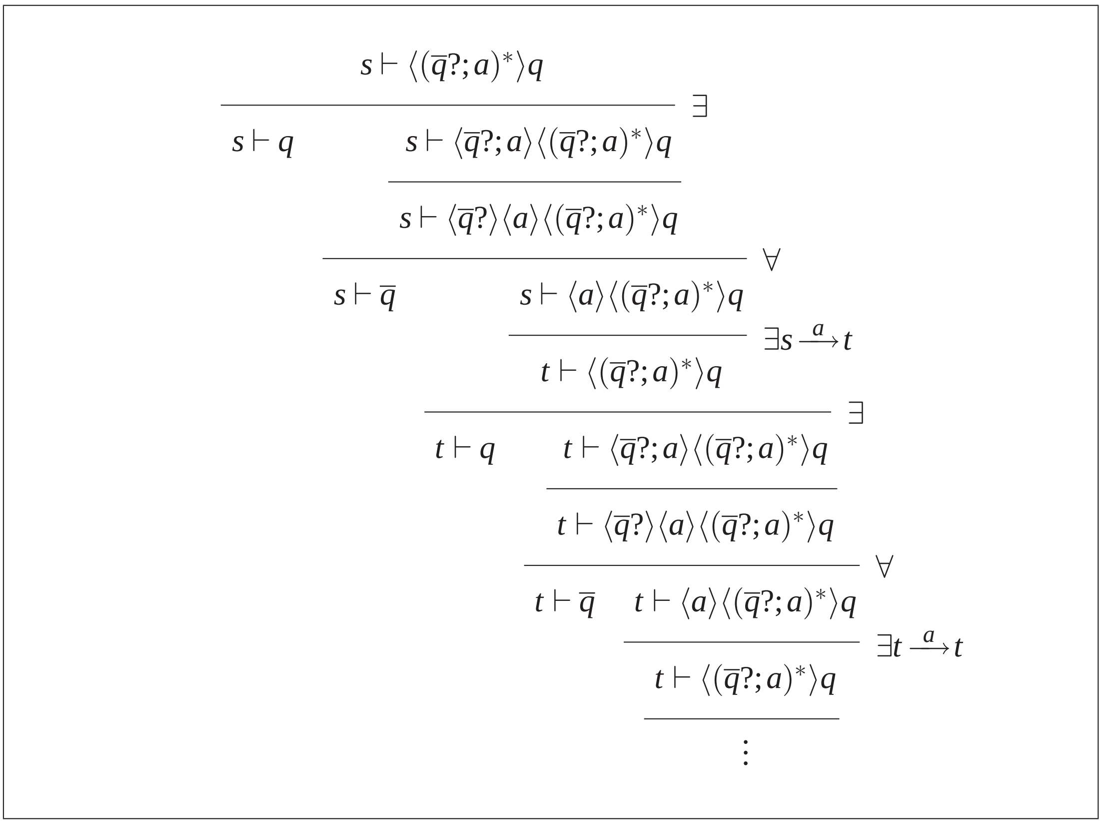


Figure 4.2: The full game tree for Example 42.

corresponding configuration since the players only choose subformulas. Rules $(\langle a\rangle)$ and $([a])$ may not be applicable in case there is no corresponding transition to choose in the underlying transition system.

Thus, a finite play must end in a configuration of either of the forms $t \vdash q$, $t \vdash \langle a\rangle\psi$ or $t \vdash [a]\psi$. In the second case, winning condition 2 determines the winner. Winning condition 5 does the same for the third case. For the first case, note that either $q \in L(t)$ or $q \notin L(t)$. Therefore, the winner is uniquely determined by winning condition 1 or 4, too.

Suppose now that the play at hand is of infinite length. According to Fact 43, this is only possible if rule $(\langle * \rangle)$ or $([*])$ is played infinitely often since all other rules genuinely decrease the size of the configuration or a program occurring in it. Moreover, the players must choose the option that increases the size of the actual configuration infinitely often.

Note that, if player $\exists$ chooses $\varphi$ in the unfolding of $\langle\alpha^*\rangle\varphi$ for example, $\langle\alpha^*\rangle\varphi$ cannot occur in the play again. Otherwise it would be a genuine subformula of itself. The same holds for player $\forall$ and $[\alpha^*]\varphi$.

Thus, in an infinite play they will almost always choose $\langle\alpha\rangle\langle\alpha^*\rangle\varphi$, resp. $[\alpha][\alpha^*]\varphi$, in applications of rules $(\langle*\rangle)$ and $([*])$. Suppose both are played infinitely often, i.e. there are $\langle\alpha^*\rangle\varphi$ and $[\beta^*]\psi$ that occur infinitely often in a play. One of them must be a subformula of the other, say $[\beta^*]\psi \in Sub(\varphi)$. But if player $\exists$ always chooses $\langle\alpha\rangle\langle\alpha^*\rangle\varphi$ in an application of rule $(\langle*\rangle)$ then $\varphi$ will never occur as the formula component of a configuration in the play. Consequently, $[\beta^*]\psi$ cannot either.

Hence, in every infinite play either a $\langle\alpha^*\rangle\varphi$ or a $[\alpha^*]\varphi$ occurs infinitely often and the winner of this play is uniquely determined by winning condition 3 or 6. ∎

**Definition 45** Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with $s,t \in \mathcal{S}$, $\varphi \in$ PDL and $\psi \in Sub(\varphi)$. A configuration $t \vdash \psi$ of the game $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ is called *true* if $t \models \psi$ and *false* otherwise.

**Lemma 46** *Player $\exists$ preserves falsity and can preserve truth with her choices. Player $\forall$ preserves truth and can preserve falsity with his choices.*

PROOF First consider rule $(\vee)$. Take a configuration

$$C = t \vdash \varphi_0 \vee \varphi_1$$

Suppose $C$ is false, i.e. $t \not\models \varphi_0$ and $t \not\models \varphi_1$. Regardless of which $i$ player $\exists$ chooses, the configuration $t \vdash \varphi_i$ will be false. On the other hand, suppose $C$ is true. Then $t \models \varphi_0$ or $t \models \varphi_1$, and player $\exists$ can preserve truth by choosing $i$ accordingly. The proofs for rules $(\wedge)$, $(\langle\cup\rangle)$ and $([\cup])$ are similar or dual. The cases of rules $(\langle*\rangle)$, $([*])$, $(\langle?\rangle)$ and $([?])$ can be reduced to the boolean connectives.

Consider now a configuration

$$C = t \vdash \langle a\rangle\psi$$

Suppose $C$ is false. Then either $t \stackrel{a}{\not\to}$ or for every $t' \in \mathcal{S}$: if $t \xrightarrow{a} t'$ then $t' \not\models \psi$. I.e. if $t$ has an $a$-successor then player $\exists$ cannot make the following configuration true. If $t$ does not have an $a$-successor then there will be no next configuration and consequently player $\exists$ cannot make it true either.

Suppose now that $C$ is true. Then there is a $t' \in \mathcal{S}$ s.t. $t \xrightarrow{a} t'$ and $t' \models \psi$. By choosing this $t'$, player $\exists$ can preserve truth. The case of rule $([a])$ is dual. ■

Note that the deterministic rules $(\langle;\rangle)$ and $([;])$ preserve both truth and falsity.

Preserving truth, resp. falsity, is going to play an important role in the following proofs of soundness and completeness, Theorems 48 and 49. Consequently, it is going to be an important part of player $\exists$'s, resp. player $\forall$'s, winning strategies. However, this alone is not enough as the next example shows.

**Example 47** Take the transition system $\mathcal{T}$ consisting of one state $s$ only with an $a$-loop to itself, i.e. $s \xrightarrow{a} s$. Consider the formula

$$\varphi \;=\; \langle a^*\rangle\langle a\rangle\mathtt{tt}$$

which postulates the existence of a finite path whose transitions are labelled with $a$ and which has at least two states.

The game $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ only consists of unfolding $\varphi$ and choosing the only possible transition $s \xrightarrow{a} s$. Note that $s \models \psi$ for all $\psi \in Sub(\varphi)$, i.e. regardless of player $\exists$'s choices with rule $(\langle *\rangle)$, she will always preserve truth. However, in order to win she needs to choose $\langle a\rangle\mathtt{tt}$ at some point, otherwise player $\forall$ would win with his winning condition 3.

Therefore it is part of both players' strategies to choose the smaller of two formulas if both preserve truth, resp. falsity.

**Theorem 48 (Soundness)** *If* $\mathcal{T}, s \not\models \varphi$ *then player* $\forall$ *wins* $\mathcal{G}_{\mathcal{T}}(s,\varphi)$.

PROOF If $s \not\models \varphi$ then the starting configuration of every play of $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ is false. We build a game tree for player $\forall$ preserving falsity. I.e. whenever a rule requires him to make a choice the tree will contain the successor configuration that preserves falsity according to Lemma 46. All of player $\exists$'s choices are contained in the tree.

Player $\exists$ cannot win a finite play of this tree since she only wins finite plays that end in true configurations. Suppose she wins an infinite play. Then it must contain infinitely many false configurations of the form $C_i = t_i \vdash [\alpha^*]\psi$ for $i = 0, 1, \ldots$. Consider the first of these. By falsity

$$t_0 \;\not\models\; [\alpha^*]\psi$$

According to Lemma 15 of Chapter 2, there must be a smallest $k \in \mathbb{N}$ s.t.

$$t_0 \;\not\models\; [\alpha^k]\psi$$

If $C_1 = t_1 \vdash [\alpha^*]\psi$ is reached it can be interpreted as

$$t_1 \vdash [\alpha^{k-1}]\psi$$

The argument is iterated with $C_1$.

By preservation of falsity the play must eventually reach a false configuration $C_k$ interpreted as

$$t_k \;\vdash\; [\alpha^0]\psi$$

But $[\alpha^0]\psi \equiv \mathtt{tt}$, i.e. $C_k$ cannot be false.

We conclude that the assumption of $t_0 \vdash [\alpha^*]\psi$ being false was wrong and that therefore player $\exists$ cannot win an infinite play either. Hence, player $\forall$ wins $\mathcal{G}_{\mathcal{T}}(s, \varphi)$. ∎

**Theorem 49 (Completeness)** *If $\mathcal{T}, s \models \varphi$ then player $\exists$ wins $\mathcal{G}_{\mathcal{T}}(s, \varphi)$.*

PROOF According to Lemma 13, PDL is closed under negation. Furthermore, the class of PDL model checking games is closed under dual games since for every game rule there is a dual rule and for every winning condition there is a dual winning conditions, too.

Suppose now that $\mathcal{T}, s \models \varphi$, i.e. $\mathcal{T}, s \not\models \overline{\varphi}$. According to Theorem 48, player $\forall$ wins $\mathcal{G}_{\mathcal{T}}(s, \overline{\varphi})$. But then player $\exists$ wins $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ by Theorem 39, the duality principle. ∎

Theorems 48 and 49 show that the PDL model checking games are *determined*, i.e. for every game one of the players has a winning strategy.

**Corollary 50 (Determinacy)** *Player $\forall$ wins $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ iff player $\exists$ does not win $\mathcal{G}_{\mathcal{T}}(s, \varphi)$.*

**Theorem 51 (Winning strategies)** *The winning strategies for the PDL model checking games are history-free.*

PROOF Consider player $\exists$'s winning strategies. According to the proof of Theorem 49, she needs to preserve truth. Note that the truth value of a configuration only depends on its state and its formula component and not on the history of a play.

Furthermore, if she has the choice between two different successor configurations and both are true, she chooses the smaller one. But the size of a successor configuration does not depend on the history either.

The situation for player $\forall$ is dual. Thus, his winning strategies are history-free as well. ■

## PDL over Finite State Transition Systems

The completeness proof of the PDL satisfiability games that will be presented in Section 6.3 depends on the fact that satisfiable PDL formulas have finite models.

**Theorem 52 (Finite model property)** *PDL has the finite model property.*

PROOF Suppose $\varphi_0 \in$ PDL is satisfiable. Then it has a model $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with $s_0 \in \mathcal{S}$. Furthermore, there is a successful game tree $T$ for player $\exists$ and the game $\mathcal{G}_{\mathcal{T}}(s_0, \varphi_0)$. We construct another tree $T'$ and show that it is a successful game tree for player $\exists$ as well. Note that for every infinite branch $C_0, C_1, \ldots$ in $T$ there is a $[\alpha^*]\psi \in Sub(\varphi_0)$ s.t. the branch contains infinitely many configurations $C_{i_0}, C_{i_1}, \ldots$ with

$$C_{i_j} \;=\; t_j \vdash [\alpha^*]\psi \quad \text{for some } t_j \in \mathcal{S}$$

To obtain a graph $T_1$ from $T$ we do the following. For every such branch in $T$ we discard the entire subtree beginning with $C_{i_1}$ and add an edge from $C_{i_1-1}$ to $C_{i_0}$. Let $T'$ be the unravelling of $T_1$ with respect to the starting configuration $C_0$:

$$T' \;=\; \mathcal{R}_{C_0}(T_1)$$

In order to show that $T'$ is a game tree we need to consider the added edges from a $C_{i_1-1}$ to a $C_{i_0}$. Regardless of which rule was applied to $C_{i_1-1}$ to obtain $C_{i_1}$, the pair $(C_{i_1-1}, C_{i_0})$ is a valid instance of this rule as well. This is because the formula components of $C_{i_0}$ and $C_{i_1}$ are the same.

Moreover, $T'$ is also a game tree for player $\exists$. Note that she has a winning strategy for the subgames starting in any configuration of $T$, in particular $C_{i_0}$ and $C_{i_1}$ for any branch. According to Theorem 51, winning strategies are history-free. Thus, the subgame starting with $C_{i_1}$ can be replaced by the subgame starting with $C_{i_0}$ without effecting the winner of the entire game.

Note that there are only finitely many different states of $\mathcal{T}$ occurring in a configuration of $T'$. Thus, it is possible to define a finite transition system $\mathcal{T}' = (\mathcal{S}', \{\xrightarrow{a}{}' \mid a \in \mathcal{A}\}, L)$ by

$$\mathcal{S}' \;:=\; \{\, t \in \mathcal{S} \mid \text{there is a } \psi \in Sub(\varphi_0) \text{ s.t. } t \vdash \psi \text{ is a configuration in } T_1 \,\}$$

with transitions given by

$$t_1 \xrightarrow{a}{}' t_2 \quad \text{iff} \quad \begin{array}{l} \text{there is a configuration } t_1 \vdash \langle a \rangle \psi \text{ or } t_1 \vdash [a]\psi, \text{ and} \\ \text{a configuration } t_2 \vdash \psi \text{ s.t. rule } (\langle a \rangle) \text{ or } ([a]) \text{ was played} \\ \text{between them} \end{array}$$

The labelling of the states is taken from their respective labellings in $\mathcal{T}$.

In fact, $T'$ is a successful game tree for player $\exists$ and the game $\mathcal{G}_{\mathcal{T}'}(s_0, \varphi_0)$. Then $\mathcal{T}', s_0 \models \varphi_0$ by Theorem 48, i.e. $\varphi_0$ has a finite model. ■

If the underlying transition system is finite the winning conditions can be modified to result in finite plays only. The game rules remain the same. Player $\forall$ wins the play $C_0, \ldots, C_n$ iff

1. $C_n \;=\; t \vdash q$ and $q \notin L(t)$, or
2. $C_n \;=\; t \vdash \langle a \rangle \psi$ and $t \stackrel{a}{\not\rightarrow}$, or
3. $C_n \;=\; t \vdash \langle \alpha^* \rangle \psi$ and there is a $C_i$ with $i < n$ and $C_i = C_n$.

Player $\exists$ wins the play $C_0, \ldots, C_n$ iff

4. $C_n \;=\; t \vdash q$ and $q \in L(t)$, or

5. $C_n \;=\; t \vdash [a]\psi$ and $t \stackrel{a}{\not\rightarrow}$, or

6. $C_n \;=\; t \vdash [\alpha^*]\psi$ and there is a $C_i$ with $i < n$ and $C_i = C_n$.

The new winning conditions are simplified versions of the ones for arbitrary transition systems. Winning conditions 1,2,4 and 5 are just the same. By Theorem 51, winning strategies are history-free, and the new winning conditions 3 and 6 result from the old ones by regarding plays in the game graph instead of in the game tree, see Section 2.7.

### Complexity

One way of analysing the complexity of finding winning strategies in the PDL model checking games is to use the results on alternating complexity classes. It is not hard to see that a play of a game $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ can be played using space that is logarithmic in the size of the input only. This is done by encoding a configuration using two pebbles. One of them is placed on a state of the transition system, the other on a subformula of the input formula. The pebbles can be stored as counters which need logarithmic space in the size of the transition system and the formula.

Therefore, the winner of $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ can be decided in alternating LOGSPACE which is the same as PTIME according to [CKS81]. However, using a more explicit analysis this result can be improved.

**Theorem 53 (Complexity)** *Deciding the winner of a PDL model checking game is in LINTIME.*

PROOF We sketch a global algorithm that decides the winner of $\mathcal{G}_{\mathcal{T}}(s,\varphi)$. Since winning strategies are history-free the game can be represented by the game graph. The algorithm simply labels nodes of this graph with either $\forall$ or $\exists$ depending on which player can win the game starting with the configuration at hand. This is done in a bottom-up manner.

The game graph can be partitioned into blocks and these blocks can be enumerated s.t. every path of the graph either stays in one block or leaves a block into another one whose index is strictly greater than the first one's. This block structure is induced by the formula component of a configuration only. A block is in fact a strongly connected component of the graph. And strongly connected components can be computed in linear time using Tarjan's algorithm for example, [Tar72].

Remember that most rules of the games reduce the size of the formula at hand. Exceptions are formulas of the form $\langle\alpha^*\rangle\psi$ and $[\alpha^*]\psi$. These can cause the game graph to have loops. Paths cannot lead back into a block they have been in because once a play reaches a formula, say, $\langle\alpha^*\rangle\psi$ it can never reach a proper superformula of it again. Furthermore, each block can only have loops of one type, a $\langle\alpha^*\rangle\psi$ or a $[\alpha^*]\psi$.

Thus, the graph of blocks is directed and acyclic. It can be processed starting at those blocks which are furthest away from the starting configuration

$$C_0 \;=\; s \vdash \varphi$$

The configurations in such a block can be labelled in the following way. Terminal configurations, i.e. those that end a play with winning conditions 1,2,4 or 5 are labelled with the corresponding winner. The last configuration of a path that exhibits a repeat is labelled $\forall$ if the type of this block is $\langle\alpha^*\rangle\psi$ and with $\exists$ otherwise. The other configurations can be labelled in a bottom-up manner depending on which player has a choice in the configuration at hand and whether there is a successor configuration that is labelled with their name already.

The algorithm only needs to visit each node of the game graph once. For a transition system with state set $\mathcal{S}$ and a formula $\varphi$ the size of the game graph is $|\mathcal{S}| \cdot |\varphi|$. Thus, the claim follows. ■

This is essentially the same technique that is used to show that model checking for the alternation free $\mu$-calculus $\mathcal{L}_\mu^0$ can be done in linear time as well, [And94, CS92, BC96].

## Extensions of PDL

The PDL model checking games can be extended in a straight-forward way in order to capture extensions of PDL like Converse-PDL, [Str81], and PDL-$\Delta$, [Str85], as defined in Section 2.5.

$$\frac{s \vdash \langle\overline{\alpha \cup \beta}\rangle\varphi}{s \vdash \langle\overline{\alpha} \cup \overline{\beta}\rangle\varphi} \qquad \frac{s \vdash [\overline{\alpha \cup \beta}]\varphi}{s \vdash [\overline{\alpha} \cup \overline{\beta}]\varphi} \qquad \frac{s \vdash \langle\overline{\alpha;\beta}\rangle\varphi}{s \vdash \langle\overline{\beta};\overline{\alpha}\rangle\varphi}$$

$$\frac{s \vdash [\overline{\alpha;\beta}]\varphi}{s \vdash [\overline{\beta};\overline{\alpha}]\varphi} \qquad \frac{s \vdash \langle\overline{\alpha^*}\rangle\varphi}{s \vdash \langle\overline{\alpha}^*\rangle\varphi} \qquad \frac{s \vdash [\overline{\alpha^*}]\varphi}{s \vdash [\overline{\alpha}^*]\varphi}$$

$$\frac{s \vdash \langle\overline{a}\rangle\varphi}{t \vdash \varphi} \; \exists \; t \xrightarrow{a} s \qquad \frac{s \vdash [\overline{a}]\varphi}{t \vdash \varphi} \; \forall \; t \xrightarrow{a} s \qquad \frac{s \vdash \mathit{repeat}(\alpha)}{s \vdash \langle\alpha\rangle\mathit{repeat}(\alpha)}$$

Figure 4.3: The rules for extensions of PDL.

To allow converse of programs and the repeat operator in the PDL model checking games one simply has to add the rules of Figure 4.3. They mimic the equivalences for converse programs and use the unfolding characterisation of the *repeat* construct as given in Section 2.5.

The winning conditions have to be extended, too. There are two for the case of the converse of an atomic program which cannot be executed in a particular state. I.e. player $\forall$ wins the play $C_0, \ldots$ if there is a $n \in \mathbb{N}$ s.t.

$$C_n \;=\; t \vdash \langle\overline{a}\rangle\psi$$

for some $t$ and $\psi$ and there is no state $s$ s.t. $s \xrightarrow{a} t$. Consequently, player $\exists$ wins if player $\forall$ gets stuck in a configuration

$$C_n \;=\; t \vdash [\overline{a}]\psi$$

and there is no $s$ s.t. $s \xrightarrow{a} t$.

The repeat construct requires an additional winning condition for player $\exists$. She wins an infinite play if there are infinitely many configurations $C_0, C_1, \ldots$ and a program $\alpha$ s.t.

$$C_i \;=\; t_i \vdash \mathit{repeat}(\alpha)$$

for some $t_i \in \mathcal{S}$ and every $i \in \mathbb{N}$.

These extensions do not effect the complexity of game-based PDL model checking. It is still possible in linear time.

# Chapter 5

# Model Checking Games for Branching Time Logics

*But I remembered a voice from my past*
*'Gambling only pays when you're winning'*
—
GENESIS

## 5.1 Focus Games and Sets of Formulas

Some of the games in this and the following chapters will use a special tool called *focus*. Mathematically, the focus simply is a function from a set to its elements. It is used to highlight, resp. focus on one particular element in a set of formulas. A *focus game* is a model checking or satisfiability game that makes use of a focus. A configuration in such a game involves a focus on a set of formulas. This is for example written as

$$\left[\varphi\right], \Phi$$

and is to be understood in the following way: $\varphi$ is a single formula, $\Phi$ is a set of formulas. The configuration at hand is, or at least contains, the disjoint union $\Phi \cup \{\varphi\}$ as a set of formulas in which $\varphi$ is highlighted.

Confluence is a potential problem for the games whose configurations contain or are sets of formulas. The rules of all the games in this thesis however deal with *principle formulas*: there is usually one formula in a configuration which gets replaced by a subformula of it in an application of this rule. The other formulas which are present at this moment get discarded or copied into the next configuration. However, when using sets, there are several candidates for a principle formula and, hence, more than one rule might be admissible at a certain moment.

Informally, a game is called *confluent* if the order in which admissible rules are applied does not effect the outcome of a play. The games of the following chapters are confluent because of a simple argument. One only needs to consider possible conflicts between rules that require different players to choose a particular subformula of a possible principle formula. By doing so, another subformula might be discarded. Later this could turn out to have been a bad choice since the other player discarded a subformula of another principle formula of this moment which might be necessary for the first player to win. On the other hand, if the first player had performed a different choice, the opponent might have reacted differently as well.

This is, however, not possible for the games of the following chapters which use sets of formulas since it is always at most one player who has the possibility to discard formulas.

As it was mentioned in Section 2.6 already, regenerating fixed point constructs play an important role in games that use sets of formulas.

**Definition 54** In a play $C_0, C_1, \dots$ of a game, a formula $\varphi$ is called *regenerating* between $C_k$ and $C_n$ for $k < n$ iff

- $\varphi$ is a fixed point construct, i.e. there is a $\psi \in Sub(\varphi)$ s.t. $\varphi \in Sub(\psi)$, and
- $\varphi \in C_k$ and $\varphi \in C_n$ and for all $i$ with $k \leq i < n$: $(C_i, C_{i+1})$ is an instance of a rule that either preserved $\varphi$, resp. its unfolding, or replaced it by a subformula of it.

The fixed point construct $\varphi$ gets regenerated *infinitely often* in a play $C_0, \ldots$ if there is an $i \in \mathbb{N}$ s.t. $\varphi$ gets regenerated between $C_i$ and $C_n$ for all $n > i$.

## 5.2 Model Checking Games for CTL$^*$

Given a total LTS $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$, $s \in \mathcal{S}$ and a CTL$^*$ formula $\varphi$, the CTL$^*$ *model checking game* $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ is a *focus game* in the above sense. Player $\exists$ wants to show that $\mathcal{T}, s \models \varphi$ whereas player $\forall$ tries to show that $\mathcal{T}, s \not\models \varphi$.

The set of configurations of $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ is

$$\mathcal{C} \;=\; \mathcal{S} \times \{\mathtt{E}, \mathtt{A}\} \times Sub(\varphi) \times 2^{Sub(\varphi)}$$

A configuration is written

$$t \;\vdash\; Q(\left[\psi\right], \Phi) \tag{5.1}$$

where $t \in \mathcal{S}$, $Q \in \{\mathtt{E}, \mathtt{A}\}$, $\psi \in Sub(\varphi)$ and $\Phi \subseteq Sub(\varphi)$. With such a configuration we associate a player $p$ called the *path player*. This is $p := \forall$ if $Q = \mathtt{A}$ and $p := \exists$ if $Q = \mathtt{E}$. The path player's opponent $\overline{p}$ will also be called the *focus player*.

We will simply write $t \vdash Q(\Phi)$ if there is a $\psi \in \Phi$ in focus that does not need explicit mentioning.

The intuitive meaning of the configuration in (5.1) is as follows: The path player $p$ constructs a path $\pi$ in $\mathcal{T}$ starting with $t$ in a state-by-state manner. The focus player $\overline{p}$ tries to highlight a particular formula $\psi$ from the set of all formulas in this configuration s.t. $\pi \not\models \psi$ if $\overline{p} = \forall$, and $\pi \models \psi$ if $\overline{p} = \exists$.

In other words, if $Q = \mathtt{E}$, then player $\exists$ wants to show that there is a path $\pi = t \ldots$ s.t.

$$\pi \;\models\; \psi \wedge \bigwedge_{\varphi \in \Phi} \varphi$$

although player $\forall$ believes that $\pi \not\models \psi$. If $Q = \mathtt{A}$ then player $\forall$ wants to show that there is a path $\pi = t \ldots$ s.t.

$$\pi \;\not\models\; \psi \vee \bigvee_{\varphi \in \Phi} \varphi$$

although player $\exists$ believes that $\pi \models \psi$.

The *side formulas*, i.e. those that are not in focus, can be seen as an insurance for the focus player to redo a move that she has done before. This is necessary because the path player is allowed to choose the path stepwise along which a formula is examined.

At each configuration the set of side formulas together with the formula in focus can be understood as a disjunction, resp. conjunction, of formulas in case the path player is player $\forall$, resp. $\exists$. This is also justified by the equivalences

$$\mathtt{E}(\varphi \vee \psi) \equiv \mathtt{E}\varphi \vee \mathtt{E}\psi \qquad \text{and} \qquad \mathtt{A}(\varphi \wedge \psi) \equiv \mathtt{A}\varphi \wedge \mathtt{A}\psi$$

Each play of $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ begins with the configuration

$$s \vdash A(\boxed{\varphi})$$

Note that $\varphi$ as a starting formula is a state formula and therefore equivalent to $\mathtt{A}\varphi$. Rule $(\mathtt{A})$ would set the path player to $\forall$ anyway. If $\varphi = \mathtt{E}\psi$, as it will be in Example 57 later on, rule $(\mathtt{E})$ sets the path player to $\exists$ in the next move. This reflects the equivalence $\mathtt{AE}\psi \equiv \mathtt{E}\psi$.

From then on, the play proceeds according to the rules given in Figures 5.1 and 5.2. In addition to the rule schemes introduced in Chapter 4 we will use another one. A rule of the form

$$(r) \ \ \frac{C}{C'} \ \ p$$

is to be read as follows: player $p$ *can* play this rule in a configuration that matches $C$ but does not have to.

We will motivate the CTL$^*$ model checking game rules in the following. Suppose player $\exists$ is constructing a path $\pi$ and there is a $\varphi_0 \vee \varphi_1$ in the actual configuration. Since player $\exists$ believes that $\pi \models \varphi_0 \vee \varphi_1$ she can choose one of the disjuncts and the other one can be discarded. This is formalised in rules $(\mathtt{E}[\vee])$ and $(\mathtt{E}\vee)$.

Suppose there is a $\varphi_0 \wedge \varphi_1$. Player $\forall$ believes that $\pi \not\models \varphi_0 \wedge \varphi_1$ and has to pick the conjunct $\varphi_i$ that fails by setting the focus to it, see rule $(\mathtt{E}[\wedge])$. However, since he does not know which path player $\exists$ is going to choose, the other conjunct $\varphi_{1-i}$ is preserved. Consequently, if the conjunction was not in focus there is no choice at all, see rule $(\mathtt{E}\wedge)$. Later rule $(\mathtt{FC})$ will allow him to pick out $\varphi_{1-i}$ if player $\exists$ was constructing a

$$(\mathtt{A}[\wedge])\ \frac{s \vdash \mathtt{A}(\left[\varphi_0 \wedge \varphi_1\right], \Phi)}{s \vdash \mathtt{A}(\left[\varphi_i\right], \Phi)}\ \forall i \qquad (\mathtt{E}[\vee])\ \frac{s \vdash \mathtt{E}(\left[\varphi_0 \vee \varphi_1\right], \Phi)}{s \vdash \mathtt{E}(\left[\varphi_i\right], \Phi)}\ \exists i$$

$$(\mathtt{A}[\vee])\ \frac{s \vdash \mathtt{A}(\left[\varphi_0 \vee \varphi_1\right], \Phi)}{s \vdash \mathtt{A}(\left[\varphi_i\right], \varphi_{1-i}, \Phi)}\ \exists i \qquad (\mathtt{E}[\wedge])\ \frac{s \vdash \mathtt{E}(\left[\varphi_0 \wedge \varphi_1\right], \Phi)}{s \vdash \mathtt{E}(\left[\varphi_i\right], \varphi_{1-i}, \Phi)}\ \forall i$$

$$(\mathtt{A}\wedge)\ \frac{s \vdash \mathtt{A}(\left[\psi\right], \varphi_0 \wedge \varphi_1, \Phi)}{s \vdash \mathtt{A}(\left[\psi\right], \varphi_i, \Phi)}\ \forall i \qquad (\mathtt{E}\vee)\ \frac{s \vdash \mathtt{E}(\left[\psi\right], \varphi_0 \vee \varphi_1, \Phi)}{s \vdash \mathtt{E}(\left[\psi\right], \varphi_i, \Phi)}\ \exists i$$

$$(\mathtt{A}\vee)\ \frac{s \vdash \mathtt{A}(\left[\psi\right], \varphi_0 \vee \varphi_1, \Phi)}{s \vdash \mathtt{A}(\left[\psi\right], \varphi_0, \varphi_1, \Phi)} \qquad (\mathtt{E}\wedge)\ \frac{s \vdash \mathtt{E}(\left[\psi\right], \varphi_0 \wedge \varphi_1, \Phi)}{s \vdash \mathtt{E}(\left[\psi\right], \varphi_0, \varphi_1, \Phi)}$$

$$(\mathtt{A})\ \frac{s \vdash Q(\left[\mathtt{A}\varphi\right], \Phi)}{s \vdash \mathtt{A}(\left[\varphi\right])} \qquad (\mathtt{E})\ \frac{s \vdash Q(\left[\mathtt{E}\varphi\right], \Phi)}{s \vdash \mathtt{E}(\left[\varphi\right])}$$

$$(\not{q})\ \frac{s \vdash Q(\left[\varphi\right], q, \Phi)}{s \vdash Q(\left[\varphi\right], \Phi)}\ \overline{p} \qquad (\not{Q})\ \frac{s \vdash Q'(\left[\varphi\right], Q\psi, \Phi)}{s \vdash Q'(\left[\varphi\right], \Phi)}\ \overline{p}$$

$$(\mathtt{FC})\ \frac{s \vdash Q(\left[\varphi\right], \psi, \Phi)}{s \vdash Q(\left[\psi\right], \varphi, \Phi)}\ \overline{p} \qquad (\mathtt{X})\ \frac{s \vdash Q(\left[\mathtt{X}\varphi_0\right], \mathtt{X}\varphi_1, \ldots, \mathtt{X}\varphi_k)}{t \vdash Q(\left[\varphi_0\right], \varphi_1, \ldots, \varphi_k)}\ p\ \ s \to t$$

Figure 5.1: The model checking games rules for CTL$^*$.

$$([\mathtt{U}])\ \frac{s \vdash Q(\left[\varphi \mathtt{U} \psi\right], \Phi)}{s \vdash Q(\left[\psi \vee (\varphi \wedge \mathtt{X}(\varphi \mathtt{U} \psi))\right], \Phi)}$$

$$([\mathtt{R}])\ \frac{s \vdash Q(\left[\varphi \mathtt{R} \psi\right], \Phi)}{s \vdash Q(\left[\psi \wedge (\varphi \vee \mathtt{X}(\varphi \mathtt{R} \psi))\right], \Phi)}$$

$$(\mathtt{U})\ \frac{s \vdash Q(\left[\chi\right], \varphi \mathtt{U} \psi, \Phi)}{s \vdash Q(\left[\chi\right], \psi \vee (\varphi \wedge \mathtt{X}(\varphi \mathtt{U} \psi)), \Phi)}$$

$$(\mathtt{R})\ \frac{s \vdash Q(\left[\chi\right], \varphi \mathtt{R} \psi, \Phi)}{s \vdash Q(\left[\chi\right], \psi \wedge (\varphi \vee \mathtt{X}(\varphi \mathtt{R} \psi)), \Phi)}$$

Figure 5.2: The unfolding rules for the CTL$^*$ model checking games.

path on which $\varphi_i$ actually holds. Rules $(\mathtt{A}[\wedge])$, $(\mathtt{A}[\vee])$, $(\mathtt{A}\wedge)$, and $(\mathtt{A}\vee)$ cover the dual situations.

Once the focus player has decided to prove, resp. refute, a path quantified formula a new path $\pi$ needs to be chosen. The new path player depends on the new path quantifier. The set of side formulas will be discarded since they were only relevant for the old path, see rules $(\mathtt{A})$ and $(\mathtt{E})$.

Rule $(\not{q})$ allows the path player to discard propositions if they do not prove, resp. refute, the current disjunction, resp. conjunction, of formulas. The same is possible for path quantified formulas using rule $(\not{Q})$.

Using the fixed point characterisation of the temporal operators $\mathtt{U}$ and $\mathtt{R}$ they simply get unfolded with rules $([\mathtt{U}])$, $([\mathtt{R}])$, $(\mathtt{U})$, and $(\mathtt{R})$.

Applying these rules consecutively can result in a configuration in which every formula is of the form $\mathtt{X}\psi$, i.e. speaks about the next state of the underlying path. Thus, the path player has to choose the next state and the according formulas are examined on this one, see rule $(\mathtt{X})$. Note that the $p$ in this rule denotes the actual path player which depends on $Q$.

Finally, the focus player $\overline{p}$ – again depending on the $Q$ of the actual configuration – is allowed to reset the focus at any moment of the play using rule $(\mathtt{FC})$. This might be necessary whenever the path player reveals a further state of the path he or she is going to choose. The focus player is always given the chance to reset the focus, particularly *before* a play is finished. Note that there are situations in which a play can get stuck if he does not change it.

**Definition 55** A configuration is called *terminal* if it is of the form

$$s \vdash Q(\left[q\right], \Phi)$$

for some $q \in \mathcal{P}$ and some $\Phi$, and the focus player refuses or is unable to use rule $(\mathtt{FC})$. If $\Phi = \emptyset$ then the focus player is unable to use rule $(\mathtt{FC})$. Moreover, remember that he or she is given the chance to reset the focus after every application of another rule. Thus, they refuse to change the focus if they do not make use of this possibility. This is useful if the configuration at hand makes the focus player win the current play. Therefore there is no need to change the focus.

**Definition 56** A formula $\varphi \mathtt{U} \psi$ is called *present* in a configuration $s \vdash Q(\Phi)$ iff

$$\{\ \varphi \mathtt{U} \psi,\ \psi \vee (\varphi \wedge \mathtt{X}(\varphi \mathtt{U} \psi)),\ \varphi \wedge \mathtt{X}(\varphi \mathtt{U} \psi),\ \mathtt{X}(\varphi \mathtt{U} \psi)\ \} \ \cap \ \Phi \ \neq \ \emptyset$$

A $\varphi \mathtt{R} \psi$ is called *present* in a configuration $s \vdash Q(\Phi)$ iff

$$\{\ \varphi \mathtt{R} \psi,\ \psi \wedge (\varphi \vee \mathtt{X}(\varphi \mathtt{R} \psi)),\ \varphi \vee \mathtt{X}(\varphi \mathtt{R} \psi),\ \mathtt{X}(\varphi \mathtt{R} \psi)\ \} \ \cap \ \Phi \ \neq \ \emptyset$$

Player $\forall$ wins the play $C_0, C_1, \ldots$ of $\mathcal{G}_{\mathcal{T}}(s, \varphi_0)$ iff

1. it reaches a terminal configuration $C_n = t \vdash Q(\left[q\right], \Phi)$ and $q \notin L(t)$, or
2. there is a $\varphi \mathtt{U} \psi \in Sub(\varphi_0)$ and infinitely many configurations $C_{i_0}, C_{i_1}, \ldots$ s.t. for every $j \in \mathbb{N}$:

- $C_{i_j} = t_{i_j} \vdash \mathtt{E}(\Phi)$ for some $t_{i_j} \in \mathcal{S}$ and $\Phi$, and
- $\left[\varphi \mathtt{U} \psi\right]$ is in focus in every $C_{i_j}$, and
- after $C_{i_0}$ player $\forall$ has not used rule $(\mathtt{FC})$, or

3. there are infinitely many configurations $C_{i_0}, C_{i_1}, \ldots$ s.t. for all $j \in \mathbb{N}$ there are $t_{i_j} \in \mathcal{S}$ and $\Phi \subseteq Sub(\varphi_0)$ and $C_{i_j} = t_{i_j} \vdash \mathtt{A}(\Phi)$, and either
   - player $\exists$ has used rule $(\mathtt{FC})$ infinitely often, or
   - there is a $\varphi \mathtt{U} \psi$ that is present and in focus in infinitely many $C_{i_j}$.

Player $\exists$ wins the play $C_0, C_1, \ldots$ of $\mathcal{G}_{\mathcal{T}}(s, \varphi_0)$ iff

4. it reaches a terminal configuration $C_n = t \vdash Q(\left[q\right], \Phi)$ and $q \in L(t)$, or
5. there is a $\varphi \mathtt{R} \psi \in Sub(\varphi_0)$ and infinitely many configurations $C_{i_0}, C_{i_1}, \ldots$ s.t. for every $j \in \mathbb{N}$:
   - $C_{i_j} = t_{i_j} \vdash \mathtt{A}(\Phi)$ for some $t_{i_j} \in \mathcal{S}$ and $\Phi$, and
   - $\left[\varphi \mathtt{R} \psi\right]$ is in focus in every $C_{i_j}$, and
   - after $C_{i_0}$ player $\exists$ has not used rule $(\mathtt{FC})$, or
6. there are infinitely many configurations $C_{i_0}, C_{i_1}, \ldots$ s.t. for all $j \in \mathbb{N}$ there are $t_{i_j} \in \mathcal{S}$ and $\Phi \subseteq Sub(\varphi_0)$ and $C_{i_j} = t_{i_j} \vdash \mathtt{E}(\Phi)$, and either
   - player $\forall$ has used rule $(\mathtt{FC})$ infinitely often, or
   - there is a $\varphi \mathtt{R} \psi$ that is present and in focus in infinitely many $C_{i_j}$.

The motivation for the conditions for infinite plays is the following. Condition 2 is winning for player $\forall$ because in this situation he managed to show the regeneration of an $\mathtt{U}$ formula along a path that player $\exists$ chose. Condition 3 is winning for him since player $\exists$ failed to show the regeneration of a $\mathtt{R}$ formula along a path he chose. The conditions for player $\exists$ are dual.

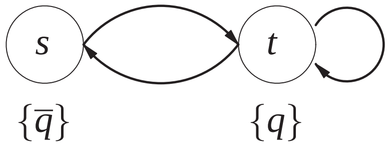


Figure 5.3: The transition system for Example 57.

To illustrate the games we give an example that makes use of an abbreviated $\mathtt{G}$ formula. The simplified game rules for this construct and for an $\mathtt{F}$ formula can easily be derived from rules $([\mathtt{U}])$, $([\mathtt{R}])$, $(\mathtt{U})$ and $(\mathtt{R})$ and are

$$\frac{s \vdash Q(\left[\mathtt{F}\varphi\right], \Phi)}{s \vdash Q(\left[\varphi \vee \mathtt{XF}\varphi\right], \Phi)} \qquad \frac{s \vdash Q(\left[\mathtt{G}\varphi\right], \Phi)}{s \vdash Q(\left[\varphi \wedge \mathtt{XG}\varphi\right], \Phi)}$$

$$\frac{s \vdash Q(\left[\mathtt{X}\psi\right], \mathtt{F}\varphi, \Phi)}{s \vdash Q(\left[\mathtt{X}\psi\right], \varphi \vee \mathtt{XF}\varphi, \Phi)} \qquad \frac{s \vdash Q(\left[\mathtt{X}\psi\right], \mathtt{G}\varphi, \Phi)}{s \vdash Q(\left[\mathtt{X}\psi\right], \varphi \wedge \mathtt{XG}\varphi, \Phi)}$$

**Example 57** Let $\mathcal{T}$ be the transition system of Figure 5.3. The formula under consideration is

$$\varphi \;:=\; \mathtt{E}(\overline{q}\mathtt{U}(\mathtt{G}q))$$

The property described by $\varphi$ is: "There exists a path with a finite prefix and an infinite suffix. On the prefix $q$ never holds, on the suffix it always does." Confer also Example 12. $\mathcal{T}$ with starting state $s$ satisfies $\varphi$. The game tree for player $\exists$ is depicted in Figure 5.4. Note that in the second configuration player $\exists$ becomes the path player which makes player $\forall$ the focus player.

Since plays are of infinite length we can only depict them partially. Here, all plays feature a repeating configuration. Later we will prove that winning strategies are history-free. Thus, we can argue in the following way.

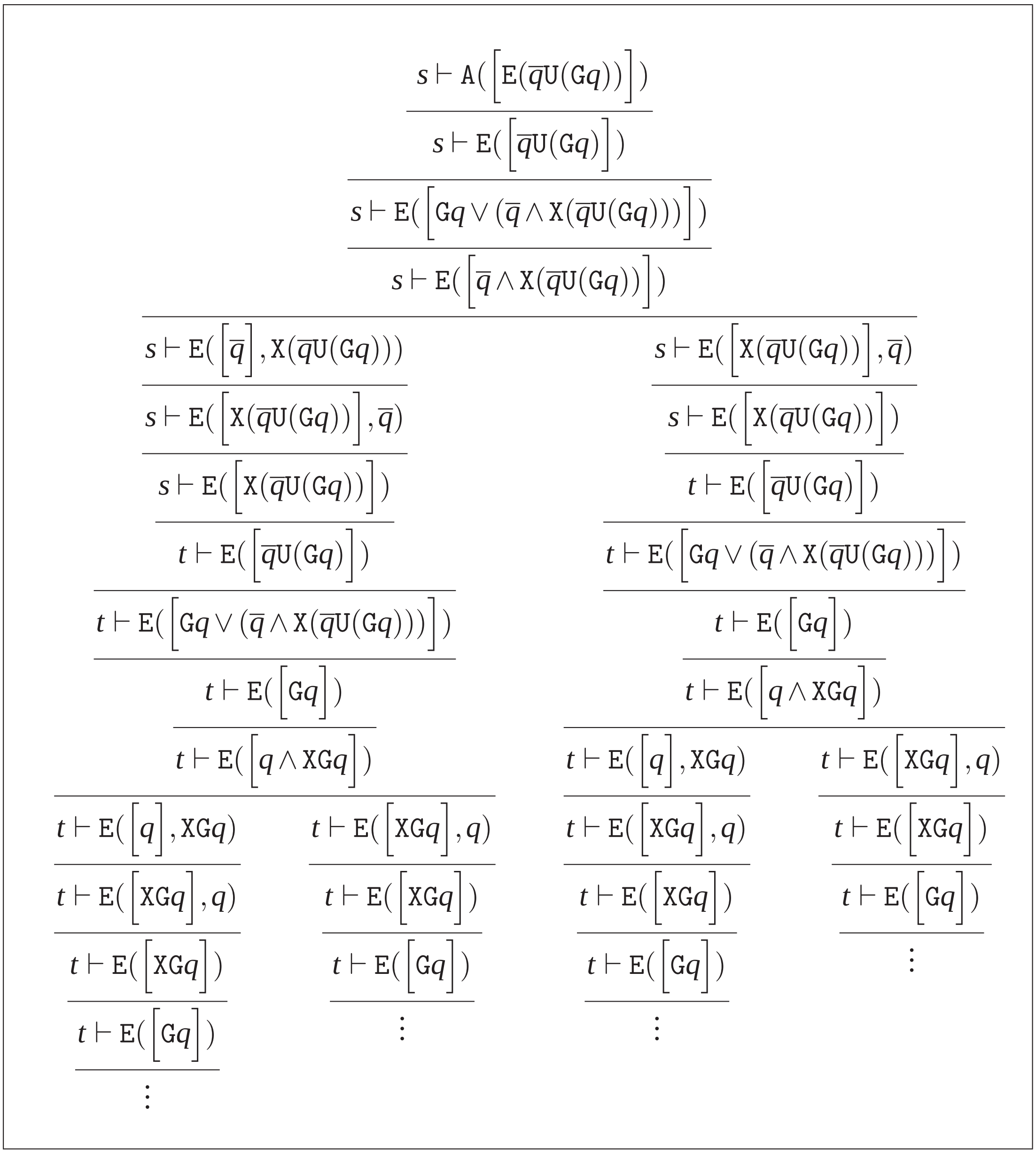


Figure 5.4: The game tree for player ∃ of Example 57.

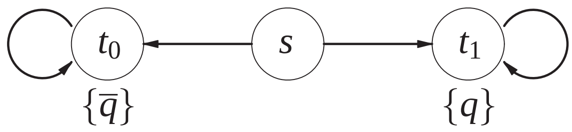


Figure 5.5: The transition system for Example 58.

Player $\exists$ wins the plays that proceed like the leftmost branch or the second from the right with winning condition 6 since player $\forall$ changes focus. She wins the others, i.e. the rightmost path and the second from the left with winning condition 6 as well. Here, player $\forall$ is not able to show the regeneration of an $\mathtt{U}$ formula along the path player $\exists$ selects.

Before we proceed to prove correctness of the games we give two further examples that illustrate why a configuration in the model checking game needs to be a set of formulas and, moreover, why the focus on this set is needed, too.

**Example 58** Consider the CTL$^*$ formula

$$\varphi \ := \ \mathtt{A}(\mathtt{X}q \vee \mathtt{X}\overline{q})$$

from Example 12. $\varphi$ says that every path's next state is labelled with either $q$ or $\overline{q}$. $\varphi$ is a tautology, so player $\forall$ should not win the game on any transition system, in particular the one shown in Figure 5.5. Note that the labelling of $s$ is unimportant for this example.

However, if we require configurations to contain one formula only, player $\exists$ cannot win $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ anymore. This is because player $\exists$ has to choose one of the disjuncts *before* player $\forall$ chooses a transition from $s$ to $t_i$, $i \in \{0,1\}$. If player $\exists$ selects $\mathtt{X}q$ for example he would choose $t_0$ and vice versa. Thus, configurations containing one formula only can make the path player too strong provided paths are chosen stepwise.

**Example 59** This example justifies the use of the focus structure on sets of formulas. Consider

$$\varphi \ := \ \mathtt{E}(\mathtt{F}q \wedge \mathtt{GF}q)$$

$$
\dfrac{\dfrac{\vdots}{s \vdash \mathtt{E}(\mathtt{F}q, \mathtt{XGF}q)}}{\dfrac{\vdots}{s \vdash \mathtt{E}(\mathtt{F}q, \mathtt{XGF}q)}}
\qquad\qquad
\dfrac{\dfrac{\vdots}{s \vdash \mathtt{E}(\mathtt{F}q, \mathtt{GF}q)}}{\dfrac{\vdots}{s \vdash \mathtt{E}(\mathtt{F}q, \mathtt{GF}q)}}
$$

Figure 5.6: The plays without focus of Example 59.

from Example 12 and the two following transition systems. $\mathcal{T}_1$ and $\mathcal{T}_2$ consist of one state $s$ and one transition $s \to s$ only. The labelling function of $\mathcal{T}_1$ assigns $q$ to $s$ whereas the one of $\mathcal{T}_2$ assigns $\overline{q}$ to $s$.

$\mathcal{T}_1, s \models \varphi$ but $\mathcal{T}_2, s \not\models \varphi$ since $\varphi$ postulates the existence of a path which visits a state satisfying $q$ infinitely often. However, without an additional structure like the focus on the set of formulas the games $\mathcal{G}_{\mathcal{T}_1}(s,\varphi)$ and $\mathcal{G}_{\mathcal{T}_2}(s,\varphi)$ would look like the ones depicted in Figure 5.6.

The difference between $\mathcal{G}_{\mathcal{T}_1}(s,\varphi)$, depicted on the left, and $\mathcal{G}_{\mathcal{T}_2}(s,\varphi)$ is the generation of $\mathtt{F}q$. In the first case it is generated from the $\mathtt{XGF}q$ above, in the second it regenerates itself. Hence, in that case player $\forall$ can keep the focus on $\mathtt{F}q$ and explicitly show this regeneration.

## Correctness

**Fact 60** *Rules* $(\mathtt{A}\wedge)$, $(\mathtt{E}\vee)$, $(\not{q})$, $(\not{Q})$ *and* $(\mathtt{X})$ *reduce the size of the actual configuration. Rules* $(\mathtt{A}\vee)$ *and* $(\mathtt{E}\wedge)$ *reduce the number of connectives in the actual configuration. Rules* $(\mathtt{A}[\wedge])$, $(\mathtt{E}[\vee])$, $(\mathtt{A}[\vee])$ *and* $(\mathtt{E}[\wedge])$ *reduce the size of the formula in focus and, hence, the size of the entire configuration. Rules* $(\mathtt{A})$ *and* $(\mathtt{E})$ *reduce the number of path quantifiers in the actual configuration and, hence, its size. Rules* $([\mathtt{U}])$, $([\mathtt{R}])$, $(\mathtt{U})$ *and* $(\mathtt{R})$ *increase the size of the actual configuration. Rule* $(\mathtt{FC})$ *is the only one that preserves both the size and the number of connectives in a configuration.*

**Lemma 61** *The path player can only change a finite number of times in a play.*

PROOF The path player can only change with the rules $(\mathtt{A})$ and $(\mathtt{E})$. But these discard the entire set of present sideformulas. Let $Q_1, Q_2 \in \{\mathtt{A}, \mathtt{E}\}$ with $Q_1 \neq Q_2$. Suppose $Q_1\varphi$ is in focus and the path player changes. If after that $Q_2\psi$ gets into focus to change the path player again, then $Q_2\psi$ is a genuine subformula of $\varphi$ and thus is shorter than $Q_1\varphi$. But the formula to start with is of finite length. Hence, this can only occur finitely often. ■

Note that Lemma 61 can be generalised slightly by considering all applications of rule $(\mathtt{A})$ and $(\mathtt{E})$ and not just those that change the path player.

**Theorem 62** *Every play has a uniquely determined winner.*

PROOF A play is either finite or infinite. It is only finite if it ends in a terminal configuration

$$s \;\vdash\; Q(\left[q\right], \Phi)$$

Then either $q \in L(s)$ in which case player $\exists$ wins or $q \notin L(s)$ in which case player $\forall$ wins.

Consider now an infinite play. According to Lemma 61, the path player can only change finitely many times, therefore in every infinite sequence of configurations one of the players can only occur finitely many times as the path player. Thus we can speak of *the* path player for a particular infinite play as the player who is almost always the path player in a configuration. Note that this also determines *the* focus player.

Moreover, for a play to be of infinite length there must be a formula of the form $\varphi\mathtt{U}\psi$ or $\varphi\mathtt{R}\psi$ that gets regenerated infinitely many times. Note that according to Fact 60, at least one of the rules $([\mathtt{U}])$, $([\mathtt{R}])$, $(\mathtt{U})$ and $(\mathtt{R})$ must be played infinitely often since the starting configuration is of finite size and all other rules reduce at least a component of the configuration. But then there are only finitely many possibilities for a $\mathtt{U}$ or $\mathtt{R}$ formula to get unfolded with these rules.

Thus, in every infinite play there is a $\varphi\mathtt{U}\psi$ or a $\varphi\mathtt{R}\psi$ that is present infinitely many times. Now, the focus player can change the focus finitely or infinitely many times. In

| focus player | (FC) infinitely often | present formula | winner | condition |
|---|---|---|---|---|
| $\forall$ | no | $\varphi \mathtt{U} \psi$ | $\forall$ | 2 |
| | | $\varphi \mathtt{R} \psi$ | $\exists$ | 6 |
| | yes | | $\exists$ | 6 |
| $\exists$ | no | $\varphi \mathtt{R} \psi$ | $\exists$ | 5 |
| | | $\varphi \mathtt{U} \psi$ | $\forall$ | 3 |
| | yes | | $\forall$ | 3 |

Figure 5.7: The winning conditions for the CTL$^*$ model checking games.

the latter case no U or R formula ever needs to occur in focus since the focus player can always avoid it. However, in the former case, a $\varphi \mathtt{U} \psi$ or a $\varphi \mathtt{R} \psi$ must almost always be present and in focus for otherwise Fact 60 shows that the size of the formula in focus would infinitely often get reduced.

Figure 5.7 depicts this as a nested case distinction and shows which winning condition determines the winner in which case. Every possible infinite play is covered by one of the cases. The first case distinction is on the player who eventually becomes and remains the focus player. The second is on the question of whether he or she uses rule (FC) infinitely often or not. Finally, the third case split concerns the question of whether there is a $\varphi \mathtt{U} \psi$ or a $\varphi \mathtt{R} \psi$ that is present infinitely often. Note that this becomes irrelevant if the focus is changed infinitely often since this behaviour determines the focus player as the loser of the play already. ■

The next result reestablishes an observation from [EL87] in terms of games: CTL$^*$ model checking can be polynomially reduced to LTL model checking. However, it needs a technical definition first.

**Definition 63** Let $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ with $s \in \mathcal{S}$. A *block* of a game graph for a game $\mathcal{G}_{\mathcal{T}}(s, \varphi_0)$ is a subset $\mathcal{B} \subseteq \mathcal{C}$ of the configurations of $\mathcal{G}_{\mathcal{T}}(s, \varphi_0)$ s.t. either

- for all $C \in \mathcal{B}$: $C = t \vdash \mathtt{A}(\Phi)$ for some $t \in \mathcal{S}$ and $\Phi \subseteq Sub(\varphi_0)$, or
- for all $C \in \mathcal{B}$: $C = t \vdash \mathtt{E}(\Phi)$ for some $t \in \mathcal{S}$ and $\Phi \subseteq Sub(\varphi_0)$.

**Lemma 64** *Let $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ with $s \in \mathcal{S}$. The game graph for $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ can be partitioned into a finite set of blocks $\mathcal{B}_1, \ldots, \mathcal{B}_n$, s.t. every play never leaves a block $i$ into a block $j$ with $j < i$. Moreover, $n \leq \frac{|\varphi|}{2}$.*

PROOF We sketch an algorithm that finds this partition. It is basically the same as the standard algorithm for finding a topological order on the set of connected components of a directed graph.

At the beginning let $i := 1$ and add $C_0$ to $\mathcal{B}_i$. Do the same repeatedly with its successor configurations unless one of them is reached via an application of rule $(\mathtt{E})$ or $(\mathtt{A})$. If so, then increase $i$ by 1 and continue with the respective successors.

According to Lemma 61, on every path through the game graph the path player eventually remains the same, in fact no further application of rule $(\mathtt{E})$ or $(\mathtt{A})$ is encountered. Note that even if the underlying transition system is not image-finite, only a finite number of blocks is needed to cover the entire game graph. This is because an infinite branching in the transition system is only reflected in the game graph at a position in which rule $(\mathtt{X})$ is played. However, there the actual configuration and its successors are put into the same block.

No transitions from a block with a higher index to one with a lower index are possible as they would correspond to an application of a game rule that strictly increases the number of path quantifiers in a configuration. According to Fact 60, this is impossible since there is no such rule.

Finally, $\varphi$ can contain at most $\frac{|\varphi|}{2}$ irredundant path quantifiers because of the equivalence $Q_1 Q_2 \psi \equiv Q_2 \psi$ for all $Q_1, Q_2 \in \{\mathtt{A}, \mathtt{E}\}$. ■

Rules $(\not{Q})$ and $(\not{q})$ suggest that path quantified formulas bear a similarity to propositions in the way they are treated in a game. Indeed, since an application of rule $(\mathtt{A})$ or $(\mathtt{E})$ discards all present sideformulas, processing path quantified formulas can be seen as starting a new subgame. Each of these subgames can be regarded as a game for an LTL formula, either universally or existentially path quantified.

**Definition 65** Take a state $s$ of a transition system $\mathcal{T}$ and an ordered sequence $\varphi_1, \ldots, \varphi_n$ of formulas. Assume that

$$s \not\models \mathtt{E}(\varphi_1 \wedge \ldots \wedge \varphi_n)$$

i.e. no path $\pi$ starting with $s$ satisfies all $\varphi_i$. With each $\varphi_i$ and each such state $s$ we associate a set $P'_{\varphi_i}(s)$ of finite prefixes of paths starting with $s$ in the following way. Let $\sigma = s \ldots t$ be a finite sequence of states in $\mathcal{T}$. Since $\mathcal{T}$ is assumed to be total, $\sigma$ is not maximal.

$$\sigma \in P'_{\varphi_i}(s) \quad \text{iff} \quad \text{there is a path } \pi = \sigma\pi' \text{ s.t. } \pi \not\models \varphi_i$$

Let $P_i(s) \subseteq P'_i(s)$ be defined by

$$\sigma \in P_{\varphi_i}(s) \quad \text{iff} \quad \sigma \in P'_{\varphi_i}(s) \text{ and for all } j < i : \sigma \notin P'_{\varphi_j}(s)$$

Informally, $P'_{\varphi_i}(s)$ consist of all finite prefixes of a path starting in $s$ which can be extended to an infinite path not satisfying $\varphi_i$. $P_{\varphi_i}(s)$ is the subset of $P'_{\varphi_i}(s)$ containing all those finite prefixes that do not occur in a $P'_{\varphi_j}(s)$ for a smaller index already. This makes

$$\{ \, P_{\varphi_i}(s) \mid i \in \{1, \ldots, n\} \, \}$$

a partition on the set of finite sequences of paths starting in $s$.

Next we show that a finite sequence of states $\sigma$ can never occur in a set with a smaller index than those containing prefixes of $\sigma$. This gives the focus player an optimal strategy in a CTL$^*$ model checking game.

**Lemma 66** *Take two formulas $\varphi_i, \varphi_j$ of an ordered sequence of formulas. Let $\sigma_1, \sigma_2$ be finite prefixes of a path starting in $s$, s.t. $\sigma_2 = \sigma_1\sigma$ for some $\sigma$. If $\sigma_1 \in P_{\varphi_i}(s)$ and $\sigma_2 \in P_{\varphi_j}(s)$ then $j \geq i$.*

PROOF Suppose $\sigma_1 \in P_{\varphi_i}(s)$ for some $i$, $\sigma_2 \in P_{\varphi_j}(s)$ for some $j$ and $j < i$. By definition $\sigma_2$ can be extended to a path $\pi = \sigma_2 \ldots$ s.t. $\pi \not\models \varphi_j$ for the according $\varphi_j$. But then $\sigma_1$ can be extended to $\pi$ as well and therefore $\sigma_1 \in P'_{\varphi_j}(s)$. Thus, $\sigma_1 \in P_{\varphi_i}(s)$ is impossible since $i > j$ is assumed. ∎

The next lemma shows that it does not matter whether the sets $P_\varphi(s)$ are calculated at the beginning and the focus is set according to these sets or whether they are recalculated after every application of rule $(\mathtt{X})$.

**Lemma 67** *Take formulas $\mathtt{X}\varphi_1,\ldots,\mathtt{X}\varphi_n$ and two states $s,t$ of a transition system $\mathcal{T}$ s.t. $s \to t$. Consider the sets $P_{\mathtt{X}\varphi_1}(s),\ldots,P_{\mathtt{X}\varphi_n}(s)$ and $P_{\varphi_1}(t),\ldots,P_{\varphi_n}(t)$. Let $\sigma' = t\ldots$ be some finite sequence of states and $\sigma = s\sigma'$. If $\sigma \in P_{\mathtt{X}\varphi_i}(s)$ and $\sigma' \in P_{\varphi_j}(t)$ then $j \geq i$.*

PROOF Suppose $\sigma \in P_{\mathtt{X}\varphi_i}(s)$, $\sigma' \in P_{\varphi_j}(t)$ and $j < i$. Then $\sigma'$ can be extended to a $\pi'$ s.t. $\pi' \not\models \varphi_j$. But then take $\pi := s\pi'$. Clearly, $\pi \not\models \mathtt{X}\varphi_j$. Therefore $\sigma \in P_{\mathtt{X}\varphi_j}(s)$ which contradicts the assumption that $\sigma \in P_{\mathtt{X}\varphi_i}(s)$. ■

The main correctness proof of the CTL$^*$ model checking games proceeds by induction on the path quantifier depth of the input formula. The next two theorems form the induction base case, i.e. we will prove soundness and completeness for input formulas $\varphi_0$ of the form $\mathtt{A}\varphi$ or $\mathtt{E}\varphi$ where $\varphi$ is a pure linear time formula.

**Theorem 68 (Soundness)** *Let $\mathcal{T} = (\mathcal{S},\to,L)$ with $s_0 \in \mathcal{S}$ and $\varphi_0 \in CTL^*$ s.t. $\varphi_0 = Q\varphi$ for a $Q \in \{\mathtt{E},\mathtt{A}\}$ and a $\varphi$ not containing any path quantifiers. If $s_0 \not\models \varphi_0$ then player $\forall$ wins $\mathcal{G}_{\mathcal{T}}(s_0,\varphi_0)$.*

PROOF There are two distinguishable cases depending on the path quantifier of $\varphi_0$. First, let $\varphi_0 = \mathtt{A}\varphi$. This means there is a path $\pi = s_0 s_1 \ldots$ s.t. $\pi \not\models \varphi$. We construct a game tree for player $\forall$ using this path. Note that disjuncts are preserved and conjuncts are chosen since player $\forall$ is the path player, i.e. the set of formulas of the configuration at hand is interpreted disjunctively.

Whenever rule $(\mathtt{X})$ has to be played player $\forall$ chooses the next state $s_i$ of $\pi$. It is not hard to see that the following invariant holds true: if the play visits a configuration $s_i \vdash \mathtt{A}(\Phi)$ then for all $\psi \in \Phi$: $\pi^i \not\models \psi$.

Remember that at the beginning there is only $\varphi$ which is not fulfilled by $\pi$. Unfolding $\mathtt{U}$ and $\mathtt{R}$ formulas does not change this. Both disjuncts of a disjunction are not satisfied, otherwise the disjunction would be satisfied on the remainder $\pi^i$ of the path chosen at the beginning. And if

$$\pi^i \quad \not\models \quad \mathtt{X}\psi_1 \vee \ldots \vee \mathtt{X}\psi_n$$

then

$$\pi^{i+1} \not\models \psi_1 \vee \ldots \vee \psi_n$$

Thus, applications of rule $(\mathtt{X})$ preserve this invariant. Whenever a conjunction occurs he chooses the false conjunct. If both are false he chooses the smaller one.

It is impossible for player $\exists$ to win with winning condition 4 since this requires a formula to be present that is fulfilled on the remaining path.

Suppose player $\exists$ wins a play of this game with her winning condition 5, i.e. she eventually keeps the focus on a $\chi\mathtt{R}\psi$. More precisely, there is a configuration

$$C = s_i \vdash \mathtt{A}(\left[\chi\mathtt{R}\psi\right], \Phi)$$

after which she does not use rule $(\mathtt{FC})$ anymore. According to the invariant described above, $\pi^i \not\models \chi\mathtt{R}\psi$. By Lemma 10 of Chapter 2 there is a $k \in \mathbb{N}$ s.t.

$$\pi^i \not\models \chi\mathtt{R}^k\psi$$

At some point, player $\forall$ will choose the next state $s_{i+1}$ of $\pi$ when playing rule $(\mathtt{X})$. Since player $\exists$ keeps the focus on $\chi\mathtt{R}\psi$ it will still be present in the configuration

$$s_{i+1} \vdash \mathtt{A}(\left[\chi\mathtt{R}\psi\right], \Phi')$$

and

$$\pi^{i+1} \not\models \chi\mathtt{R}\psi$$

holds by the invariant. But

$$\chi\mathtt{R}^k\psi \equiv \psi \wedge (\chi \vee \mathtt{X}(\chi\mathtt{R}^{k-1}\psi))$$

Remember that in case of two false conjuncts player $\forall$ chooses the smaller one. Clearly, $\psi$ is smaller than $\chi \vee \mathtt{X}(\chi\mathtt{R}\psi)$. But we can assume that he did not choose $\psi$ since it would immediately contradict the assumption that player $\exists$ wins with her winning condition 5. Therefore we can assume the other conjunct to be false. Then, by definition of the approximants

$$\pi^{i+1} \not\models \chi\mathtt{R}^{k-1}\psi$$

This argument can be iterated until the state $s_{i+k}$ is reached with the condition

$$\pi^{i+k} \not\models \chi \mathtt{R}^0 \psi$$

But $\chi\mathtt{R}^0\psi \equiv \mathtt{tt}$ which is satisfied by $\pi^{i+k}$. We conclude that player $\exists$ cannot win with her winning condition 5.

Player $\exists$ cannot win a play of this game with her winning condition 6 since it requires her to be the path player which she is not.

The second case is $\varphi_0 = \mathtt{E}\varphi$. Since $s_0 \not\models \varphi_0$, every path $\pi = s_0 \ldots$ does not satisfy $\varphi$. Now, player $\exists$ is the path player and thus, conjuncts are preserved and disjuncts are chosen. Setting the focus is the only thing that player $\forall$ has control over. We use Lemma 66 as a basis for player $\forall$'s strategy.

At any point in the play, player $\exists$ will have outlined a finite prefix $\sigma = s_0 \ldots s_i$ of a path starting with $s_0$. The invariant we use in this case is the following: there is always at least one $\psi_j$ in the actual configuration $s_i \vdash \mathtt{E}(\Phi)$ s.t. $P_{\psi_j}(s_i) \neq \emptyset$.

At the beginning $\varphi$ is such a formula. Note that no path satisfies $\varphi$. Thus, if a disjunction occurs player $\exists$ cannot choose a disjunct and a corresponding path that satisfies it. If a conjunction occurs then one of the conjuncts must be false regardless of which path player $\exists$ is going to follow. Unfolding $\mathtt{U}$ and $\mathtt{R}$ formulas preserves this invariant.

Hence, at any stage $s_i \vdash \mathtt{E}(\psi_1, \ldots, \psi_k)$ of the play there is at least one $\psi_j \in \Phi$ s.t. player $\exists$ cannot find a path $\pi = s_i \ldots$ with $\pi \models \psi_j$. In other words, $P_{\psi_j}(s_i) \neq \emptyset$.

Player $\forall$ sets the focus to this $\psi_j$. Since at any later point player $\exists$ will have outlined an extension of $s_0 \ldots s_i$, Lemma 66 applies. It shows that player $\forall$ only needs to change the focus finitely many times because there are only finitely many subformulas of $\varphi$ and, hence, only finitely many sets $P_{\psi_j}(s)$ for any $s \in \mathcal{S}$. Remember the lemma says that player $\forall$ can change the focus in such a way that the index of $P_{\psi_j}$ always gets increased.

This shows that player $\exists$ cannot win a play with the first part of her winning condition 6 because this requires player $\forall$ to change the focus infinitely often. To avoid defeat with the second part of this winning condition, player $\forall$ must eventually keep the focus on a

$\chi \mathtt{U} \psi$. Again, if the focus remains on a particular formula then it must be a regenerating one. Thus, it can only be a $\chi \mathtt{U} \psi$ or $\chi \mathtt{R} \psi$. Suppose the latter is true, i.e. there is a configuration

$$s_i \;\vdash\; \mathtt{E}(\left[\chi \mathtt{R} \psi\right], \Phi)$$

after which player $\forall$ does not change focus anymore. As in the first case of this proof one can show that there is a path $\pi = s_i \ldots$ s.t. $\pi \models \chi \mathtt{R} \psi$. Therefore, $\chi \mathtt{R} \psi$ was not a false formula, and there must have been a different one that player $\forall$ could have set the focus to.

Player $\exists$ cannot win with her winning condition 4 since it requires a $q$ to be present in a terminal configuration $s_i \vdash \mathtt{E}(\left[q\right], \Phi)$ s.t. $q \in L(s_i)$. But then $P_q(s_i) = \emptyset$ since every extension of this finite sequence of states trivially satisfies $q$ at $s_i$. Therefore player $\forall$ would not have ended up with the focus on $q$ in the first place.

Player $\exists$ cannot win a play with winning condition 5 either, since it requires player $\forall$ to be the path player which he is not.

Since player $\forall$ has strategies for both cases of path quantified formulas that disable winning plays for player $\exists$ he must win the game $\mathcal{G}_{\mathcal{T}}(s_0, \varphi_0)$. ■

Completeness of the CTL$^*$ model checking games can be proved using the duality principle Theorem 39, and the soundness Theorem 68. However, since this is based on Definition 65 and Lemma 66 it is necessary to dualise these first.

**Definition 69** Take a state $s$ of a transition system $\mathcal{T}$ and an ordered sequence $\varphi_1, \ldots, \varphi_n$ of satisfiable formulas. Assume that $s \models \mathtt{A}(\varphi_1 \vee \ldots \vee \varphi_n)$, i.e. every path $\pi$ starting with $s$ satisfies at least one $\varphi_i$. With each $\varphi_i$ and each such state $s$ we associate a set $P'_{\varphi_i}(s)$ of finite prefixes of paths starting with $s$ in the following way. Let $\sigma = s \ldots t$ be a finite sequence of states in $\mathcal{T}$.

$$\sigma \in P'_{\varphi_i}(s) \quad \text{iff} \quad \text{there is a path } \pi = \sigma\pi' \text{ s.t. } \pi \models \varphi_i$$

Let $P_i(s) \subseteq P'_i(s)$ be defined by

$$\sigma \in P_{\varphi_i}(s) \quad \text{iff} \quad \sigma \in P'_{\varphi_i}(s) \text{ and for all } j < i : \sigma \notin P'_{\varphi_j}(s)$$

Here, $P'_{\varphi_i}(s)$ consist of all finite prefixes of a path starting in $s$ which can be extended to an infinite path satisfying $\varphi_i$. Again, $P_{\varphi_i}(s)$ is its subset containing only those elements that are not included in a set with a smaller index.

The next lemma is proved exactly in the same way as Lemma 66 for the soundness part.

**Lemma 70** *Take two formulas $\varphi_i, \varphi_j$ of an ordered sequence of formulas. Let $\sigma_1, \sigma_2$ be finite prefixes of a path starting in $s$, s.t. $\sigma_2 = \sigma_1\sigma$ for some $\sigma$. If $\sigma_1 \in P_{\varphi_i}(s)$ and $\sigma_2 \in P_{\varphi_j}(s)$ then $j \geq i$.*

**Theorem 71 (Completeness)** *Let $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ with $s_0 \in \mathcal{S}$ and $\varphi_0 \in CTL^*$ s.t. $\varphi_0 = Q\varphi$ for a $Q \in \{\mathtt{E}, \mathtt{A}\}$ and a $\varphi$ not containing any path quantifiers. If $s_0 \models \varphi_0$ then player $\exists$ wins $\mathcal{G}_{\mathcal{T}}(s_0, \varphi_0)$.*

PROOF Note that CTL$^*$ is closed under negation and that the class of CTL$^*$ model checking games is closed under dual games. Furthermore, the negation of a $\varphi_0$ with one path quantifier at the top-level position only is a $\overline{\varphi_0}$ of the same form.

Suppose now that $s_0 \models \varphi_0$, i.e. $s_0 \not\models \overline{\varphi_0}$. According to Theorem 68, player $\forall$ wins $\mathcal{G}_{\mathcal{T}}(s_0, \overline{\varphi_0})$. But then player $\exists$ wins $\mathcal{G}_{\mathcal{T}}(s_0, \varphi_0)$ according to Theorem 39. ■

The next theorem proves general correctness of the CTL$^*$ model checking games. $\varphi_0$ can be an arbitrary CTL$^*$ formula now.

**Theorem 72 (Correctness)** *Let $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ with $s \in \mathcal{S}$. Player $\exists$ wins $\mathcal{G}_{\mathcal{T}}(s, \varphi_0)$ iff $s \models \varphi_0$.*

PROOF This is true if $\varphi_0$ is an atomic proposition. For formulas with one path quantifier only the claim is proved in Theorems 68 and 71. Suppose $\varphi_0$ has path quantifier depth $k$. By induction the claim is true for formulas with path quantifier depth less than $k$.

In general, $\mathcal{G}_{\mathcal{T}}(s, \varphi_0)$ has configurations $t \vdash Q'(\Phi)$ with a $Q\varphi \in \Phi$. $Q\varphi$ is a state formula with a path quantifier depth strictly less than $\varphi_0$'s because it is a genuine subformula of $\varphi_0$. Since it is a state formula, either $t \models Q\varphi$ or $t \not\models Q\varphi$ holds. By hypothesis either of the players has a winning strategy for the game $\mathcal{G}_{\mathcal{T}}(t, Q\varphi)$.

Suppose it is the one who is also the focus player at the current moment in the game at hand. He or she can set the focus to $Q\varphi$ with rule $(\mathtt{FC})$ and play rule $(\mathtt{E})$ or $(\mathtt{A})$ depending on $Q$. Note that the resulting configuration is of the same form as a general starting configuration. Furthermore, Lemma 64 shows that in the following a repeat on an earlier configuration cannot occur anymore. Thus, in fact they play the game for $t$ and $Q\varphi$. By hypothesis player $\exists$ wins this one iff $t \models Q\varphi$. Thus, if the actual focus player wins this game he or she also has a strategy for the game on $\varphi_0$.

Suppose the focus player does not win $\mathcal{G}_{\mathcal{T}}(t, Q\varphi)$. Then he or she can discard it by playing rule $(\emptyset)$. The following configuration corresponds to a state formula with path quantifier depth strictly less than $k$. Thus, the claim follows by hypothesis as well. ■

As in the PDL case, Theorem 72 shows that for every CTL$^*$ model checking game one of the players has a winning strategy.

**Corollary 73 (Determinacy)** *Player $\forall$ wins $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ iff player $\exists$ does not win $\mathcal{G}_{\mathcal{T}}(s,\varphi)$.*

The proofs of Theorems 68 and 71 show that the games can be simplified regarding the positioning of the focus.

- It suffices to allow focus change moves immediately after an application of rule $(\mathtt{X})$ only.
- Player $\forall$ only needs to consider formulas that contain a $\varphi\mathtt{U}\psi$ to set the focus to. Dually, player $\exists$ can do the same with formulas containing a $\varphi\mathtt{R}\psi$.

**Theorem 74 (Winning strategies)** *The winning strategies for the CTL$^*$ model checking games are history-free.*

PROOF Again, first we regard formulas with one path quantifier only which is at the top-level position. Consider player $\forall$'s winning strategies. Suppose the formula at hand is $\varphi_0 = \mathtt{A}\varphi$. Then he is the path player. One part of his strategy consists of choosing a path $\pi$ in the underlying transition system that does not satisfy $\psi$. This path does not depend on the play.

Furthermore, whenever a conjunction occurs he chooses the conjunct that is not satisfied by $\pi$ or its remaining suffix. If both conjuncts are false he chooses the smaller one. This is necessary for $\mathtt{R}$ formulas that are not fulfilled along the path that a play follows. Note that $\varphi\mathtt{R}\psi$ unfolds to a conjunction in which $\psi$ is one of the conjuncts. Suppose $\pi \not\models \varphi\mathtt{R}\psi$ where $\pi$ is the path he is going to choose for the remainder of the play. Then $\pi \not\models \psi$ but also $\pi \not\models \varphi \vee \mathtt{X}(\varphi\mathtt{R}\psi)$, i.e. player $\forall$ has no choice but to preserve falsity. However, only the first choice guarantees him to win. If he infinitely often postpones to refute $\psi$, i.e. always makes the second choice, then player $\exists$ is going to win with her winning condition 5 since she can leave the focus on $\varphi\mathtt{R}\psi$.

The choices of this strategy only depend on the formula and state component of the actual configuration, but not on the history of a play. Thus, this strategy is history-free.

Suppose now $\varphi_0 = \mathtt{E}\varphi$. Player $\forall$'s actions are reduced to setting the focus to a formula which he believes is not satisfied by the path that player $\exists$ is going to reveal. He can order all possibly occurring subformulas at the beginning of the play, s.t.

$$\varphi_i \in Sub(\varphi_j) \quad \text{implies} \quad j > i$$

where a $\varphi\mathtt{U}\psi$ or a $\varphi\mathtt{R}\psi$ is identified with their unfoldings. Then, at any point

$$t \ \vdash \ \mathtt{E}(\psi_1, \ldots, \psi_n)$$

during the play he can compute the sets $P_{\psi_1}(t), \ldots, P_{\psi_n}(t)$ according to Definition 65. His strategy simply tells him to set the focus to the formula with the least index whose corresponding set is non-empty. Lemma 67 shows that even if he forgets and recalculates these sets each time rule $(\mathtt{X})$ is played, this still guarantees that he does not need to change the focus back to a formula as long as he preserves the order of the subformulas he chose at the beginning. According to the proof of Theorem 68, he only needs to change the focus after an application of rule $(\mathtt{X})$, i.e. whenever he calculates the sets $P_{\psi_i}(t)$.

Between applications of rule $(\mathtt{X})$ he might have to set the focus to a particular conjunct. Suppose the actual configuration is

$$t \ \vdash \ \mathtt{E}(\left[\psi_i \wedge \psi_j\right], \Phi)$$

with path sets $P_{\psi_i}(t)$ and $P_{\psi_j}(t)$. Setting the focus to either of these will at most increase the index of the associated set since both conjuncts are obviously subformulas of the conjunction. Therefore it is safe for player $\forall$ to set the focus to the conjunct with the smaller index according to the order he chose at the start of the game. This choice does not depend on the history of the play either.

By duality, player $\exists$'s winning strategies are history-free, too.

Finally, strategies for games involving formulas with more than one path quantifier can be composed inductively in the same way as subgames are in the proof of Theorem 72. Correctness of this construction is guaranteed by the hypothesis of having a history-free winning strategy for subgames on formulas with a smaller quantifier depth. More importantly, the composition of history-free winning strategies is history-free. ■

In order to prove the small model property for LTL and CTL in Chapter 6 based on their satisfiability games we show that CTL$^*$ possesses the finite model property. It is based on the following lemma.

**Lemma 75** *Let $\mathcal{T} = (\mathcal{S}, \to, L)$ with $s \in \mathcal{S}$ and $\mathcal{R}_s(\mathcal{T})$ be its unravelling with respect to s. Then for all $\varphi \in CTL^*$: $\mathcal{T}, s \models \varphi$ iff $\mathcal{R}_s(\mathcal{T}) \models \varphi$.*

PROOF For every path in $\mathcal{T}$ there is a path in $\mathcal{R}_s(\mathcal{T})$ with the same state labellings and vice versa. Moreover, $\mathcal{T} \sim \mathcal{R}_s(\mathcal{T})$ and CTL$^*$ cannot distinguish bisimilar states. ■

This lemma is in fact nothing more than the tree model property for CTL$^*$ according to Section 2.1.

**Lemma 76** *Let $\mathcal{T} = (\mathcal{S}, \to, L)$ with $s_0 \in \mathcal{S}$, $\varphi_0 \in CTL^*$, s.t. $\mathcal{T}, s_0 \models \varphi_0$. Let $T$ be a successful game tree for player $\exists$ and the game $\mathcal{G}_{\mathcal{T}}(s_0, \varphi_0)$. Then there exists a finite tree prefix $T'$ of $T$, s.t. every maximal branch $P = C_0, \ldots, C_n$ through $T'$ satisfies one of the following properties.*

1. *$C_n$ is terminal, or*
2. *$C_n = t \vdash Q(\left[\varphi \mathtt{R} \psi\right], \Phi)$ and there is an $i < n$ s.t. $C_i = s \vdash Q(\left[\varphi \mathtt{R} \psi\right], \Phi)$ and there is no application of rule $(\mathtt{FC})$ between $C_i$ and $C_n$, or*

3. *$C_n = t \vdash \mathtt{E}(\left[\varphi\right], \Phi)$ and there is an $i < n$ s.t. $C_i = s \vdash \mathtt{E}(\left[\varphi\right], \Phi)$ and there is an application of rule $(\mathtt{FC})$ between $C_i$ and $C_n$.*

PROOF $T$ is a successful game tree for player $\exists$. Thus, every path through $T$ is a winning play for her. The finite tree prefix $T'$ can be constructed by cutting paths at appropriate positions.

If the corresponding play is won with condition 4 then it is finite and included in $T'$. It fulfils the first condition of the claim.

Suppose it is won with condition 5, i.e. from some point on player $\exists$ keeps the focus on a $\varphi \mathtt{R} \psi$. By finiteness of $Sub(\varphi_0)$ this play can be cut to fulfil the second condition of the claim.

Finally, if it is won with condition 6 there must be a moment after which player $\forall$ has used rule $(\mathtt{FC})$ and the play can be cut to satisfy the third condition of the claim. Or he left the focus on a $\varphi \mathtt{R} \psi$ in which case the second condition of the claim can be fulfilled.

Note that, if $T$ is finite then every path in it fulfils condition 1 above and therefore $T$ itself is the required finite tree prefix already. ∎

**Theorem 77 (Finite model property)** *CTL$^*$ has the finite model property.*

PROOF Suppose $\varphi_0 \in$ CTL$^*$ is satisfiable. Then it has a model $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ with $s_0 \in \mathcal{S}$. By Theorem 72 there is a game tree $T$ for player $\exists$ for the game $\mathcal{G}_{\mathcal{T}}(s_0, \varphi_0)$. If $|\mathcal{S}| < \infty$ then the claim is proved already.

Suppose therefore that $|\mathcal{S}| = \infty$. In general, $T$ will be infinite as well. According to Lemma 76, there is a finite tree prefix $T_1$ of $T$. We will amend this to an infinite game tree and show that it is a successful game tree for player $\exists$.

According to Lemma 76, every path in $T_1$ either ends in a terminal configuration or in a leaf $C_n$ that has a companion $C_i$, $i < n$, that differs from $C_n$ only in the state component.

In the next step we remove each such $C_n$ and add a transition from $C_{n-1}$ to the companion $C_i$ instead. Note that this represents a valid application of a game rule since the formula components of $C_i$ and $C_n$ are equal.

This construction yields a finite graph $T'$ with loops. Consider now its unravelling $\mathcal{R}_{C_0}(T')$ with respect to the starting configuration $C_0$. Every path in $\mathcal{R}_{C_0}(T')$ represents a play of a game $\mathcal{G}_{\mathcal{T}'}(s_0, \varphi_0)$ where $\mathcal{T}' = (\mathcal{S}', \{\xrightarrow{a}{}' \mid a \in \mathcal{A}\}, L)$ is defined by

$$\mathcal{S}' \;:=\; \{\, t \in \mathcal{S} \mid \text{there is a configuration } t \vdash Q(\Phi) \text{ in } T' \,\}$$

with transitions given by

$$t_1 \xrightarrow{a}{}' t_2 \quad \text{iff} \quad \begin{array}{l} \text{there are configurations } t_1 \vdash Q(\mathtt{X}\psi_1, \ldots, \mathtt{X}\psi_m) \\ \text{and } t_2 \vdash Q(\psi_1, \ldots, \psi_m) \text{ in } T' \text{ s.t. rule } (\mathtt{X}) \\ \text{was played between them} \end{array}$$

The labelling of the states is taken from their respective labellings in $\mathcal{T}$.

It remains to be seen that $\mathcal{R}_{C_0}(T')$ is a successful game tree for player $\exists$. Every finite path in $\mathcal{R}_{C_0}(T')$ fulfils her winning condition 4 since it is taken from her successful game tree $T$. Each infinite path in $T'$ is eventually cyclic and was constructed to fulfil winning condition 5 or 6 depending on which condition of Lemma 76 the underlying finite part fulfils.

As $\mathcal{R}_{C_0}(T')$ is a successful game tree for player $\exists$, $\mathcal{T}'$ with starting state $s_0$ must be a model for $\varphi_0$, according to Theorem 68. But $\mathcal{T}'$ consists of those states only that occur in the finite tree prefix $T_1$. Thus, $\varphi_0$ has a finite model. ■

## CTL$^*$ over Finite State Transition Systems

Similar to the model checking games for PDL from Chapter 4 the winning conditions for the CTL$^*$ model checking games can be simplified if the underlying transition system is finite. Then, player $\forall$ wins the play $C_0, \ldots, C_n$ of $\mathcal{G}_{\mathcal{T}}(s, \varphi_0)$ iff

1. $C_n \;=\; t \vdash Q(\left[q\right], \Phi)$ is terminal and $q \notin L(t)$, or
2. there is are $i < n$, $t \in \mathcal{S}$, $\varphi, \psi \in Sub(\varphi_0)$ and $\Phi \subseteq Sub(\varphi_0)$ s.t.
    - $C_i \;=\; C_n \;=\; t \vdash \mathtt{E}(\left[\varphi \mathtt{U} \psi\right], \Phi)$, and
    - between $C_i$ and $C_n$ player $\forall$ has not used rule $(\mathtt{FC})$, or

3. there are $i < n$, $t \in \mathcal{S}$, $\varphi \in Sub(\varphi_0)$ and $\Phi \subseteq Sub(\varphi_0)$ s.t. $C_n = t \vdash \mathtt{A}(\left[\varphi\right], \Phi)$ and $C_i = C_n$ and either
   - player $\exists$ has used rule $(\mathtt{FC})$ between $C_i$ and $C_n$, or
   - $\varphi$ is of the form $\chi \mathtt{U} \psi$.

Player $\exists$ wins the play $C_0, \ldots, C_n$ of $\mathcal{G}_{\mathcal{T}}(s, \varphi_0)$ iff

4. $C_n = t \vdash Q(\left[q\right], \Phi)$ is terminal and $q \in L(t)$, or
5. there are $i < n$, $t \in \mathcal{S}$, $\varphi, \psi \in Sub(\varphi_0)$ and $\Phi \subseteq Sub(\varphi_0)$ s.t.
   - $C_i = C_n = t \vdash \mathtt{A}(\left[\varphi \mathtt{R} \psi\right], \Phi)$, and
   - between $C_i$ and $C_n$ player $\exists$ has not used rule $(\mathtt{FC})$, or
6. there are $i < n$, $t \in \mathcal{S}$, $\varphi \in Sub(\varphi_0)$ and $\Phi \subseteq Sub(\varphi_0)$ s.t. $C_n = t \vdash \mathtt{E}(\left[\varphi\right], \Phi)$ and $C_i = C_n$ and either
   - player $\forall$ has used rule $(\mathtt{FC})$ between $C_i$ and $C_n$, or
   - $\varphi$ is of the form $\chi \mathtt{R} \psi$.

The new winning conditions for finite transition systems are equivalent to the old ones for arbitrary transition systems. If the underlying transition system is finite then there are only finitely many possible configurations. Since the winning strategies are history-free, see Theorem 74, the game tree can be represented as a graph according to Section 2.7. The new winning conditions then simply are a reformulation of the old ones on graphs.

We can give an upper complexity bound for game-based CTL* model checking that matches the upper bound from [CES83] and the lower bound from [SC85].

**Lemma 78** *Let $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ be finite with $s \in \mathcal{S}$ and $\varphi \in CTL^*$. Every play of $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ according to the winning conditions for games with underlying finite transition systems has length at most $|\mathcal{S}| \cdot |\varphi| \cdot 2^{|\varphi|} + 3$.*

PROOF There are $|\mathcal{S}| \cdot |\varphi| \cdot 2^{|\varphi|}$ many different configurations for the game $\mathcal{G}_{\mathcal{T}}(s, \varphi)$. Note that the two different possibilities for the $Q$ component of a configuration are annulled by the fact that there are only $2^{|\varphi|-1}$ many possible sets of subformulas of $\varphi$ not containing the actual formula in focus. Hence, every play of length more than this must repeat on a configuration.

This does not meet the requirements in the winning conditions 2 and 5 exactly. The formula in focus can possibly be the unfolding of a $\mathtt{U}$ or a $\mathtt{R}$ formula. In this case at most three more steps are necessary to obtain a situation to which one of the winning conditions applies. ∎

**Theorem 79 (Complexity)** *Deciding the winner of a CTL$^*$ model checking game is in PSPACE.*

PROOF An alternating algorithm can easily be extracted from the games by letting player $\exists$ make nondeterministic choices and player $\forall$ universal ones. However, this would result in an alternating PSPACE procedure which, by [CKS81], can only be transformed into a deterministic EXPTIME algorithm. This would be suboptimal because CTL$^*$ model checking is PSPACE-complete. To obtain a PSPACE procedure we need to determinise one of the player's choices without using more than polynomial space.

First, we describe a nondeterministic algorithm that decides whether or not the path player has a winning strategy for a game on a formula $\varphi_0$ with one top-level path quantifier only.

Suppose $\varphi_0 = \mathtt{E}\psi$, i.e. player $\exists$ is the path player. The algorithm nondeterministically chooses disjuncts and successor states whenever rule $(\mathtt{E}[\vee])$, $(\mathtt{E}\vee)$ or $(\mathtt{X})$ is played. Remember that player $\forall$'s choices in such a game are reduced to setting the focus and finding a configuration with an $\mathtt{U}$ formula that gets repeated upon s.t. he did not change the focus between the two occurrences of this configuration.

First we describe how to determinise the positioning of the focus. The only formulas that are interesting for him are of the form $\varphi \mathtt{U} \psi$. The algorithm maintains a list of all $\mathtt{U}$ subformulas of the input formula. At the beginning the formulas occur in the list in decreasing order of size. At any point in the play, the focus is placed on the first $\varphi \mathtt{U} \psi$

formula in the list that is present in the actual configuration. Once player $\exists$ discards it by choosing $\psi$ after the unfolding, it is moved to the end of the list. The focus is placed onto the present formula that is next in the new list. Whenever there is a conjunction in focus, he puts it onto the conjunct that contains the next $\mathtt{U}$ formula from the list. This strategy guarantees that every possibly reoccurring $\mathtt{U}$ formula occurred in focus before the play can perform a repeat. Moreover, it is deterministic.

We let the algorithm store two configurations: the actual one which gets overwritten each time a game rule is played, and a configuration $C_r$ to find a repeat upon. At the beginning $C_r$ is set to the starting configuration.

The algorithm needs to store a binary flag to indicate whether or not the focus has been changed after a possibly repeating configuration was stored. At last, it needs to store a counter that measures the length of the play at hand to terminate it in case the play does not repeat on $C_r$. The maximal length of a play without a repeat is

$$|\mathcal{S}| \cdot |\varphi_0| \cdot 2^{|\varphi_0|} \ + \ 3$$

according to Lemma 78. Thus the size of the counter is bounded by

$$|\varphi_0| \ + \ \log|\mathcal{S}| \ + \ \log|\varphi_0| \ + \ \mathit{const}$$

Furthermore, the actual value of the counter is stored whenever $C_r$ is set.

The algorithm returns "$\forall$" if at some point the actual configuration equals the stored one and the focus change flag is set to false. It returns "?" if in this situation the flag is true or the counter reaches its maximal value. In this case the game is restarted with $C_r$ and the stored counter value, i.e. $C_r$ gets overwritten by the next configuration and the algorithm attempts to show that there is a repeat on the new $C_r$. If the counter value stored with $C_r$ reaches the maximal value

$$|\mathcal{S}| \cdot |\varphi_0| \cdot 2^{|\varphi_0|} \ + \ 3$$

it outputs "$\exists$". In this case, there was no chance for player $\forall$ to show that he could enforce a play with a regenerating $\varphi\mathtt{U}\psi$, hence, player $\exists$ wins with condition 6.

If the input formula is of the form $\varphi_0 = \mathtt{A}\psi$ then the algorithm to be used is simply the dual of the one described. It universally chooses conjuncts, and the maintained list consists of $\mathtt{R}$ formulas. The return values are swapped.

Both algorithms are either nondeterministic or co-nondeterministic and use space which is polynomial in the size of the input: two configurations, two counters and a flag. By Savitch's Theorem, there is also a deterministic PSPACE procedure that decides the winner of $\mathcal{G}_{\mathcal{T}}(s,\varphi_0)$, [Sav69].

For arbitrary formulas the appropriate algorithm above can be called for every block of the game graph. There can only be $\frac{|\varphi_0|}{2}$ irredundant path quantifiers in $\varphi_0$. Thus, there can only be $\frac{|\varphi_0|}{2}$ blocks in the game graph, and the algorithms need to be called at most $\frac{|\varphi_0|}{2}$ many times. The space they need can be reused for every call. Hence, deciding the winner of $\mathcal{G}_{\mathcal{T}}(s,\varphi_0)$ is in PSPACE for arbitrary $\varphi_0$. ■

### Comparing Automata and Games for CTL$^*$ Model Checking

[KVW00] uses *hesitant alternating automata* HAA to do space-efficient model checking for CTL$^*$.

It is possible to view the games of this section as automata as well. Configurations of the games correspond to states of an automaton and winning conditions become acceptance conditions. However, the winning conditions proposed here depend on the position of the focus which is not easily translatable into a Büchi acceptance condition for example. The reason for this is the fact that Büchi acceptance conditions are only concerned with states but do not consider what happens in an automaton's run between two visits of a certain state.

Another difference between the games of this section and the HAA of [KVW00] is the fact that configurations of the games are sets of formulas whereas states of the automata are single subformulas only. It is known that alternating automata can be transformed into nondeterministic ones at the cost of an exponential blow-up. For alternating automata with single formulas as components of their states this means the nondeterministic version will have states featuring sets of formulas.

The games of this section compare to something between alternating and nondeterministic automata. One of the player's choices regarding boolean connectives have been eliminated by using sets of formulas. Alternating automata branch nondeterministically or universally at these points. However, the games are not like nondeterministic automata either since not all of one player's choices have been determinised. Instead, the problem of detecting whether there is a regenerating fixed point construct has been built into the games as a task for one of the players. For automata, this question is answered on the level of deciding non-emptiness of the accepted language.

The acceptance condition for HAA's is a combination of a Rabin and a Streett condition, especially tailored to the requirements of model checking branching time logics. The idea of using a mixture of two different acceptance conditions can be found in the games as well where they appear as winning conditions for two different players.

There is one thing that the games of this section and the HAA's have in common. Being *hesitant* means the automaton's state set can be partitioned into blocks such that transitions only lead to blocks with a lower index and each block is either existential or universal. This idea was in essence formulated in Lemma 64. It is also used in tableau-based model checking for CTL$^*$ in [BCG95]. In fact, this property is a feature of the logic rather than the method with which model checking is decided, and the key to the observation that LTL and CTL$^*$ model checking are polynomially interreducible.

## 5.3 Model Checking Games for CTL

In the case of a model checking game on a CTL formula no sets of formulas and hence no focus are needed. Since every temporal operator is immediately preceded by a path quantifier situations like the ones in Examples 58 and 59 cannot occur. Moreover, whenever a temporal operator is handled the corresponding quantifier would cause all side formulas to be erased from a configuration anyway. Thus, the model checking game rules can be simplified vastly for the CTL case. In this section, $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ denotes a model checking game according to the rules presented in Figure 5.8.

The set of configurations for a game on $\mathcal{T} = (\mathcal{S}, \rightarrow, L), s \in \mathcal{S}$ and $\varphi_0 \in$ CTL is

$$\mathcal{C} \;=\; \mathcal{S} \times Sub(\varphi_0)$$

Every play begins with $C_0 \;=\; s \vdash \varphi_0$. Player $\forall$ wins the play $C_0, C_1, \ldots$ iff

$$\frac{s \vdash \varphi_0 \wedge \varphi_1}{s \vdash \varphi_i}\ \forall i \qquad \frac{s \vdash \varphi_0 \vee \varphi_1}{s \vdash \varphi_i}\ \exists i$$

$$\frac{s \vdash \mathtt{AX}\varphi}{t \vdash \varphi}\ \forall\ s \to t \qquad \frac{s \vdash \mathtt{EX}\varphi}{t \vdash \varphi}\ \exists\ s \to t$$

$$\frac{s \vdash Q(\varphi \mathtt{U} \psi)}{s \vdash \psi \vee (\varphi \wedge Q\mathtt{X}Q(\varphi \mathtt{U} \psi))} \qquad \frac{s \vdash Q(\varphi \mathtt{R} \psi)}{s \vdash \psi \wedge (\varphi \vee Q\mathtt{X}Q(\varphi \mathtt{R} \psi))}$$

Figure 5.8: The rules for the CTL model checking games.

1. there is an $n \in \mathbb{N}$ s.t. $C_n = t \vdash q$ and $q \notin L(t)$, or
2. there are infinitely many configurations $C_{i_0}, C_{i_1}, \ldots$ and $\varphi, \psi \in Sub(\varphi_0)$ s.t. for all $j \in \mathbb{N}$: $C_{i_j} = t_{i_j} \vdash Q(\varphi \mathtt{U} \psi)$ for some $t_{i_j} \in \mathcal{S}$.

Player $\exists$ wins the play $C_0, C_1, \ldots$ iff

3. there is an $n \in \mathbb{N}$ s.t. $C_n = t \vdash q$ and $q \in L(t)$, or
4. there are infinitely many configurations $C_{i_0}, C_{i_1}, \ldots$ and $\varphi, \psi \in Sub(\varphi_0)$ s.t. for all $j \in \mathbb{N}$: $C_{i_j} = t_{i_j} \vdash Q(\varphi \mathtt{R} \psi)$ for some $t_{i_j} \in \mathcal{S}$.

**Lemma 80** *Every play has a uniquely determined winner.*

PROOF The winning conditions are mutually exclusive, i.e. a play can be won by at most one player. Moreover, formulas of the form $Q(\varphi \mathtt{U} \psi)$ and $Q(\varphi \mathtt{R} \psi)$ are exactly those that do not reduce the size of the actual configuration. Thus, every play must either reach an atomic proposition in which case it either holds or does not hold in the actual state. Or it proceeds ad infinitum with one of these formulas being visited infinitely often. ∎

**Theorem 81 (Correctness)** *Let $\mathcal{T} = (\mathcal{S}, \to, L)$, $s \in \mathcal{S}$, $\varphi \in CTL$. $\mathcal{T}, s \models \varphi$ iff player $\exists$ wins $\mathcal{G}_{\mathcal{T}}(s, \varphi)$.*

PROOF Every rule in a CTL game can be seen as a combination of rules of a CTL$^*$ game, and the winning conditions are simply amended to these combined rules and simplified configurations.

Note that the CTL winning conditions are the same as the winning conditions for the CTL$^*$ games if configurations only contain the formula in focus. In this case the focus itself can be discarded of course.

If the CTL$^*$ game rules are applied to CTL formulas then no sideformula can persist. In fact, whenever they occur they will be discarded immediately. Take a conjunction for example that occurs in a CTL$^*$ game configuration

$$t \;\vdash\; Q(\left[\psi_0 \wedge \psi_1\right], \Phi$$

If $Q = \mathtt{A}$ then player $\forall$ chooses one of the conjuncts like he does in a CTL game. If $Q = \mathtt{E}$ then he chooses an $i \in \{0, 1\}$ and the focus is set to $\psi_i$ while $\psi_{1-i}$ is added to the sideformulas. But $\psi_i$ is a CTL formula, too, i.e. it is of the form $Q'\chi$ with $Q' \in \{\mathtt{E}, \mathtt{A}\}$. Rule $(\mathtt{E})$ or $(\mathtt{A})$ causes the sideformulas including $\psi_{1-i}$ to be discarded in the next step. Thus, in the CTL$^*$ game the next configuration would be $t \vdash Q'(\chi)$ which is written as $t \vdash Q'\chi$ in the CTL game.

All the other cases are similar or dual to this one. ■

**Corollary 82 (Determinacy)** *Player $\forall$ wins $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ iff player $\exists$ does not win $\mathcal{G}_{\mathcal{T}}(s, \varphi)$.*

History-freeness of the winning strategies carries over from the CTL$^*$ model checking games.

**Corollary 83 (Winning strategies)** *The winning strategies for the CTL model checking games are history-free.*

### CTL over Finite State Transition Systems

If the underlying transition system is finite, the winning conditions can be reformulated as in the CTL$^*$ case. Player $\forall$ wins the play $C_0, \dots, C_n$ iff

1. $C_n = t \vdash q$ and $q \notin L(t)$, or
2. there is an $i < n$ and a $t \in \mathcal{S}$ s.t. $C_i = C_n = t \vdash Q(\varphi \mathtt{U} \psi)$ for some $\varphi, \psi$ and $Q \in \{\mathtt{A}, \mathtt{E}\}$.

Player $\exists$ wins the play $C_0, \dots, C_n$ iff

3. $C_n = t \vdash q$ and $q \in L(t)$, or
4. there is an $i < n$ and a $t \in \mathcal{S}$ s.t. $C_i = C_n = t \vdash Q(\varphi \mathtt{R} \psi)$ for some $\varphi, \psi$ and $Q \in \{\mathtt{A}, \mathtt{E}\}$.

Correctness of these winning conditions follows from Theorems 74 and 81.

Regarding a CTL formula as a CTL$^*$ formula does not result in an optimal model checking procedure. Considering the fact that no focus changes occur and that every configuration is of size linear in the input formula would still result in a PSPACE procedure. However, this does not take into account the special structure of CTL formulas. In particular, every block of the game graph is of constant size.

Using the alternation results from [CKS81] it is easy to see that the winner of a CTL model checking game can be determined in polynomial time. Again, the games give rise to an alternating algorithm that needs logarithmic space only. However, this can be improved even further by using a more explicit approach.

**Theorem 84 (Complexity)** *Deciding the winner of a CTL model checking game is in LINTIME.*

PROOF It makes more sense to use the same notion of *block* as it was introduced in Chapter 4 for PDL model checking. Here, blocks of the game graph are given by the formula component s.t. every path traverses blocks in increasing order of index and eventually remains in one block only. Note that the index of a block basically measures how far it is away from the starting configuration.

This is possible since formulas of the form $Q(\varphi \mathtt{U} \psi)$ and $Q(\varphi \mathtt{R} \psi)$ are the only ones that do not reduce the size of a configuration. Also, blocks can have loops induced by one of these formulas only. Thus, each block has a type $\mathtt{U}$ or $\mathtt{R}$. The global CTL model checking procedure works bottom-up just like the one for PDL. It also needs to visit each node of the game graph at most once. Remember that the size of the game graph is $|\mathcal{S}| \cdot |\varphi|$ for a transition system with state set $\mathcal{S}$ and a formula $\varphi$. ■

### Comparing Automata and Games for CTL Model Checking

Similar to the CTL$^*$ case, *hesitant alternating automata* have also been used in [KVW00] to decide the model checking problem for CTL based on *weak alternating automata*, WAA, [MSS88]. Their state set is partitioned into blocks like those of HAA. However, they accept with a simple Büchi condition.

Given that the CTL model checking games feature single formulas in their configurations only, there is a certain similarity between them and the WAA for CTL model checking. Also, the choices made by the two players correspond to the nondeterministic and universal branches in an alternating automaton. It is not hard to see that the winning conditions of the CTL model checking games can be modelled by a Büchi condition. The configurations that can be visited infinitely often are exactly those of the form $t \vdash Q(\varphi \mathtt{R} \psi)$.

This similarity is not surprising since the main difference between games and HAA is due to the use of the focus, but the CTL model checking game is not a focus game.

## 5.4 Model Checking Games for CTL$^+$

Since CTL$^+$ is known to be exponentially more succinct than CTL, [Wil99, AI01], one cannot expect the same radical simplifications from CTL$^*$ games to CTL games. Example 58 suggests that configurations in a model checking game $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ for a CTL$^+$ formula $\varphi$ must contain sets of subformulas. However, since the formula of

$$(\mathtt{A}\wedge)\ \frac{s \vdash \mathtt{A}(\varphi_0 \wedge \varphi_1, \Phi)}{s \vdash \mathtt{A}(\varphi_i, \Phi)}\ \forall i \qquad (\mathtt{U})\ \frac{s \vdash Q(\varphi \mathtt{U} \psi, \Phi)}{s \vdash Q(\psi \vee (\varphi \wedge Q\mathtt{X}Q(\varphi \mathtt{U} \psi)), \Phi)}$$

$$(\mathtt{E}\vee)\ \frac{s \vdash \mathtt{E}(\varphi_0 \vee \varphi_1, \Phi)}{s \vdash \mathtt{E}(\varphi_i, \Phi)}\ \exists i \qquad (\mathtt{R})\ \frac{s \vdash Q(\varphi \mathtt{R} \psi, \Phi)}{s \vdash Q(\psi \wedge (\varphi \vee Q\mathtt{X}Q(\varphi \mathtt{R} \psi)), \Phi)}$$

$$(\mathtt{A}\vee)\ \frac{s \vdash \mathtt{A}(\varphi_0 \vee \varphi_1, \Phi)}{s \vdash \mathtt{A}(\varphi_0, \varphi_1, \Phi)} \qquad (\mathtt{X})\ \frac{s \vdash Q(\mathtt{X}\varphi_0, \ldots, \mathtt{X}\varphi_k)}{t \vdash Q(\varphi_0, \ldots, \varphi_k)}\ p\ \ s \longrightarrow t$$

$$(\mathtt{E}\wedge)\ \frac{s \vdash \mathtt{E}(\varphi_0 \wedge \varphi_1, \Phi)}{s \vdash \mathtt{E}(\varphi_0, \varphi_1, \Phi)} \qquad (\not q)\ \frac{s \vdash Q(q, \Phi)}{s \vdash Q(\Phi)}\ \overline{p}$$

$$(\mathtt{A})\ \frac{s \vdash Q(\mathtt{A}\varphi, \Phi)}{s \vdash \mathtt{A}(\varphi)}\ \overline{p} \qquad (\mathtt{A})\ \frac{s \vdash Q(\mathtt{A}\varphi, \Phi)}{s \vdash Q(\Phi)}\ \overline{p},\ \text{if } \Phi \neq \emptyset$$

$$(\mathtt{E})\ \frac{s \vdash Q(\mathtt{E}\varphi, \Phi)}{s \vdash \mathtt{E}(\varphi)}\ \overline{p} \qquad (\mathtt{E})\ \frac{s \vdash Q(\mathtt{E}\varphi, \Phi)}{s \vdash Q(\Phi)}\ \overline{p},\ \text{if } \Phi \neq \emptyset$$

Figure 5.9: The rules for the $\text{CTL}^+$ model checking games.

Example 59 that justifies the use of the focus is not in $\text{CTL}^+$ the question of whether a focus is needed for $\text{CTL}^+$ games is reasonable to ask. $\text{CTL}^+$ does not allow nested temporal operators, therefore the answer is no.

Configurations of a $\text{CTL}^+$ model checking game are

$$\mathcal{C} \;=\; \mathcal{S} \times \{\mathtt{A}, \mathtt{E}\} \times 2^{Sub(\varphi)}$$

containing at least one subformula.

The game rules are given in Figure 5.9. In addition to the rule schemes introduced in Chapter 4 and Section 5.2, a rule of the form

$$(r)\ \frac{C}{C'}\ \ p\ c,\ \ d$$

is only applicable if the condition $d$ is met in the actual configuration.

Here, this applies to rules $(\mathtt{A})$ and $(\mathtt{E})$. Note that there are two cases for each of them. A quantified formula or an atomic proposition can only be discarded if least one side formula is present. There is no requirement for discarding sideformulas. However, in both cases the rule operates on the same formula. Therefore, we consider them to be one rule only. The other rules result from the CTL$^*$ model checking game rules in Figure 5.1 by disregarding the focus.

Every play of $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ starts with $C_0 = s \vdash \mathtt{A}(\varphi)$. Let $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ with $s_0 \in \mathcal{S}$. Player $\forall$ wins the play $C_0, C_1, \ldots$ of $\mathcal{G}_{\mathcal{T}}(s_0,\varphi_0)$ iff

1. there is an $n \in \mathbb{N}$ s.t. $C_n = t \vdash Q(q,\Phi)$ and $q \notin L(t)$, or
2. there are infinitely many configurations $C_{i_0}, C_{i_0}, \ldots$ and $\varphi, \psi \in Sub(\varphi_0)$ s.t. for all $j \in \mathbb{N}$: $C_{i_j} = t_{i_j} \vdash \mathtt{E}(\varphi \mathtt{U} \psi, \Phi)$ for some $t_{i_j} \in \mathcal{S}$ and $\Phi$.
3. there are infinitely many configurations $C_{i_0}, C_{i_0}, \ldots$ and $\Phi \subseteq Sub(\varphi_0)$ s.t. for all $j \in \mathbb{N}$:
   - $C_{i_j} = t_{i_j} \vdash \mathtt{A}(\Phi)$ for some $t_{i_j} \in \mathcal{S}$, and
   - no formula of the form $\chi \mathtt{R} \psi$ is present in $\Phi$.

Player $\exists$ wins the play $C_0, C_1, \ldots$ of $\mathcal{G}_{\mathcal{T}}(s_0,\varphi_0)$ iff

4. there is an $n \in \mathbb{N}$ s.t. $C_n = t \vdash Q(q,\Phi)$ and $q \in L(t)$, or
5. there are infinitely many configurations $C_{i_0}, C_{i_0}, \ldots$ and $\varphi, \psi \in Sub(\varphi_0)$ s.t. for all $j \in \mathbb{N}$: $C_{i_j} = t_{i_j} \vdash \mathtt{A}(\varphi \mathtt{R} \psi, \Phi)$ for some $t_{i_j} \in \mathcal{S}$ and $\Phi$.
6. there are infinitely many configurations $C_{i_0}, C_{i_0}, \ldots$ and $\Phi \subseteq Sub(\varphi_0)$ s.t. for all $j \in \mathbb{N}$:
   - $C_{i_j} = t_{i_j} \vdash \mathtt{E}(\Phi)$ for some $t_{i_j} \in \mathcal{S}$, and
   - no formula of the form $\chi \mathtt{U} \psi$ is present in $\Phi$.

Note that the path player's opponent must be allowed to discard atomic propositions *before* one of the winning conditions can apply.

**Theorem 85 (Correctness)** *Let* $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ *with* $s \in \mathcal{S}$ *and* $\varphi \in CTL^+$. $\mathcal{T}, s \models \varphi$ *iff player* $\exists$ *wins* $\mathcal{G}_{\mathcal{T}}(s, \varphi)$.

PROOF The game rules and winning conditions for the CTL$^+$ games arise from the CTL$^*$ games by removing the focus. Thus, it suffices to show that, whenever a play is infinite, there is no ambiguity about the regeneration of $\mathtt{U}$ or $\mathtt{R}$ formulas. Assume a play like

$$\frac{\dfrac{\dfrac{s \vdash \mathtt{A}(\varphi)}{\vdots}}{t \vdash \mathtt{E}(\chi \mathtt{U} \psi, \Phi)}}{\dfrac{\vdots}{t' \vdash \mathtt{E}(\chi \mathtt{U} \psi, \Phi)}} \qquad \vdots$$

We will show that in this case $\chi \mathtt{U} \psi$ in the lower configuration can only stem from itself in the upper one. Suppose it does not, i.e. there is a $\varphi' \in \Phi$ s.t. $\chi \mathtt{U} \psi \in Sub(\varphi')$. Between these two configurations rule $(\mathtt{X})$ has been played at least once, otherwise nothing has been done to $\chi \mathtt{U} \psi$ and it trivially stems from itself.

Remember that CTL$^+$ does not allow temporal operators to be nested. Therefore, $\chi \mathtt{U} \psi$ occurred in the scope of a path quantifier $\mathtt{E}$ or $\mathtt{A}$ in $\varphi'$. In order for $\chi \mathtt{U} \psi$ to appear in the lower configuration, rule $(\mathtt{E})$ or $(\mathtt{A})$ must have been played between the two configurations at hand. But they either cause the $\chi \mathtt{U} \psi$ or all present sideformulas to be discarded. In particular, $\varphi'$ cannot have regenerated itself. Thus, either it is wrong to assume that $\Phi$ occurs again, or there is a superformula of $\varphi'$ that generated $\varphi'$ again. But then the argument applies to this one and there are only finitely many superformulas of a formula. Therefore, if $\chi \mathtt{U} \psi$ occurs infinitely often it must be the case that it regenerates itself. The same holds of course for a $\chi \mathtt{R} \psi$. ■

**Corollary 86 (Determinacy)** *Player* $\forall$ *wins* $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ *iff player* $\exists$ *does not win* $\mathcal{G}_{\mathcal{T}}(s, \varphi)$.

The winning conditions can be simplified as in the CTL$^*$ and the CTL case if the underlying transition system is finite.

Again, since the $\text{CTL}^+$ games are only a special case of the $\text{CTL}^*$ model checking games their winning strategies are history-free as well.

**Corollary 87 (Winning strategies)** *The winning strategies for the $\text{CTL}^+$ model checking games are history-free.*

Deciding the winner of a $\text{CTL}^+$ model checking play can be at most as hard as it is for $\text{CTL}^*$ formulas. However, simply ignoring the focus in a $\text{CTL}^*$ model checking game to obtain a $\text{CTL}^+$ model checking does not effect the complexity of deciding the winner.

**Theorem 88 (Complexity)** *Deciding the winner of a $\text{CTL}^+$ model checking game is in PSPACE.*

This is slightly worse than the known upper and lower bound of $\Delta_2$ from [LMS01]. It seems like a far more explicit analysis of the structure of a $\text{CTL}^+$ model checking graph etc. is needed to obtain a better game based complexity bound than PSPACE. Just ignoring the focus also does not make use of the special structure of $\text{CTL}^+$ formulas as opposed to arbitrary $\text{CTL}^*$ formulas.

To the best of our knowledge, the model checking problem for $\text{CTL}^+$ has not attracted a great deal of attention. In particular, there is no special class of automata which have been shown to be applicable directly to the $\text{CTL}^+$ model checking problem without translation $\text{CTL}^+$ formulas into CTL first. Note that the upper complexity bound in [LMS01] has been established by a reduction technique.

## 5.5 Model Checking Games for BLTL

Since BLTL formulas can contain arbitrary nestings of path operators together with boolean connectives the focus approach on sets of formulas is needed in that case, too. However, according to Lemma 64, the game graph for an BLTL formula consists of one block only. Therefore it is not necessary to memorise the path player explicitly. Rules $(\mathtt{A})$, $(\mathtt{E})$, $(\not{Q})$ and $(\not{q})$ never apply, and in rule $(\mathtt{X})$ it is always player $\forall$ who chooses the next state from the transition system.

These optimisations do not provide better complexity results of game based BLTL model checking compared to $CTL^*$ model checking. The proof of the next theorem is the same as the proof of Theorem 79. The fact that the model checking procedure only needs to be called once does not affect the space complexity of the problem. Again, this result matches the known lower and upper bounds.

**Theorem 89 (Complexity)** *Deciding the winner of a BLTL model checking game is in PSPACE.*

### Comparing Automata and Games for BLTL Model Checking

The automata-theoretic approach to BLTL model checking has been studied in detail, for example in [VW86a]. First, nondeterministic Büchi automata were used for this task based on the observation that BLTL formulas can be translated into these at the cost of an exponential blow-up. This is not suboptimal since BLTL model checking is PSPACE-complete and, hence, is very likely to require exponential time. The non-emptiness problem for these automata, to which BLTL model checking is reduced is decidable in polynomial time and nondeterministic logarithmic space.

A different approach is taken in [Var96] which proposes the use of alternating automata for this task as well. Similar to automata-theoretic $CTL^*$ model checking, a BLTL formula is translated into an alternating Büchi automaton which is possible in linear time. However, this translation makes the non-emptiness problem for these automata PSPACE-hard. In fact it is PSPACE-complete.

Comparing these automata with the BLTL games leads to the same conclusions as those that were made for $CTL^*$ in Section 5.2. The two main differences between the games and the automata are the following.

- The automata feature more alternation by branching universally at conjunctions and nondeterministically at disjunctions. The games however determinise one of these by using sets of formulas for disjunctions.
- The question of whether or not a run of an alternating automaton is accepting is decided on top of the automata. It is done graph-theoretically by solving a

certain reachability problem. For the games this is implicitly done by giving player ∃ control over the focus and making the focus setting behaviour a part of the winning conditions. The algorithmics used for deciding whether there is a successful game tree is simpler than the one used for automata.

Having the same conclusions as those for $\mathrm{CTL}^*$ is not surprising since the $\mathrm{CTL}^*$ games basically consist of several BLTL games played consecutively. This is reflected on the automata side as well. For BLTL, normal alternating automata suffice. The property of being weak or hesitant is only needed for branching time logics since the linear time logic does not impose a block structure on the game graph, resp. the automaton.

# Chapter 6

# Satisfiability Games for LTL, CTL and PDL

*Let no one ignorant of*
*Mathematics enter here.*
—
PLATO

## 6.1 Satisfiability Games for LTL

Given an LTL formula $\varphi_0$ the *satisfiability game* $\mathcal{G}(\varphi_0)$ is played to determine whether $\varphi_0$ has a model or not. It is player $\exists$'s task to show that it does, whereas player $\forall$ wants to show that there is no path $\pi$ of any total transition system s.t. $\pi \models \varphi_0$.

Configurations of $\mathcal{G}(\varphi_0)$ are nonempty sets of subformulas of $\varphi_0$ with a focus like the one in Chapter 5,

$$\mathcal{C} = Sub(\varphi_0) \times 2^{Sub(\varphi_0)}$$

Every play of $\mathcal{G}(\varphi_0)$ starts with $C_0 = \left[\varphi_0\right]$. It is always player $\forall$ who has control over the position of the focus.

There are two possibilities for a $\varphi_0$ to be unsatisfiable. Either it inevitably forces a state of a possible model to be labelled with $\mathtt{ff}$ or a proposition $q$ and its complement $\overline{q}$, or it does not enable a least fixed point operator, i.e. an $\mathtt{U}$ formula, to be fulfilled at some point. The inevitability of one of these situations is reflected in a possible winning strategy for player $\forall$. The situations themselves are modelled by the winning conditions.

A configuration $\left[\varphi \mathtt{U} \psi\right], \Phi$ is to be read as: Player $\exists$ wants to build a model for

$$(\varphi \mathtt{U} \psi) \wedge \bigwedge_{\chi \in \Phi} \chi$$

while player $\forall$ tries to show that $\varphi \mathtt{U} \psi$ does not get fulfilled along the play. Player $\forall$ is allowed to set the focus to formulas of other forms. This is obviously necessary if there is no $\varphi \mathtt{U} \psi$ present in the actual configuration.

The game rules are given in Figure 6.1. Rules $([\vee])$ and $(\vee)$ are justified by the fact that a disjunction is satisfiable iff one of the disjuncts is satisfiable. For a conjunction to be satisfiable the combination of both conjuncts must be satisfiable. Thus, rules $([\wedge])$ and $(\wedge)$ simply flatten conjunctions to sets. Fixed point operators are unfolded with rules $([\mathtt{U}])$, $(\mathtt{U})$, $([\mathtt{R}])$ and $(\mathtt{R})$. Finally, player $\forall$ controls the position of the focus with rules $(\mathtt{FC})$ and $([\wedge])$.

**Definition 90** A configuration is *terminal* if it is of the form $\left[q\right], \Phi$ and player $\forall$ refuses or is unable to move the focus.

Next we define the outcome of a play. Player $\forall$ wins the play $C_0, \ldots, C_n$ iff

1. $C_n = \left[q\right], \Phi$ is terminal and $q = \mathtt{ff}$ or $\overline{q} \in \Phi$, or
2. $C_n = \left[\varphi \mathtt{U} \psi\right], \Phi$ and there is an $i \in \mathbb{N}$, s.t. $i < n$ and $C_i = C_n$, and player $\forall$ has not used rule $(\mathtt{FC})$ between $C_i$ and $C_n$.

$$([\vee])\ \frac{\left[\varphi_0 \vee \varphi_1\right], \Phi}{\left[\varphi_i\right], \Phi}\ \exists i \qquad (\vee)\ \frac{\left[\psi\right], \varphi_0 \vee \varphi_1, \Phi}{\left[\psi\right], \varphi_i, \Phi}\ \exists i$$

$$([\wedge])\ \frac{\left[\varphi_0 \wedge \varphi_1\right], \Phi}{\left[\varphi_i\right], \varphi_{1-i}, \Phi}\ \forall i \qquad (\wedge)\ \frac{\left[\psi\right], \varphi_0 \wedge \varphi_1, \Phi}{\left[\psi\right], \varphi_0, \varphi_1, \Phi}$$

$$([\mathtt{U}])\ \frac{\left[\varphi \mathtt{U} \psi\right], \Phi}{\left[\psi \vee (\varphi \wedge \mathtt{X}(\varphi \mathtt{U} \psi))\right], \Phi} \qquad ([\mathtt{R}])\ \frac{\left[\varphi \mathtt{R} \psi\right], \Phi}{\left[\psi \wedge (\varphi \vee \mathtt{X}(\varphi \mathtt{R} \psi))\right], \Phi}$$

$$(\mathtt{U})\ \frac{\left[\chi\right], \varphi \mathtt{U} \psi, \Phi}{\left[\chi\right], \psi \vee (\varphi \wedge \mathtt{X}(\varphi \mathtt{U} \psi)), \Phi} \qquad (\mathtt{R})\ \frac{\left[\chi\right], \varphi \mathtt{R} \psi, \Phi}{\left[\chi\right], \psi \wedge (\varphi \vee \mathtt{X}(\varphi \mathtt{R} \psi)), \Phi}$$

$$(\mathtt{X})\ \frac{\left[\mathtt{X}\varphi_1\right], \ldots, \mathtt{X}\varphi_k, q_1, \ldots, q_n}{\left[\varphi_1\right], \ldots, \varphi_k} \qquad (\mathtt{FC})\ \frac{\left[\varphi\right], \psi, \Phi}{\left[\psi\right], \varphi, \Phi}\ \forall$$

Figure 6.1: The satisfiability game rules for LTL.

Player $\exists$ wins the play $C_0, \ldots, C_n$ iff

3. $C_n = \left[q\right], \Phi$ is terminal, $q \neq \mathtt{ff}$ and $\overline{q} \notin \Phi$, or
4. $C_n = \left[\varphi\right], \Phi$ and there is an $i \in \mathbb{N}$, s.t. $i < n$ and $C_i = C_n$, and player $\forall$ has used rule $(\mathtt{FC})$ between $C_i$ and $C_n$.
5. $C_n = \left[\varphi \mathtt{R} \psi\right], \Phi$ and there is an $i \in \mathbb{N}$, s.t. $i < n$ and $C_i = C_n$, and player $\forall$ has not used rule $(\mathtt{FC})$ between $C_i$ and $C_n$.

To illustrate the satisfiability games we consider a formula that is very similar to the CTL$^*$ formula of Example 59 which was used to justify the use of a focus. Again, it is very easy to extend the game rules to handle abbreviated $\mathtt{F}$ and $\mathtt{G}$ formulas explicitly. The rules are

$$\frac{\left[\mathtt{F}\varphi\right],\Phi}{\left[\varphi\vee\mathtt{XF}\varphi\right],\Phi} \qquad \frac{\left[\mathtt{G}\varphi\right],\Phi}{\left[\varphi\wedge\mathtt{XG}\varphi\right],\Phi}$$

$$\frac{\left[\psi\right],\mathtt{F}\varphi,\Phi}{\left[\psi\right],\varphi\vee\mathtt{XF}\varphi,\Phi} \qquad \frac{\left[\psi\right],\mathtt{G}\varphi,\Phi}{\left[\psi\right],\varphi,\mathtt{XG}\varphi,\Phi}$$

**Example 91** Let

$$\varphi \ := \ \mathtt{F}q\wedge\mathtt{GF}q$$

$\varphi$ is satisfiable as Example 59 shows. An excerpt of the full game tree is depicted in Figure 6.2. Since player $\forall$ is allowed to use rule $(\mathtt{FC})$ at any moment in the game the entire game tree has more branches. We only include "sensible" choices for the positioning of the focus, i.e. those that do not make him lose immediately.

Indeed, player $\exists$ has a winning strategy for this game. It consists of enforcing either the leftmost play or the right one of the pair in the middle. She wins both of these with winning condition 4 since player $\forall$ had to change the focus at some point. The left play of the two in the middle and both plays at the right side are won by player $\forall$ with winning condition 2. This is because player $\exists$ never fulfilled the $\mathtt{F}q$ although she could have and, hence, it stayed in focus.

## Correctness

Before we can prove correctness of the games we need to establish a few facts about the rules and prove a few lemmas.

**Fact 92** $(\mathtt{FC})$ *is the only rule that maintains the size of a configuration. Rules* $([\vee])$, $(\vee)$, $([\wedge])$, $(\wedge)$ *and* $(\mathtt{X})$ *reduce the number of connectives in a configuration, while rules* $([\mathtt{U}])$, $(\mathtt{U})$, $([\mathtt{R}])$ *and* $(\mathtt{R})$ *increase the number of connectives.*

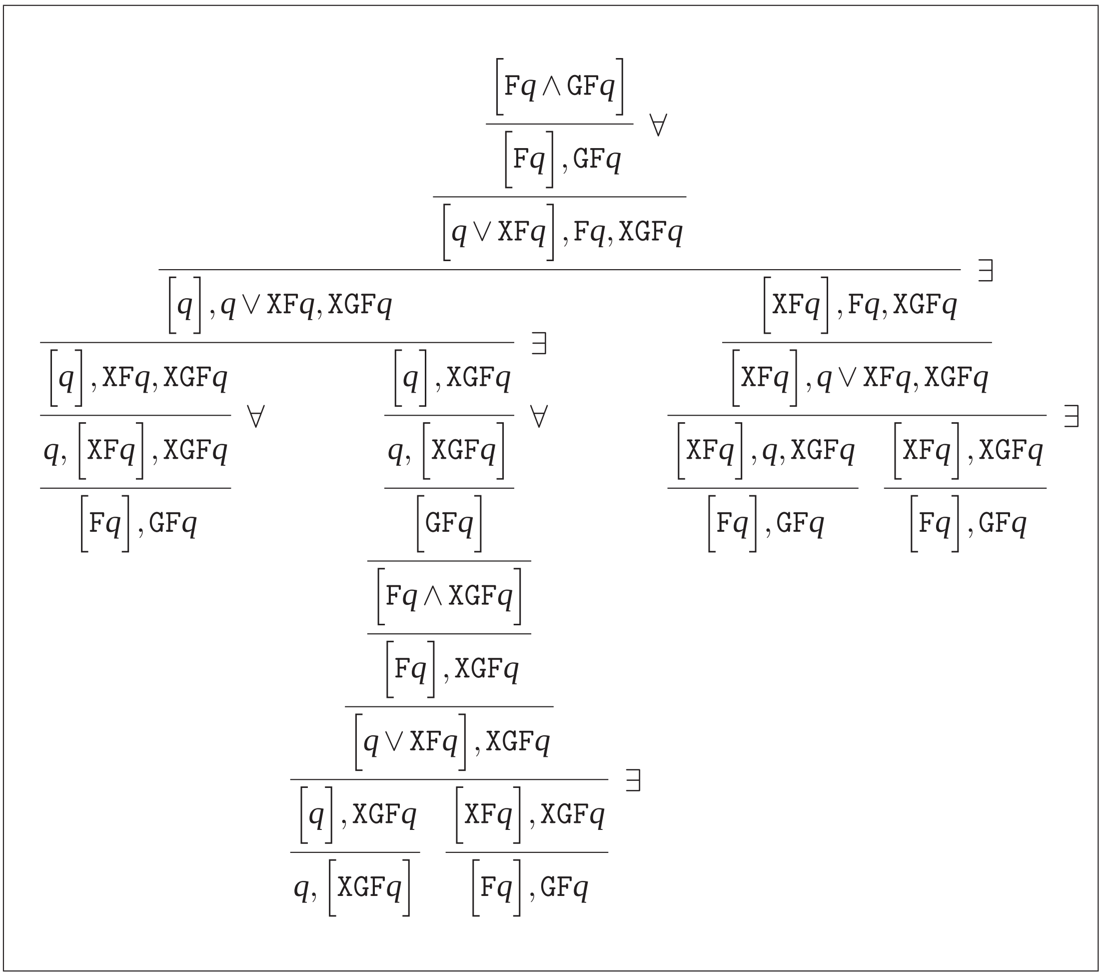

Figure 6.2: The interesting part of the game tree of Example 91.

**Lemma 93** *Every play of* $\mathcal{G}(\varphi)$ *has finite length less than* $|\varphi| \cdot 2^{|\varphi|} + 3$.

PROOF $|\varphi| \cdot 2^{|\varphi|}$ is the maximal number of possible different configurations in a play of $\mathcal{G}(\varphi)$. Therefore, every play of length more than $|\varphi| \cdot 2^{|\varphi|}$ must have a repeat on a configuration.

There are two possibilities for such a play. Either, player $\forall$ has used rule $(\mathtt{FC})$ between the two occurrences of a repeating configuration. Or he has not used rule $(\mathtt{FC})$ in between. But then at least one other rule must have been played. By Fact 92, all other rules either reduce or increase the number of connectives in a configuration. Thus, one of the unfolding rules $([\mathtt{U}])$, $(\mathtt{U})$, $([\mathtt{R}])$ or $(\mathtt{R})$ must have been applied to obtain a repeat. Note that an $\mathtt{U}$ or a $\mathtt{R}$ is guarded by an $\mathtt{X}$ in its unfolding. Thus, there must have been at least one application of rule $(\mathtt{X})$ between the repeating configurations. This reduces the size of the formula in focus.

Since the focus change rule has not been used rule $([\mathtt{U}])$ or $([\mathtt{R}])$ must have been played in fact. This means that the focus has been kept on an $\mathtt{U}$ or a $\mathtt{R}$ and their respective unfoldings. Then there also are configurations of the form $\left[\varphi\mathtt{U}\psi\right],\Phi$, resp. $\left[\varphi\mathtt{R}\psi\right],\Phi$ that the play repeats on.

The repeat on these configurations must occur at most 3 steps later because the formula in focus can be at most 3 connectives larger than a $\varphi\mathtt{U}\psi$ or $\varphi\mathtt{R}\psi$. ∎

**Lemma 94** *Every play has a uniquely determined winner.*

PROOF A play either ends in a terminal configuration or performs a repeat. In the first case, winning conditions 1 and 3 determine the winner. Note that they are mutually exclusive and cover all possible scenarios.

In the second case player $\forall$ either has used rule $(\mathtt{FC})$ between the repeating configurations or not. If he has, player $\exists$ wins with winning condition 4. If he has not, then the winner is determined by the formula that remained in focus while being regenerated. According to the proof of Lemma 93, it is either a $\varphi\mathtt{U}\psi$ or a $\varphi\mathtt{R}\psi$. In the first case he wins with winning condition 2, in the second case player $\exists$ wins with condition 5. ∎

**Corollary 95 (Determinacy)** *Player $\forall$ wins $\mathcal{G}(\varphi)$ iff player $\exists$ does not win $\mathcal{G}(\varphi)$.*

PROOF The "only if" part is trivial. The "if" part follows from Theorem 37 of Section 2.6 and Lemmas 93 and 94. ∎

**Lemma 96** *The game rules preserve unsatisfiability.*

PROOF Player $\forall$ preserves unsatisfiability since his moves are only concerned with the position of the focus.

Player $\exists$ preserves unsatisfiability with her moves as well since the only thing she does is to choose disjuncts. Suppose

$$(\psi_0 \vee \psi_1) \wedge \Phi$$

is unsatisfiable. Then so are $\psi_0 \wedge \Phi$ and $\psi_1 \wedge \Phi$. Consequently, player $\exists$ cannot force the play into a satisfiable configuration.

Unfolding $\mathtt{U}$ and $\mathtt{R}$ formulas preserves unsatisfiability because they are replaced by a logically equivalent formula.

Finally, consider an application of rule $(\mathtt{X})$. Suppose $\psi_1 \wedge \ldots \wedge \psi_k$ is satisfiable, i.e. it has a model $\pi$. Suppose furthermore that $q_1 \wedge \ldots \wedge q_n$ is satisfiable. Let $\pi' := s\pi$ for some state $s$ with $L(s) = \{q_1, \ldots, q_n\}$. Then,

$$\pi' \models \mathtt{X}\psi_1, \ldots, \mathtt{X}\psi_k, q_1, \ldots, q_n$$

In other words, if $\mathtt{X}\psi_1, \ldots, \mathtt{X}\psi_k, q_1, \ldots, q_n$ is unsatisfiable then so is $q_1, \ldots, q_n$ or $\psi_1, \ldots, \psi_k$. In the first case player $\forall$ can change the focus to the $q_i$ that causes unsatisfiability and the resulting terminal configuration is unsatisfiable. In the latter case rule $(\mathtt{X})$ is applied deterministically and the next configuration is unsatisfiable as well. ■

Next we describe a strategy for player $\forall$ and the game $\mathcal{G}(\varphi_0)$ and prove that it is optimal.

**Definition 97 (Priority list strategy)** Let $l$ be a *priority list* of all $\mathtt{U}$ subformulas of the input formula $\varphi_0$ in decreasing order of size, i.e.

$$l = \varphi_1 \mathtt{U} \psi_1, \ldots, \varphi_n \mathtt{U} \psi_n$$

with

$$\varphi_i \mathtt{U} \psi_i \in Sub(\varphi_j \mathtt{U} \psi_j) \quad \text{and} \quad \varphi_i \mathtt{U} \psi_i \neq \varphi_j \mathtt{U} \psi_j \quad \text{implies} \quad j < i$$

In that case $\varphi_j \mathtt{U} \psi_j$ is said to have higher priority than $\varphi_i \mathtt{U} \psi_i$.

We say that $\varphi \mathtt{U} \psi$ is present in a configuration $C$ if

$$\{\, \varphi \mathtt{U} \psi,\ \psi \vee (\varphi \wedge \mathtt{X}(\varphi \mathtt{U} \psi)),\ \varphi \wedge \mathtt{X}(\varphi \mathtt{U} \psi),\ \mathtt{X}(\varphi \mathtt{U} \psi) \,\} \cap C \neq \emptyset$$

Player $\forall$ starts with the focus on $\varphi_0$. If the formula in focus is a $\varphi \mathtt{R} \psi$ formula and there is a $\varphi' \mathtt{U} \psi' \in Sub(\psi)$ then $\forall$ sets the focus to $\psi$ when $\varphi \mathtt{R} \psi$ gets unfolded with rule $([\mathtt{R}])$ or $(\mathtt{R})$. If the formula in focus is a conjunction then $\forall$ chooses the conjunct that contains the $\mathtt{U}$ formula with the highest priority in $l$ if possible. If the focus remains on a $\mathtt{R}$ formula or ends up on a propositional constant then $\forall$ changes focus to avoid defeat by winning condition 3 or 5. He sets the focus to the formula with highest priority in $l$ or a superformula of it.

If the focus is on a $\varphi \mathtt{U} \psi$ then he keeps it there until it becomes "fulfilled", i.e. player $\exists$ chooses the disjunct $\psi$ when it is unfolded. $\varphi \mathtt{U} \psi$ is then moved to the end of $l$ and gets the lowest priority. Again, player $\forall$ changes focus to the formula with highest priority that is present in the actual configuration if possible.

If at any point the actual configuration $C$ contains an atomic contradiction, i.e. there is a $q \in C$ and a $\overline{q} \in C$ then player $\forall$ immediately sets the focus to one of them and wins with condition 1. The same holds for a $\mathtt{ff} \in C$.

**Lemma 98** ***(Optimality)*** *If player $\forall$ wins $\mathcal{G}(\varphi_0)$ then he wins it with the priority list strategy.*

PROOF Suppose he wins $\mathcal{G}(\varphi_0)$, i.e. he is always able to enforce a play that is winning for himself. If he wins it with his winning condition 1 then he does so with the priority list strategy since it requires him to check at any moment whether he can do so.

Suppose therefore that it is won with his winning condition 2, i.e. $\varphi_0$ contains a $\varphi \mathtt{U} \psi$ that does not get fulfilled during the play. W.l.o.g. we assume that it is the biggest, i.e. there is no superformula of it which is an $\mathtt{U}$ formula as well and which does not get fulfilled either. At the beginning, $\varphi \mathtt{U} \psi$ is inserted into the priority list. Note that the formulas before it in the list can be assumed to be superformulas of $\varphi \mathtt{U} \psi$.

Player $\forall$'s optimal strategy tells him to set the focus to the earliest element of the list that is present in the actual configuration and to keep it there. By assumption, this $\mathtt{U}$ formula gets fulfilled at some point and he changes focus to the next one. Since $\varphi \mathtt{U} \psi$ is assumed to regenerate it must be present at any time and therefore, there must be a moment when player $\forall$ sets the focus to it. Since it does not get fulfilled he leaves the focus there and, by Lemma 94, wins eventually with his winning condition 2.

Note that he never changes the focus back to an $\mathtt{U}$ that has been in focus already before he has tried all other present $\mathtt{U}$ formulas. This is because fulfilled $\mathtt{U}$ formulas get appended to the end of the priority list. ■

**Definition 99** ***(Minimal formula)*** Let $P = C_0, \dots, C_n$ be a play of $\mathcal{G}(\varphi_0)$. Assume every $C_i$ is unsatisfiable and given as a sequence of formulas in increasing order of size, i.e.

$$C_i \;=\; \varphi_{i,0}, \dots, \varphi_{i,n_i} \quad \text{with } \varphi_{i,j} \in Sub(\varphi_{i,k}) \text{ implies } j \leq k$$

for each $i \in \{0, \dots, n\}$. Let $\chi_{C_i}$ denote the $\varphi_{i,k}$ in $C_i$ s.t.

$$\bigwedge_{j<k} \varphi_{i,j} \quad \text{is satisfiable, but} \quad \bigwedge_{j \leq k} \varphi_{i,j} \quad \text{is unsatisfiable.}$$

The *minimal formula causing unsatisfiability* in $P$ is the syntactically smallest formula that occurs first among the $\chi_{C_i}$ for every $C_i$.

$$\chi_P \;:=\; \chi_{C_k} \quad \text{s.t. } \forall i = 0, \dots, n : |\chi_{C_k}| \leq |\chi_{C_i}| \text{ and } \forall j < k : |\chi_{C_k}| < |\chi_{C_j}|$$

**Lemma 100** *Let $\varphi_0$ be unsatisfiable and $P$ be a play of $\mathcal{G}(\varphi_0)$. Then $\chi_P$ exists and is unique.*

PROOF According to Lemma 96, all configurations $C_i$ of $P$ must be unsatisfiable. Thus, each $\chi_{C_i}$ exists. The syntactically smallest among them exists but may not be unique. However, the indices of the configurations are linearly ordered, and $\chi_P$ is the $\chi_{C_i}$ with the smallest $i$ among them. Thus, it is unique. ■

**Lemma 101** *Let $\varphi_0$ be unsatisfiable and $P$ be a play of $\mathcal{G}(\varphi_0)$. Then $\chi_P$ is either atomic or of the form $\varphi \mathtt{U} \psi$.*

PROOF Let $P = C_0, \dots, C_n$ and $C_i = \Phi_i$ with some formula in focus. For a configuration $C_i$ whose elements can be ordered as $\varphi_{i,0}, \dots, \varphi_{i,n_i}$ and with $\chi_{C_i} = \varphi_{i,k}$ for some $k$ according to Definition 99 we let $\Phi_i$ denote the smallest satisfiable part of $C_i$, i.e.

$$\Phi_i \;:=\; \bigwedge_{j<k} \varphi_{i,j}$$

Note that $\models \Phi_i \rightarrow \overline{\chi_{C_i}}$ for every $i = 0, \dots, n$.

Let $k = \min \{\, i \mid \chi_P \in C_i \,\}$ be the index of the earliest configuration containing $\chi_P$. We will show the claim by case analysis on $\chi_P$.

Suppose $\chi_P = q$. $\models \Phi_k \rightarrow \overline{q}$ only if $\overline{q} \in \Phi_k$. Note that $\overline{q}$ could occur in another formula of $\Phi_k$, for example $\Phi_k = \overline{q}\mathtt{R}\overline{q}$. But then there would be a smaller formula, namely $\overline{q}$, which causes unsatisfiability since the game rules remove connectives whilst preserving unsatisfiability. The smallest such formula is $\overline{q}$ itself since it occurs within the scope of no connective, i.e. $\overline{q} \in \Phi_k$.

Suppose $\chi_P = \psi_0 \vee \psi_1$. $\models \Phi_k \rightarrow \overline{(\psi_0 \vee \psi_1)}$ only if $\models \Phi_k \rightarrow \overline{\psi_0}$ and $\models \Phi_k \rightarrow \overline{\psi_1}$. But if rule $([\vee])$ or $(\vee)$ was applied to $\psi_0 \vee \psi_1$ the following configuration will contain either $\psi_0$ or $\psi_1$ which are both syntactically smaller than $\chi_P$ and cause unsatisfiability.

Suppose $\chi_P = \psi_0 \wedge \psi_1$. Then $\models \Phi_k \rightarrow \overline{\psi_0}$ or $\models \Phi_k \rightarrow \overline{\psi_1}$. Note that conjuncts are preserved with rules $([\wedge])$ and $(\wedge)$. Thus, $\chi_P$ cannot be the smallest formula occurring earliest that causes unsatisfiability.

Suppose $\chi_P = \mathtt{X}\psi$. Either $\Phi_k$ consists of atomic propositions and formulas of the form $\mathtt{X}\psi'$ only, or there is a later configuration that does. This is because the game rules eventually produce a configuration to which rule $(\mathtt{X})$ is applicable. But

$$\models \mathtt{X}\psi_1, \ldots, \mathtt{X}\psi_m, q_1, \ldots, q_l \rightarrow \overline{\mathtt{X}\psi}$$

only if

$$\models \psi_1, \ldots, \psi_m \rightarrow \overline{\psi}$$

Hence, the configuration following the next application of rule $(\mathtt{X})$ contains a smaller candidate for $\chi_P$.

Suppose $\chi_P = \varphi\mathtt{R}\psi$. $\overline{\varphi\mathtt{R}\psi} \equiv \overline{\varphi}\mathtt{U}\overline{\psi}$. Therefore, $\models \Phi_k \rightarrow \overline{\varphi\mathtt{R}\psi}$ only if $\models \Phi_k \rightarrow \overline{\varphi}\mathtt{U}\overline{\psi}$. Note that rules $([\mathtt{R}])$ and $(\mathtt{R})$ unfold $\chi_P$ to a conjunction in which $\psi$ is one of the conjuncts. Conjuncts are preserved which means that $\psi$ is present and the other conjunct will either generate $\psi$ after the next application of rule $(\mathtt{X})$ or get replaced by $\varphi$. In the first case there will be a configuration $C_m = \psi, \Phi_m$ s.t. $m > k$ and $\models \Phi_m \rightarrow \overline{\psi}$ which shows that $\chi_P$ was not smallest. In the second case $C_m = \varphi, \Phi_m$ with $m > k$ and $\models \Phi_m \rightarrow \overline{\varphi}$. Again, there would be a smaller formula than $\chi_P$ that causes unsatisfiability.

Finally, suppose $\chi_P = \varphi\mathtt{U}\psi$. $\models \Phi_k \rightarrow \overline{\varphi\mathtt{U}\psi}$ means either there is an $m > k$ s.t.

$$\models \Phi_m \rightarrow \overline{\varphi \vee \psi} \quad \text{but} \quad \not\models \Phi_j \rightarrow \overline{\varphi} \ \text{ for all } k \leq j < m$$

or for all $m \geq k$:

$$\not\models \Phi_m \rightarrow \overline{\varphi} \quad \text{and} \quad \models \Phi_m \rightarrow \overline{\psi}$$

In the first case both $\varphi$ and $\psi$ are smaller formulas than $\chi_P$ and cause unsatisfiability as well. Remember that as long as $\varphi \mathtt{U} \psi$ is unfolded either $\varphi$ or $\psi$ occurs in a configuration. As long as it occurs it must result from the unfolding of $\chi_P$.

However, the second case does not contradict the assumption that $\chi_P$ is syntactically smallest. It results from a play in which player $\exists$ never fulfils $\varphi \mathtt{U} \psi$ s.t. $\varphi$ occurs between each two unfoldings but $\psi$ never does. ■

**Theorem 102 (Soundness)** *If $\varphi_0$ is unsatisfiable then player $\forall$ wins $\mathcal{G}(\varphi_0)$.*

PROOF Assume $\varphi_0$ is unsatisfiable. We show that player $\forall$ wins $\mathcal{G}(\varphi_0)$ by using the priority list strategy.

Take any play $C_0, \ldots, C_n$ of $\mathcal{G}(\varphi_0)$. By Lemma 96, each $C_i$ is unsatisfiable, in particular $C_n$. Thus, player $\exists$ cannot win this play with her winning condition 3 since it requires the last configuration of the play to be satisfiable if player $\forall$ is unable to change the focus. It is impossible for him simply to refuse to do so even though he would be able to as this is excluded by his priority list strategy.

Since $\varphi_0$ is assumed to be unsatisfiable, Lemma 101 applies. Regardless of which play is played, $\chi_P$ is either atomic or an $\mathtt{U}$ formula. Let $C_k$ be the earliest configuration containing $\chi_P$ s.t.

$$C_k \;=\; \chi_P, \Phi_k \qquad \text{and} \qquad \models \Phi_k \rightarrow \overline{\chi_P}$$

If $\chi_P = q$ then $\overline{q}$ must implicitly be present in $\Phi_k$, e.g. in the form $\overline{q}\mathtt{R}\overline{q}$. But the rules remove connectives whilst preserving unsatisfiability and $\overline{q}$ cannot be in the scope of a $\mathtt{X}$. Note that a $\mathtt{X}$ is the only unary connective of LTL. Therefore, after at most $\log|\varphi_0|$ steps the priority list strategy causes player $\forall$ to win the play since he will set the focus to either $q$ or $\overline{q}$ once $\overline{q}$ becomes present.

Suppose $\chi_P$ is of the form $\varphi \mathtt{U} \psi$. If player $\forall$ sets the focus to $\chi_P$ when $C_k$ is reached then he wins the resulting play with his winning condition 2. Note that player $\exists$ can never fulfil $\chi_P$ by assumption. Thus, player $\forall$ can leave the focus on it.

Suppose this is not the case, i.e.

$$C_k \;=\; \left[\varphi'\right], \chi_P, \Phi$$

$\varphi'$ is an $\mathtt{U}$ formula as well since player $\forall$'s strategy only allows him to set the focus to a formula other than that if no $\mathtt{U}$ formula is present. But $\chi_P$ is going to remain present since player $\exists$ cannot fulfil it. Moreover, $\chi_P$ is a member of the priority list at this moment.

We can assume $\varphi'$ to get fulfilled at some point. If it does not then player $\forall$ will win with condition 2 just as he does in the preceding case.

The moment it gets fulfilled it is moved to the end of the priority list and player $\forall$ resets the focus to the $\mathtt{U}$ formula which has highest priority and is present. Note that $\chi_P$ is present and that two formulas only swap their priority order if the one with the higher priority gets fulfilled. Therefore, there are only finitely many $\mathtt{U}$ formulas other than $\chi_P$ the focus can be set to. As soon as one of them persists, player $\forall$ wins with winning condition 2. Eventually, this will be $\chi_P$ unless another one did beforehand.

Note that this argumentation holds for every play of $\mathcal{G}(\varphi_0)$. Thus, player $\forall$ will win each play with his winning condition 1 or 2 if he uses his priority list strategy. ■

**Theorem 103 (Completeness)** *If $\varphi_0$ is satisfiable then player $\exists$ wins $\mathcal{G}(\varphi_0)$.*

PROOF If $\varphi_0$ is satisfiable then it has a model $\pi = s_0, s_1, \ldots$. The LTL formula $\varphi_0$ can be regarded as a CTL$^*$ path formula interpreted over the transition system $\pi$. Since $\pi$ consists of a single path only, it is also a model for the proper CTL$^*$ formula $\mathtt{E}\varphi_0$. According to Theorem 77, $\pi$ can be assumed to be a finite representation of an infinite path, and according to Theorem 72, player $\exists$ wins the CTL$^*$ model checking game $\mathcal{G}_\pi(s_0, \mathtt{E}\varphi_0)$. We will use this game to construct a winning strategy for player $\exists$ in the satisfiability game $\mathcal{G}(\varphi_0)$.

$\mathtt{E}\varphi_0$ contains one CTL$^*$ path quantifier only, therefore each play stays in one single block according to Lemma 64. Player $\exists$ is the path player but her choices with model checking game rule $(\mathtt{X})$ are deterministic since every state $s_i$ has a unique successor $s_{i+1}$. Player $\forall$ has control over the focus in the satisfiability game and the model checking game.

The model checking game starts with rule $(\mathtt{E})$ which removes the existential path quantifier and yields the configuration

$$s_0 \ \vdash \ \mathtt{E}(\left[\varphi_0\right])$$

From then on, every move in the satisfiability game is guided by the model checking game. If $\mathcal{G}(\varphi_0)$ reaches a configuration with a disjunction then player $\exists$ uses her winning strategy in $\mathcal{G}_\pi(s_0,\varphi_0)$ to choose a disjunct and makes the same choice in $\mathcal{G}(\varphi_0)$. Conjunctions are flattened in both games. However, player $\forall$ can set the focus in $\mathcal{G}(\varphi_0)$ to a different formula than the one in focus in $\mathcal{G}_\pi(s_0,\varphi_0)$. But the rules of the model checking game allow him to reset the focus at any point. This means that there is a position in the model checking game tree for player $\exists$ which corresponds to the actual position in $\mathcal{G}(\varphi_0)$, s.t. player $\exists$ has a winning strategy for the game continuing with this configuration.

Suppose the model checking play visits a configuration

$$s_j \ \vdash \ \mathtt{E}(\left[\psi\right],\Phi)$$

after it visited

$$s_i \ \vdash \ \mathtt{E}(\left[\psi\right],\Phi)$$

This is not a repeat since these configurations differ in the state component. However, such a play would correspond to a repeat in $\mathcal{G}(\varphi_0)$. To maintain a full correspondence between the model checking and the satisfiability game we restart the construction of player $\exists$'s game tree for $\mathcal{G}(\varphi_0)$ at the first occurrence of the position

$$\left[\psi\right],\Phi$$

Note that this is only done if the model checking play visits two configurations with different state components.

Then there is a repeat in $\mathcal{G}(\varphi_0)$ iff there is a repeat in $\mathcal{G}_\pi(s_0,\varphi_0)$. By assumption, $\pi$ is a finite representation of an infinite path, therefore the model checking play will eventually perform a repeat. Thus, the restarting process for the satisfiability play will eventually terminate.

Player $\forall$ cannot win a play of $\mathcal{G}(\varphi_0)$ with his winning condition 2. Remember that this means he is able to keep the focus on a $\varphi \mathtt{U} \psi$ until the play performs a repeat. But this would only be possible if he was also able to do this in $\mathcal{G}_\pi(s_0, \varphi_0)$ which contradicts the assumption that player $\exists$ is the winner of this.

He cannot win by condition 1 either since this would enable him to win the model checking play by setting the focus to a proposition that is not satisfied by the actual state. Remember that state labellings are total, i.e. for every $q \in \mathcal{P}$ and every state $s$ either $q \in L(s)$ or $\overline{q} \in L(s)$. But his winning condition 1 requires both of them to be present in a configuration which cannot occur in player $\exists$'s model checking game tree for $\mathcal{G}_\pi(s_0, \varphi_0)$.

By Corollary 95, player $\exists$ wins $\mathcal{G}(\varphi_0)$. ■

**Theorem 104 (Small model property)** *If $\varphi_0 \in LTL$ is satisfiable then it has a model of size less than $|\varphi_0| \cdot 2^{|\varphi_0|}$.*

PROOF Suppose $\varphi_0$ is satisfiable. By Theorem 103, player $\exists$ wins $\mathcal{G}(\varphi_0)$. Let $C_0, \ldots, C_n$ be the resulting play. We define a finite representation of a possibly infinite path $\pi$ of a transition system in the following way. The states of $\pi$ are equivalence classes $[C_i]$ of the set of all occurring configurations $C_0, \ldots, C_n$ under the equivalence relation

$$C_i \sim C_j \quad \text{iff} \quad \text{between } C_i \text{ and } C_j \text{ there is no application of rule } (\mathtt{X}).$$

Then $[C_i] := \{C_j \mid C_j \sim C_i\}$. Transitions in $\pi$ are defined as

$$[C_i] \rightarrow [C_k] \quad \text{iff} \quad C_i \not\sim C_k \text{ and there is a } j \in \mathbb{N} \text{ s.t. } C_i \sim C_j \text{ and } C_{j+1} \sim C_k.$$

The labelling of the states of $\pi$ is defined as

$$q \in L([C_i]) \quad \text{iff there is a } j \in \mathbb{N} \text{ s.t. } C_i \sim C_j \text{ and } q \in C_j.$$

$\pi$ is an eventually cyclic finite representation of an infinite path if the corresponding play was won with winning condition 4 or 5. It is a finite path if it was won with winning condition 3. In that case add an arbitrary loop to its end to fulfil the totality requirement.

Lemma 93 shows that the size of this representation of $\pi$ is bounded by $|\varphi_0| \cdot 2^{|\varphi_0|}$.

We claim that $\pi$ is a model for $\varphi_0$. Let $\pi^{([C_i])}$ denote the suffix of $\pi$ that begins with $[C_i]$. We show by induction on the formula structure that for all $i, j < n$:

$$\pi^{([C_i])} \models \psi \qquad \text{for all} \quad \psi \in C_j \quad \text{if} \quad C_i \sim C_j$$

This is true for atomic propositions $q$ because of the way the labellings of the states were chosen. Suppose it is true for $\varphi$ and $\psi$.

If $\varphi \vee \psi \in C_j$ for some $j$ then game rules $([\vee])$ and $(\vee)$ guarantee that there is a $C_i$ with $C_i \sim C_j$ and either $\varphi \in C_i$ or $\psi \in C_i$. But then $\pi^{([C_i])} \models \varphi$ or $\pi^{([C_i])} \models \psi$ by hypothesis and, hence, $\pi^{([C_i])} \models \varphi \vee \psi$. The cases of $\varphi \wedge \psi$ and $\mathtt{X}\varphi$ are similar.

If $\varphi \mathtt{U} \psi \in C_j$ for some $j$ then player $\exists$'s winning strategy guarantees that there is a $C_k$ with $\psi \in C_k$ because she has fulfilled all occurring $\mathtt{U}$ formulas. Otherwise player $\forall$ would have won the corresponding play with his winning condition 2 according to Lemma 98. The induction hypothesis yields $\pi^{([C_k])} \models \psi$. Furthermore, for every $i$ with $j \leq i < k'$ where $k'$ is chosen least s.t. $C_k \sim C_{k'}$, there is an $i'$ s.t. $C_i \sim C_{i'}$ and $\varphi \in C_{i'}$. But then $\pi^{([C_i])} \models \varphi \mathtt{U} \psi$.

Finally, the case of $\varphi \mathtt{R} \psi \in C_j$ is similar. Now note that $\varphi_0 \in C_0$ and $\pi^{([C_0])} = \pi$. Thus, $\pi \models \varphi_0$. ■

History-freeness of player $\exists$'s winning strategy carries over from the CTL$^*$ model checking game to the LTL satisfiability game. The situation for player $\forall$ is different.

**Theorem 105 (Winning strategies)**

*a) Player $\exists$'s winning strategies are history-free.*

*b) Player $\forall$'s priority list strategies are LVR strategies.*

PROOF Player $\exists$'s winning strategy for $\mathcal{G}(\varphi_0)$ with a satisfiable $\varphi_0$ consists of choosing a model $\pi$ for $\varphi_0$ and playing according to her strategy for the CTL$^*$ model checking game $\mathcal{G}_\pi(s_0, \mathtt{E}\varphi_0)$. The choice of the model does not depend on the play, and by Theorem 74, her winning strategy for $\mathcal{G}_\pi(s_0, \mathtt{E}\varphi_0)$ is history-free. Hence, so is her winning strategy for $\mathcal{G}(\varphi_0)$.

Player $\forall$'s winning strategy for $\mathcal{G}(\varphi_0)$ with an unsatisfiable $\varphi_0$ is different. Remember that he is only concerned with the position of the focus. His priority list is in fact a latest visitation record. According to Definition 41 of Chapter 2, the set of interesting configurations for him is the set of all possible configurations of the form $\left[\varphi U \psi\right], \Phi$ for every $\varphi U \psi \in Sub(\varphi_0)$.

The priority list of Definition 97 is a succinct representation of this LVR. Note that it is essential but also sufficient for player $\forall$ to keep the focus on a $\varphi U \psi$. Maintaining a priority list of all configurations would not give him more information than is needed to win.

Player $\forall$ only moves elements from positions in the list to its end. At the beginning, no element occurs twice. Thus, the requirements for a LVR are fulfilled. ■

## Complexity

The close correspondence between CTL$^*$ model checking games and LTL satisfiability games is reflected in the analysis of its complexity. The proof of the following theorem is similar to the one of Theorem 79.

**Theorem 106 (Complexity)** *Deciding the winner of an LTL satisfiability game is in PSPACE.*

PROOF A game based satisfiability checking algorithm for LTL can make use of the priority list strategy described in the proof of Theorem 102. This determinises player $\forall$'s moves. What remains is a nondeterministic game since the existential player is left with some choices. To find a winning play for player $\exists$ the algorithm needs to store two configurations: the actual one which gets overwritten each time a game rule is played, and one which is used to find a repeat on.

It is up to player $\forall$ to find a repeat on a configuration $\left[\varphi U \psi\right], \Phi$ without changing focus between the two occurrences. Therefore, he would at some point universally choose to store such a configuration. However, the result would be an alternating procedure which will not have optimal complexity. Therefore, we determinise player $\forall$'s choice of the configuration to repeat on, similar to the proof of Theorem 79.

Let $\varphi$ be the input formula. The algorithm maintains a counter to measure the length of the play at hand. It starts by storing the first configuration. With this it stores the counter value. It proceeds to check whether there is a play that repeats on this configuration. A simple flag is used to indicate whether the focus was changed in between or not. If there is a repeat and the focus was not changed it returns $\forall$ as the winner. If there is no repeat then a counter is used to terminate the play at hand. The algorithm returns ? as soon as the counter value reaches $|\varphi| \cdot 2^{|\varphi|} + 3$ which is justified by Lemma 93. Then it restarts with the successor of the stored configuration and the stored counter value increased by one.

If the stored counter value reaches $|\varphi| \cdot 2^{|\varphi|} + 3$ then the algorithm terminates and returns $\exists$ as the winner.

The size of the counter is polynomial in the size of the input. The size of the memory needed to store two configurations is polynomial in the size of the input as well. Therefore, LTL satisfiability checking can be done in nondeterministic PSPACE. According to [Sav69], there is a deterministic PSPACE algorithm for this problem as well. ■

### Comparing Automata and Games for LTL Satisfiability Checking

The first automata to be used for deciding satisfiability of LTL formulas were nondeterministic Büchi automata. There is of course an obvious difference to the games of this section since 2-player games correspond to alternating rather than nondeterministic automata. However, as Theorem 106 shows, applying the priority list strategy determinises player $\forall$'s moves and leaves a nondeterministic game.

These nondeterministic automata guess truth values for each subformula of the input formula at each state of a possible model and verify these guesses with their transition relation. The games of this section are more flexible in this respect since configurations only contain necessary subformulas. The verification of the correctness of the automaton's choices can be seen as the automata-theoretic counterpart to Lemma 96 which states that unsatisfiability is preserved.

There is an intimate relationship between the model checking problem for CTL$^*$

and the satisfiability checking problem for LTL. This is not only reflected in their computational complexities – both are PSPACE-complete – but also in the similarities between the BLTL model checking games of Chapter 5 and the LTL satisfiability games of this section. Remember that a CTL$^*$ model checking games is in fact a collection of BLTL model checking games.

The proof of Theorem 103 is based heavily on the close relationship between these games. In fact, an LTL satisfiability game is a BLTL model checking game without the state component in a configuration.

Since these relationships are merely a feature of the games but a property of the logics, the automata that have been used for BLTL model checking can also be used for LTL satisfiability checking. This means that the alternating automata from [Var96] are useful for the satisfiability problem as well, see the comparisons in Section 5.5.

Again, the games of this section can be seen as an intermediate step between the alternating automata and their translation into nondeterministic ones. Remember that the alternating automata's states consist of single formulas only.

The question of whether there is a regenerating $\mathtt{U}$ formula is answered in the non-emptiness test of the language accepted by the automaton. This problem is PSPACE-complete. For the games this question is easier to answer as the proof of Theorem 106 shows. It is simply done by querying the value of a boolean flag. However, it is only easier since a game's configuration is more complex than an automaton's state. But it is the size of a configuration in a game that causes the PSPACE complexity.

## 6.2 Satisfiability Games for CTL

The *satisfiability game* $\mathcal{G}(\varphi_0)$ for a CTL formula $\varphi_0$ is defined along the lines of Section 6.1. Player $\exists$ attempts to show that $\varphi_0$ has a model, whereas player $\forall$ tries to show that there is none. Here, a model is a total transition system $\mathcal{T}$. The set of configurations of the focus game $\mathcal{G}(\varphi_0)$ is

$$\mathcal{C} \;=\; Sub(\varphi_0) \times 2^{Sub(\varphi_0)}$$

The game rules are depicted in Figure 6.3. Boolean combinators are handled in the same way as they are in the LTL games, and so is the focus. Rules $([Q\mathtt{U}])$, $([Q\mathtt{R}])$, $(Q\mathtt{U})$ and $(Q\mathtt{R})$ are justified by the unfoldings of the temporal operators in CTL.

Because of the path quantifiers, applying the above rules will result in a configuration in which every formula is either propositional or of the form $\mathtt{EX}\psi$ or $\mathtt{AX}\psi$. Such a configuration postulates the existence of several successor states, each of them satisfying all of the universally quantified formulas and at least one existentially quantified formula, s.t. every $\mathtt{EX}\varphi$ is covered by one successor state. This is modelled in the rules $(\mathtt{EX})$ and $(\mathtt{AX})$.

The winning conditions for the CTL games are similar to those for the LTL games, and so is the definition of a terminal configuration. Player $\forall$ wins the play $C_0,\ldots,C_n$ iff

1. $C_n = \Big[q\Big],\Phi$ is terminal and $q = \mathtt{ff}$ or $\overline{q} \in \Phi$, or
2. $C_n = \Big[Q(\varphi\mathtt{U}\psi)\Big],\Phi$ for some $Q \in \{\mathtt{E},\mathtt{A}\}$ and there is an $i \in \mathbb{N}$, s.t. $i < n$ and $C_i = C_n$, and player $\forall$ has not used rule $(\mathtt{FC})$ between $C_i$ and $C_n$.

Player $\exists$ wins the play $C_0,\ldots,C_n$ iff

3. $C_n = \Big[q\Big],\Phi$ is terminal, $q \neq \mathtt{ff}$ and $\overline{q} \notin \Phi$, or
4. $C_n = \Big[\varphi\Big],\Phi$ and there is an $i \in \mathbb{N}$, s.t. $i < n$ and $C_i = C_n$, and player $\forall$ has used rule $(\mathtt{FC})$ between $C_i$ and $C_n$.
5. $C_n = \Big[Q(\varphi\mathtt{R}\psi)\Big],\Phi$ for some $Q \in \{\mathtt{E},\mathtt{A}\}$ and there is an $i \in \mathbb{N}$, s.t. $i < n$ and $C_i = C_n$, and player $\forall$ has not used rule $(\mathtt{FC})$ between $C_i$ and $C_n$.

## Correctness

As in the LTL case we need to establish a few facts and lemmas before we can proceed to prove the games correct.

$$([\vee])\ \frac{\left[\varphi_0 \vee \varphi_1\right], \Phi}{\left[\varphi_i\right], \Phi}\ \exists i \qquad ([Q\mathtt{U}])\ \frac{\left[Q(\varphi \mathtt{U} \psi)\right], \Phi}{\left[\psi \vee (\varphi \wedge Q\mathtt{X}Q(\varphi \mathtt{U} \psi))\right], \Phi}$$

$$([\wedge])\ \frac{\left[\varphi_0 \wedge \varphi_1\right], \Phi}{\left[\varphi_i\right], \varphi_{1-i}, \Phi}\ \forall i \qquad ([Q\mathtt{R}])\ \frac{\left[Q(\varphi \mathtt{R} \psi)\right], \Phi}{\left[\psi \wedge (\varphi \vee Q\mathtt{X}Q(\varphi \mathtt{R} \psi))\right], \Phi}$$

$$(\vee)\ \frac{\left[\psi\right], \varphi_0 \vee \varphi_1, \Phi}{\left[\psi\right], \varphi_i, \Phi}\ \exists i \qquad (Q\mathtt{U})\ \frac{\left[\chi\right], Q(\varphi \mathtt{U} \psi), \Phi}{\left[\chi\right], \psi \vee (\varphi \wedge Q\mathtt{X}Q(\varphi \mathtt{U} \psi)), \Phi}$$

$$(\wedge)\ \frac{\left[\psi\right], \varphi_0 \wedge \varphi_1, \Phi}{\left[\psi\right], \varphi_0, \varphi_1, \Phi} \qquad (Q\mathtt{R})\ \frac{\left[\chi\right], Q(\varphi \mathtt{R} \psi), \Phi}{\left[\chi\right], \psi \wedge (\varphi \vee Q\mathtt{X}Q(\varphi \mathtt{R} \psi)), \Phi}$$

$$(\mathtt{EX})\ \frac{\mathtt{AX}\psi_1, \ldots, \mathtt{AX}\psi_m, \left[\mathtt{EX}\varphi_1\right], \ldots, \mathtt{EX}\varphi_k, q_1, \ldots, q_n}{\psi_1, \ldots, \psi_m, \left[\varphi_1\right]} \qquad (\mathtt{FC})\ \frac{\left[\varphi\right], \psi, \Phi}{\left[\psi\right], \varphi, \Phi}\ \forall$$

$$(\mathtt{AX})\ \frac{\left[\mathtt{AX}\psi_1\right], \ldots, \mathtt{AX}\psi_m, \mathtt{EX}\varphi_1, \ldots, \mathtt{EX}\varphi_k, q_1, \ldots, q_n}{\left[\psi_1\right], \ldots, \psi_m, \varphi_i}\ \forall i$$

Figure 6.3: The satisfiability game rules for CTL.

**Fact 107** $(\mathtt{FC})$ *is the only rule that maintains the size of a configuration. Rules* $([\vee])$, $(\vee)$, $([\wedge])$, $(\wedge)$, $(\mathtt{EX})$ *and* $(\mathtt{AX})$ *reduce the number of connectives in a configuration, while rules* $([Q\mathtt{U}])$, $(Q\mathtt{U})$, $([Q\mathtt{R}])$ *and* $(Q\mathtt{R})$ *increase the number of connectives.*

**Lemma 108** *Every play of* $\mathcal{G}(\varphi)$ *has finite length less than* $|\varphi| \cdot 2^{|\varphi|} + 3$ *and a uniquely determined winner.*

This is proved exactly like Lemmas 93 and 94 for LTL. Consequently, determinacy follows for the CTL satisfiability games in the same way.

**Corollary 109 (Determinacy)** *Player* $\forall$ *wins* $\mathcal{G}(\varphi)$ *iff player* $\exists$ *does not win* $\mathcal{G}(\varphi)$.

**Lemma 110** *Player* $\exists$ *preserves unsatisfiability with her rules. Player* $\forall$ *can preserve unsatisfiability.*

PROOF The rules for boolean connectives and the focus change rule are present in the LTL games as well. Since player $\exists$ only chooses disjuncts the claim follows for her from Lemma 96 already.

Preservation of unsatisfiability with the deterministic unfolding rules $([Q\mathtt{U}])$, $(Q\mathtt{U})$, $([Q\mathtt{R}])$ and $(Q\mathtt{R})$ follows from the unfolding characterisation of $\mathtt{U}$ and $\mathtt{R}$ formulas in CTL which was presented in Section 2.4.

The only new cases are those of rules $(\mathtt{EX})$ and $(\mathtt{AX})$. They do not need to be looked at separately. Suppose

$$\psi_1, \ldots, \psi_m, \varphi_i$$

is satisfiable for every $\varphi_i$, $i = 1, \ldots, k$. Thus each has a model $\mathcal{T}_i$ with a state $s_i$ s.t. $s_i \models \varphi_i$ and $s_i \models \psi_j$ for $j = 1, \ldots, m$. Define a new LTS $\mathcal{T}'$ as the disjoint union over all $\mathcal{T}_i$ with a new state $s'$ s.t. $s' \to s_i$ for each $i = 1, \ldots, k$. Let $s'$ be consistently labelled with $L(s') = \{q_1, \ldots, q_n\}$. Then,

$$\mathcal{T}', s' \models \mathtt{AX}\psi_1, \ldots, \mathtt{AX}\psi_m, \mathtt{EX}\varphi_1, \ldots, \mathtt{EX}\varphi_k, q_1, \ldots, q_n$$

Thus, this formula is satisfiable. Therefore, if it is unsatisfiable then one of the $\varphi_i, \psi_1, \ldots, \psi_m$ must be unsatisfiable as well. In rule $(\mathtt{AX})$ player $\forall$ can choose it accordingly and preserve unsatisfiability.

In order to preserve unsatisfiability with rule $(\mathtt{EX})$ he might have to change focus to the $\mathtt{EX}\varphi_i$ that causes the unsatisfiability before the rule is played. Note that he is allowed to change focus at any moment in the play. ■

As in the LTL case, we will describe a priority list strategy for player $\forall$. The difference to Definition 97 is the fact that player $\forall$ has to use several lists in a game $\mathcal{G}(\varphi_0)$.

**Definition 111 (Priority list strategy)** Let $l$ be a *priority list* of all $\mathtt{U}$ subformulas of $\varphi_0$ in decreasing order of size, i.e.

$$l \;=\; Q_1(\varphi_1\mathtt{U}\psi_1),\ldots,Q_n(\varphi_n\mathtt{U}\psi_n)$$

with

$$Q_i(\varphi_i\mathtt{U}\psi_i) \in Sub(Q_j(\varphi_j\mathtt{U}\psi_j)) \quad \text{and} \quad Q_i(\varphi_i\mathtt{U}\psi_i) \neq Q_j(\varphi_j\mathtt{U}\psi_j) \quad \text{implies} \quad j < i$$

In that case $Q_j(\varphi_j\mathtt{U}\psi_j)$ is said to have higher priority than $Q_i(\varphi_i\mathtt{U}\psi_i)$. We say that $Q(\varphi\mathtt{U}\psi)$ is present in a configuration $C$ if

$$\{\ Q(\varphi\mathtt{U}\psi),\ \psi\vee(\varphi\wedge Q\mathtt{X}Q(\varphi\mathtt{U}\psi)),\ \varphi\wedge Q\mathtt{X}Q(\varphi\mathtt{U}\psi)\in C,\ Q\mathtt{X}Q(\varphi\mathtt{U}\psi)\ \}\ \cap\ C\ \neq\ \emptyset$$

Player $\forall$ uses the priority list as it is described in Definition 97. He attempts to set the focus to the $\mathtt{U}$ formula with the highest priority that is present or a superformula of it. Here, an $\mathtt{U}$ formula means a formula of the form $\mathtt{E}(\varphi\mathtt{U}\psi)$ or $\mathtt{A}(\varphi\mathtt{U}\psi)$. Fulfilled $\mathtt{U}$ formulas get appended to the end of the list. At any moment he checks whether he can win by setting the focus to an atomic proposition.

Note an essential difference between the LTL and the CTL satisfiability games. In the LTL case player $\forall$ only chooses the position of the focus. This is entirely determinised by the priority list strategy. Here, player $\forall$ also makes choices with rule $(\mathtt{AX})$. This is unaffected by the priority list strategy. His overall strategy is therefore composed of several priority list strategies, each corresponding to a certain sequence of choices he makes with rule $(\mathtt{AX})$ in a play. In addition, whenever this rule has to be played he chooses the $\mathtt{EX}\varphi_i$ that preserves unsatisfiability if the actual configuration is unsatisfiable. If it is satisfiable he can choose any formula.

Thus, in a game tree for player $\exists$, player $\forall$ will have used several priority list strategies since the presence of an $\mathtt{U}$ formula generally depends on the choices made with rule $(\mathtt{AX})$.

We will speak of *the* priority list strategy to denote his overall strategy that combines the priority list idea with the preservation of unsatisfiability.

The next lemma is proved in the same way as Lemma 98 for the LTL games.

**Lemma 112** ***(Optimality)*** *If player $\forall$ wins $\mathcal{G}(\varphi_0)$ then he wins it with the priority list strategy.*

The *minimal formula* $\chi_P$ causing unsatisfiability in a play $P$ of a CTL satisfiability game is defined just as it is in Definition 99 for LTL. $\chi_P$ is the syntactically least formula that causes unsatisfiability and that occurs earliest in a configuration of $P$.

**Lemma 113** *Let $\varphi_0$ be unsatisfiable and $P$ be a play of $\mathcal{G}(\varphi_0)$ in which player $\forall$ uses his priority list strategy. Then $\chi_P$ is either atomic or of the form $Q(\varphi\mathtt{U}\psi)$ for a $Q \in \{\mathtt{E},\mathtt{A}\}$.*

PROOF This is proved by case analysis on $\chi_P$ as well. Note that the cases of atomic propositions, disjunctions and conjunctions are the same as the ones in the proof of Lemma 101.

$\chi_P = \mathtt{EX}\psi$ is impossible as well as $\chi_p = \mathtt{AX}\psi$ since rules $(\mathtt{EX})$ and $(\mathtt{AX})$ produce the syntactically smaller $\psi$ in the latter case anyway and in the former case if the priority list strategy is used.

The cases of $\chi_P = Q(\varphi\mathtt{R}\psi)$ are similar to the case of a $\mathtt{R}$ formula in the proof of Lemma 101. Regardless of $Q$, the syntactically smaller $\psi$ will always be present in later configurations and eventually cause unsatisfiability unless $\varphi$ does.

The remaining cases are those of $\chi_P = Q(\varphi\mathtt{U}\psi)$, $Q \in \{\mathtt{E},\mathtt{A}\}$. Let $C_k = \chi_P, \Phi_k$ be the first configuration in $P$ containing $\chi_P$, s.t. $\models \Phi_k \to \overline{Q(\varphi\mathtt{U}\psi)}$. Again, $\Phi_k$ denotes the satisfiable part of $C_k$ in the sense of Lemma 101. Then either there is an $m > k$ s.t.

$$\models \Phi_m \to \overline{\varphi \vee \psi} \quad \text{but} \quad \not\models \Phi_j \to \overline{\varphi} \ \text{ for all } k \leq j < m$$

or for all $m \geq k$:

$$\not\models \Phi_m \rightarrow \overline{\varphi} \quad \text{and} \quad \models \Phi_m \rightarrow \overline{\psi}$$

In the first case both $\varphi$ and $\psi$ are smaller formulas than $\chi_P$ and cause unsatisfiability as well. Remember that as long as $Q(\varphi \mathtt{U} \psi)$ is unfolded either $\varphi$ or $\psi$ occurs in a configuration. As long as it occurs it must result from the unfolding of $\chi_P$.

Again, the second case does not contradict the assumption that $\chi_P$ is syntactically smallest. It is found in a play in which player $\exists$ never fulfils $Q(\varphi \mathtt{U} \psi)$ s.t. $\varphi$ occurs between each two unfoldings but $\psi$ never does. ■

**Theorem 114 (Soundness)** *If $\varphi_0$ is unsatisfiable then player $\forall$ wins $\mathcal{G}(\varphi_0)$.*

PROOF Assume $\varphi_0$ is unsatisfiable. As in the proof of Theorem 102, we show that player $\forall$ wins $\mathcal{G}(\varphi_0)$ by using his priority list strategy.

Consider a play $C_0, \ldots, C_n$ of $\mathcal{G}(\varphi_0)$. By Lemma 96, each $C_i$ is unsatisfiable, in particular $C_n$. Thus, player $\exists$ cannot win this play with her winning condition 3 since it requires the last configuration of the play to be satisfiable. Remember that the case of player $\forall$ simply refusing to continue to play is excluded by using the priority list strategy.

Since $\varphi_0$ is assumed to be unsatisfiable, Lemma 113 applies. Regardless of which play $P$ is played, $\chi_P$ is either atomic or of the form $Q(\varphi \mathtt{U} \psi)$. Let $C_k$ be the earliest configuration containing $\chi_P$ s.t.

$$C_k \;=\; \chi_P, \Phi_k \qquad \text{and} \qquad \models \Phi_k \rightarrow \overline{\chi_P}$$

If $\chi_P = q$ then the priority list strategy causes player $\forall$ to win the play since he will set the focus to either $q$ or $\overline{q}$. Note that $\overline{q}$ must either be present in $C_k$ or occur at most $\log|\varphi_0|$ steps later.

Suppose $\chi_P$ is of the form $Q(\varphi \mathtt{U} \psi)$. If player $\forall$ sets the focus to $\chi_P$ when $C_k$ is reached then he wins the resulting play with his winning condition 2. Note that player $\exists$ can never fulfil $\chi_P$ by assumption. Thus, player $\forall$ can leave the focus on it.

Suppose this is not the case, i.e.

$$C_k \;=\; \left[\varphi'\right], \chi_P, \Phi$$

$\varphi'$ is an $\mathtt{U}$ formula as well since player $\forall$'s strategy only allows him to set the focus to a formula other than that if no $\mathtt{U}$ formula is present. But if the $\mathtt{U}$ formula is of the form $\mathtt{A}(\varphi\mathtt{U}\psi)$ then $\chi_P$ is going to remain present since player $\exists$ cannot fulfil it. Moreover, $\chi_P$ is a member of the priority list at this moment. If it is of the form $\mathtt{E}(\varphi\mathtt{U}\psi)$ then it could theoretically be discarded by an application of rule $(\mathtt{EX})$. But since player $\forall$ is assumed to preserve unsatisfiability it would not be the $\chi_P$ for the play $P$ at hand. Hence, player $\exists$ cannot fulfil it either and it is also a member of the priority list.

We can assume $\varphi'$ to get fulfilled at some point. If it does not then player $\forall$ will win with condition 2 just as he does in the preceding case.

The moment it gets fulfilled it is moved to the end of the priority list and player $\forall$ resets the focus to the $\mathtt{U}$ formula which has highest priority and is present. Note that $\chi_P$ is present and that two formulas only swap their priority order if the one with the higher priority gets fulfilled. Therefore, there are only finitely many $\mathtt{U}$ formulas other than $\chi_P$ the focus can be set to. As soon as one of them persists, player $\forall$ wins with winning condition 2. Eventually, this will be $\chi_P$ unless another one did beforehand.

Note that the argumentation above holds for every play of $\mathcal{G}(\varphi_0)$. Thus, player $\forall$ will win each play either with his winning condition 1 or 2 if he uses his priority list strategy. ■

Similar to the proof of Theorem 103, we will relate the satisfiability games for CTL to its model checking games of Section 5.3 and obtain completeness in this way. However, one satisfiability play must be related to several model checking plays since configurations of the latter contain single formulas only.

**Theorem 115 (Completeness)** *If $\varphi_0$ is satisfiable then player $\exists$ wins $\mathcal{G}(\varphi_0)$.*

PROOF Suppose $\varphi_0$ is satisfiable, i.e. it has a model $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ with $s_0 \in \mathcal{S}$ s.t. $s_0 \models \varphi_0$. $\varphi_0$ is also a CTL$^*$ formula. Thus, by Theorem 77, $\mathcal{T}$ can be assumed to be finite. Player $\exists$'s moves in $\mathcal{G}(\varphi_0)$ will be guided by her moves in the CTL model checking games $\mathcal{G}_{\mathcal{T}}(s, \psi)$ where $s \in \mathcal{S}$ and $\psi \in Sub(\varphi_0)$. Remember that the CTL model checking game is not a focus game.

The starting positions for both plays are $\left[\varphi_0\right]$ and $s_0 \vdash \varphi_0$. Suppose the actual formula in focus is a disjunction. Then player $\exists$ uses her winning strategy in

$\mathcal{G}_{\mathcal{T}}(s_0, \varphi_0)$ to choose the disjunct that guarantees her to win the remaining play. In the satisfiability game she chooses the same disjunct. Unfolding of temporal operators is deterministically done in the same way in both plays.

The only interesting case is the one of a conjunction in $\mathcal{G}(\varphi_0)$. Consider the first occurrence of such a situation in $\mathcal{G}(\varphi_0)$. At this moment no sideformula can be present. Let therefore $\left[\psi_0 \wedge \psi_1\right]$ be such a configuration. This must correspond to a position

$$s \ \vdash \ \psi_0 \wedge \psi_1$$

in the model checking play. Since player $\exists$ is assumed to have a winning strategy for this game she must have winning strategies for both $\mathcal{G}_{\mathcal{T}}(s, \psi_0)$ and $\mathcal{G}_{\mathcal{T}}(s, \psi_1)$. In $\mathcal{G}(\varphi_0)$ player $\forall$ sets the focus to one of the conjuncts, say $\psi_0$. Then all choices regarding formulas in focus are matched by choices in $\mathcal{G}_{\mathcal{T}}(s, \psi_0)$, whereas all choices regarding sideformulas correspond to choices in $\mathcal{G}_{\mathcal{T}}(s, \psi_1)$. Thus, at any moment in the satisfiability game a configuration containing $n$ formulas is matched by $n$ model checking plays. Furthermore, the state component of all model checking plays is always the same. Changing focus does not alter the situation on the model checking side at all.

Finally, if the satisfiability play reaches a configuration

$$\mathtt{AX}\psi_1, \ldots, \mathtt{AX}\psi_m, \left[\mathtt{EX}\varphi_1\right], \ldots, \mathtt{EX}\varphi_k, q_1, \ldots, q_n$$

the model checking plays corresponding to $\mathtt{EX}\varphi_2, \ldots, \mathtt{EX}\varphi_k$ are discarded. Player $\exists$ chooses a successor state $t$ in the play for $\mathtt{EX}\varphi_1$. By assumption she has winning strategies for the remaining model checking games

$$\mathcal{G}_{\mathcal{T}}(t, \varphi_1),\ \mathcal{G}_{\mathcal{T}}(t, \psi_1),\ \ldots,\ \mathcal{G}_{\mathcal{T}}(t, \psi_m)$$

The same argument holds for a configuration with the focus on an $\mathtt{AX}\psi$, the only difference being player $\forall$ who determines the model checking play in which player $\exists$ chooses a successor state.

It is possible for the satisfiability play to perform a repeat on a configuration $\left[\psi_0\right], \Phi$ while the set of model checking plays does not. Let $\Phi = \psi_1, \ldots, \psi_n$. Whenever the model checking plays are at stages

$$t \ \vdash \ \psi_i \qquad \text{for } i = 1, \ldots, n$$

after they were at stages $s \vdash \psi_i$, and $s \neq t$, and the focus is on the same formula, then the satisfiability game is restarted at the first occurrence of $\left[\psi_0\right], \Phi$. This is not done if $s = t$.

Suppose now that player $\forall$ wins the satisfiability game. If he does so with winning condition 1 then there must be two configurations

$$t \ \vdash \ q \qquad \text{and} \qquad t \ \vdash \ \overline{q}$$

in the set of model checking plays. Player $\exists$ cannot win both of the model checking plays as she is assumed to. Suppose therefore that player $\forall$ wins with condition 2. But a repeat with a $Q(\varphi \mathtt{U} \psi)$ in focus corresponds to a model checking play that repeats on $Q(\varphi \mathtt{U} \psi)$ as well and would be won by player $\forall$, too.

We conclude therefore that player $\forall$ cannot win any play of $\mathcal{G}(\varphi_0)$ and by Corollary 109 that player $\exists$ must have a winning strategy for $\mathcal{G}(\varphi_0)$. ∎

**Theorem 116 (Small model property)** *If $\varphi_0 \in CTL$ is satisfiable then it has a model of size less than $|\varphi_0| \cdot 2^{|\varphi_0|}$.*

PROOF Suppose $\varphi_0$ is satisfiable. By Theorem 115, player $\exists$ has a winning strategy for the game $\mathcal{G}(\varphi_0)$. We extract a transition system $\mathcal{T}$ from player $\exists$'s game tree. $\mathcal{T}$ will be a tree-like structure. A play in the game tree will be transformed into a branch $\pi$ of $\mathcal{T}$. For each play $C_0, C_1, \ldots, C_n$ the branch $\pi$ consists of states which are equivalence classes $[C_i]$ of the set of all occurring configurations $C_i$ under the equivalence relation

$$C_i \sim C_j \quad \text{iff} \quad \text{between } C_i \text{ and } C_j \text{ there is no application of rule } (\mathtt{EX}) \text{ or } (\mathtt{AX}).$$

Then $[C_i] := \{C_j \mid C_j \sim C_i\}$. Transitions in $\mathcal{T}$ are defined as

$$[C_i] \to [C_k] \quad \text{iff} \quad C_i \not\sim C_k \text{ and there is a } j \in \mathbb{N} \text{ s.t. } C_i \sim C_j \text{ and } C_{j+1} \sim C_k.$$

The labelling of the states of $\mathcal{T}$ is defined as

$$q \in L([C_i]) \quad \text{iff} \quad \text{there is a } j \in \mathbb{N} \text{ s.t. } C_i \sim C_j \text{ and } q \in C_j.$$

Each $\pi$ is an eventually cyclic finite representation of an infinite path unless it resulted from a play which is won by player $\exists$ with her winning condition 3. In that case $\pi$ can be made into an eventually cyclic path model by appending another state $[C_{n+1}]$ with

$$[C_n] \rightarrow [C_{n+1}] \qquad \text{and} \qquad [C_{n+1}] \rightarrow [C_{n+1}]$$

The paths can be put together to obtain a finite representation $\mathcal{T}$ of an infinite tree. Note that each path $\pi$ starts with $[C_0]$.

Since there are only $|\varphi_0| \cdot 2^{|\varphi_0|}$ many different configurations of $\mathcal{G}(\varphi_0)$ this is also an upper bound on the number of equivalence classes $[C_i]$ and, hence, the size of $\mathcal{T}$.

It remains to be seen that $\mathcal{T}, [C_0]$ is a model for $\varphi_0$. In fact, the following stronger proposition holds for all $i, j < n$:

$$[C_i] \;\models\; \psi \qquad \text{for all} \quad \psi \in C_j \quad \text{if} \quad C_i \sim C_j$$

This is done by induction on $\psi$ similar to the proof of Theorem 104 for LTL. Note that the cases of $\psi = \mathtt{EX}\varphi$ and $\psi = \mathtt{AX}\varphi$ hold because all of player $\forall$'s choices with rule $(\mathtt{AX})$ are contained in player $\exists$'s game tree.

Finally, $\varphi_0 \in C_0$ and, thus, $[C_0] \models \varphi_0$. ∎

A consequence of this proof is the tree model property for CTL. However, it also follows from Lemma 75 which shows the tree model property for CTL$^*$.

**Corollary 117 (Tree model property)** *CTL has the tree model property.*

For two LVRs $l_1$ and $l_2$ the *interleaving* of $l_1$ and $l_2$ is a sequence containing each element of $l_1$ and $l_2$ exactly once such that the order of the elements in $l_1$ and $l_2$ is preserved.

**Lemma 118** *The interleaving of two disjoint LVRs is an LVR.*

PROOF Let $l = C_1, \ldots, C_n$ be the interleaving of $l_1$ and $l_2$ which are LVRs over the disjoint $I_1$ and $I_2$. Then $C_i \in I_1 \cup I_2$ for every $i = 1, \ldots, n$. $C_i \neq C_j$ for all $1 \leq i < j \leq n$

because $I_1 \cap I_2 = \emptyset$. Finally,

$$n \;=\; |l_1| + |l_2| \;\leq\; |I_1| + |I_2| \;=\; |I_1 \cup I_2|$$

Thus, $l$ is an LVR. ■

**Theorem 119 (Winning strategies)**
*a) Player ∃'s winning strategies are history-free.*
*b) Player ∀'s winning strategies are LVR strategies.*

PROOF Player ∃'s winning strategy for $\mathcal{G}(\varphi_0)$ with a satisfiable $\varphi_0$ consists of choosing a model $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ for $\varphi_0$ and playing according to her strategies in the corresponding CTL model checking games $\mathcal{G}_{\mathcal{T}}(s, \psi)$ for $s \in \mathcal{S}$ and $\psi \in Sub(\varphi_0)$. The choice of the model does not depend on the play, and by Theorem 83, her model checking winning strategies are history-free. They add up to a history-free strategy for $\mathcal{G}(\varphi_0)$ since there is no interaction between the model checking games.

Player ∀'s winning strategies are latest visitation record strategies. Note that he uses essentially the same strategy as in the LTL games. The fact that his overall strategy consists of using one priority list for each sequence of applications of rules $(\mathtt{AX})$ and $(\mathtt{EX})$ does not change this. All the lists can be interleaved to one overall list in which the origin of each U formula is used as an annotation to fulfil the requirements of being a LVR.

Note that he simply ignores unimportant parts of the LVR because he always changes focus to present U formulas only. According to Lemma 118, the result is a LVR, too.

The only conceptual difference to the LTL games is the additional choice he has with rule $(\mathtt{AX})$. Choosing an $\mathtt{EX}\varphi_i$ determines the state of his priority list. In terms of the proof of Theorem 114 it determines which list to continue his strategy with. But all he needs to do with rule $(\mathtt{AX})$ to win the remaining game is to preserve unsatisfiability. This choice does not depend on the history of the play.

Thus, his overall strategy consisting of preserving unsatisfiability and maintaining a priority list is a LVR strategy. ■

### Complexity

**Theorem 120 (Complexity)** *Deciding the winner of a CTL satisfiability game is in EXPTIME.*

PROOF Unlike the proofs of Theorems 79 and 106, here player ∀'s moves cannot be determinised using the priority list strategy described in the proof of Theorem 114. The reason is game rule (AX) which requires player ∀ to make a choice other than positioning the focus.

An alternating algorithm only needs to store two configurations and a counter to find a winning play for one of the players. Again, the configurations are the actual one and one chosen by player ∀ on which he tries to find a repeat. The counter is bounded by $|\varphi| \cdot 2^{|\varphi|} + 3$ and, hence, requires space which is polynomial in the size of the input $\varphi$. Therefore, checking whether a CTL formula is satisfiable is in APSPACE which is the same as EXPTIME, [CKS81]. ■

## 6.3 Satisfiability Games for PDL

The set of configurations of the *satisfiability game* $\mathcal{G}(\varphi_0)$ for a PDL formula $\varphi_0$ is

$$\mathcal{C} \;=\; Sub(\varphi_0) \times 2^{Sub(\varphi_0)}$$

Again, $\mathcal{G}(\varphi_0)$ is a focus game like the ones of Sections 6.1 and 6.2 with the difference that the model $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ for $\varphi_0$ which player ∃ implicitly attempts to construct need not be total.

The presentation of the game rules is split into two sets. The first set deals with boolean combinators and modalities with atomic programs. They can be found in Figure 6.4. Rules ($[\vee]$), ($\vee$), ($[\wedge]$) and ($\wedge$) are the usual ones for disjunctions and conjunctions. There also is the focus change rule (FC) that player ∀ can use at any point in a play. Rules ($\langle a \rangle$) and ($[a]$) are parametrised by the action $a$ and are the PDL counterparts to the CTL rules (EX) and (AX). Note that PDL, unlike CTL, distinguishes transitions with different labels. Therefore only those formulas that speak about successor states

$$([\vee])\ \frac{\left[\varphi_0 \vee \varphi_1\right], \Phi}{\left[\varphi_i\right], \Phi}\ \exists i \qquad ([\wedge])\ \frac{\left[\varphi_0 \wedge \varphi_1\right], \Phi}{\left[\varphi_i\right], \varphi_{1-i}, \Phi}\ \forall i \qquad (\mathtt{FC})\ \frac{\left[\varphi\right], \psi, \Phi}{\left[\psi\right], \varphi, \Phi}\ \forall$$

$$(\vee)\ \frac{\left[\psi\right], \varphi_0 \vee \varphi_1, \Phi}{\left[\psi\right], \varphi_i, \Phi}\ \exists i \qquad (\wedge)\ \frac{\left[\psi\right], \varphi_0 \wedge \varphi_1, \Phi}{\left[\psi\right], \varphi_0, \varphi_1, \Phi}$$

$$(\langle a_1 \rangle)\ \frac{\left[\langle a_1 \rangle \varphi_1\right], \ldots, \langle a_n \rangle \varphi_n, [b_1]\psi_1, \ldots, [b_m]\psi_m, q_1, \ldots, q_l}{\left[\varphi_1\right], \Psi}$$

where for all $i = 1, \ldots, m : \psi_i \in \Psi$ iff $b_i = a_1$

$$([a_i])\ \frac{\langle a_1 \rangle \varphi_1, \ldots, \langle a_n \rangle \varphi_n, \left[[b_1]\psi_1\right], \ldots, [b_m]\psi_m, q_1, \ldots, q_l}{\varphi_i, \left[\psi_1\right], \Psi}\ \forall i$$

where $a_i = b_1$ and for all $j = 2, \ldots, m : \psi_j \in \Psi$ iff $b_j = b_1$

Figure 6.4: The PDL satisfiability game rules for formulas.

which can be reached with the same atomic program are included in an application of rule $(\langle a \rangle)$ or $([a])$.

The second set of rules deals with non-atomic programs. Basically, they apply the equivalences given in Section 2.5 to obtain formulas with smaller programs. Rules $([\langle \cup \rangle])$, $(\langle \cup \rangle)$, $([[\cup]])$ and $([\cup])$ have been optimised in the sense that the corresponding equivalences yield a disjunction or a conjunction, and the following choice by one of the players has been built into the rule already. Note that $\overline{\psi}$ in rules $([[?]])$ and $([?])$ denotes the complement of $\psi$ according to Lemma 13 of Section 2.5.

$$([\langle\cup\rangle])\ \frac{\Big[\langle\alpha_0\cup\alpha_1\rangle\varphi\Big],\Phi}{\Big[\langle\alpha_i\rangle\varphi\Big],\Phi}\ \exists i \qquad ([[\cup]])\ \frac{\Big[[\alpha_0\cup\alpha_1]\varphi\Big],\Phi}{\Big[[\alpha_i]\varphi\Big],[\alpha_{1-i}]\varphi,\Phi}\ \forall i$$

$$(\langle\cup\rangle)\ \frac{\Big[\psi\Big],\langle\alpha_0\cup\alpha_1\rangle\varphi,\Phi}{\Big[\psi\Big],\langle\alpha_i\rangle\varphi,\Phi}\ \exists i \qquad ([\cup])\ \frac{\Big[\psi\Big],[\alpha_0\cup\alpha_1]\varphi,\Phi}{\Big[\psi\Big],[\alpha_0]\varphi,[\alpha_1]\varphi,\Phi}$$

$$([\langle;\rangle])\ \frac{\Big[\langle\alpha_0;\alpha_1\rangle\varphi\Big],\Phi}{\Big[\langle\alpha_0\rangle\langle\alpha_1\rangle\varphi\Big],\Phi} \qquad ([\langle?\rangle])\ \frac{\Big[\langle\psi?\rangle\varphi\Big],\Phi}{\psi,\Big[\varphi\Big],\Phi} \qquad (\langle?\rangle)\ \frac{\Big[\chi\Big],\langle\psi?\rangle\varphi,\Phi}{\Big[\chi\Big],\psi,\varphi,\Phi}$$

$$([[;]])\ \frac{\Big[[\alpha_0;\alpha_1]\varphi\Big],\Phi}{\Big[[\alpha_0][\alpha_1]\varphi\Big],\Phi} \qquad ([[?]])\ \frac{\Big[[\psi?]\varphi\Big],\Phi}{\Big[\overline{\psi}\vee\varphi\Big],\Phi} \qquad ([?])\ \frac{\Big[\chi\Big],[\psi?]\varphi,\Phi}{\Big[\chi\Big],\overline{\psi}\vee\varphi,\Phi}$$

$$(\langle;\rangle)\ \frac{\Big[\psi\Big],\langle\alpha_0;\alpha_1\rangle\varphi,\Phi}{\Big[\psi\Big],\langle\alpha_0\rangle\langle\alpha_1\rangle\varphi,\Phi} \qquad ([;])\ \frac{\Big[\psi\Big],[\alpha_0;\alpha_1]\varphi,\Phi}{\Big[\psi\Big],[\alpha_0][\alpha_1]\varphi,\Phi}$$

$$([\langle*\rangle])\ \frac{\Big[\langle\alpha^*\rangle\varphi\Big],\Phi}{\Big[\varphi\Big],\Phi\quad|\quad\Big[\langle\alpha\rangle\langle\alpha^*\rangle\varphi\Big],\Phi}\ \exists \qquad ([[*]])\ \frac{\Big[[\alpha^*]\varphi\Big],\Phi}{\Big[\varphi\wedge[\alpha][\alpha^*]\varphi\Big],\Phi}$$

$$(\langle*\rangle)\ \frac{\Big[\psi\Big],\langle\alpha^*\rangle\varphi,\Phi}{\Big[\psi\Big],\varphi,\Phi\quad|\quad\Big[\psi\Big],\langle\alpha\rangle\langle\alpha^*\rangle\varphi,\Phi}\ \exists \qquad ([*])\ \frac{\Big[\psi\Big],[\alpha^*]\varphi,\Phi}{\Big[\psi\Big],\varphi,[\alpha][\alpha^*]\varphi,\Phi}$$

Figure 6.5: The PDL satisfiability game rules for programs.

**Definition 121** A configuration $C$ of $\mathcal{G}(\varphi_0)$ is called *terminal* if

- $C = \left[[a_1]\psi_1\right], \ldots, [a_n]\psi_n, q_1, \ldots, q_k$, or
- $C = \left[q\right], \Phi$ and player $\forall$ refuses or is unable to move the focus with rule $(\mathtt{FC})$.

The slightly different definition of a terminal configuration compared to those in LTL or CTL games is due to the fact that models of PDL formulas are not required to be total.

The winning conditions for the PDL games are similar to those for the CTL games. Player $\forall$ wins the play $C_0, \ldots, C_n$ iff

1. $C_n = \left[q\right], \Phi$ is terminal, and $q = \mathtt{ff}$ or $\overline{q} \in \Phi$, or
2. $C_n = \left[\langle\alpha^*\rangle\varphi\right], \Phi$ and there is an $i \in \mathbb{N}$, s.t. $i < n$ and $C_i = C_n$, and player $\forall$ has not used rule $(\mathtt{FC})$ between $C_i$ and $C_n$.

Player $\exists$ wins the play $C_0, \ldots, C_n$ iff

3. $C_n$ is terminal, and $\mathtt{ff} \notin C$ and for every $q \in C$: $\overline{q} \notin C$, or
4. $C_n = \left[\varphi\right], \Phi$ and there is an $i \in \mathbb{N}$, s.t. $i < n$ and $C_i = C_n$, and player $\forall$ has used rule $(\mathtt{FC})$ between $C_i$ and $C_n$.
5. $C_n = \left[[\alpha^*]\varphi\right], \Phi$ and there is an $i \in \mathbb{N}$, s.t. $i < n$ and $C_i = C_n$, and player $\forall$ has not used rule $(\mathtt{FC})$ between $C_i$ and $C_n$.

## Correctness

Again, finiteness of every play and uniqueness of their winners are proved in the same way as they are for LTL and CTL, see Lemmas 93, 94 and 108. The same holds for determinacy.

**Fact 122** $(\mathtt{FC})$ *is the only rule that maintains the size of a configuration. Rules* $([\langle * \rangle])$*,* $(\langle * \rangle)$*,* $([[*]])$ *and* $([*])$ *increase the number of connectives in a configuration. All other rules either reduce the number of connectives in a program* $\alpha$ *or the number of boolean connectives and modalities in a formula.*

**Lemma 123** *Every play of* $\mathcal{G}(\varphi)$ *has finite length less than* $|\varphi| \cdot 2^{|\varphi|} + 3$ *and a uniquely determined winner.*

**Corollary 124 (Determinacy)** *Player* $\forall$ *wins* $\mathcal{G}(\varphi)$ *iff player* $\exists$ *does not win* $\mathcal{G}(\varphi)$*.*

**Lemma 125** *Player* $\exists$ *preserves unsatisfiability with her rules. Player* $\forall$ *can preserve unsatisfiability.*

PROOF The cases of the rules for boolean connectives have been dealt with in the LTL or CTL version of this lemma already (Lemmas 96 and 110). All the deterministic rules preserve unsatisfiability because the apply equivalences for PDL formulas.

Note that player $\exists$'s choices are all special instances of configurations containing disjunctions. Player $\forall$'s choice with rule $([[\cup]])$ is an instance of a conjunctive choice. Preservation of unsatisfiability with rule $(\mathtt{FC})$ is trivial.

For the remaining cases of rules $(\langle a_i \rangle)$ and $([a_i])$ suppose that $\varphi_i, \psi_{i,1}, \ldots, \psi_{i,m_i}$ is satisfiable for every $\varphi_i$ with $i \in \{1, \ldots, n\}$. Then each of them has a model $\mathcal{T}_i, s_i$ s.t.

$$\mathcal{T}_i, s_i \models \varphi_i \wedge \psi_{i,1} \wedge \ldots \wedge \psi_{i,m_i} \quad \text{for all } i \in \{1, \ldots, n\}$$

We define $\mathcal{T}'$ as the disjoint union over all $\mathcal{T}_i$ with a new state $s$ and transitions $s \xrightarrow{a_i} s_i$, s.t. $a_i \neq a_j$ if $i \neq j$, and a consistent labelling $L(s) = \{q_1, \ldots, q_l\}$. Then

$$\mathcal{T}', s \models \bigwedge_{i=1}^{l} q_i \wedge \bigwedge_{i=1}^{n} (\langle a_i \rangle \varphi_i \wedge \bigwedge_{j=1}^{m_i} [a_i] \psi_{i,j})$$

The conjuncts can be permuted into the form that is presented in rule $(\langle a_i \rangle)$ or $([a_i])$.

Conversely, if this formula is unsatisfiable then there must be an $i \in \{1, \ldots, n\}$ s.t.

$$\varphi_i, \ \psi_{i,1}, \ \ldots, \ \psi_{i,m_i}$$

is unsatisfiable. This shows that player $\forall$ can preserve unsatisfiability with rules $([a_i])$ and with rule $(\langle a_i \rangle)$ by possibly changing focus accordingly before it is applied. ∎

As in the CTL case, we describe a priority list strategy for player $\forall$. Again, the actual list during a play depends on player $\forall$'s choices with game rule $([a_i])$.

**Definition 126 (Priority list strategy)** Let $l$ be a *priority list* of all subformulas of $\varphi_0$ of the form $\langle\alpha^*\rangle\psi$ for some program $\alpha$ in decreasing order of size, i.e.

$$l \;=\; \langle\alpha_1^*\rangle\psi_1, \ldots, \langle\alpha_n^*\rangle\psi_n$$

with

$$\langle\alpha_i^*\rangle\psi_i \in Sub(\langle\alpha_j^*\rangle\psi_j) \quad \text{and} \quad \langle\alpha_i^*\rangle\psi_i \neq \langle\alpha_j^*\rangle\psi_j \quad \text{implies} \quad j < i$$

In that case $\langle\alpha_j^*\rangle\psi_j$ is said to have higher priority than $\langle\alpha_i^*\rangle\psi_i$. We say that $\langle\alpha^*\rangle\psi$ is present in a configuration $C$ if $\langle\alpha^*\rangle\psi \in C$ or $\langle\alpha\rangle\langle\alpha^*\rangle\psi \in C$.

Player $\forall$ uses the priority list as it is described in Definition 97. He attempts to set the focus to the $\langle\alpha^*\rangle\psi$ with the highest priority that is present or a superformula of it. $\langle\alpha^*\rangle\psi$ gets appended to the end of the list when player $\exists$ chooses $\psi$ in the unfolding instead of $\langle\alpha\rangle\langle\alpha^*\rangle\psi$. At any moment player $\forall$ checks whether he can win by setting the focus to an atomic proposition.

The next lemma is proved in the same way as Lemma 98 for the LTL games. Note that, as in the CTL case, we speak of *the* priority list strategy as his overall strategy that includes the preservation of unsatisfiability according to Lemma 125.

**Lemma 127** ***(Optimality)*** *If player $\forall$ wins $\mathcal{G}(\varphi_0)$ then he wins it with the priority list strategy.*

The *minimal formula* $\chi_P$ causing unsatisfiability in a play $P$ of a PDL satisfiability game is defined just as it is in Definition 99 for LTL. $\chi_P$ is the syntactically least formula that causes unsatisfiability and that occurs earliest in a configuration of $P$.

Here, $\langle\alpha\rangle\varphi$ counts as smaller than $\langle\beta\rangle\psi$ if the number of connectives in $\alpha$ is less than the number of connectives in $\beta$. The same holds for formulas of the form $[\alpha]\varphi$. This is important for applications of rule $([\langle;\rangle])$ for example that replace formulas with others that have the same number of connectives but with a reduced number of connectives inside a modality.

**Lemma 128** *Let $\varphi_0$ be unsatisfiable and $P$ be a play of $\mathcal{G}(\varphi_0)$ in which player $\forall$ uses his priority list strategy. Then $\chi_P$ is either atomic or of the form $\langle\alpha^*\rangle\psi$.*

PROOF This is proved by case analysis on $\chi_P$. Note that the cases of atomic propositions, disjunctions and conjunctions are the same as the ones in the proof of Lemma 101.

Thus, $\chi_P$ can only be of the form $\langle\beta\rangle\varphi$ or $[\beta]\varphi$. The following cases are excluded:

$$\langle\alpha_0 \cup \alpha_1\rangle\varphi, \quad [\alpha_0 \cup \alpha_1]\varphi, \quad \langle\psi?\rangle\varphi \quad \text{and} \quad [\psi?]\varphi$$

They can all be reduced to the case of a disjunction or a conjunction. For $[\psi?]\varphi$ note that $\overline{\psi}$ is of the same size as $\psi$.

If $\beta = a$ for some $a \in \mathcal{A}$ then rule $(\langle a\rangle)$ or $([a])$ will eventually remove the modality from $\varphi$ which is then a better candidate for $\chi_P$. Compare this case also to the case of a $\mathtt{X}\varphi$ in LTL and a $Q\mathtt{X}\varphi$ in CTL.

Next, there are the cases of $\langle\alpha_0;\alpha_1\rangle\varphi$ and $[\alpha_0;\alpha_1]\varphi$. Game rules $([\langle;\rangle])$, $(\langle;\rangle)$, $([[;]])$ and $([;])$ applied to a formula of this form produce a semantically equivalent formula that is smaller by convention since the sequential composition operator is removed. Thus, they cannot be minimal formulas causing unsatisfiability either.

The two remaining cases are $[\alpha^*]\varphi$ and $\langle\alpha^*\rangle\varphi$. The former can be excluded again since it only causes unsatisfiability if the smaller $\varphi$ causes unsatisfiability later on in the play. Note that the unfolding of $[\alpha^*]\varphi$ guarantees $\varphi$ or a subformula of it to be present at any moment in the play.

Finally, suppose $\chi_P = \langle\alpha^*\rangle\varphi$. Thus, there is a configuration $C_k = \chi_P, \Phi_k$ in the play s.t.

$$\models \quad \Phi_k \rightarrow \overline{\langle\alpha^*\rangle\varphi}$$

This means $\models \Phi_k \rightarrow [\alpha^*]\overline{\varphi}$. Remember rule $(\langle *\rangle)$ for the unfolding of $\langle\alpha^*\rangle\varphi$. Player $\exists$ chooses either $\varphi$ or $\langle\alpha\rangle\langle\alpha^*\rangle\varphi$. In the first case, $\varphi$ contradicts the assumption that $\chi_P$ is of the form $\langle\alpha^*\rangle\varphi$ since $\varphi$ is syntactically smaller. This is because

$$[\alpha^*]\overline{\varphi} \quad \equiv \quad \overline{\varphi} \wedge [\alpha][\alpha^*]\overline{\varphi}$$

and, hence,

$$\models \quad \Phi_k \to \overline{\varphi}$$

However, the case where player $\exists$ always chooses $\langle\alpha\rangle\langle\alpha^*\rangle\varphi$ instead does not contradict the assumption since $\langle\alpha\rangle\langle\alpha^*\rangle\varphi$ is not syntactically smaller than $\langle\alpha^*\rangle\varphi$. In this case, $\langle\alpha^*\rangle\varphi$ is the smallest formula causing unsatisfiability. ■

**Theorem 129 (Soundness)** *If $\varphi_0$ is unsatisfiable then player $\forall$ wins $\mathcal{G}(\varphi_0)$.*

PROOF Assume $\varphi_0$ is unsatisfiable. As in the proofs of Theorems 102 and 114, we show that player $\forall$ wins $\mathcal{G}(\varphi_0)$ by using his priority list strategy and preserving unsatisfiability.

Let the two players play a play $C_0, \ldots, C_n$ of $\mathcal{G}(\varphi_0)$. By Lemma 125, each $C_i$ is unsatisfiable, in particular $C_n$. Thus, player $\exists$ cannot win this play with her winning condition 3 since it requires the last configuration of the play to be satisfiable.

Since $\varphi_0$ is assumed to be unsatisfiable, Lemma 128 applies. Regardless of which play $P$ is played, $\chi_P$ is either atomic or of the form $\langle\alpha^*\rangle\varphi$. Let $C_k$ be the earliest configuration containing $\chi_P$ s.t.

$$C_k \;=\; \chi_P, \Phi_k \qquad \text{and} \qquad \models \; \Phi_k \to \overline{\chi_P}$$

If $\chi_P = q$ then the priority list strategy causes player $\forall$ to win the play since he will set the focus to either $q$ or $\overline{q}$ in $C_k$ or at most $\log|\varphi_0|$ steps later.

Suppose $\chi_P$ is of the form $\langle\alpha^*\rangle\varphi$. If player $\forall$ sets the focus to $\chi_P$ when $C_k$ is reached then he wins the resulting play with his winning condition 2. Note that player $\exists$ can never fulfil $\chi_P$ by assumption. Thus, player $\forall$ can leave the focus on it.

Suppose this is not the case, i.e.

$$C_k \;=\; \left[\varphi'\right], \chi_P, \Phi$$

$\varphi'$ is of the form $\langle\alpha^*\rangle\varphi$ as well since player $\forall$'s strategy only allows him to set the focus to a formula other than that if no $\langle\alpha^*\rangle\varphi$ formula is present. But $\chi_P$ is going to remain present since player $\exists$ cannot fulfil it. Moreover, $\chi_P$ is a member of the priority list at this moment.

We can assume $\varphi'$ to get fulfilled at some point. If it does not then player $\forall$ will win with condition 2 just as he does in the preceding case.

The moment it gets fulfilled it is moved to the end of the priority list and player $\forall$ resets the focus to the $\langle\alpha^*\rangle\varphi$ formula which has highest priority and is present. Note that $\chi_P$ is present and that two formulas only swap their priority order if the one with the higher priority gets fulfilled. Therefore, there are only finitely many formulas of the form $\langle\alpha^*\rangle\varphi$ other than $\chi_P$ that the focus can be set to. As soon as one of them persists, player $\forall$ wins with winning condition 2. Eventually, this will be $\chi_P$ unless another one did beforehand.

Note that the argumentation above holds for every play of $\mathcal{G}(\varphi_0)$. Thus, player $\forall$ will win each play either with his winning condition 1 or 2 if he uses his priority list strategy. ∎

Similar to the proofs of Theorems 103 and 115, we will relate the satisfiability games for PDL to its model checking games of Chapter 4 and obtain completeness in this way. Again, one satisfiability play must be related to several model checking plays since configurations of the latter contain single formulas only.

**Theorem 130 (Completeness)** *If $\varphi_0$ is satisfiable then player $\exists$ wins $\mathcal{G}(\varphi_0)$.*

PROOF Suppose $\varphi_0$ is satisfiable. Then it has a model $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with $s_0 \in \mathcal{S}$. By Theorem 52, $\mathcal{T}$ can be assumed to be finite. Player $\exists$'s moves in $\mathcal{G}(\varphi_0)$ will be guided by her winning strategies in the PDL model checking games $\mathcal{G}_\mathcal{T}(s, \psi)$ where $s \in \mathcal{S}$ and $\psi \in Sub(\varphi_0)$. Remember that a PDL model checking game is not a focus game.

The starting positions for both plays are $\left[\varphi_0\right]$ and $s_0 \vdash \varphi_0$. Suppose the actual formula in focus is a disjunction. Then player $\exists$ uses her winning strategy in $\mathcal{G}_\mathcal{T}(s_0, \varphi_0)$ to choose the disjunct that guarantees her to win the remaining play. In the satisfiability game she chooses the same disjunct. Unfolding of temporal operators is deterministically done in the same way in both plays.

The only interesting case is a conjunction in $\mathcal{G}(\varphi_0)$. Consider the first occurrence of such a situation in $\mathcal{G}(\varphi_0)$. At this moment no sideformula can be present. Let therefore

$\left[\psi_0 \wedge \psi_1\right]$ be such a configuration. This must correspond to a position

$$s \;\vdash\; \psi_0 \wedge \psi_1$$

in the model checking play. Since player $\exists$ is assumed to have a winning strategy for this game she must have winning strategies for both $\mathcal{G}_{\mathcal{T}}(s,\psi_0)$ and $\mathcal{G}_{\mathcal{T}}(s,\psi_1)$. In $\mathcal{G}(\varphi_0)$ player $\forall$ sets the focus to one of the conjuncts, say $\psi_0$. Then all choices regarding formulas in focus are matched by choices in $\mathcal{G}_{\mathcal{T}}(s,\psi_0)$, whereas all choices regarding sideformulas correspond to choices in $\mathcal{G}_{\mathcal{T}}(s,\psi_1)$. Thus, at any moment in the satisfiability game a configuration containing $n$ formulas is matched by $n$ model checking plays. Furthermore, the state component of all model checking plays is always the same. Changing focus does not alter the situation on the model checking side at all.

Finally, if the satisfiability play reaches a configuration

$$\left[\langle a_1\rangle\varphi_1\right],\ldots,\langle a_n\rangle\varphi_n,[b_1]\psi_1,\ldots,[b_m]\psi_m,q_1,\ldots,q_l$$

the model checking plays corresponding to $\langle a_i\rangle\varphi_i$ are discarded for all $i = 2,\ldots,n$, as well as those for $[b_j]\psi_j$ for all $j = 1,\ldots,m$ with $b_j \neq a_1$. Player $\exists$ chooses a successor state $t$ of $s$ in the play for $\langle a_1\rangle\varphi_1$. This guarantees that a transition $s \xrightarrow{a_1} t$ exists.

By assumption she has winning strategies for the remaining model checking games $\mathcal{G}_{\mathcal{T}}(t,\varphi_1)$ and $\mathcal{G}_{\mathcal{T}}(t,\psi_j)$ for all $j = 1,\ldots,m$ with $b_j = a_1$. This is because $s \xrightarrow{a_1} t$ and player $\exists$ wins the model checking games $\mathcal{G}_{\mathcal{T}}(s,[b_j]\psi_j)$.

The same argument holds for a configuration with the focus on a $[b]\psi$, the only difference being player $\forall$ who determines the model checking play in which player $\exists$ chooses a successor state.

It is possible for the satisfiability play to perform a repeat on a configuration $\left[\psi_0\right],\Phi$ while the set of model checking plays does not. Let $\Phi = \psi_1,\ldots,\psi_n$. Whenever the model checking plays are at stages $t \vdash \psi_i$, for $i = 1,\ldots,n$, after they were at stages $s \vdash \psi_i$, and $s \neq t$, and the focus is on the same formula, then the satisfiability game is restarted at the first occurrence of $\left[\psi_0\right],\Phi$. This is not done if $s = t$. Since $\mathcal{T}$ is assumed to be finite this iteration process will eventually terminate.

Suppose now that player $\forall$ wins the satisfiability game. If he does with winning condition 1 then there must be two configurations $t \vdash q$ and $t \vdash \overline{q}$ in the set of model checking plays. Player $\exists$ cannot win both of the model checking plays as she is assumed to. Suppose therefore that player $\forall$ wins with condition 2. But a repeat with a $\langle\alpha^*\rangle\varphi$ in focus corresponds to a model checking play that repeats on $\langle\alpha^*\rangle\varphi$ as well and would be won by player $\forall$, too.

We conclude therefore that player $\forall$ cannot win any play of $\mathcal{G}(\varphi_0)$ and by Corollary 109 that player $\exists$ must have a winning strategy for $\mathcal{G}(\varphi_0)$. ∎

**Theorem 131 (Small model property)** *If $\varphi_0 \in PDL$ is satisfiable then it has a model of size less than $|\varphi_0| \cdot 2^{|\varphi_0|}$.*

PROOF Suppose $\varphi_0$ is satisfiable. By Theorem 130, player $\exists$ has a winning strategy for the game $\mathcal{G}(\varphi_0)$. Her game tree is used to build a model $\mathcal{T}$ for $\varphi_0$. Let two configurations that occur in the same play be equivalent if they denote the same state in a model.

$$C_i \sim C_j \text{ iff there is no application of rule } (\langle a\rangle) \text{ or } ([a]) \text{ in between}$$

Again, $[C_i]$ is the equivalence class of $C_i$. States of $\mathcal{T}$ are collapsed configurations under the relation $\sim$. Transitions in $\mathcal{T}$ are defined by

$$[C_i] \xrightarrow{a} [C_k] \quad \text{iff} \quad \begin{array}{l}\text{there is a } j \in \mathbb{N} \text{ s.t. } C_i \sim C_j, C_{j+1} \sim C_k \text{ and} \\ \text{between } C_j \text{ and } C_{j+1} \text{ rule } (\langle a\rangle) \text{ or } ([a]) \text{ has been played}\end{array}$$

The states in $\mathcal{T}$ are labelled as follows.

$$q \in L([C_i]) \text{ iff there is a } j \in \mathbb{N} \text{ s.t. } C_i \sim C_j \text{ and } q \in C_j$$

As in the proofs of Theorems 104 and 116 it is possible to show the following by induction on the structure of $\psi$ for all $i, j < n$:

$$[C_i] \models \psi \qquad \text{for all } \psi \in C_j \text{ if } C_i \sim C_j$$

Again, the fact that $\mathcal{T}$ arises from player $\exists$'s game graph guarantees that this holds for formulas of the form $\langle\alpha\rangle\varphi$ and $[\alpha]\varphi$, and that each $\langle\alpha^*\rangle\varphi$ gets fulfilled in $\mathcal{T}$. Thus, $[C_0] \models \varphi_0$.

Lemma 123 shows that the size of the constructed model is bounded by $|\varphi_0| \cdot 2^{|\varphi_0|}$ since this is the maximal number of different configurations in $\mathcal{G}(\varphi_0)$. ■

**Corollary 132 (Tree model property)** *PDL has the tree model property.*

**Theorem 133 (Winning strategies)**
*a) Player $\exists$'s winning strategies are history-free.*
*b) Player $\forall$'s winning strategies are LVR strategies.*

PROOF This is proved in the same way as Theorem 119 for CTL. Player $\exists$'s winning strategy for $\mathcal{G}(\varphi_0)$ with a satisfiable $\varphi_0$ consists of choosing a model for $\varphi_0$ and playing according to her strategies in the corresponding PDL model checking games $\mathcal{G}_{\mathcal{T}}(s,\psi)$ for $s \in \mathcal{S}$ and $\psi \in Sub(\varphi_0)$. By Theorem 51, her model checking winning strategies are history-free.

Player $\forall$'s winning strategies are latest visitation record strategies since he uses the same strategy as he does in the CTL games. ■

## Complexity

The proofs of soundness and completeness show that the satisfiability problem for PDL is very similar to the satisfiability problem for CTL. This is reflected in their computational complexities as well.

**Theorem 134 (Complexity)** *Deciding the winner of a PDL satisfiability game is in EXPTIME.*

PROOF As in the proof of Theorem 120, player $\forall$'s moves cannot be determinised at no additional cost. The priority list strategy from the proof of Theorem 129 is applicable but leaves choices with rule $([a])$.

As in the CTL case, an alternating algorithm only needs to store two configurations and a counter to find a winning play for one of the players. Again, the configurations are the actual one and one chosen by player $\forall$ on which he tries to find a repeat. The maximal counter value is bounded by

$$|\varphi| \cdot 2^{|\varphi|} \ + \ 3$$

Thus, the counter requires space polynomial in the size of the input $\varphi$. Therefore, checking whether a CTL formula is satisfiable is in APSPACE which is the same as EXPTIME, [CKS81]. ∎

# Chapter 7

# Complete Axiomatisations for LTL, CTL and PDL

*And now for something*
*completely different . . .*

—

MONTY PYTHON

This chapter provides an example of the usefulness of satisfiability games. By using a different technique to prove completeness in Theorems 103, 115 and 130 we can extract complete axiomatisations for LTL, CTL and PDL from the satisfiability games.

In all cases, complete axiomatisations already exist. The axiom systems presented and developed here do not have any advantages over the existing ones as such. It is the game-based approach to satisfiability checking which bears advantages over other approaches because it provides a uniform way of creating complete axiomatisations for different logics.

We will make use of the fact that LTL, CTL and PDL are closed under negation. However, here we prefer the semantical notation $\neg\varphi$ of the negation of $\varphi$.

**Definition 135** An *axiom system* $\mathtt{A}$ for a logic $\mathcal{L}$ is a set of *axioms* and *rules*, s.t. every axiom is of the form $\vdash \varphi$ for a $\varphi \in \mathcal{L}$, and every rule is of the form

$$\text{if} \quad \vdash \varphi \quad \text{then} \quad \vdash \psi$$

In both cases, $\varphi$ and $\psi$ are allowed to contain formula variables. In this case they are interpreted as formula schemes.

An $\mathtt{A}$-proof of a formula $\varphi \in L$ is a finite sequence $\varphi_0, \ldots, \varphi_n$ of formulas of $L$, s.t. $\varphi = \varphi_n$ and for all $i = 0, \ldots, n$:

- $\varphi_i$ is an instance of an axiom in $\mathtt{A}$, or
- there is a $j < i$ s.t. $\varphi_i$ follows from $\varphi_j$ as an instance of a rule in $\mathtt{A}$.

We will write $\vdash_{\mathtt{A}} \varphi$ to indicate that $\varphi$ is provable in $\mathtt{A}$. If the axiom system can be derived from the context we drop the index and simply write $\vdash \varphi$. A formula $\varphi$ whose negation cannot be proved in $\mathtt{A}$, $\not\vdash_{\mathtt{A}} \neg\varphi$, is called $\mathtt{A}$-*consistent* or *consistent* for short.

An axiom system is called *sound* if every provable formula is valid, i.e.

$$\vdash \varphi \quad \text{implies} \quad \models \varphi$$

for every $\varphi \in \mathcal{L}$. It is called *complete* if the converse holds, i.e.

$$\models \varphi \quad \text{implies} \quad \vdash \varphi$$

for every $\varphi \in \mathcal{L}$.

Completeness of an axiomatisation is an important property since it guarantees that every validity of a logic can be captured syntactically. Note that being valid is a semantical property.

Soundness is equally important since an axiomatisation that allows non-valid formulas to be proved is not very useful. Soundness is often very easy to establish. The standard technique is rule induction on the structure of $\mathtt{A}$.

Completeness is usually harder to prove. One possibility is proof by contraposition. If the underlying logic is closed under negation then completeness can be rephrased as

$$\not\vdash \neg\varphi \quad \text{implies} \quad \not\models \neg\varphi$$

This means if every consistent formula is satisfiable then the axiom system is complete.

In the following sections we will give alternative proofs of the completeness of the satisfiability games in the previous chapter. This technique will not make use of a model of a formula. Instead it changes the games slightly to rule out plays in a game tree for player $\exists$ that are won by player $\forall$. In these modified games, player $\forall$ cannot win a single play on a satisfiable formula.

The task is completed by extracting axioms from the game rules and winning conditions such that the rules preserve consistency. Then, player $\forall$ cannot win a play on a consistent formula which, by soundness of the games, means that the formula must be satisfiable. Hence, the axiomatisation is complete.

Finally, the axiom systems need to be proved sound which is very easy in all three cases.

## 7.1 A Complete Axiomatisation for LTL

In this section $\mathcal{G}(\varphi)$ always refers to a satisfiability game for an LTL formula $\varphi$ in the sense of Section 6.1.

**Lemma 136** *If* $\chi \wedge (\varphi \mathtt{U} \psi)$ *is satisfiable then*

$$\chi \wedge (\psi \vee (\varphi \wedge \mathtt{X}((\varphi \wedge \neg\chi)\mathtt{U}(\psi \wedge \neg\chi))))$$

*is satisfiable.*

PROOF Suppose there is a model $\pi$ for $\chi \wedge (\varphi \mathtt{U} \psi)$, i.e. $\pi \models \chi$ and $\pi \models \varphi \mathtt{U} \psi$. Then there is a $k \in \mathbb{N}$ s.t. $\pi^k \models \psi$ and for all $j < k$: $\pi^j \models \varphi$. Suppose furthermore, that

$$\chi \wedge (\psi \vee (\varphi \wedge \mathtt{X}((\varphi \wedge \neg\chi)\mathtt{U}(\psi \wedge \neg\chi))))$$

is not satisfiable. This means

$$\models \quad \chi \rightarrow (\neg\psi \wedge (\neg\varphi \vee \mathtt{X}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))))$$

$k = 0$ is impossible since $\pi \models \chi$ implies $\pi \models \neg\psi$. But if $k > 0$ then $\pi \models \varphi$ and therefore

$$\pi \quad \models \quad \mathtt{X}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))$$

But this means $\pi^1 \models \neg\psi \vee \chi$, and

$$\pi^1 \models \neg\varphi \vee \chi \quad \text{or} \quad \pi^1 \models \mathtt{X}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))$$

If $\pi^1 \models \chi$ then $\pi^1 \models \neg\psi$. But $\pi^1 \models \varphi$ because of $\pi \models \varphi\mathtt{U}\psi$, and therefore

$$\pi^1 \models \mathtt{X}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))$$

by the assumed validity. If $\pi^1 \not\models \chi$ then a contradiction to $\pi \models \varphi\mathtt{U}\psi$ is reached immediately.

This argument can be iterated starting with $\pi^1$ instead of $\pi$ now. At some point, $\pi^k$ must be reached. By assumption $\pi^k \models \psi$, and the iteration yielded $\pi^k \models \chi$. But the latter implies $\pi^k \models \neg\psi$ which contradicts the assumption. We conclude that the validity above cannot hold and that therefore

$$\chi \wedge (\psi \vee (\varphi \wedge \mathtt{X}((\varphi \wedge \neg\chi)\mathtt{U}(\psi \wedge \neg\chi))))$$

must be satisfiable. ■

Now we change the LTL satisfiability games from Section 6.1 slightly. The goal is the following: player $\forall$ should not be able to win a single play on a satisfiable formula anymore. Note that with the original games this is possible, for example if player $\exists$ delays the fulfilling of an $\mathtt{U}$ formula for too long.

We allow player $\exists$ to subscript $\mathtt{U}$ formulas in a very restricted way. Whenever a play of $\mathcal{G}(\varphi_0)$ reaches a configuration $\left[\varphi\mathtt{U}\psi\right], \Phi$ she takes a note of the context $\Phi$ at the $\mathtt{U}$ after it has been unfolded. This means the next configuration will be

$$\left[\psi \vee (\varphi \wedge \mathtt{X}(\varphi\mathtt{U}_\Phi\psi))\right], \Phi$$

Since configurations in the satisfiability games are understood conjunctively we simply write $\neg\Phi$ to denote $\neg\bigwedge_{\varphi\in\Phi}\varphi$. The subscripted formula $\varphi\mathtt{U}_\Phi\psi$ is to be interpreted as

$$(\varphi \wedge \neg\Phi)\mathtt{U}(\psi \wedge \neg\Phi)$$

Note that multiple subscripts are possible, i.e. a subscripted $\mathtt{U}$ formula can be subscripted again.

There are two reasons for using subscripts instead of spelling the formulas out. A play according to the amended game rules should be finished if and only if the corresponding play without subscripts is finished. If the strengthening of an $\mathtt{U}$ formula is spelled out then a repeat on a configuration does not necessarily occur at the same moment anymore. I.e. an occurrence of a configuration $\left[\varphi \mathtt{U}_{\Psi} \psi\right], \Phi$ should count as a repeat if for example the configuration $\left[\varphi \mathtt{U} \psi\right], \Phi$ was visited before.

Moreover, once an $\mathtt{U}$ formula is subscripted with a $\Phi$ for example, it should not be possible anymore to change $\Phi$ through the game rules, i.e. to play on it.

Formally, the amended LTL satisfiability game is obtained from the one of Section 6.1 by replacing rule $([\mathtt{U}])$ with

$$([\mathtt{U}]) \quad \frac{\left[\varphi \mathtt{U}_{\Psi} \psi\right], \Phi}{\left[\psi \vee (\varphi \wedge \mathtt{X}(\varphi \mathtt{U}_{\Psi,\Phi} \psi))\right], \Phi}$$

and by adding the following instance to rule $(\mathtt{FC})$

$$(\mathtt{FC}) \quad \frac{\left[\varphi \mathtt{U}_{\Psi} \psi\right], \chi, \Phi}{\left[\chi\right], \varphi \mathtt{U} \psi, \Phi} \quad \forall$$

The winning conditions are the same except that an $\mathtt{U}$ formula can be arbitrarily subscripted.

**Lemma 137** *Player $\exists$ can preserve satisfiability with the rules of the amended LTL games. Player $\forall$ preserves satisfiability.*

PROOF Player $\forall$ preserves satisfiability since he is only concerned with the position of the focus. Suppose

$$(\varphi_0 \vee \varphi_1) \wedge \Phi$$

is satisfiable, then either $\varphi_0 \wedge \Phi$ or $\varphi_1 \wedge \Phi$ is satisfiable which shows that player $\exists$ can preserve satisfiability by choosing disjuncts accordingly.

Suppose

$$\mathtt{X}\psi_1 \wedge \ldots \wedge \mathtt{X}\psi_n \wedge q_1 \wedge \ldots \wedge q_k$$

is satisfiable, i.e. it has a model $\pi$. Then $\pi^1$ is a model for $\psi_1 \wedge \ldots \wedge \psi_n$ which shows that rule $(\mathtt{X})$ preserves satisfiability, too. So does unfolding of $\mathtt{R}$ formulas and $\mathtt{U}$ formulas that are not in focus.

Unfolding $\mathtt{U}$ formulas in focus and subscripting preserves satisfiability, too, as it is shown in Lemma 136. ■

**Theorem 138 (Completeness II)** *If $\varphi_0$ is satisfiable then player $\exists$ wins $\mathcal{G}(\varphi_0)$.*

PROOF Suppose $\varphi_0$ is satisfiable. According to Lemma 137, player $\exists$ can play in a way such that every reached configuration is satisfiable. Whenever player $\forall$ sets the focus to an $\mathtt{U}$ formula in a configuration

$$\left[\varphi \mathtt{U} \psi\right], \Phi$$

she adds the sideformulas to the index of the $\mathtt{U}$ after it has been unfolded. The indices are dropped if player $\forall$ removes the focus from this $\mathtt{U}$ formula.

By Lemma 137, player $\forall$ cannot win a play with his winning condition 1 since the final configuration of this play would be unsatisfiable. However, if the starting formula is satisfiable then he cannot win a play by a repeat on an $\mathtt{U}$ formula in focus either.

Suppose a play visits a position $\left[\varphi \mathtt{U} \psi\right], \Phi$ twice such that player $\forall$ has not changed focus in between. Then, at the second time this configuration is

$$C \;=\; \left[\varphi \mathtt{U}_{\Phi_1,\ldots,\Phi_k} \psi\right], \Phi$$

where $\Phi_1, \ldots, \Phi_k$ for some $k \in \mathbb{N}$ are all the sets of sideformulas that were present every time $\varphi \mathtt{U} \psi$ was unfolded. Therefore there is a $j \in \{1, \ldots, k\}$ s.t. $\Phi = \Phi_j$. But then $C$ is unsatisfiable since

$$\models \;\; (\varphi \wedge \neg\Phi_1 \wedge \ldots \wedge \neg\Phi_k) \;\mathtt{U}\; (\psi \wedge \neg\Phi_1 \wedge \ldots \wedge \neg\Phi_k) \;\rightarrow\; \neg\Phi_j$$

for all $j = 1, \ldots, k$. But this contradicts the assumption according to Lemma 137. We therefore conclude that player $\exists$ must win $\mathcal{G}(\varphi_0)$. ■

All that remains to be done in order to obtain a complete axiomatisation for LTL is to extract an axiom system from the game rules. This is done rule by rule such that Lemma 137 holds if "satisfiability" is replaced by "consistency".

**Example 139** We will exemplarily do this for rule $(\mathtt{X})$. The goal is the following proposition. If $\varphi_1 \wedge \ldots \wedge \varphi_k$ is inconsistent but $q_1 \wedge \ldots \wedge q_n$ is consistent then

$$\mathtt{X}\varphi_1 \wedge \ldots \wedge \mathtt{X}\varphi_k \wedge q_1 \wedge \ldots \wedge q_n \tag{7.1}$$

is inconsistent. Suppose there is a proof of

$$\vdash \varphi_2 \wedge \ldots \wedge \varphi_k \;\rightarrow\; \neg\varphi_1$$

First of all we need to put an X in front. Therefore we need a rule like (XGen). Then we can prove

$$\vdash \mathtt{X}(\varphi_2 \wedge \ldots \wedge \varphi_k \;\rightarrow\; \neg\varphi_1)$$

With (MP) and the two axioms 4 and 5 we are able to prove

$$\vdash \mathtt{X}\varphi_2 \wedge \ldots \wedge \mathtt{X}\varphi_k \;\rightarrow\; \mathtt{X}\neg\varphi_1$$

By propositional reasoning we can add a consistent set of propositional constants and prove

$$\vdash q_1 \wedge \ldots \wedge q_n \wedge \mathtt{X}\varphi_2 \wedge \ldots \wedge \mathtt{X}\varphi_k \;\rightarrow\; \mathtt{X}\neg\varphi_1$$

Finally, we need an axiom that switches the position of the X and the $\neg$ symbol, namely axiom 3. Then we can prove

$$\vdash q_1 \wedge \ldots \wedge q_n \wedge \mathtt{X}\varphi_2 \wedge \ldots \wedge \mathtt{X}\varphi_k \;\rightarrow\; \neg\mathtt{X}\varphi_1$$

which means we have a proof of the inconsistency of the formula in (7.1).

The axiom system that results if this is done to all the rules is presented in Figure 7.1.

**Lemma 140** *Let* A *be the LTL axiom system of Figure 7.1. The game rules of the amended LTL satisfiability games preserve* A*-consistency.*

PROOF Preservation of consistency by rule $(\wedge)$ is trivial. Suppose $\varphi_0 \vee \varphi_1, \Phi$ is consistent. By axiom 1 and rule (MP), $\varphi_i, \Phi$ is consistent for some $i \in \{0,1\}$. The unfolding of a R or an U formula that is not in focus preserves consistency using axiom 2 and 3.

Preservation of consistency by rule $(\mathtt{X})$ was already shown in Example 139.

Finally, rule (Rel) and axiom 7 are used to capture player $\exists$'s winning strategy and to prove that indexing formulas preserves consistency too. ∎

Axioms:

1. any propositional tautology
2. $\varphi \mathtt{U} \psi \rightarrow \psi \vee (\varphi \wedge \mathtt{X}(\varphi \mathtt{U} \psi))$
3. $\varphi \mathtt{R} \psi \rightarrow \psi \wedge (\varphi \vee \mathtt{X}(\varphi \mathtt{R} \psi))$
4. $\neg \mathtt{X} \varphi \leftrightarrow \mathtt{X} \neg \varphi$
5. $\mathtt{X} \varphi \wedge \mathtt{X} \psi \rightarrow \mathtt{X}(\varphi \wedge \psi)$
6. $\mathtt{X}(\varphi \rightarrow \psi) \rightarrow \mathtt{X} \varphi \rightarrow \mathtt{X} \psi$
7. $\neg(\varphi \mathtt{U} \psi) \leftrightarrow (\neg \varphi) \mathtt{R} (\neg \psi)$

Rules:

(MP) if $\vdash \varphi$ and $\vdash \varphi \rightarrow \psi$ then $\vdash \psi$

(XGen) if $\vdash \varphi$ then $\vdash \mathtt{X} \varphi$

(Rel) if $\vdash \chi \rightarrow \psi \wedge (\varphi \vee \mathtt{X}((\varphi \vee \chi) \mathtt{R} (\psi \vee \chi)))$ then $\vdash \chi \rightarrow \varphi \mathtt{R} \psi$

Figure 7.1: A complete axiomatisation for LTL.

**Theorem 141 (Completeness)** *The LTL axiom system* A *of Figure 7.1 is complete.*

PROOF Suppose $\varphi$ is A-consistent. Player $\exists$ wins the game $\mathcal{G}(\varphi)$ because every winning position for player $\forall$ is A-inconsistent. By Lemma 140, $\varphi$ can only be A-consistent if all winning positions are. By Theorem 102, $\varphi$ is satisfiable. ■

**Theorem 142 (Soundness)** *The LTL axiom system* A *of Figure 7.1 is sound.*

PROOF Validity of axiom 1 is trivial. Validity of the other axioms has been shown in Section 2.4 already. Rule (MP) preserves validity. Suppose $\not\models \mathtt{X}\varphi$. Then $\neg \mathtt{X} \varphi$ has a model $\pi$ s.t. $\pi^1 \models \neg \varphi$. Thus, $\not\models \varphi$ which proves preservation of validity in rule (XGen). Finally, Lemma 136 shows that rule (Rel) preserves validity. ■

Another axiom system DUX for LTL was proposed in [GPSS80]. It is presented in Figure 7.2. Its completeness was shown using maximal consistent sets of formulas.

| | |
|---|---|
| A1. | $\mathtt{ffR}(\varphi \rightarrow \psi) \rightarrow (\mathtt{ffR}\varphi \rightarrow \mathtt{ffR}\psi)$ |
| A2. | $\neg\mathtt{X}\varphi \leftrightarrow \mathtt{X}\neg\varphi$ |
| A3. | $\mathtt{X}(\varphi \rightarrow \psi) \rightarrow \mathtt{X}\varphi \rightarrow \mathtt{X}\psi$ |
| A4. | $\mathtt{ffR}\varphi \rightarrow \varphi \wedge \mathtt{X}(\mathtt{ffR}\varphi)$ |
| A5. | $\mathtt{ffR}(\varphi \wedge \mathtt{X}\varphi) \rightarrow (\varphi \rightarrow \mathtt{ffR}\varphi)$ |
| U1. | $\varphi \mathtt{U} \psi \rightarrow \mathtt{F}\psi$ |
| U2. | $\varphi \mathtt{U} \psi \rightarrow \psi \vee (\varphi \wedge \mathtt{X}(\varphi \mathtt{U} \psi))$ |
| R1. | any propositional tautology |
| R2. | if $\vdash \varphi$ and $\vdash \varphi \rightarrow \psi$ then $\vdash \psi$ |
| R3. | if $\vdash \psi$ then $\vdash \mathtt{ffR}\psi$ |

Figure 7.2: A complete axiomatisation for LTL from [GPSS80].

Soundness of DUX and completeness of A ensure that if $\vdash_{\mathtt{DUX}} \varphi$ then $\vdash_{\mathtt{A}} \varphi$, i.e. every formula that is provable in DUX is also provable in A. This holds in particular for the axioms and rules of DUX. Nevertheless, we will show how they can be derived in A.

**Theorem 143** *For all $\varphi \in LTL$: if $\vdash_{\mathtt{DUX}} \varphi$ then $\vdash_{\mathtt{A}} \varphi$.*

PROOF We show that the DUX axioms are provable in A and that the DUX rules can be simulated in A.

A2,A3,U2,R1 and R1 are present in A. A4 is an instance of axiom 3 and U1 simply reflects our abbreviation of a F formula.

R3 can be simulated as follows. We use induction on the length of a proof in DUX. Suppose there is a proof using R3. Then there is a shorter proof of $\vdash \psi$ in DUX. By hypothesis, $\vdash_{\mathtt{A}} \psi$. Instantiate rule $(\mathtt{Rel})$ with $\chi = \mathtt{tt}$ and $\varphi = \mathtt{ff}$. Then

$$\vdash_{\mathtt{A}} \mathtt{ffR}\psi \quad \text{if} \quad \vdash_{\mathtt{A}} \psi \wedge \mathtt{Xtt}$$

But this is provable using the hypothesis, axiom 1 and rule $(\mathtt{XGen})$.

Axioms A1 and A5 are more complicated to prove in A. We will show that player ∀ wins $\mathcal{G}(\neg\text{A5})$. The negation of axiom A5 is

$$\varphi \wedge (\mathtt{ffR}(\varphi \wedge \mathtt{X}\varphi)) \wedge (\mathtt{ttU}\neg\varphi)$$

Let $\varphi' = \varphi \wedge (\mathtt{ffR}(\varphi \wedge \mathtt{X}\varphi))$. The winning play for player ∀ is

$$\cfrac{\cfrac{\cfrac{\varphi, \mathtt{ffR}(\varphi \wedge \mathtt{X}\varphi), \left[\mathtt{ttU}\neg\varphi\right]}{\varphi, \mathtt{X}\varphi, \mathtt{X}(\mathtt{ffR}(\varphi \wedge \mathtt{X}\varphi)), \left[\neg\varphi \vee \mathtt{X}(\mathtt{ttU}_{\varphi'}\neg\varphi)\right]}}{\varphi, \mathtt{X}\varphi, \mathtt{X}(\mathtt{ffR}(\varphi \wedge \mathtt{X}\varphi)), \left[\mathtt{X}(\mathtt{ttU}_{\varphi'}\neg\varphi)\right]}}{\varphi, \mathtt{ffR}(\varphi \wedge \mathtt{X}\varphi), \left[\mathtt{ttU}_{\varphi'}\neg\varphi\right]}$$

The game rules used for this play are $(\mathtt{R})$, $([\mathtt{U}])$ with indexing, $([\vee])$ and $(\mathtt{X})$. Therefore the axioms and rules needed to prove A5 are 1 and $(\mathtt{MP})$ for $([\vee])$, 2 and 3 for the unfoldings, 4 – 6 and $(\mathtt{XGen})$ for $(\mathtt{X})$, 7 for the negation of A5, and $(\mathtt{Rel})$ to describe the winning condition.

Axiom A1 can be shown to be provable in A in the same way. ■

## 7.2 A Complete Axiomatisation for CTL

In this section $\mathcal{G}(\varphi)$ always refers to a satisfiability game for a CTL formula φ in the sense of Section 6.2.

**Lemma 144**

*a) If* $\chi \wedge \mathtt{E}(\varphi\mathtt{U}\psi)$ *is satisfiable then so is* $\chi \wedge (\psi \vee (\varphi \wedge \mathtt{EXE}((\varphi \wedge \neg\chi)\mathtt{U}(\psi \wedge \neg\chi))))$.

*b) If* $\chi \wedge \mathtt{A}(\varphi\mathtt{U}\psi)$ *is satisfiable then so is* $\chi \wedge (\psi \vee (\varphi \wedge \mathtt{AXA}((\varphi \wedge \neg\chi)\mathtt{U}(\psi \wedge \neg\chi))))$.

PROOF a) Suppose there is a model $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ for $\chi \wedge \mathtt{E}(\varphi\mathtt{U}\psi)$, i.e. there is a state $s \in \mathcal{S}$ s.t. $s \models \chi$ and $s \models \mathtt{E}(\varphi\mathtt{U}\psi)$. Then there is a path $\pi = s_0, s_1, \ldots$ in $\mathcal{T}$ s.t. $s_0 = s$ and for some $k \in \mathbb{N}$: $s_k \models \psi$ and $s_j \models \varphi$ for every $j < k$. Suppose furthermore, that

$$\chi \wedge (\psi \vee (\varphi \wedge \mathtt{EXE}((\varphi \wedge \neg\chi)\mathtt{U}(\psi \wedge \neg\chi))))$$

is not satisfiable, i.e.

$$\models \quad \chi \rightarrow (\neg\psi \wedge (\neg\varphi \vee \mathtt{AXA}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))))$$

$k = 0$ is impossible since $s_0 \models \chi$ implies $s_0 \models \neg\psi$. But if $k > 0$ then $s_0 \models \varphi$ and therefore

$$s_0 \quad \models \quad \mathtt{AXA}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))$$

But this means that $s_1 \models \neg\psi \vee \chi$, and

$$s_1 \quad \models \quad \neg\varphi \vee \chi \qquad \text{or} \qquad s_1 \quad \models \quad \mathtt{AXA}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))$$

If $s_1 \models \chi$ then $s_1 \models \neg\psi$ and $s_1 \models \varphi$ because of $\pi \models \varphi\mathtt{U}\psi$. But then

$$s_1 \quad \models \quad \mathtt{AXA}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))$$

by the assumed validity. If $s_1 \not\models \chi$ then a contradiction to $\pi \models \varphi\mathtt{U}\psi$ is encountered immediately.

This argument can be iterated along $\pi$. At some point, $s_k$ must be reached. By assumption $s_k \models \psi$, and the iteration yields $s_k \models \chi$. But the latter implies $s_k \models \neg\psi$ which contradicts the assumption. We conclude that the validity above cannot hold and that therefore

$$\chi \wedge (\psi \vee (\varphi \wedge \mathtt{EXE}((\varphi \wedge \neg\chi)\mathtt{U}(\psi \wedge \neg\chi))))$$

must be satisfiable.

b) Suppose there is a model $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ for $\chi \wedge \mathtt{A}(\varphi\mathtt{U}\psi)$, i.e. there is a state $s_0 \in \mathcal{S}$ s.t. $s_0 \models \chi$ and $s_0 \models \mathtt{A}(\varphi\mathtt{U}\psi)$. This means $\pi \models \varphi\mathtt{U}\psi$ for every path $\pi$ with $\pi^{(0)} = s_0$. Suppose furthermore, that

$$\chi \wedge (\psi \vee (\varphi \wedge \mathtt{AXA}((\varphi \wedge \neg\chi)\mathtt{U}(\psi \wedge \neg\chi))))$$

is not satisfiable, i.e.

$$\models \quad \chi \rightarrow (\neg\psi \wedge (\neg\varphi \vee \mathtt{EXE}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))))$$

$s_0 \models \neg\psi$ because of $s_0 \models \chi$. Then, $s_0 \models \varphi$ because of $s_0 \models \mathtt{A}(\varphi\mathtt{U}\psi)$. But from the validity above follows

$$s_0 \quad \models \quad \mathtt{EXE}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))$$

I.e. there is a state $s_1$ s.t. $s_0 \to s_1$ and

$$s_1 \models \mathtt{E}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))$$

Then, $s_1 \models \neg\psi \vee \chi$, and

$$s_1 \models \neg\varphi \vee \chi \qquad \text{or} \qquad s_1 \models \mathtt{EXE}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))$$

If $s_1 \not\models \chi$ then $s_1 \models \neg\psi$ and

$$s_1 \models \mathtt{EXE}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))$$

since $s_1 \models \varphi$ is impossible. If $s_1 \models \chi$ then by the assumed validity, $s_1 \models \neg\psi$ and

$$s_1 \models \mathtt{EXE}((\neg\varphi \vee \chi)\mathtt{R}(\neg\psi \vee \chi))$$

holds, too. Now this argument can be iterated with states $s_2, s_3, \ldots$ s.t. $s_i \models \neg\psi$ for all $i \in \mathbb{N}$. But $s_i \to s_{i+1}$ for all $i \in \mathbb{N}$. By limit closure, $\pi := s_0, s_1, s_2, \ldots$ is a path in $\mathcal{T}$ s.t. $\pi \not\models \varphi\mathtt{U}\psi$ which contradicts the assumption. We conclude that the assumed validity cannot be true and that therefore

$$\chi \wedge (\psi \vee (\varphi \wedge \mathtt{AXA}((\varphi \wedge \neg\chi)\mathtt{U}(\psi \wedge \neg\chi))))$$

must be satisfiable. ∎

Now we amend the CTL satisfiability games from Section 6.2. Again, the goal is to disable winning plays for player $\forall$ on a satisfiable input formula.

We allow player $\exists$ to subscript $Q(\varphi\mathtt{U}\psi)$ formulas in the same way as in Section 7.1. Whenever a play of $\mathcal{G}(\varphi_0)$ reaches a configuration $\left[Q(\varphi\mathtt{U}\psi)\right], \Phi$ she takes a note of the context $\Phi$ at the $\mathtt{U}$ after it has been unfolded. This means the next configuration will be

$$\left[\psi \vee (\varphi \wedge Q\mathtt{X}Q(\varphi\mathtt{U}_{\Phi}\psi))\right], \Phi$$

Changing focus discards the collected indices.

**Lemma 145** *Player $\exists$ can preserve satisfiability with the rules of the amended CTL games. Player $\forall$ preserves satisfiability with his choices.*

PROOF Most of this was already proved in Lemma 137 for the amended LTL games. Suppose

$$\mathtt{EX}\varphi_1 \wedge \ldots \wedge \mathtt{EX}\varphi_n \wedge \mathtt{AX}\psi_1 \wedge \ldots \wedge \mathtt{AX}\psi_m \wedge q_1 \wedge \ldots \wedge q_k$$

is satisfiable. According to Corollary 117, it has a tree model $\mathcal{T}$. This must contain subtrees which are models for

$$\varphi_i \wedge \psi_1 \wedge \ldots \wedge \psi_m$$

for each $i = 1, \ldots, n$, which shows that rule $(\mathtt{EX})$ preserves satisfiability as well as rule $(\mathtt{AX})$ regardless of player $\forall$'s choice.

Preservation of satisfiability with the new rule for indexing unfolded $\mathtt{U}$ formulas in CTL is shown in Lemma 144. ■

**Theorem 146 (Completeness II)** *If $\varphi_0$ is satisfiable then player $\exists$ wins $\mathcal{G}(\varphi_0)$.*

PROOF Suppose $\varphi_0$ is satisfiable. According to Lemma 145, player $\exists$ can play in a way such that every reached configuration is satisfiable. Whenever player $\forall$ sets the focus to a $Q(\varphi\mathtt{U}\psi)$ formula in a configuration

$$\Big[Q(\varphi\mathtt{U}\psi)\Big], \Phi$$

she adds the sideformulas to the indices of the $\mathtt{U}$ after it has been unfolded. They are dropped if player $\forall$ removes the focus from this $\mathtt{U}$ formula.

By Lemma 145, player $\forall$ cannot win a play with his winning condition 1 since the final configuration of this play would be unsatisfiable. However, if the starting formula is satisfiable then he cannot win a play by a repeat on a $Q(\varphi\mathtt{U}\psi)$ in focus either.

Suppose a play visits a position

$$\Big[Q(\varphi\mathtt{U}\psi)\Big], \Phi$$

twice such that player $\forall$ has not changed focus in between. Then, at the second time this configuration is

$$C = \left[Q(\varphi \mathtt{U}_{\Phi_1,\ldots,\Phi_k} \psi)\right], \Phi$$

where $\Phi_1, \ldots, \Phi_k$ for some $k \in \mathbb{N}$ are all the sets of sideformulas that were present whenever $Q(\varphi \mathtt{U} \psi)$ has been unfolded. Therefore there is a $j \in \{1, \ldots, k\}$ s.t. $\Phi = \Phi_j$. But then $C$ is unsatisfiable since

$$\models \quad Q((\varphi \wedge \neg\Phi_1 \wedge \ldots \wedge \neg\Phi_k) \; \mathtt{U} \; (\psi \wedge \neg\Phi_1 \wedge \ldots \wedge \neg\Phi_k)) \;\to\; \neg\Phi_j$$

for all $j = 1, \ldots, k$. But this contradicts the assumption according to Lemma 145. We therefore conclude that player $\exists$ must win $\mathcal{G}(\varphi_0)$. ■

To obtain a complete axiomatisation for CTL we need to translate the game rules into an axiom system. Again, the axiom system must be chosen such that Lemma 145 holds if “satisfiability” is replaced by “consistency”. It is presented in Figure 7.3.

**Lemma 147** *Let* A *be the CTL axiom system of Figure 7.3. The game rules of the amended CTL satisfiability games preserve* A*-consistency.*

PROOF Preservation of consistency by rule $(\wedge)$ and $(\vee)$ is the same as in the proof of Lemma 140. The same holds for the rules that unfold $Q(\varphi \mathtt{U} \psi)$ and $Q(\varphi \mathtt{R} \psi)$.

Suppose now that $\varphi_0, \ldots, \varphi_k$ is inconsistent, i.e.

$$\vdash \quad \varphi_1 \wedge \ldots \wedge \varphi_k \;\to\; \neg\varphi_0$$

By rule (AXGen)

$$\vdash \quad \mathtt{AX}(\varphi_1 \wedge \ldots \wedge \varphi_k \;\to\; \neg\varphi_0)$$

Then

$$\vdash \quad \mathtt{AX}\varphi_1 \wedge \ldots \wedge \mathtt{AX}\varphi_k \;\to\; \neg\mathtt{EX}\varphi_0$$

by rule (MP) and axioms 4,6 and 7. This proves preservation of consistency by rules $(\mathtt{AX})$ and $(\mathtt{EX})$. Axiom 5 is used instead of 4 if there are no $\mathtt{EX}\psi$ formulas in the actual configuration.

Finally, rule (Rel) and axioms 8 and 9 are used to capture player $\exists$’s winning strategy and to prove that indexing formulas preserves consistency too. ■

Axioms:

1. any propositional tautology
2. $Q(\varphi \mathtt{U} \psi) \to \psi \vee (\varphi \wedge Q\mathtt{X}Q(\varphi \mathtt{U} \psi))$
3. $Q(\varphi \mathtt{R} \psi) \to \psi \wedge (\varphi \vee Q\mathtt{X}Q(\varphi \mathtt{R} \psi))$
4. $\neg \mathtt{AX}\varphi \leftrightarrow \mathtt{EX}\neg\varphi$
5. $\mathtt{AX}\neg\varphi \to \neg\mathtt{AX}\varphi$
6. $\mathtt{AX}\varphi \wedge \mathtt{AX}\psi \to \mathtt{AX}(\varphi \wedge \psi)$
7. $\mathtt{AX}(\varphi \to \psi) \to \mathtt{AX}\varphi \to \mathtt{AX}\psi$
8. $\neg\mathtt{A}(\varphi \mathtt{U} \psi) \leftrightarrow \mathtt{E}(\neg\varphi)\mathtt{R}(\neg\psi)$
9. $\neg\mathtt{E}(\varphi \mathtt{U} \psi) \leftrightarrow \mathtt{A}(\neg\varphi)\mathtt{R}(\neg\psi)$

Rules:

(MP) if $\vdash \varphi$ and $\vdash \varphi \to \psi$ then $\vdash \psi$

(AXGen) if $\vdash \varphi$ then $\vdash \mathtt{AX}\varphi$

(Rel) if $\vdash \chi \to \psi \wedge (\varphi \vee Q\mathtt{X}Q((\varphi \vee \chi)\mathtt{R}(\psi \vee \chi)))$ then $\vdash \chi \to Q(\varphi \mathtt{R} \psi)$

Figure 7.3: A complete axiomatisation for CTL.

**Theorem 148 (Completeness)** *The CTL axiom system* A *of Figure 7.3 is complete.*

PROOF Suppose $\varphi$ is consistent. Then player $\exists$ wins the game $\mathcal{G}(\varphi)$. This is because all of player $\forall$'s winning positions in $\mathcal{G}(\varphi)$ are A-inconsistent. But according to Lemma 147, $\varphi$ can only be consistent if all winning positions in $\mathcal{G}(\varphi)$ are. By Theorem 114, $\varphi$ is satisfiable in this case. ■

**Theorem 149 (Soundness)** *The CTL axiom system* A *of Figure 7.3 is sound.*

PROOF This is proved in the same way as Theorem 142: the axioms are valid and the rules preserve validity. The only interesting case of the latter part is Lemma 144. ■

| | |
|---|---|
| Ax1. | any propositional tautology |
| Ax2. | $\mathtt{E}(\mathtt{tt}\mathtt{U}\psi) \leftrightarrow \mathtt{EF}\psi$ |
| Ax3. | $\mathtt{A}(\mathtt{tt}\mathtt{U}\psi) \leftrightarrow \mathtt{AF}\psi$ |
| Ax4. | $\mathtt{EX}(\varphi \vee \psi) \leftrightarrow \mathtt{EX}\varphi \vee \mathtt{EX}\psi$ |
| Ax5. | $\mathtt{AX}\varphi \leftrightarrow \neg\mathtt{EX}\neg\varphi$ |
| Ax6. | $\mathtt{E}(\varphi\mathtt{U}\psi) \leftrightarrow \psi \vee (\varphi \wedge \mathtt{EXE}(\varphi\mathtt{U}\psi))$ |
| Ax7. | $\mathtt{A}(\varphi\mathtt{U}\psi) \leftrightarrow \psi \vee (\varphi \wedge \mathtt{AXA}(\varphi\mathtt{U}\psi))$ |
| Ax8. | $\mathtt{EXtt} \wedge \mathtt{AXtt}$ |
| R1. | if $\vdash \varphi \rightarrow \psi$ then $\vdash \mathtt{EX}\varphi \rightarrow \mathtt{EX}\psi$ |
| R2. | if $\vdash \chi \rightarrow \psi \wedge \mathtt{EX}\chi$ then $\vdash \chi \rightarrow \mathtt{E}(\mathtt{ff}\mathtt{R}\psi)$ |
| R3. | if $\vdash \chi \rightarrow \psi \wedge \mathtt{AX}(\chi \vee \mathtt{A}(\varphi\mathtt{R}\chi))$ then $\vdash \chi \rightarrow \mathtt{A}(\varphi\mathtt{R}\psi)$ |
| R4. | if $\vdash \varphi$ and $\vdash \varphi \rightarrow \psi$ then $\vdash \psi$ |

Figure 7.4: A complete axiomatisation for CTL from [EH85].

Another axiom system B for CTL was proposed in [EH85]. It is presented in Figure 7.4. Soundness of B and completeness of A ensure that if $\vdash_{\mathtt{B}} \varphi$ then $\vdash_{\mathtt{A}} \varphi$, i.e. every formula that is provable in B is also provable in A. This holds in particular for the axioms and rules of B.

**Theorem 150** *For all* $\varphi \in$ *CTL: if* $\vdash_{\mathtt{B}} \varphi$ *then* $\vdash_{\mathtt{A}} \varphi$.

PROOF We show that the B axioms are provable in A and that the B rules can be simulated in A.

Axioms Ax1, Ax5, Ax6 and Ax7 as well as rule R4 are present in A. We have introduced Ax2 and Ax3 as abbreviations. An A-proof of Ax8 is the following.

$$\mathtt{tt},\ \mathtt{AXtt},\ \mathtt{AXtt} \rightarrow \neg\mathtt{AXff},\ \neg\mathtt{AXff},\ \neg\mathtt{AXff} \rightarrow \mathtt{EXtt},\ \mathtt{EXtt},\ \mathtt{AXtt} \wedge \mathtt{EXtt}$$

It uses axioms 1, 4 and 5 and rules $(\mathtt{MP})$ and $(\mathtt{AXGen})$. In a similar way, axioms 1 and 6 – 9, and rule $(\mathtt{MP})$ are needed to prove Ax4. R2 is an instance of rule $(\mathtt{Rel})$ with $Q = \mathtt{E}$ and $\varphi = \mathtt{ff}$. R1 is simulated using $(\mathtt{AXGen})$, 9, $(\mathtt{MP})$ and 7.

Finally, R3 is simulated using rule $(\mathtt{Rel})$ with $Q = \mathtt{A}$. By hypothesis there is an A-proof for

$$\vdash \ \chi \to \psi \wedge \mathtt{AX}(\chi \vee \mathtt{A}(\varphi \mathtt{R} \psi))$$

It is used to obtain a proof for

$$\vdash \ \psi \wedge (\varphi \vee \mathtt{AXA}((\varphi \vee \chi)\mathtt{R}(\psi \vee \chi)))$$

using 1, 3 and $(\mathtt{MP})$. Then, $\vdash \chi \to \mathtt{A}(\varphi \mathtt{R} \psi)$ follows with rule $(\mathtt{Rel})$. ■

## 7.3 A Complete Axiomatisation for PDL

Here, $\mathcal{G}(\varphi)$ always refers to a satisfiability game for a PDL formula $\varphi$ in the sense of Section 6.3.

**Lemma 151** *If* $\chi \wedge \langle \alpha^* \rangle \varphi$ *is satisfiable then*

$$\chi \wedge (\varphi \vee \langle \alpha \rangle \langle ((\neg\chi)?;\alpha)^* \rangle (\varphi \wedge \neg\chi))$$

*is satisfiable.*

PROOF Suppose there is a model $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ for $\chi \wedge \langle \alpha^* \rangle \varphi$, i.e. there is a state $s \in \mathcal{S}$ s.t. $s \models \chi$ and $s \models \langle \alpha^* \rangle \varphi$. Then there is a path $\pi = s_0, s_1, \ldots$ in $\mathcal{T}$ s.t. $s_0 = s$ and for some $k \in \mathbb{N}$: $s_k \models \varphi$, $s_k \models \neg\chi$ and for every $j < k$: $s_j \xrightarrow{\alpha} s_{j+1}$. Suppose furthermore, that

$$\chi \wedge (\varphi \vee \langle \alpha \rangle \langle ((\neg\chi)?;\alpha)^* \rangle (\varphi \wedge \neg\chi))$$

is not satisfiable, i.e.

$$\models \ \chi \to (\neg\varphi \wedge [\alpha][((\neg\chi)?;\alpha)^*](\neg\varphi \vee \chi))$$

Thus, $s_0 \models \chi$ implies $s_0 \models \neg\varphi$ and

$$s_0 \ \models \ [\alpha][((\neg\chi)?;\alpha)^*](\neg\varphi \vee \chi))$$

Then,

$$s_1 \ \models \ [((\neg\chi)?;\alpha)^*](\neg\varphi \vee \chi)$$

because $s_0 \xrightarrow{\alpha} s_1$, i.e. $s_1 \models \neg\varphi$ or $s_1 \models \chi$, and

$$s_1 \models [(\neg\chi)?][\alpha][((\neg\chi)?;\alpha)^*](\neg\varphi \vee \chi)$$

This is equivalent to $s_1 \models \chi$ or

$$s_1 \models [\alpha][((\neg\chi)?;\alpha)^*](\neg\varphi \vee \chi)$$

Thus, if $s_1 \not\models \chi$ then $s_1 \models \neg\varphi$ and

$$s_1 \models [\alpha][((\neg\chi)?;\alpha)^*](\neg\varphi \vee \chi)$$

On the other hand, if $s_1 \models \chi$ then, by the assumed validity, $s_1 \models \neg\varphi$ and

$$s_1 \models [\alpha][((\neg\chi)?;\alpha)^*](\neg\varphi \vee \chi)$$

holds, too. Thus, $s_1 \not\models \varphi$ and, in particular,

$$s_2 \models [((\neg\chi)?;\alpha)^*](\neg\varphi \vee \chi)$$

This argument can now be iterated along the path $\pi$ showing that $s_i \not\models \varphi$ for all $i \in \mathbb{N}$. But this contradicts the assumption $s_k \models \varphi$ for some $k \in \mathbb{N}$. We conclude that the assumed validity cannot be true and that therefore

$$\chi \wedge (\varphi \vee \langle\alpha\rangle\langle((\neg\chi)?;\alpha)^*\rangle(\varphi \wedge \neg\chi)$$

must be satisfiable. ■

Again we amend the PDL satisfiability games from Section 6.3. We allow player $\exists$ to take a note of the sideformulas in a configuration $\left[\langle\alpha^*\rangle\varphi\right], \Phi$ after $\langle\alpha^*\rangle\varphi$ has been unfolded to

$$\left[\varphi \vee \langle\alpha\rangle\langle\alpha^*\rangle_\Phi\varphi\right], \Phi$$

In such a case, $\langle\alpha^*\rangle_\Phi\varphi$ will be interpreted as

$$\langle((\neg\Phi)?;\alpha)^*\rangle(\varphi \wedge \neg\Phi)$$

Again, adding new subscripts to already existing ones is allowed. We interpret multiply subscripted formulas $\langle\alpha^*\rangle_{\Phi_1,\ldots,\Phi_n}\varphi$ as

$$\langle((\neg\Phi_1)?;\ldots;(\neg\Phi_n)?;\alpha)^*\rangle(\varphi \wedge \neg\Phi_1 \wedge \ldots \wedge \neg\Phi_n)$$

**Lemma 152** *Player $\exists$ can preserve satisfiability with the rules of the amended PDL games. Player $\forall$ preserves satisfiability with his choices.*

PROOF The cases of rules ($[\wedge]$) and ($[\vee]$) as well as ($\wedge$) and ($\vee$) are proved as in Lemma 137 or 145. The cases of unfolding a $\langle\alpha^*\rangle\varphi$ if it is not in focus or a $[\alpha^*]\varphi$ are trivial. So are all the cases that deal with game rules for programs. This is because the game rules are derived from the PDL equivalences introduced in Section 2.5. In some cases, a following choice of a disjunct is built into the rule already. This does not affect preservation of satisfiability.

Rules ($\langle a\rangle$) and ($[a]$) remain to be analysed. Suppose that a configuration

$$C \;=\; \langle a_1\rangle\varphi_1,\ldots,\langle a_n\rangle\varphi_n,[b_1]\psi_1,\ldots,[b_m]\psi_m,q_1,\ldots,q_l$$

is satisfiable in which the position of the focus does not matter. Then its model $s$ of an LTS $\mathcal{T}$ must have successor states for every $\langle a_i\rangle\varphi_i \in C$. These states must be reachable through a $\xrightarrow{a_i}$ transition and must satisfy $\varphi_i$. Furthermore for every $[a_i]\psi_j$ these states must satisfy $\psi_j$. Note that $a_i = b_j$ for some $j \leq m$ is possible. Therefore, the following configuration $\varphi_i,\psi_{j_1},\ldots,\psi_{j_k}$ will be satisfiable regardless of player $\forall$'s choice with rule ($[a_i]$).

Finally, preservation of satisfiability by the amended unfolding of a $\langle\alpha^*\rangle\varphi$ was proved in Lemma 151 already. ■

**Theorem 153 (Completeness II)** *If $\varphi_0$ is satisfiable then player $\exists$ wins $\mathcal{G}(\varphi_0)$.*

PROOF Suppose $\varphi_0$ is satisfiable. According to Lemma 152, player $\exists$ can play in a way such that every reached configuration is satisfiable. Whenever player $\forall$ sets the focus to a $\langle\alpha^*\rangle\varphi$ formula in a configuration

$$\left[\langle\alpha^*\rangle\varphi\right],\Phi$$

she adds the sideformulas to the index of $\langle\alpha^*\rangle\varphi$ after it has been unfolded. The indices are dropped if player $\forall$ removes the focus from this formula.

By Lemma 152 player $\forall$ cannot win a play with his winning condition 1 since the final configuration of this play would be unsatisfiable. However, if the starting formula is satisfiable then he cannot win a play by a repeat on a $\langle\alpha^*\rangle$ in focus either.

Suppose a play visits a position $\left[\langle\alpha^*\rangle\varphi\right],\Phi$ twice such that player $\forall$ has not changed focus in between. Then, at the second time this configuration is

$$C \;=\; \left[\langle\alpha^*\rangle_{\Phi_1,\ldots,\Phi_k}\varphi\right],\Phi$$

where $\Phi_1,\ldots,\Phi_k$ for some $k \in \mathbb{N}$ are all the sets of sideformulas that were present whenever $\langle\alpha^*\rangle\varphi$ has been unfolded. Therefore there is a $j \in \{1,\ldots,k\}$ s.t. $\Phi = \Phi_j$. But then $C$ is unsatisfiable since

$$\begin{aligned}\langle\alpha^*\rangle_{\Phi_1,\ldots,\Phi_k}\varphi \quad &\equiv \quad (\varphi \wedge \neg\Phi_1 \wedge \ldots \wedge \neg\Phi_k) \;\vee \\ &\quad\;\; (\neg\Phi_1 \wedge \ldots \wedge \neg\Phi_k \wedge \langle\alpha\rangle\langle\alpha^*\rangle_{\Phi_1,\ldots,\Phi_k}\varphi)\end{aligned}$$

Hence,

$$\models \; \langle\alpha^*\rangle_{\Phi_1,\ldots,\Phi_k}\varphi \;\rightarrow\; \neg\Phi$$

which means the final configuration of such a play is not satisfiable. But this contradicts the assumption according to Lemma 145. We therefore conclude that player $\exists$ must win $\mathcal{G}(\varphi_0)$. ■

All that remains to be done in order to obtain a complete axiomatisation for PDL is to translate the game rules into an axiom system. Again, it must be chosen such that Lemma 152 holds if “satisfiability” is replaced by “consistency”. The result is presented in Figure 7.5.

**Lemma 154** *Let* A *be the PDL axiom system of Figure 7.5. The game rules of the amended PDL satisfiability games preserve* A*-consistency.*

PROOF Preservation of consistency by rules ($[\wedge]$), ($[\vee]$), ($\wedge$) and ($\vee$) is the same as in the proofs of Lemmas 140 and 147. Axioms 1,2,6 and rule (MP) are used to prove that an unfolding of a $\langle\alpha^*\rangle\varphi$ which is not in focus and a $[\alpha^*]\varphi$ preserves consistency.

The other rules for PDL programs preserve consistency by axioms 1 and 3 – 6 and rule (MP).

Axioms:

1. any propositional tautology
2. $\neg\langle\alpha\rangle\varphi \leftrightarrow [\alpha]\neg\varphi$
3. $\langle\alpha\cup\beta\rangle\varphi \leftrightarrow \langle\alpha\rangle\varphi \vee \langle\beta\rangle\varphi$
4. $\langle\alpha;\beta\rangle\varphi \leftrightarrow \langle\alpha\rangle\langle\beta\rangle\varphi$
5. $\langle\alpha^*\rangle\varphi \leftrightarrow \varphi \vee \langle\alpha\rangle\langle\alpha^*\rangle\varphi$
6. $\langle\psi?\rangle\varphi \leftrightarrow \psi \wedge \varphi$
7. $[a]\varphi \wedge [a]\psi \rightarrow [a](\varphi \wedge \psi)$
8. $[a](\varphi \rightarrow \psi) \rightarrow [a]\varphi \rightarrow [a]\psi$

Rules:

(MP) if $\vdash \varphi$ and $\vdash \varphi \rightarrow \psi$ then $\vdash \psi$

(Gen) if $\vdash \varphi$ then $\vdash [a]\varphi$ for any $a \in \mathcal{A}$

($[\alpha^*]$) if $\vdash \chi \rightarrow \varphi \wedge [\alpha][((\neg\chi)?;\alpha)^*](\varphi \vee \chi)$ then $\vdash \chi \rightarrow [\alpha^*]\varphi$

Figure 7.5: A complete axiomatisation for PDL.

Suppose now that $\varphi_0, \ldots, \varphi_k$ is inconsistent, i.e.

$$\vdash \quad \varphi_1 \wedge \ldots \wedge \varphi_k \rightarrow \neg\varphi_0$$

By rule (Gen)

$$\vdash \quad [a](\varphi_1 \wedge \ldots \wedge \varphi_k \rightarrow \neg\varphi_0)$$

for any $a \in \mathcal{A}$. Then

$$\vdash \quad [a]\varphi_1 \wedge \ldots \wedge [a]\varphi_k \rightarrow \neg\langle a\rangle\varphi_0$$

by rule (MP) and axioms 2,7 and 8. This proves preservation of consistency by rules ($\langle a\rangle$) and ($[a]$).

Finally, rule (Rel) and axioms 7 and 8 are used to capture player $\exists$'s winning strategy and to prove that indexing formulas preserves consistency too. ∎

| | |
|---|---|
| S1. | any propositional tautology |
| S2. | $\langle\alpha\rangle\varphi \wedge [\alpha]\psi \rightarrow \langle\alpha\rangle(\varphi \wedge \psi)$ |
| S3. | $\langle\alpha\rangle(\varphi \vee \psi) \leftrightarrow \langle\alpha\rangle\varphi \vee \langle\alpha\rangle\psi$ |
| S4. | $\langle\alpha \cup \beta\rangle\varphi \leftrightarrow \langle\alpha\rangle\varphi \vee \langle\beta\rangle\varphi$ |
| S5. | $\langle\alpha;\beta\rangle\varphi \leftrightarrow \langle\alpha\rangle\langle\beta\rangle\varphi$ |
| S6. | $\langle\psi?\rangle\varphi \leftrightarrow \psi \wedge \varphi$ |
| S7. | $\varphi \vee \langle\alpha\rangle\langle\alpha^*\rangle\varphi \rightarrow \langle\alpha^*\rangle\varphi$ |
| S8. | $\langle\alpha^*\rangle\varphi \rightarrow \varphi \vee \langle\alpha^*\rangle(\neg\varphi \wedge \langle\alpha\rangle\varphi)$ |
| R1. | if $\vdash \varphi$ and $\vdash \varphi \rightarrow \psi$ then $\vdash \psi$ |
| R2. | if $\vdash \varphi$ then $\vdash [\alpha]\varphi$ for any $\alpha$ |

Figure 7.6: The Segerberg axiomatisation for PDL.

**Theorem 155 (Completeness)** *The PDL axiom system* A *of Figure 7.5 is complete.*

PROOF Suppose $\varphi$ is consistent. Then player $\exists$ wins the game $\mathcal{G}(\varphi)$. This is because all of player $\forall$'s winning positions in $\mathcal{G}(\varphi)$ are A-inconsistent. But according to Lemma 154, $\varphi$ can only be consistent if all winning positions in $\mathcal{G}(\varphi)$ are. According to Theorem 129, $\varphi$ is satisfiable in this case. ∎

**Theorem 156 (Soundness)** *The PDL axiom system* A *of Figure 7.5 is sound.*

PROOF This is proved in the same way as Theorems 142 and 149: the axioms are valid and the rules preserve validity. The only interesting case of the latter part is Lemma 151. ∎

Another axiom system S for PDL was proposed in [Seg77], usually called the *Segerberg axiom system*. It is presented in Figure 7.6.

Soundness of S and completeness of A ensure that if $\vdash_S \varphi$ then $\vdash_A \varphi$, i.e. every formula that is provable in S is also provable in A. This holds in particular for the axioms and rules of S. Nevertheless, we will show how they can be derived in A.

**Theorem 157** *For all* $\varphi \in PDL$*: if* $\vdash_{S} \varphi$ *then* $\vdash_{\mathtt{A}} \varphi$.

PROOF Axioms S1, and S4 – S7 as well as rule R1 are present in A. S2 and S3 are proved using axioms 1,2,7,8 and rule $(\mathtt{MP})$. Note the difference between the S-rule R2 and $(\mathtt{Gen})$ in A. To prove R2 for arbitrary $\alpha$ with $(\mathtt{Gen})$ for atomic $a \in \mathcal{A}$ only one can use induction on the structure of $\alpha$. The cases $\alpha = \alpha_1;\alpha_2$, $\alpha = \alpha_1 \cup \alpha_2$ and $\alpha = \psi?$ need axioms 1,2, rules $(\mathtt{MP})$ and $(\mathtt{Gen})$ and the corresponding axiom 3,4 or 6. For the case of $\alpha = \beta^*$, rule $([\alpha^*])$ is needed with the instantiation $\chi = \mathtt{tt}$. It reduces to proving

$$\vdash \quad \varphi \wedge [\beta][(\mathtt{ff}?;\beta)^*]\mathtt{tt}$$

under the hypothesis of having a proof for $\vdash \varphi$. But the other conjunct is equivalent to $\mathtt{tt}$ and can be derived in A. Note that by induction hypothesis $\vdash [\beta]\varphi$ if $\vdash \varphi$ since $\beta$ is syntactically smaller than $\alpha$.

To show that A can derive axiom S8 we consider player $\forall$'s strategy for $\mathcal{G}(\neg S8)$. The negation of axiom S8 is

$$\langle\alpha^*\rangle\varphi \wedge \neg\varphi \wedge [\alpha^*](\varphi \vee [\alpha]\neg\varphi)$$

Let $\varphi' := \neg\varphi \wedge [\alpha^*](\varphi \vee [\alpha]\neg\varphi)$. Player $\forall$'s winning play looks like

$$\dfrac{\dfrac{\dfrac{\Big[\langle\alpha^*\rangle\varphi\Big], \neg\varphi, [\alpha^*](\varphi \vee [\alpha]\neg\varphi)}{\Big[\varphi \vee \langle\alpha\rangle\langle\alpha^*\rangle_{\varphi'}\varphi\Big], \neg\varphi, \varphi \vee [\alpha]\neg\varphi, [\alpha][\alpha^*](\varphi \vee [\alpha]\neg\varphi)}}{\Big[\langle\alpha\rangle\langle\alpha^*\rangle_{\varphi'}\varphi\Big], \neg\varphi, [\alpha]\neg\varphi, [\alpha][\alpha^*](\varphi \vee [\alpha]\neg\varphi)}}{\Big[\langle\alpha^*\rangle_{\varphi'}\varphi\Big], \neg\varphi, [\alpha^*](\varphi \vee [\alpha]\neg\varphi)}$$

The game rules used in this play are $([\langle*\rangle])$,$([*])$,$(\wedge)$,$([\vee])$,$(\vee)$ and $(\langle a\rangle)$ depending on the exact structure of $\alpha$. The axioms and rules corresponding to these game rules are listed in the proof of Lemma 154. ∎

# Chapter 8

# Satisfiability Games for CTL$^*$

*This isn't 'Nam. This is*
*bowling. There are rules.*

—

WALTER SOBCHAK

Satisfiability games for CTL$^*$ are played by player $\forall$ and $\exists$ in the same sense as the games for LTL, CTL and PDL of Chapter 6. Note that models of CTL$^*$ formulas are total transition systems $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$.

However, configurations of the CTL$^*$ games are more complicated. The 2-EXPTIME-hardness of the satisfiability checking problem for CTL$^*$ proved in [VS85] suggests that simple sets of subformulas do not suffice. Instead, one has to use sets of sets of formulas.

We will use the following abbreviations: $\Gamma$ and $\Sigma$ are nonempty sets $\{\varphi_0, \ldots, \varphi_n\}$ of formulas that are interpreted conjunctively. $\mathcal{E}$ denotes a possibly empty set $\mathtt{E}\Sigma_1, \ldots, \mathtt{E}\Sigma_n$ of such $\Sigma$s preceded by existential path quantifiers. $\mathcal{A}$ stands for either the empty set or an $\mathtt{A}(\Gamma_1; \ldots; \Gamma_n)$ with $n \geq 1$. We will also use this notation with $n = 0$ to denote the empty set. A semicolon is interpreted as a disjunction. $\Pi$ is a maximally consistent finite set of atomic propositions, i.e. for all $q \in \mathit{Prop}$: $\mathtt{tt} \in \Pi$, and $q \in \Pi$ iff $\overline{q} \notin \Pi$.

To indicate that a $\Gamma$ or $\Sigma$ consists solely of formulas of the form $\mathtt{X}\psi$ we write $\mathtt{X}\Gamma$, resp. $\mathtt{X}\Sigma$. If $\mathtt{X}\Gamma$ or $\mathtt{X}\Sigma$ occurs in a rule then $\Gamma$, resp. $\Sigma$, consists of all $\psi$ s.t. $\mathtt{X}\psi \in \mathtt{X}\Gamma$, resp. $\mathtt{X}\Sigma$.

The basis for a configuration $C$ of the satisfiability game $\mathcal{G}(\varphi_0)$ for a CTL$^*$ formula $\varphi_0$ is a set of sets of formulas and is written

$$\mathtt{A}(\Gamma_1;\ldots;\Gamma_n),\mathtt{E}\Sigma_1,\ldots,\mathtt{E}\Sigma_m,\Pi \tag{8.1}$$

possibly using the abbreviations introduced above. The $\Gamma_i$ are permutable, and so are the $\mathtt{E}\Sigma_i$. For example,

$$\mathtt{A}(\Gamma_1;\Gamma_2),\mathtt{E}\Sigma_1,\mathtt{E}\Sigma_2,\Pi$$

is not distinguished from

$$\mathtt{A}(\Gamma_2;\Gamma_1),\mathtt{E}\Sigma_2,\mathtt{E}\Sigma_1,\Pi$$

As usual, we omit curly set brackets and write $\varphi_1,\ldots,\varphi_n$ instead of $\{\varphi_1,\ldots,\varphi_n\}$ as well as $\Gamma_1;\ldots;\Gamma_m$ instead of $\{\Gamma_1;\ldots;\Gamma_m\}$.

The meaning of a configuration $C$ like the one in (8.1) is: if $C$ is satisfiable then it is satisfied by a state $s$ of a transition system $\mathcal{T}$ s.t. $s$ is labelled with $\Pi$. There are $m$ paths $\pi_1,\ldots,\pi_m$ starting in $s$ s.t. $\pi_i \models \Sigma_i$ for $i=1,\ldots,m$. Furthermore, for all $i=1,\ldots,m$ there is a $j \in \{1,\ldots,n\}$, s.t. $\pi_i \models \Gamma_j$.

Since every configuration of a game will be of this form it can be seen as a *normal form* for CTL$^*$ formulas.

Like the games of Chapter 6, the CTL$^*$ satisfiability games are *focus games*. We omitted the focus in the sample configuration basis (8.1) above because there are several possible positions it can be placed onto. It can either be on a single formula of a conjunction inside the universally path quantified part $\mathcal{A}$,

$$\mathtt{A}(\boxed{\psi},\Gamma_1;\ldots;\Gamma_m),\mathtt{E}\Sigma_1,\ldots,\mathtt{E}\Sigma_n,\Pi$$

or on a single formula inside an existentially path quantified conjunction.

$$\mathtt{A}(\Gamma_1;\ldots;\Gamma_m),\mathtt{E}(\boxed{\psi},\Sigma_1),\ldots,\mathtt{E}\Sigma_n,\Pi$$

Furthermore, it can be placed on the $\mathcal{A}$ part of a configuration

$$\left[\mathtt{A}(\Gamma_1;\ldots;\Gamma_m)\right],\mathtt{E}\Sigma_1,\ldots,\mathtt{E}\Sigma_n,\Pi$$

or on a disjunct inside of it.

$$\mathtt{A}(\left[\Gamma_1\right];\ldots;\Gamma_m),\mathtt{E}\Sigma_1,\ldots,\mathtt{E}\Sigma_n,\Pi$$

It can never be on $\Pi$, one of its elements, or on an entire $\mathtt{E}\Sigma_i$.

For a configuration $C$ we write $\psi \in C$ if

$$C \;=\; \mathtt{A}(\Gamma_1;\ldots;\Gamma_m),\mathtt{E}\Sigma_1,\ldots,\mathtt{E}\Sigma_n,\Pi$$

and $\psi \in \Gamma_i$ for some $i \in \{1,\ldots,m\}$, or $\psi \in \Sigma_i$ for some $i \in \{1,\ldots,n\}$, or $\psi \in \Pi$. The case of a $\left[\psi\right] \in C$ is defined analogously. However, $\left[\psi\right] \in \Pi$ is impossible.

To start a play of $\mathcal{G}(\varphi_0)$ player $\exists$ chooses a maximally consistent set $\Pi$ of propositional constants and the first configuration is

$$\left[\mathtt{A}(\varphi_0)\right],\Pi$$

Note that putting $\varphi_0$ into the universally path quantified part does not impose a restriction on the formulas since $\varphi_0$ is a state formula by definition, and therefore $\varphi_0 \equiv \mathtt{A}\varphi_0$.

To reduce the number of rules we use the $\ulcorner\varphi\urcorner$ construct. A rule containing this should be read as at least two different rules. The first rule is obtained by replacing every $\ulcorner\varphi\urcorner$ with $\left[\varphi\right]$. The other rules result from this rule scheme by imagining any other $\psi$ with $\psi \neq \varphi$ in the upper configuration to be in focus and remain there for the lower configuration. For example,

$$\frac{\mathtt{A}(\ulcorner\varphi_0 \wedge \varphi_1\urcorner,\Gamma;\ldots),\mathcal{E},\Pi}{\mathtt{A}(\ulcorner\varphi_i\urcorner,\varphi_{1-i},\Gamma;\ldots),\mathcal{E},\Pi}\;\forall i$$

abbreviates the following rules.

$$\frac{\mathtt{A}(\left[\varphi_0 \wedge \varphi_1\right],\Gamma;\ldots),\mathcal{E},\Pi}{\mathtt{A}(\left[\varphi_i\right],\varphi_{1-i},\Gamma;\ldots),\mathcal{E},\Pi}\;\forall i \qquad \text{and} \qquad \frac{\mathtt{A}(\varphi_0 \wedge \varphi_1,\Gamma;\Gamma';\ldots),\mathtt{E}\Sigma,\mathcal{E},\Pi}{\mathtt{A}(\varphi_0,\varphi_1,\Gamma;\Gamma';\ldots),\mathtt{E}\Sigma,\mathcal{E},\Pi}$$

$$(\mathtt{A}\wedge)\ \frac{\mathtt{A}(\ulcorner \varphi_0 \wedge \varphi_1 \urcorner, \Gamma; \ldots), \mathcal{E}, \Pi}{\mathtt{A}(\ulcorner \varphi_i \urcorner, \varphi_{1-i}, \Gamma; \ldots), \mathcal{E}, \Pi}\ \forall i \qquad (\mathtt{E}\wedge)\ \frac{\mathcal{A}, \mathtt{E}(\ulcorner \varphi_0 \wedge \varphi_1 \urcorner, \Sigma_1), \mathcal{E}, \Pi}{\mathcal{A}, \mathtt{E}(\ulcorner \varphi_i \urcorner, \varphi_{1-i}, \Sigma_1), \mathcal{E}, \Pi}\ \forall i$$

$$(\mathtt{A}\vee)\ \frac{\mathtt{A}(\ulcorner \varphi_0 \vee \varphi_1 \urcorner, \Gamma; \ldots), \mathcal{E}, \Pi}{\mathtt{A}(\ulcorner \varphi_i \urcorner, \Gamma; \varphi_{1-i}, \Gamma; \ldots), \mathcal{E}, \Pi}\ \exists i \qquad (\mathtt{E}\vee)\ \frac{\mathcal{A}, \mathtt{E}(\ulcorner \varphi_0 \vee \varphi_1 \urcorner, \Sigma), \mathcal{E}, \Pi}{\mathcal{A}, \mathtt{E}(\ulcorner \varphi_i \urcorner, \Sigma), \mathcal{E}, \Pi}\ \exists i$$

Figure 8.1: The CTL$^*$ satisfiability game rules for boolean operators.

with a $[\psi]$ in $\Gamma$, $\Gamma'$ or $\Sigma$, or the focus on the $\mathcal{A}$ part or on a disjunct inside. Note that player $\forall$'s choice becomes obsolete in the second case if $\varphi_0 \wedge \varphi_1$ is not in focus.

The game rules are presented in Figures 8.1 – 8.5. Figure 8.1 contains the rules for boolean connectives. Rules $(\mathtt{A}\wedge)$, $(\mathtt{E}\wedge)$ and $(\mathtt{E}\vee)$ are very similar to those of the satisfiability games in Chapter 6. However, disjunctions inside an $\mathcal{A}$ are preserved. Rule $(\mathtt{A}\vee)$ handles this and transforms the formulas inside $\mathcal{A}$ into disjunctive normal form. The reason for this preservation is the inequivalence

$$\mathtt{A}(\varphi \vee \psi) \quad \not\equiv \quad \mathtt{A}\varphi \vee \mathtt{A}\psi$$

I.e. in order to construct a model for $\mathtt{A}(\varphi \vee \psi)$ it is not possible to discard one of the disjuncts since some paths in the model might satisfy $\varphi$ while others satisfy $\psi$. Moreover, compare this to the model checking games for CTL$^*$ of Chapter 5 where disjuncts are preserved if player $\forall$ is the path player.

Figure 8.2 contains the rules for path quantified formulas. Basically, they are moved outside and merged with an existing $\mathcal{A}$, resp. $\mathcal{E}$, in order to maintain the normal form and obtain a configuration in which all formulas inside these parts are preceded by a $\mathtt{X}$ operator.

Note that there are two rule schemata labelled $(\mathtt{EA})$ for universally path quantified formulas inside an $\mathtt{E}\Sigma$. Since they operate on the same formula in the same position

$$(\mathtt{EA})\quad \frac{\mathtt{A}(\Gamma_1;\ldots;\Gamma_n),\mathtt{E}(\boxed{\mathtt{A}\varphi},\Sigma),\mathcal{E},\Pi}{\mathtt{A}(\varphi,\Gamma_1;\ldots;\boxed{\varphi},\Gamma_i;\ldots;\varphi,\Gamma_n),\mathtt{E}\Sigma,\mathcal{E},\Pi}\ \exists i\quad \text{if } \Sigma\neq\emptyset$$

$$(\mathtt{EA})\quad \frac{\mathtt{A}(\Gamma_1;\ldots;\Gamma_n),\mathtt{E}(\boxed{\mathtt{A}\varphi}),\mathcal{E},\Pi}{\mathtt{A}(\varphi,\Gamma_1;\ldots;\boxed{\varphi},\Gamma_i;\ldots;\varphi,\Gamma_n),\mathcal{E},\Pi}\ \exists i$$

$$(\mathtt{EE})\quad \frac{\mathcal{A},\mathtt{E}(\boxed{\mathtt{E}\varphi},\Sigma),\mathcal{E},\Pi}{\mathcal{A},\mathtt{E}(\boxed{\varphi}),\mathtt{E}\Sigma,\mathcal{E},\Pi}\quad \text{if } \Sigma\neq\emptyset \qquad (\mathtt{EE})\quad \frac{\mathcal{A},\mathtt{E}(\boxed{\mathtt{E}\varphi}),\mathcal{E},\Pi}{\mathcal{A},\mathtt{E}(\boxed{\varphi}),\mathcal{E},\Pi}$$

$$(\mathtt{AE})\quad \frac{\mathtt{A}(\boxed{\mathtt{E}\varphi},\Gamma;\ldots),\mathcal{E},\Pi}{\mathtt{A}(\Gamma;\ldots),\mathtt{E}(\boxed{\varphi}),\mathcal{E},\Pi}\ \exists\quad \text{if } \Gamma\neq\emptyset \qquad (\mathtt{AE})\quad \frac{\mathtt{A}(\boxed{\mathtt{E}\varphi}),\mathcal{E},\Pi}{\mathtt{E}(\boxed{\varphi}),\mathcal{E},\Pi}$$

$$(\mathtt{AE})\quad \frac{\mathtt{A}(\boxed{\mathtt{E}\varphi};\Gamma;\ldots),\mathcal{E},\Pi}{\mathtt{A}(\Gamma;\ldots),\mathtt{E}(\boxed{\varphi}),\mathcal{E},\Pi}\ \exists\quad \text{if } \Gamma\neq\emptyset$$

$$(\mathtt{AA})\quad \frac{\mathtt{A}(\boxed{\mathtt{A}\varphi},\Gamma_1;\ldots;\Gamma_n),\mathcal{E},\Pi}{\mathtt{A}(\boxed{\varphi},\Gamma_1;\ldots;\varphi,\Gamma_n),\mathcal{E},\Pi}\ \exists \qquad (\mathtt{AA})\quad \frac{\mathtt{A}(\boxed{\mathtt{A}\varphi}),\mathcal{E},\Pi}{\mathtt{A}(\boxed{\varphi}),\mathcal{E},\Pi}$$

$$(\not\Gamma)\quad \frac{\mathtt{A}(\Gamma;\Gamma';\ldots),\mathcal{E},\Pi}{\mathtt{A}(\Gamma';\ldots),\mathcal{E},\Pi}\ \exists\quad \text{if no } \left[\psi\right]\in\Gamma,\ \ \Gamma'\neq\emptyset$$

Figure 8.2: The game rules for path quantified formulas.

and only vary in the condition $\Sigma = \emptyset$, resp. $\Sigma \neq \emptyset$, we can regard them as one rule only. Thus, whenever rule $(\mathtt{EA})$ is used it will in fact be one of the two cases. Note that these cases do not result in different configurations in the sense that the action performed on the particular $\mathtt{A}\varphi$ is the same.

Similarly, there are two cases for existentially path quantified formulas at these positions, see rule $(\mathtt{EE})$. These formulas are moved outside into the present $\mathcal{E}$. Depending on $\Sigma$, one of the $\mathtt{E}$ quantifiers might become redundant.

There are three cases for existentially path quantified formulas inside an $\mathcal{A}$ with rule $(\mathtt{AE})$. In the simplest case it is just moved outside and joins the current $\mathcal{E}$. If there are no other formulas in its disjunct then this disjunct disappears. If this is the case and there are no other disjuncts, the entire $\mathcal{A}$ disappears. This reflects the equivalence $\mathtt{AE}\varphi \equiv \mathtt{E}\varphi$.

Finally, if a universally path quantified formula $\mathtt{A}\varphi$ appears inside $\mathcal{A}$ then $\varphi$ gets distributed over all the present disjuncts. Note that this is a choice for player $\exists$. The reason for this is the following. If she believes the disjunct containing $\mathtt{A}\varphi$ to be true then all paths in a possible model for the entire configuration must satisfy $\varphi$ regardless of which other $\Gamma_i$ they satisfy as well. If she believes $\mathtt{A}\varphi$ to be false then she can discard the whole disjunct containing it with rule $(\not\Gamma)$. However, this is only possible if at least one more disjunct is present. Otherwise she could make an unsatisfiable configuration trivially satisfiable.

Again, there is a second case for rule $(\mathtt{AA})$ in which no other formulas are present inside $\mathcal{A}$. According to the equivalence $\mathtt{AA}\varphi \equiv \mathtt{A}\varphi$, the outer path quantifier is simply removed. In this case there is nothing to choose for player $\exists$, neither the position of the focus nor whether to discard or keep the disjunct.

Figure 8.3 lists the rules that deal with atomic propositions occurring anywhere else than in a $\Pi$. Here the basic consensus is: true propositions, i.e. those that occur in the actual $\Pi$, are removed from $\mathcal{A}$ or $\mathcal{E}$ to obtain a configuration in which every formula apart from the propositions in $\Pi$ begins with a $\mathtt{X}$ operator. This is done with all instances of rules $(\mathtt{A}q)$ and $(\mathtt{E}q)$. Note that all rules are deterministic but only applicable if the corresponding condition is met.

$$(\mathtt{A}q)\ \frac{\mathtt{A}(q,\Gamma;\ldots),\mathcal{E},\Pi}{\mathtt{A}(\Gamma;\ldots),\mathcal{E},\Pi}\quad \text{if } q\in\Pi,\ \ \Gamma\neq\emptyset$$

$$(\mathtt{A}q)\ \frac{\mathtt{A}(q;\Gamma;\ldots),\mathcal{E},\Pi}{\mathtt{A}(\Gamma;\ldots),\mathcal{E},\Pi}\quad \text{if } q\in\Pi,\ \ \Gamma\neq\emptyset \qquad (\mathtt{A}q)\ \frac{\mathtt{A}(q),\mathcal{E},\Pi}{\mathcal{E},\Pi}\quad \text{if } q\in\Pi$$

$$(\not q)\ \frac{\mathtt{A}(q,\Gamma;\Gamma';\ldots),\mathcal{E},\Pi}{\mathtt{A}(\Gamma';\ldots),\mathcal{E},\Pi}\quad \text{if } \overline{q}\in\Pi,\ \ \Gamma'\neq\emptyset$$

$$(\mathtt{E}q)\ \frac{\mathcal{A},\mathtt{E}(q,\Sigma),\mathcal{E},\Pi}{\mathcal{A},\mathtt{E}\Sigma,\mathcal{E},\Pi}\quad \text{if } q\in\Pi,\ \ \Sigma\neq\emptyset \qquad (\mathtt{E}q)\ \frac{\mathcal{A},\mathtt{E}(q),\mathcal{E},\Pi}{\mathcal{A},\mathcal{E},\Pi}\quad \text{if } q\in\Pi$$

Figure 8.3: The game rules for propositions.

If there are no other formulas besides the atomic proposition $q$ in its conjunction, resp. in its disjunct and there are no other disjuncts, then the corresponding path quantifier is removed together with the $q$. This reflects the equivalences $\mathtt{A}q \equiv q \equiv \mathtt{E}q$.

False propositions, i.e. those that are not included in the actual $\Pi$, cannot simply be discarded. If they occur inside an $\mathtt{E}\Sigma$ then they witness the fact that this $\mathtt{E}\Sigma$ together with $\Pi$ is unsatisfiable. However, if they occur in a $\Gamma$ inside $\mathcal{A}$ which contains at least one more $\Gamma'$ then $\Gamma$ is unsatisfiable with the current $\Pi$ and can be discarded with rule $(\not\Gamma)$. This does not make an unsatisfiable configuration satisfiable since $\Gamma'$ might be satisfiable together with $\Pi$.

Figure 8.4 shows the rules regarding the temporal operators $\mathtt{U}$ and $\mathtt{R}$. They simply are unfolded regardless of their position. Again, it is easy to extend the set of rules to include $\mathtt{F}$ and $\mathtt{G}$ formulas as primitives.

$$\frac{\mathtt{A}(\ulcorner \mathtt{F}\varphi \urcorner,\Gamma;\ldots),\mathcal{E},\Pi}{\mathtt{A}(\ulcorner \varphi\vee\mathtt{XF}\varphi \urcorner,\Gamma;\ldots),\mathcal{E},\Pi} \qquad \frac{\mathcal{A},\mathtt{E}(\ulcorner \mathtt{F}\varphi \urcorner,\Sigma),\mathcal{E},\Pi}{\mathcal{A},\mathtt{E}(\ulcorner \varphi\vee\mathtt{XF}\varphi \urcorner,\Sigma),\mathcal{E},\Pi}$$

$$(\mathtt{AU})\ \frac{\mathtt{A}(\ulcorner \varphi \mathtt{U} \psi \urcorner, \Gamma; \ldots), \mathcal{E}, \Pi}{\mathtt{A}(\ulcorner \psi \vee (\varphi \wedge \mathtt{X}(\varphi \mathtt{U} \psi)) \urcorner, \Gamma; \ldots), \mathcal{E}, \Pi}$$

$$(\mathtt{EU})\ \frac{\mathcal{A}, \mathtt{E}(\ulcorner \varphi \mathtt{U} \psi \urcorner, \Sigma), \mathcal{E}, \Pi}{\mathcal{A}, \mathtt{E}(\ulcorner \psi \vee (\varphi \wedge \mathtt{X}(\varphi \mathtt{U} \psi)) \urcorner, \Sigma), \mathcal{E}, \Pi}$$

$$(\mathtt{AR})\ \frac{\mathtt{A}(\ulcorner \varphi \mathtt{R} \psi \urcorner, \Gamma; \ldots), \mathcal{E}, \Pi}{\mathtt{A}(\ulcorner \psi \wedge (\varphi \vee \mathtt{X}(\varphi \mathtt{R} \psi)) \urcorner, \Gamma; \ldots), \mathcal{E}, \Pi}$$

$$(\mathtt{ER})\ \frac{\mathcal{A}, \mathtt{E}(\ulcorner \varphi \mathtt{R} \psi \urcorner, \Sigma), \mathcal{E}, \Pi}{\mathcal{A}, \mathtt{E}(\ulcorner \psi \wedge (\varphi \vee \mathtt{X}(\varphi \mathtt{R} \psi)) \urcorner, \Sigma), \mathcal{E}, \Pi}$$

Figure 8.4: The unfolding rules for the CTL* satisfiability games.

$$\frac{\mathtt{A}(\ulcorner \mathtt{G}\varphi \urcorner, \Gamma; \ldots), \mathcal{E}, \Pi}{\mathtt{A}(\ulcorner \varphi \wedge \mathtt{XG}\varphi \urcorner, \Gamma; \ldots), \mathcal{E}, \Pi} \qquad \frac{\mathcal{A}, \mathtt{E}(\ulcorner \mathtt{G}\varphi \urcorner, \Sigma), \mathcal{E}, \Pi}{\mathcal{A}, \mathtt{E}(\ulcorner \varphi \wedge \mathtt{XG}\varphi \urcorner, \Sigma), \mathcal{E}, \Pi}$$

Applying these rules consecutively will eventually result in a configuration in which every formula inside the $\mathcal{A}$ and the $\mathcal{E}$ part is of the form $\mathtt{X}\psi$ unless a false proposition could not be discarded. Recalling the intended meaning of a configuration this situation requires the game to construct successor states of the state at hand. If the focus is inside a particular $\mathtt{E}\Sigma$ then the prospective path satisfying $\Sigma$ must be followed in order not to lose the focus. This is formalised in rule $(\mathtt{EX})$ shown in Figure 8.5. Note that the $\mathcal{A}$ part can also be empty in this case.

If the focus is inside the $\mathcal{A}$ part then every possible path can be examined in the play at hand. Thus, player $\forall$ selects one by choosing a particular $\mathtt{E}\Sigma$ with rule $(\mathtt{AX})$. After

$$(\mathtt{EX}) \quad \frac{\mathtt{A}(\mathtt{X}\Gamma_1;\ldots;\mathtt{X}\Gamma_n),\mathtt{E}(\left[\mathtt{X}\psi\right],\mathtt{X}\Sigma),\mathtt{EX}\Sigma_1,\ldots,\mathtt{EX}\Sigma_m,\Pi'}{\mathtt{A}(\Gamma_1;\ldots;\Gamma_n),\mathtt{E}(\left[\psi\right],\Sigma),\Pi} \; \exists\Pi, \quad n \geq 0, \;\; m \geq 0$$

$$(\mathtt{AX}) \quad \frac{\mathtt{A}(\left[\mathtt{X}\psi\right],\mathtt{X}\Gamma_1;\ldots;\mathtt{X}\Gamma_n),\mathtt{EX}\Sigma_1,\ldots,\mathtt{EX}\Sigma_m,\Pi'}{\mathtt{A}(\left[\psi\right],\Gamma_1;\ldots;\Gamma_n),\mathtt{E}(\psi,\Sigma_i),\Pi} \; \forall i \;\; \exists\Pi, \quad n \geq 1, \;\; m \geq 1$$

$$(\mathtt{AX}\not{\mathtt{E}}) \quad \frac{\mathtt{A}(\left[\mathtt{X}\psi\right],\mathtt{X}\Gamma_1;\ldots;\mathtt{X}\Gamma_n),\Pi'}{\mathtt{A}(\left[\psi\right],\Gamma_1;\ldots;\Gamma_n),\mathtt{E}(\psi),\Pi} \; \exists\Pi, \quad n \geq 1$$

$$(\mathrm{FM}_1) \quad \frac{\left[\mathtt{A}(\Gamma;\ldots)\right],\mathcal{E},\Pi}{\mathtt{A}(\left[\Gamma\right];\ldots),\mathcal{E},\Pi} \; \exists\Gamma \qquad (\mathrm{FM}_2) \quad \frac{\mathtt{A}(\left[\varphi,\Gamma\right];\ldots),\mathcal{E},\Pi}{\mathtt{A}(\left[\varphi\right],\Gamma;\ldots),\mathcal{E},\Pi} \; \forall\varphi$$

$$(\mathrm{FC}_1) \quad \frac{\mathtt{A}(\left[\psi\right],\Gamma;\Gamma';\ldots),\mathcal{E},\Pi}{\mathtt{A}(\psi,\Gamma;\left[\Gamma'\right];\ldots),\mathcal{E},\Pi} \; \exists \qquad (\mathrm{FC}_2) \quad \frac{\mathtt{A}(\left[\varphi\right],\psi,\Gamma;\ldots),\mathcal{E},\Pi}{\mathtt{A}(\varphi,\left[\psi\right],\Gamma;\ldots),\mathcal{E},\Pi} \; \forall$$

$$(\mathrm{FC}_3) \quad \frac{\mathtt{A}(\left[\varphi\right],\Gamma;\ldots),\mathtt{E}(\psi,\Sigma),\mathcal{E},\Pi}{\mathtt{A}(\varphi,\Gamma;\ldots),\mathtt{E}(\left[\psi\right],\Sigma),\mathcal{E},\Pi} \; \forall \qquad (\mathrm{FC}_4) \quad \frac{\mathcal{A},\mathtt{E}(\left[\varphi\right],\psi,\Sigma),\mathcal{E},\Pi}{\mathcal{A},\mathtt{E}(\varphi,\left[\psi\right],\Sigma),\mathcal{E},\Pi} \; \forall$$

$$(\mathrm{FC}_4) \quad \frac{\mathcal{A},\mathtt{E}(\left[\varphi\right],\Sigma),\mathtt{E}(\psi,\Sigma'),\mathcal{E},\Pi}{\mathcal{A},\mathtt{E}(\varphi,\Sigma),\mathtt{E}(\left[\psi\right],\Sigma'),\mathcal{E},\Pi} \; \forall \qquad (\mathrm{FC}_5) \quad \frac{\mathcal{A},\mathtt{E}(\left[\varphi\right],\Sigma),\mathcal{E},\Pi}{\left[\mathcal{A}\right],\mathtt{E}(\varphi,\Sigma),\mathcal{E},\Pi} \; \forall$$

Figure 8.5: The next-step and focus rules for the CTL$^*$ games.

that, player $\exists$ chooses a maximal consistent $\Pi$.

Rule $(\mathtt{AX}\not\exists)$ takes into account a situation without any $\mathtt{E}\Sigma$ formulas. In this case we imagine a single $\mathtt{E}(\mathtt{Xtt})$ to be present. This reflects the requirement of transition systems being total, i.e. every state has at least one successor.

In all cases player $\exists$ chooses a new maximal consistent set $\Pi$ of propositions which will serve as the labelling of the new state. Note that in an application of rule $(\mathtt{AX})$ or $(\mathtt{AX}\not\exists)$ the formula that is currently in focus gets duplicated into the chosen or created $\mathtt{E}\Sigma$. We will illustrate the reason for this with an example later on. A justification for the correctness of this move is the validity

$$\models \mathtt{A}\varphi \wedge \mathtt{E}\psi \rightarrow \mathtt{E}(\varphi \wedge \psi)$$

The remaining rules formalise the changing and positioning of the focus. Remember that every play of $\mathcal{G}(\varphi_0)$ starts with the configuration

$$\left[\mathtt{A}(\varphi_0)\right], \Pi$$

for some $\Pi$. If $\varphi_0$ is a disjunction then player $\exists$ can put the focus onto one of the disjuncts with rule $(\mathrm{FM}_1)$. She will choose the one that she believes is satisfied by the path the play will outline in a possible model. A disjunct $\Gamma$ itself is interpreted conjunctively, thus player $\forall$ puts the focus onto one formula inside $\Gamma$ using rule $(\mathrm{FM}_2)$.

At last, both players have their chances to reset the focus in order to respond to the other player's moves accordingly. Player $\exists$ is allowed to change her mind about which $\Gamma$ inside $\mathcal{A}$ is satisfied by the path that the play at hand forms. This is necessary since the path depends on player $\forall$'s choices with rule $(\mathtt{AX})$. However, in order not to make player $\exists$ too strong she is only allowed to change the focus with rule $(\mathrm{FC}_1)$ after an application of rule $(\mathtt{AX})$ or $(\mathtt{AX}\not\exists)$.

Player $\forall$ must be allowed to change the focus to respond to player $\exists$'s choices of disjuncts inside $\mathcal{E}$ and her focus moves inside $\mathcal{A}$. He can change the focus inside a $\Gamma$ to any other formula using rule $(\mathrm{FC}_2)$. He can also move it out of $\mathcal{A}$ and place it onto any formula inside $\mathcal{E}$ with rule $(\mathrm{FC}_3)$. Without this the duplication of formulas into an $\mathtt{E}\Sigma$ would become meaningless. Finally, he can move it from one $\mathtt{E}\Sigma$ into another with rule $(\mathrm{FC}_4)$ or back onto $\mathcal{A}$ with rule $(\mathrm{FC}_5)$ to let player $\exists$ put it onto a $\Gamma$ again.

Again, note that there are two instances of rule $(\mathrm{FC}_4)$. In both cases player $\forall$ changes focus from a $\varphi$ to a $\psi$ which both occur in the $\mathcal{E}$ part of the actual configuration. There is no need to distinguish the two cases in which $\varphi$ and $\psi$ occur in two different or the same $\mathtt{E}\Sigma$. The important point is the fact that player $\forall$ changes focus at all. Therefore, we list these two cases of a rule under one name.

**Definition 158** A configuration $C$ is called *terminal* if

$$C = \mathtt{A}(\boxed{q}, \Gamma; \ldots), \mathcal{E}, \Pi \qquad \text{or} \qquad C = \mathcal{A}, \mathtt{E}(\boxed{q}, \Sigma), \mathcal{E}, \Pi$$

and both players refuse to or are unable to move the focus.

Player $\forall$ wins the play $C_0, C_1, \ldots, C_n$ iff

1. $C_n = \mathtt{A}(\boxed{q}, \Gamma; \ldots), \mathcal{E}, \Pi$ or $C_n = \mathcal{A}, \mathtt{E}(\boxed{q}, \Sigma), \mathcal{E}, \Pi$, $C_n$ is terminal and $\overline{q} \in \Pi$, or
2. there is an $i < n$ s.t. $C_i = C_n$ and a $\boxed{\varphi \mathtt{U} \psi} \in C_n$ and none of the rules $(\mathrm{FC}_i)$, $i = 1, \ldots, 5$, has been used between $C_i$ and $C_n$.
3. there is an $i < n$ s.t. $C_i = C_n$ and between $C_i$ and $C_n$ player $\exists$ has used rule $(\mathrm{FC}_1)$ and player $\forall$ has not used rule $(\mathrm{FC}_3)$, $(\mathrm{FC}_4)$ or $(\mathrm{FC}_5)$.

Player $\exists$ wins the play $C_0, C_1, \ldots, C_n$ iff

4. $C_n = \mathtt{A}(\boxed{q}, \Gamma; \ldots), \mathcal{E}, \Pi$ or $C_n = \mathcal{A}, \mathtt{E}(\boxed{q}, \Sigma), \mathcal{E}, \Pi$, $C_n$ is terminal and $q \in \Pi$, or
5. there is an $i < n$ s.t. $C_i = C_n$ and a $\boxed{\varphi \mathtt{R} \psi} \in C_n$ and none of the rules $(\mathrm{FC}_i)$, $i = 1, \ldots, 5$, has been used between $C_i$ and $C_n$.
6. there is an $i < n$ s.t. $C_i = C_n$ and either
   - player $\exists$ has not used rule $(\mathrm{FC}_1)$ but player $\forall$ has used one of the rules $(\mathrm{FC}_2), \ldots, (\mathrm{FC}_5)$, or
   - player $\exists$ has used rule $(\mathrm{FC}_1)$ and player $\forall$ has used rule $(\mathrm{FC}_3)$, $(\mathrm{FC}_4)$ or $(\mathrm{FC}_5)$.

Winning conditions 1 and 4 are straightforward and similar to the winning conditions for the LTL, CTL and PDL games concerning terminal configurations.

Again, if the focus stays on an $\mathtt{U}$ formula until a repeat is found player $\forall$ should win since he managed to show that this particular $\mathtt{U}$ formula regenerates itself and, hence, that player $\exists$ did not fulfil it. Conversely, if the formula in focus is a $\mathtt{R}$ and the focus has not been changed then player $\exists$ is be the winner.

A player should also win a play in which they did not use their focus change rules whereas their opponent did. This is formalised in the first part of winning condition 6 and, to some extent, in condition 3. Player $\forall$ should win if he uses rule $(\mathrm{FC}_2)$ as long as player $\exists$ uses $(\mathrm{FC}_1)$. If the $\Gamma$ in which player $\forall$ changed focus was not false at this moment then player $\exists$ could have left the focus there in order to win with her winning condition 6.

The motivation for the second part of winning condition 6 is the following. Rules $(\mathrm{FC}_3)$ and $(\mathrm{FC}_5)$ can only occur in conjunction with each other between repeating configurations since they switch the focus between the $\mathcal{A}$ and $\mathcal{E}$ parts of a configuration. If rule $(\mathrm{FC}_4)$ was played but neither $(\mathrm{FC}_3)$ nor $(\mathrm{FC}_5)$ then player $\exists$ could not have used her focus change rule. Suppose therefore, she has and player $\forall$ has changed focus with both rules $(\mathrm{FC}_3)$ and $(\mathrm{FC}_5)$.

He has either done so after player $\exists$ changed focus forth and back between $\mathcal{A}$ and $\mathcal{E}$ or beforehand. If it was afterwards he was reluctant to show that the $\Gamma$ which player $\exists$ has put the focus to does not get satisfied during the play. If it was beforehand he refused to show unsatisfaction of another $\Gamma'$ and player $\exists$'s focus change can be seen as a response to that. Thus, in both cases she should be the winner of the underlying play.

Remember that rules $(\mathtt{AX})$ and $(\mathtt{AX}\not{E})$ copy formulas from the $\mathcal{A}$ part of a configuration into an $\mathtt{E}\Sigma$. Without this player $\forall$ would be too weak. As in the model checking games for CTL$^*$ of Section 5.2 and the satisfiability games for LTL and CTL of Chapter 6, he uses the focus to follow the unfolding of $\mathtt{U}$ formulas. But, since disjuncts inside $\mathcal{A}$ are preserved, there is never a need for player $\exists$ to fulfil a $\varphi\mathtt{U}\psi$ formula. Instead, she can always set the focus to $\psi$ and redo her choice to $\varphi \wedge \mathtt{X}(\varphi\mathtt{U}\psi)$ whenever $\psi$ does not guarantee her to win. However, if $\varphi\mathtt{U}\psi$ gets duplicated into an $\mathtt{E}\Sigma$ then player $\forall$ has the chance to follow the regeneration there.

$$\left[\mathtt{A}(\mathtt{AFG}q \wedge \mathtt{EGF}\overline{q})\right], q$$

$$\mathtt{A}(\left[\mathtt{AFG}q\right], \mathtt{EGF}\overline{q}), q$$

$$\mathtt{A}(\left[\mathtt{FG}q\right]), \mathtt{E}(\mathtt{GF}\overline{q}), q$$

$$\mathtt{A}(\left[\mathtt{G}q \vee \mathtt{XFG}q\right]), \mathtt{E}(\mathtt{F}\overline{q}, \mathtt{XGF}\overline{q}), q$$

$$\mathtt{A}(\left[q \wedge \mathtt{XG}q\right]; \mathtt{XFG}q), \mathtt{E}(\overline{q} \vee \mathtt{XF}\overline{q}, \mathtt{XGF}\overline{q}), q$$

$$\mathtt{A}(q, \left[\mathtt{XG}q\right]; \mathtt{XFG}q), \mathtt{E}(\mathtt{XF}\overline{q}, \mathtt{XGF}\overline{q}), q$$

$$\mathtt{A}(\left[\mathtt{XG}q\right]; \mathtt{XFG}q), \mathtt{E}(\mathtt{XF}\overline{q}, \mathtt{XGF}\overline{q}), q$$

$$\mathtt{A}(\left[\mathtt{G}q\right]; \mathtt{FG}q), \mathtt{E}(\mathtt{F}\overline{q}, \mathtt{GF}\overline{q}, \mathtt{G}q), q$$

$$\mathtt{A}(\mathtt{G}q; \mathtt{FG}q), \mathtt{E}(\left[\mathtt{F}\overline{q}\right], \mathtt{GF}\overline{q}, \mathtt{G}q), q$$

$$\mathtt{A}(q, \mathtt{XG}q; \mathtt{G}q; \mathtt{XFG}q), \mathtt{E}(\left[\overline{q} \vee \mathtt{XF}\overline{q}\right], \mathtt{F}\overline{q}, \mathtt{XGF}\overline{q}, q, \mathtt{XG}q), q$$

$$\mathtt{A}(\mathtt{XG}q; \mathtt{XFG}q), \mathtt{E}(\left[\mathtt{XF}\overline{q}\right], \mathtt{XGF}\overline{q}, \mathtt{XG}q), q$$

$$\mathtt{A}(\mathtt{G}q; \mathtt{FG}q), \mathtt{E}(\left[\mathtt{F}\overline{q}\right], \mathtt{GF}\overline{q}, \mathtt{G}q), q$$

Figure 8.6: Player ∀'s winning play of Example 159.

**Example 159** Let

$$\varphi \;:=\; \mathtt{AGF}q \wedge \mathtt{EFG}\overline{q}$$

φ is unsatisfiable since

$$\mathtt{AGF}q \;\equiv\; \overline{\overline{\mathtt{EFG}\overline{q}}}$$

Player ∀'s winning play is depicted in Figure 8.6. Not every move is shown explicitly. If for example a G is unfolded we implicitly flatten the created conjunction. We also omit configurations if two rules operate on different formulas and present the application as one rule.

Assume player ∃ has chosen $q$ as the propositional part to start with. We also omit tt there since it is trivially part of the Π in any configuration of every play and game.

First the formula at hand is brought into the correct form for the game. The temporal operators are unfolded and a $\wedge$ is removed. Then, player $\exists$ has the choice whether to put the focus onto $\mathtt{G}q$ or $\mathtt{XFG}q$. She chooses the former because otherwise player $\forall$ could exhibit a repeat on the F inside the latter.

Since $q$ is present outside, player $\forall$ can only set the focus to $\mathtt{XG}q$. For the same reason player $\exists$ cannot choose to fulfil the $\mathtt{F}\overline{q}$ in the $\mathcal{E}$ part.

Note that, if player $\forall$ had started with the focus in $\mathtt{E}(\mathtt{FG}\overline{q})$ then player $\exists$ would have chosen $\overline{q}$ for the propositional part at any time. She also would have fulfilled the F formula immediately. The same holds for the case where player $\forall$ changes focus into $\mathcal{E}$ at some point. In any way, the F that is not in the part containing the focus would not get fulfilled and player $\forall$ would be unable to show this.

But now the $\mathtt{G}q$ in focus gets copied into the $\mathcal{E}$ and player $\forall$ can change the focus to the F inside of it. From now on, player $\exists$ must always choose $q$ for the propositional part. Otherwise, player $\forall$ could simply change focus to the $q$ that results from the unfolding of $\mathtt{G}q$ in $\mathcal{E}$, and win with his winning condition 1.

But then she cannot fulfil the $\mathtt{F}\overline{q}$ in $\mathcal{E}$ and player $\forall$ can keep the focus on it until a repeat occurs and win with condition 2 since he does not change the focus between the repeating configurations.

**Example 160** Now, take the similar formula

$$\varphi \;:=\; \mathtt{AGEF}q \wedge \mathtt{EFG}\overline{q}$$

This time, $\varphi$ is satisfiable. The simplest possible model $\mathcal{T}$ for $\varphi$ consists of two states $s$ and $t$ with $L(s) = \{\overline{q}\}$ and $L(t) = \{q\}$. The transitions of $\mathcal{T}$ are $s \to s$, $s \to t$ and $t \to t$. $s \models \varphi$ because every path is either an infinite loop through $s$ or eventually becomes an infinite loop through $t$. Thus, there is a path $\pi$ on which $\overline{q}$ holds infinitely often, namely $\pi = ss\ldots$. On the other hand, every reachable state is the origin of a path on which $q$ holds eventually.

A simplified version of the game tree for player $\exists$ is given in Figure 8.7. Not every application of a rule is listed explicitly in order to keep the size of the tree small. Also, we omit to put the focus into the configurations. Instead we discuss what the players

$$\mathtt{A}(\mathtt{GEF}q), \mathtt{E}(\mathtt{FG}\overline{q}), \overline{q}$$
$$\overline{\mathtt{A}(\mathtt{EF}q, \mathtt{XGEF}q), \mathtt{E}(\mathtt{G}\overline{q}), \overline{q}}$$
$$\overline{\mathtt{A}(\mathtt{XGEF}q), \mathtt{E}(\mathtt{XG}\overline{q}), \mathtt{E}(\mathtt{XF}q), \overline{q}}$$
$$\overline{\mathtt{A}(\mathtt{GEF}q), \mathtt{E}(\mathtt{G}\overline{q}, \mathtt{GEF}q), \overline{q} \qquad \mathtt{A}(\mathtt{GEF}q), \mathtt{E}(\mathtt{F}q), q}$$
$$\overline{\mathtt{A}(\mathtt{XGEF}q), \mathtt{E}(\mathtt{XG}\overline{q}, \mathtt{XGEF}q), \mathtt{E}(\mathtt{F}q), \overline{q}} \qquad \overline{\mathtt{A}(\mathtt{XGEF}q), q}$$
$$\vdots \qquad\qquad\qquad \vdots$$

Figure 8.7: A simplified version of the game tree for Example 160.

can do about its position.

Player $\exists$ chooses $\overline{q}$ at the beginning. This gives her the chance to fulfil the $\mathtt{FG}\overline{q}$ and get rid of $\mathtt{XFG}\overline{q}$ in its unfolding. In the next step the $\mathtt{G}\overline{q}$ gets unfolded and, since $\overline{q}$ is present outside, only $\mathtt{XG}\overline{q}$ remains. Inside $\mathcal{A}$, the $\mathtt{EF}q$ gets promoted to the outside since there is only one disjunct, leaving $\mathtt{XGEF}q$. Player $\forall$ does not set the focus to $\mathtt{G}\overline{q}$ since player $\forall$ will always choose $\overline{q}$ and he cannot win with the repeat on $\mathtt{G}\overline{q}$. Thus, he has to set it to $\mathtt{XGEF}q$ at this point and choose one of the $\mathtt{E}(\ldots)$. If he selects $\mathtt{E}(\mathtt{XG}\overline{q})$ then $\mathtt{GEF}q$ gets copied into it, the $\mathtt{G}$ formulas get unfolded, $\mathtt{EF}q$ is put outside, etc. The resulting configuration is almost the same as the one of the third row. The difference only lies in the $\mathtt{E}(\ldots)$. Since player $\forall$ has not won so far he could only by putting the focus into $\mathtt{E}(\mathtt{XG}\overline{q}, \mathtt{XGEF}q)$. However, the play proceeds in a similar way without ever giving player $\forall$ a chance to win.

The path on the right corresponds to player $\forall$'s choice of $\mathtt{E}(\mathtt{XF}q)$. In this case player $\exists$ selects $q$ as the next propositional part and fulfils the $\mathtt{F}q$ which disappears. Player $\forall$ must keep the focus on $\mathtt{XGEF}q$. He cannot win on this formula anymore since player $\exists$ can always choose $q$ as the next proposition. The fact that in the next step another $\mathtt{E}(\mathtt{GEF}q)$ is created does not change anything about this, since the formulas inside $\mathtt{E}(\ldots)$ will always be present inside $\mathcal{A}$ as well.

## Correctness

**Fact 161** *Rules* $(\mathtt{A}\wedge)$, $(\mathtt{E}\wedge)$, $(\mathtt{A}\vee)$, $(\mathtt{E}\vee)$, $(\mathtt{EA})$, $(\mathtt{AA})$ *and* $(\mathtt{EX})$ *reduce the number of connectives in the actual configuration. Rules* $(\mathtt{E}q)$, $(\mathtt{A}q)$, $(\not{q})$ *and* $(\not{\Gamma})$ *reduce the size of the actual configuration. Rules* $(\mathtt{AE})$ *and* $(\mathtt{EE})$ *reduce the size of the actual* $\mathcal{A}$ *or a* $\Sigma$ *in the actual configuration.*

*Rules* $(\mathtt{AX})$ *and* $(\mathtt{AX}\not{\mathtt{E}})$ *can potentially increase the size of the actual configuration, but they reduce the size of its* $\mathcal{A}$ *part.*

*Rules* $(\mathtt{AU})$, $(\mathtt{AR})$, $(\mathtt{EU})$, $(\mathtt{ER})$ *increase the size and the number of connectives of the actual configuration.*

*Rules* $(\mathrm{FM}_1)$, $(\mathrm{FM}_2)$ *and* $(\mathrm{FC}_i)$, $i = 1,\ldots,5$, *preserve both size and number of connectives in a configuration. Rule* $(\mathrm{FC}_5)$ *puts an* $\mathcal{A}$ *into focus, while rules* $(\mathrm{FM}_1)$ *and* $(\mathrm{FM}_2)$ *reverse this process by reducing the object in focus to a* $\Gamma$, *resp. a single formula again.*

**Lemma 162** *Every play has a uniquely determined winner.*

PROOF Suppose a play does not visit a configuration twice. It can only be finished in a terminal configuration. But then the focus must be on a $q$ which is either inside $\mathcal{A}$ or $\mathcal{E}$. Furthermore, $q$ is either present in the actual $\Pi$ or not. These four cases are covered by the mutually exclusive winning conditions 1 and 4.

Now consider a play with a repeating configuration. There are two possibilities for such a repeat. Suppose the focus was not changed in between. This is only possible if a $\varphi\mathtt{U}\psi$ or a $\varphi\mathtt{R}\psi$ stayed in focus since a combination of rules that reduce and increase the number of connectives in a configuration must have been played. But reducing the number of connectives leads to a situation in which all formulas inside the $\mathcal{A}$ part and every $\mathtt{E}\Sigma$ are preceded by a $\mathtt{X}$ operator. Then, rule $(\mathtt{EX})$, $(\mathtt{AX})$ or $(\mathtt{AX}\not{\mathtt{E}})$ must apply, i.e. the formula in focus gets reduced as well. Only one of the unfolding rules can restore the original formula in focus eventually which means it is a $\varphi\mathtt{U}\psi$ or $\varphi\mathtt{R}\psi$.

In the first case, player $\forall$ wins with his winning condition 2. In the second case player $\exists$ wins the play at hand with her winning condition 5.

Suppose now that the focus was changed between the occurrences of a repeating configuration. Consequently, one of the players must have used one of their focus change rules. If player $\exists$ did not use hers she wins with the first part of condition 6. If she used hers but player $\forall$ did not use any of his then he wins with condition 3. If both have changed focus then it depends on which rules player $\forall$ has used. He still wins with the second part of condition 3 if it was only $(\mathrm{FC}_2)$. If it was one of the others then player $\exists$ wins with the second part of condition 6.

This shows that the winning conditions cover all possible situations and that the winner of a play is uniquely determined. ■

**Lemma 163** *Every play of $\mathcal{G}(\varphi_0)$ is of length less than $O(2^{|\varphi_0|} \cdot 2^{(2^{|\varphi_0|})})$.*

PROOF There are $2^{2^{|\varphi_0|}}$ possible sets of sets of subformulas of $\varphi_0$. Furthermore, the focus can be on any formula which can be in any set, or on $\mathcal{A}$ or any $\Gamma$ inside $\mathcal{A}$. Thus, there are at most

$$(1+|\varphi_0|+2^{|\varphi_0|}) \cdot 2^{2^{|\varphi_0|}}$$

possible different configurations in $\mathcal{G}(\varphi_0)$ and every play of length $n$ or more must visit a configuration twice.

This is an upper bound on the length of a play if it is won with winning condition 3 or 6. If the condition that applies is 2 or 5 then the additional requirement of the formula in focus being a $\varphi \mathtt{U} \psi$ or $\varphi \mathtt{R} \psi$ must be fulfilled. The proof of Lemma 162 shows that in such a case a formula of this form must stay in focus. I.e. the moment the play performs a repeat a $\varphi \mathtt{U} \psi$ or a $\varphi \mathtt{R} \psi$ must be present in focus. Note that this can be in the form of an unfolding for instance. In any case, it can have at most three connectives more than a $\varphi \mathtt{U} \psi$ or $\varphi \mathtt{R} \psi$. Therefore, after five more steps – three for the connectives and two focus moves – a situation like the ones required for winning conditions 2 or 5 to apply must be reached. Thus,

$$(1+|\varphi_0|+2^{|\varphi_0|}) \cdot 2^{(2^{|\varphi_0|})}+5 \;=\; O(2^{|\varphi_0|} \cdot 2^{(2^{|\varphi_0|})})$$

is the maximal length of a play in the game $\mathcal{G}(\varphi_0)$. ■

**Corollary 164 (Determinacy)** *Player* $\forall$ *wins* $\mathcal{G}(\varphi)$ *iff player* $\exists$ *does not win* $\mathcal{G}(\varphi)$.

PROOF By Lemmas 162 and 163, every play of $\mathcal{G}(\varphi)$ has a uniquely determined winner and is of finite length. Then, Theorem 37 applies which says that for every game $\mathcal{G}(\varphi)$ one of the players has a winning strategy. ∎

**Lemma 165** *Player* $\exists$ *preserves unsatisfiability with her choices. Player* $\forall$ *can preserve unsatisfiability with his choices.*

PROOF Player $\forall$ is mostly concerned with the position of the focus. Those moves preserve unsatisfiability. The only rule that requires him to make a genuine choice is rule $(\mathtt{AX})$. Suppose

$$\mathtt{A}(\psi,\Gamma_1;\ldots;\Gamma_n),\mathtt{E}(\psi,\Sigma_i),\Pi$$

is satisfiable for every $i \in \{1,\ldots,m\}$. Then each of them has a model $\mathcal{T}_i = (\mathcal{S}_i, \rightarrow_i, L_i)$ with an $s_i \in \mathcal{S}$ s.t. there is a path $\pi = s_i \ldots$ with

$$\pi \models \psi \wedge \varphi \qquad \text{for every } \varphi \in \Sigma_i$$

Furthermore, for every path $\pi'$ with $\pi' = s_i \ldots$ there is a $j \in \{1,\ldots,n\}$ s.t.

$$\pi' \models \varphi \qquad \text{for all } \varphi \in \Gamma_j \quad \text{and} \quad j = 1 \quad \text{implies} \quad \pi' \models \psi$$

We assume the state sets of the $\mathcal{T}_i$ to be pairwise disjoint. Take the transition system

$$\mathcal{T}' := \Big( \{s_0\} \cup \bigcup_{i=1}^{m} \mathcal{S}_i, \bigcup_{i=1}^{m} \underset{i}{\rightarrow}, \bigcup_{i=1}^{m} L_i \Big)$$

with the additional transitions

$$s_0 \rightarrow s_i \quad \text{for every } i \in \{1,\ldots,m\}$$

and $L(s_0) := \Pi'$ for some maximally consistent $\Pi'$. Then,

$$s_0 \models \mathtt{A}(\mathtt{X}\psi,\mathtt{X}\Gamma_1;\ldots;\mathtt{X}\Gamma_n),\mathtt{EX}\Sigma_1,\ldots,\mathtt{EX}\Sigma_m,\Pi' \tag{8.2}$$

i.e. this formula is satisfiable, too. Conversely, if this formula is unsatisfiable and $\Pi'$ is maximally consistent then there is an $i \in \{1,\ldots,m\}$ s.t.

$$\mathtt{A}(\psi,\Gamma_1;\ldots;\Gamma_n),\mathtt{E}(\psi,\Sigma_i),\Pi$$

is unsatisfiable. Note that, if the focus is on $\mathtt{X}\psi$, player $\forall$ can apply rule $(\mathtt{AX})$ to the configuration in (8.2). By choosing the right $i$ he can preserve unsatisfiability. This is also possible if $\mathcal{E} = \emptyset$. If the focus is in a $\Sigma_i$ that is not the one to choose he can change the focus with rule $(\mathrm{FC}_4)$ and then preserve unsatisfiability with rule $(\mathtt{EX})$.

The rules that require player $\exists$ to set the focus preserve unsatisfiability. So do those that make her choose a disjunct. The cases of rules $(\mathtt{EA})$ and $(\mathtt{EE})$ are given by the equivalences $Q_2 Q_1 \varphi \equiv Q_1 \varphi$ for $Q_1, Q_2 \in \{\mathtt{E}, \mathtt{A}\}$.

Now consider rule $(\mathtt{AE})$. Suppose there is a model $\mathcal{T}, s$ for

$$\mathtt{A}(\Gamma_1; \ldots; \Gamma_n), \mathtt{E}(\psi), \mathcal{E}, \Pi$$

Then $\mathcal{T}, s$ is also a model for

$$\mathtt{A}(\Gamma_1; \ldots; \mathtt{E}\psi, \Gamma_i; \ldots; \Gamma_n), \mathcal{E}, \Pi$$

for every $i = 1, \ldots, n$ because every path starting in $s$ satisfies $\mathtt{E}\psi$ regardless of which $\Gamma_j$ it fulfils, too. The case of rule $(\mathtt{AA})$ is similar.

Player $\exists$ also preserves unsatisfiability with her choices of a set of propositional constants in rules $(\mathtt{AX})$, $(\mathtt{EX})$ and $(\mathtt{AX}\not\mathtt{E})$.

Finally, note that the deterministic rules like unfolding of an $\mathtt{U}$ or a $\mathtt{R}$ preserve unsatisfiability as well. ∎

**Definition 166** An $\mathtt{E}\Sigma$ is called the *immediate descendant* of $\mathtt{E}\Sigma'$ in a play $C_0, \ldots, C_n$ of $\mathcal{G}(\varphi_0)$ if there are two configurations $C_i$ and $C_{i+1}$ s.t.

$$C_i \;=\; \mathcal{A}', \mathtt{E}\Sigma', \mathtt{E}\Sigma'_1, \ldots, \mathtt{E}\Sigma'_{n'}, \Pi'$$

and

$$C_{i+1} \;=\; \mathcal{A}, \mathtt{E}\Sigma, \mathtt{E}\Sigma_1, \ldots, \mathtt{E}\Sigma_n, \Pi$$

and there is a rule that transformed $\mathtt{E}\Sigma'$ into $\mathtt{E}\Sigma$. Both $n = 0$ and $n' = 0$ are possible which is needed for applications of rule $(\mathtt{EX})$ or $(\mathtt{AX}\not\mathtt{E})$ for example.

$\mathtt{E}\Sigma$ is a *descendant* of $\mathtt{E}\Sigma'$ if they are elements of the transitive closure of its immediate descendant relation.

If there are two configurations $C_i$ and $C_{i+1}$ s.t.

$$C_i \;=\; \mathtt{A}(\Gamma';\Gamma'_1;\ldots;\Gamma'_{n'}),\mathtt{E}\Sigma'_1,\ldots,\mathtt{E}\Sigma'_{m'},\Pi'$$

and

$$C_{i+1} \;=\; \mathtt{A}(\Gamma;\Gamma_1;\ldots;\Gamma_n),\mathtt{E}\Sigma,\mathtt{E}\Sigma_1,\ldots,\mathtt{E}\Sigma_m,\Pi$$

and there is a rule that transformed $\Gamma'$ into $\Gamma$ then $\Gamma$ is an immediate descendant of $\Gamma'$. Moreover, $\mathtt{A}(\Gamma;\Gamma_1;\ldots;\Gamma_n)$ is an immediate descendant of $\mathtt{A}(\Gamma';\Gamma'_1;\ldots;\Gamma'_{n'})$. Again, the descendant relation is given as the transitive closure of the immediate version.

An $\mathtt{E}\Sigma$ is called *persisting* in a play at some point if it was not discarded in the last application of rule $(\mathtt{AX})$ or $(\mathtt{EX})$ or was created in the last application of rule $(\mathtt{AX}\not\mathtt{E})$. Formally, $\mathtt{E}\Sigma$ is persisting in the configuration $C_i$ of the play $C_0,\ldots,C_n$, $0 < i \leq n$, if there is a $j \leq i$ s.t.

- between $j$ and $i$ none of the rules $(\mathtt{AX})$, $(\mathtt{EX})$ or $(\mathtt{AX}\not\mathtt{E})$ has been played, and
- $(C_{j-1},C_j)$ is an instance of rule $(\mathtt{AX})$, $(\mathtt{EX})$ or $(\mathtt{AX}\not\mathtt{E})$, and
- there is no $\mathtt{E}\Sigma' \in C_{j-1}$, or $\mathtt{E}\Sigma$ is the descendant of an $\mathtt{E}\Sigma' \in C_{j-1}$.

**Definition 167 (Top-level list strategy)** For a set $\Sigma$ of formulas let $l_\Sigma$ be a *priority list* of all *top-level* $\mathtt{U}$ subformulas in $\Sigma$, i.e.

$$l_\Sigma \;=\; \varphi_1\mathtt{U}\psi_1,\ldots,\varphi_n\mathtt{U}\psi_n$$

with $\varphi_i\mathtt{U}\psi_i \in \Sigma$ for all $i = 1,\ldots,n$. A list $l_\Gamma$ is defined for a set $\Gamma$ of formulas in the same way.

After each application of rule $(\mathrm{FM}_1)$ or $(\mathrm{FC}_1)$ with the actual configuration

$$\mathtt{A}(\left[\Gamma_1\right];\ldots;\Gamma_n),\mathtt{E}\Sigma_1,\ldots,\mathtt{E}\Sigma_n,\Pi$$

player $\forall$ creates the list $l_{\Gamma_1}$ and plays according to the following strategy. He sets the focus to the first element $\varphi_1\mathtt{U}\psi_1$ of the list and leaves it there until player $\exists$ sets the focus to $\psi_1$ after it has been unfolded. Then player $\forall$ deletes $\varphi_1\mathtt{U}\psi_1$ from $l_{\Gamma_1}$ and changes focus with rule $(\mathrm{FC}_2)$ to the next element of $l_{\Gamma_1}$.

If at some point player $\exists$ changes focus to another $\Gamma_i$ with rule $(\mathrm{FC}_1)$ he restarts this strategy with the list $l_{\Gamma_i}$. If his actual list is empty or no element of the list is present anymore he calculates $l_\Sigma$ for a persisting $\mathtt{E}\Sigma$ if one exists, changes focus to the first element of $l_\Sigma$ with rule $(\mathrm{FC}_3)$ and plays the same strategy there.

If it does not contain any top-level $\mathtt{U}$ formulas he puts the focus to the largest formula in $\Sigma$ and leaves it there until some $\mathtt{U}$ becomes top-level and calculates $l_\Sigma$ at this point. If there is none than he changes focus to the next biggest formula at the moment when the play is about to perform a repeat.

If $l_\Sigma$ becomes empty or all the top-level $\mathtt{U}$ formulas have been fulfilled he puts the focus back onto the actual $\mathcal{A}$ part with rule $(\mathrm{FC}_5)$ and restarts the entire process with the current configuration.

Of course, player $\forall$ checks at any point in the play whether he can change the focus to an atomic proposition $q$, s.t. $\overline{q} \in \Pi$ for the actual propositional part $\Pi$. This can be done with rule $(\mathrm{FC}_3)$ or $(\mathrm{FC}_4)$ if $q \in \Sigma$ for some present $\mathtt{E}\Sigma$. It is also possible with rule $(\mathrm{FC}_2)$ inside a $\Gamma$ if player $\exists$ does not take the focus away from it. Finally, if all present $\Gamma$ contain such a $q$ he can do so with rule $(\mathrm{FC}_5)$ and $(\mathrm{FM}_2)$ since player $\exists$ has to choose one of the $\Gamma$s with rule $(\mathrm{FM}_1)$ in between.

**Lemma 168** ***(Optimality)*** *Player $\forall$'s top-level list strategy is optimal.*

PROOF As in the proof of Lemma 98, we need to show that with this strategy player $\forall$ does not miss any $\mathtt{U}$ formulas. This holds only if he selects the right $\mathtt{E}\Sigma$ when using rule $(\mathrm{FC}_3)$, namely the one that contains a regenerating $\mathtt{U}$ formula. Note that during a play, $\mathtt{E}\Sigma$ components can get lost when rule $(\mathtt{AX})$ or $(\mathtt{EX})$ is played.

For the moment we assume that he nondeterministically makes the best choice when using rule $(\mathrm{FC}_3)$.

Remember that a configuration represents a combination of conjunctions and disjunctions. Therefore, not missing a regenerating $\mathtt{U}$ formula is to be interpreted in the following way. If there is one in an $\mathtt{E}\Sigma$ then player $\forall$ will eventually set the focus to it. If all $\Gamma_i$ and their descendants contain one then he will eventually keep the focus on one of them.

Suppose there is a regenerating U formula. It must become top-level in its component at some point. Suppose this is inside an $\mathtt{E}\Sigma$. It remains there since player $\exists$ cannot fulfil it. If the focus is already in $\Sigma$ it will be found unless there is another one which does not get fulfilled either. If the focus is inside $\mathcal{A}$ then player $\forall$ chooses this $\mathtt{E}\Sigma$ or its descendant when playing rule $(\mathtt{AX})$. Remember that we assume player $\forall$ to be able to guess which present $\mathtt{E}\Sigma$ is best. Eventually, he will move the focus into this $\Sigma$, find the U formula there and win with his winning condition 2.

Suppose now that all the $\Gamma$s inside the present $\mathcal{A}$ contain a regenerating U formula that is top-level already. Player $\forall$'s strategy makes him set the focus to $\mathcal{A}$ when all the formulas inside the persisting $\mathtt{E}\Sigma$ have been fulfilled. Regardless of which $\Gamma$ is chosen by player $\exists$ the regenerating U formula will be a member of $l_\Gamma$. W.l.o.g. we assume that it is the first of its kind in the list. Therefore, player $\forall$ will eventually put the focus onto it.

Unlike the case of a $\varphi \mathtt{U} \psi$ formula in a $\Sigma$, player $\exists$ can simply put the focus onto $\psi$ each time it gets unfolded. But $\varphi \mathtt{U} \psi$ is regenerating, i.e. player $\exists$ is assumed not to be able to do so if the formula was inside a $\Sigma$. This means player $\forall$ would also win with the focus inside the $\Gamma$ now containing $\psi$. This is possible if it contains another regenerating U formula like $\psi$ itself for example which has now become top-level. This reduces the argument to a smaller U formula. Finally, the regeneration of an U formula must be due to an atomic contradiction. But player $\forall$'s strategy will make him change focus and win with his condition 1 if this is the case.

Therefore player $\exists$ will at some point change focus with her rule $(\mathrm{FC}_1)$. Remember that she can only set it to another $\Gamma$ which is assumed to contain a regenerating U formula as well. This holds in particular for the other descendant of the $\Gamma$ the focus was in beforehand. With the same argument she will at some point be forced to change focus again. Eventually, she will have created a repeat and player $\forall$ wins this play with his winning condition 3.

Now consider a strategy for player $\forall$ other than the top-level list strategy and suppose that player $\exists$ has a winning strategy for the game at hand. An optimal strategy must enable player $\forall$ to try to put the focus onto every possible U formula and avoid repeating configurations for as long as possible in every play of player $\exists$'s game tree.

Candidates for possible $\mathtt{U}$ formulas are those inside an $\mathtt{E}\Sigma$ that is present and those inside a $\Gamma$ that player $\exists$ has set the focus to. Lemma 162 shows that between repeating configurations one of the rules $(\mathtt{AX})$, $(\mathtt{EX})$ or $(\mathtt{AX}\not{\mathtt{E}})$ must have been played. This will select one $\mathtt{E}\Sigma$ as persisting for each of player $\forall$'s choices that are present in player $\exists$'s game tree.

Note that the top-level list strategy behaves like two interleaved priority list strategies in the sense of Definition 97. The first formulas in a priority list are those that are top-level. If they have been processed player $\forall$'s strategy switches to the other top-level list which contains those $\mathtt{U}$ formulas that would be first in a priority list for the actual $\Gamma$, resp. $\Sigma$.

This interleaving guarantees player $\forall$ to try all possible $\mathtt{U}$ formulas before a repeat is performed. Note that he also avoids repeats by changing focus in the last moment before one occurs. ■

**Lemma 169** *Suppose* $\Pi$ *is satisfiable but*

$$\mathtt{A}(\Gamma_1 \vee \ldots \vee \Gamma_n) \wedge \mathtt{E}\Sigma_1 \wedge \ldots \wedge \mathtt{E}\Sigma_m \wedge \Pi$$

*is unsatisfiable. Then one of the following cases holds.*

1. *All* $\Sigma_i$ *and at least one* $\Gamma_j$ *is satisfiable for* $i = 1, \ldots, m$ *and* $j \in \{1, \ldots, n\}$*, but there is a* $q \in \Pi$ *s.t.* $\models \Sigma_i \rightarrow \overline{q}$ *for some* $i \in \{1, \ldots, m\}$*, or* $\models \Gamma_i \rightarrow \overline{q}$ *for all* $i = 1, \ldots, n$*, or*
2. *there is an* $i \in \{1, \ldots, m\}$ *s.t.* $\mathtt{E}\Sigma_i$ *is unsatisfiable but* $\mathtt{A}(\Gamma_1 \vee \ldots \vee \Gamma_n)$ *is satisfiable, or*
3. *for every* $i = 1, \ldots, n$*:* $\Gamma_i$ *is unsatisfiable but* $\mathtt{E}\Sigma_1 \wedge \ldots \wedge \mathtt{E}\Sigma_m$ *is satisfiable, or*
4. *there is an* $i \in \{1, \ldots, m\}$ *s.t.* $\models \mathtt{A}(\Gamma_1 \vee \ldots \vee \Gamma_n) \rightarrow \overline{\mathtt{E}\Sigma_i}$.

PROOF A conjunction is unsatisfiable if and only if one of its conjuncts is unsatisfiable or the combination of some of them imply the negation of another one. Note that the latter case is not possible for two existentially quantified formulas since they can only contradict each other in the propositional part. But then one of them has to contradict a

$q \in \Pi$ because of $\Pi$'s maximal consistency already. Note that the last case also covers a situation in which both the $\mathcal{A}$ part and an $\mathtt{E}\Sigma_i$ are unsatisfiable on their own already.

Furthermore, the $\mathcal{A}$ part can only be unsatisfiable if all of its disjuncts are unsatisfiable. ■

The next lemma analyses how these cases of unsatisfiability occur in a play.

**Lemma 170** *Suppose $\varphi_0$ is unsatisfiable. Take a play $C_0, \dots, C_n$ of $\mathcal{G}(\varphi_0)$ in which player $\forall$ uses his top-level list strategy and preserves unsatisfiability. Let*

$$C_i \quad = \quad \mathcal{A}_i, \mathcal{E}_i, \Pi_i$$

*for $i = 0, \dots, n$. Then either $C_n$ is terminal and satisfies case 1 of Lemma 169 or there is a $j$ with $0 \leq j \leq n$ s.t.*

- *$\mathcal{A}_i \wedge \Pi_i$ is unsatisfiable for all $i$ with $j \leq i \leq n$, or*
- *there is an unsatisfiable $\mathtt{E}\Sigma \in C_j$ that persists and remains unsatisfiable until $C_n$.*

*Moreover, a repeat can only occur after $C_j$.*

PROOF The existence of an unsatisfiable configuration $C_j$ is simply given by the preservation of unsatisfiability. The fact that a repeat can only occur after $C_j$ is based on the observation that two configurations which satisfy different cases of Lemma 169 cannot be syntactically equal. But the claim states that eventually one of the cases will hold generally.

Suppose $C_j$ is unsatisfiable according to the first case of Lemma 169. If there is an $\mathtt{E}\Sigma \in \mathcal{E}_j$ with $\models \Sigma \rightarrow \overline{q}$ then there must be a $\psi \in \Sigma$ s.t. $\overline{q} \in Sub(\psi)$. As in the proofs of Lemmas 101 and 113, one can show that the game rules will create a configuration with a descendant $\Sigma'$ of $\Sigma$ s.t. $\overline{q} \in \Sigma'$. But then player $\forall$'s top-level list strategy tells him to set the focus to $\overline{q}$ immediately which makes the configuration terminal.

The other part of case 1 of Lemma 169 states that $\models \Gamma_i \rightarrow \overline{q}$ for all $\Gamma_i \in \mathcal{A}_j$ and $q \in \Pi_j$. The play need not reach a terminal configuration since player $\exists$ may always be able to change the focus inside $\mathcal{A}_j$. However, because of preservation of unsatisfiability, $\overline{q}$

will always occur in a descendant of a $\Gamma_i$ after finitely many steps. As this holds for every $\Gamma_i$, player $\exists$ cannot discard them all with rule $(\not\Gamma)$ and one will remain. Game rules $(\mathtt{AX})$, $(\mathtt{EX})$ and $(\mathtt{AX}\not\mathtt{E})$ can then not be played since there is no need for player $\forall$ to put the focus to a formula of the form $\mathtt{X}\psi$. But then

$$\models \ \mathcal{A}_i \rightarrow \overline{\Pi_i}$$

for all $i$ with $j \leq i \leq n$.

Suppose now that $C_j$ is unsatisfiable because of case 2 or 3 of Lemma 169. According to Lemma 165, all $C_i$ are unsatisfiable for $i \geq j$.

Note that, if case 2 holds for $C_i$ then $C_{i+1}$ can fulfil case 3 because of rule $(\mathtt{EA})$ for example. Conversely, rule $(\mathtt{AE})$ can swap unsatisfiability from the $\mathcal{A}$ part of a configuration to an $\mathtt{E}\Sigma$. However, this can only happen if one of these parts contains a path quantified formula. Thus, applying these rules alternatingly can only happen at most $m$ times where $m$ is bounded by the quantifier depth of $\varphi_0$. Note that the $\mathcal{A}_i$ are assumed to be satisfiable and unsatisfiable alternatingly. It is not possible for an unsatisfiable $\mathtt{E}\Sigma$ to be regenerated from a $\mathtt{R}$ formula inside $\mathcal{A}_i$ for example because this would cause $\mathcal{A}_i$ and all its descendants to be unsatisfiable.

Moreover, each application of rule $(\mathtt{EA})$, i.e. each swap of unsatisfiability from an $\mathtt{E}\Sigma$ to $\mathcal{A}_i$ destroys a path quantifier in the actual configuration that cannot be regenerated. Each application of rule $(\mathtt{AE})$ reduces the number of path quantifiers inside the actual $\mathcal{A}$. Thus, a repeat is only possible in a part of a play where unsatisfiability remains with either $\mathcal{A}_i$ or an $\mathtt{E}\Sigma \in \mathcal{E}_i$.

This shows that eventually either the first case of the claim holds or each $C_i$ contains an unsatisfiable $\mathtt{E}\Sigma$ which does not become satisfiable anymore. But it is part of player $\forall$'s strategy to let an unsatisfiable $\mathtt{E}\Sigma$ persist.

What remains is case 4 of Lemma 169. Suppose that $\mathcal{A} \wedge \mathtt{E}\Sigma$ is unsatisfiable. If both conjuncts are unsatisfiable on their own then the $\mathtt{E}\Sigma$ will persist and remain unsatisfiable. Assume therefore that both conjuncts are satisfiable. Player $\forall$'s strategy will make him move the focus into $\mathcal{A}$ once all the top-level $\mathtt{U}$ formulas in $\Sigma$ are processed. Whenever rule $(\mathtt{AX})$ is played he chooses the descendant of $\mathtt{E}\Sigma$ as long as the combination of this with the present $\mathcal{A}$ is still unsatisfiable. Eventually, the important formulas from $\mathcal{A}$ will have been copied into the descendant of $\mathtt{E}\Sigma$. This will eventually make it unsatisfiable on its own and player $\forall$ will move the focus into it at some point. A repeat can only occur after that since the unsatisfiable descendant of $\mathtt{E}\Sigma$ must be syntactically different from a satisfiable $\mathtt{E}\Sigma$.

However, at some point the conjunction of the present $\mathcal{A}$ and the descendant of $\mathtt{E}\Sigma$ can become satisfiable. But this is only possible if another $\mathtt{E}\Sigma'$ appeared such that the current $\mathcal{A} \wedge \mathtt{E}\Sigma'$ is unsatisfiable. Then player $\forall$ continues with his focus strategy inside $\mathcal{A}$ but chooses $\mathtt{E}\Sigma'$ with rule $(\mathtt{AX})$. Note that at this point he loses the copied formulas.

Suppose the play proceeds as follows. We will only consider configurations before and after an application of rule $(\mathtt{AX})$. Assume there are configurations of the form

$$\mathtt{A}(\mathtt{X}\Gamma_i^1;\ldots;\mathtt{X}\Gamma_i^{n_i}),\mathtt{EX}\Sigma'_{i-1},\mathtt{EX}\Sigma_i,\Pi'$$

with successors

$$\mathtt{A}(\Gamma_i^1;\ldots;\Gamma_i^{n_i}),\mathtt{E}\Sigma_i,\Pi$$

for $i = 1,\ldots n$ and some $k \in \mathbb{N}$. Suppose that each $\mathtt{EX}\Sigma'_i$ is the descendant of $\mathtt{E}\Sigma_i$ and that each $\mathtt{A}(\mathtt{X}\Gamma_i^1;\ldots;\mathtt{X}\Gamma_i^{n_i})$ is the descendant of $\mathtt{A}(\Gamma_{i-1}^1;\ldots;\Gamma_{i-1}^{n_{i-1}})$. As usual, $\mathcal{A}_i$ stands for $\mathtt{A}(\Gamma_i^1;\ldots;\Gamma_i^{n_i})$ while $\mathcal{A}_i^{\mathtt{X}}$ is used to denote $\mathtt{A}(\mathtt{X}\Gamma_i^1;\ldots;\mathtt{X}\Gamma_i^{n_i})$. Now suppose that every $\mathcal{A}_i \wedge \mathtt{E}\Sigma_i$ is unsatisfiable whereas every $\mathcal{A}_i^{\mathtt{X}},\mathtt{EX}\Sigma'_{i-1}$ is satisfiable. This captures the situation mentioned above: every $\mathtt{E}\Sigma$ that is unsatisfiable with the present $\mathcal{A}$ becomes satisfiable while a new $\mathtt{E}\Sigma'$ appears that is unsatisfiable with the current $\mathcal{A}$.

According to Lemma 163, a repeat on a configuration must eventually occur. W.l.o.g. we assume that it is on $\mathcal{A}_1^{\mathtt{X}},\mathtt{EX}\Sigma_1$. I.e. the descendant of $\mathtt{E}\Sigma_k$ is $\mathtt{EX}\Sigma_1$, and the descendant of $\mathcal{A}_k$ is $\mathcal{A}_1^{\mathtt{X}}$. For $i = 1,\ldots,k-1$ let $\kappa_i$ denote the number of applications of rule $(\mathtt{AX})$ or $(\mathtt{EX})$ between $\mathcal{A}_i,\mathtt{E}\Sigma_i$ and $\mathcal{A}_{i+1}^{\mathtt{X}},\mathtt{EX}\Sigma'_i,\mathtt{EX}\Sigma_{i+1}$ plus 1. Note that rule $(\mathtt{AX}\not{\mathtt{E}})$ cannot be applied as there is always at least one present $\mathtt{E}\Sigma$.

Since every $\mathcal{A}_i^{\mathtt{X}} \wedge \mathtt{EX}\Sigma'_{i-1}$ is satisfiable, it has a model $T_i$. Each $T_i$ is a possibly infinite tree containing one path satisfying $\mathtt{X}\Sigma'_{i-1}$, while this and every other path satisfies at least one $\mathtt{X}\Gamma_i^j$. Therefore, each $T_i$ begins with one transition after which $\Sigma_{i-1}$ or a $\Gamma_i^j$ might require the tree to branch. We paste these $T_i$ together to form an infinite tree $\mathcal{T}$ that is finitely represented as shown in Figure 8.8.

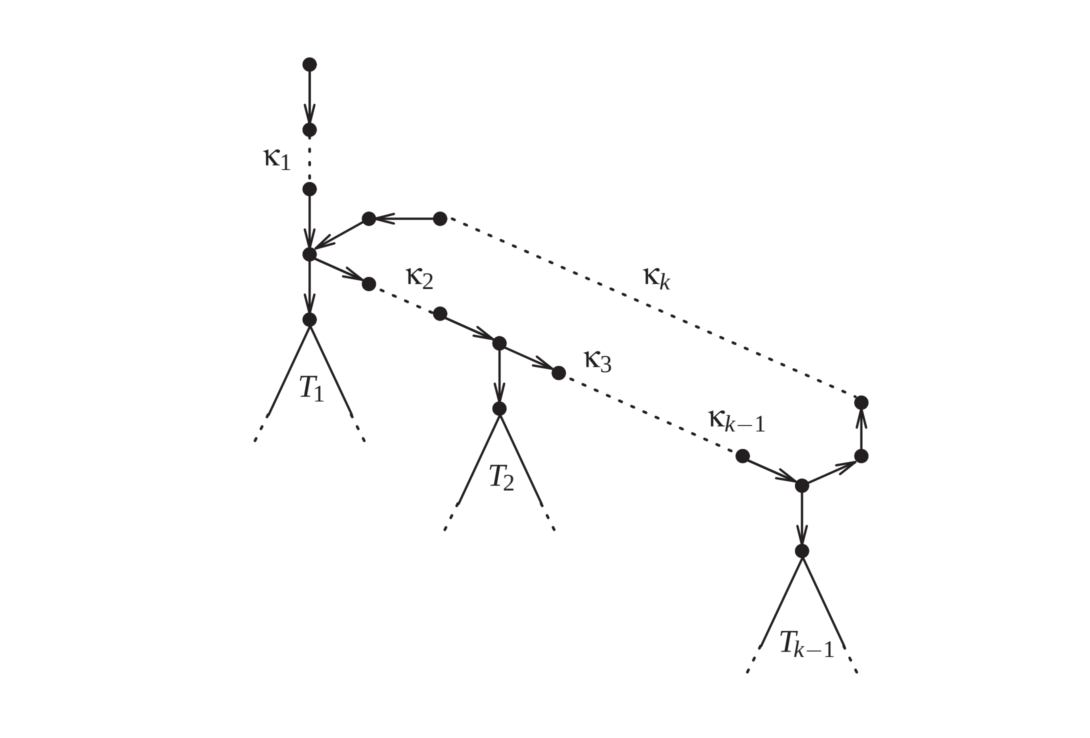


Figure 8.8: A model $\mathcal{T}$ for $\mathcal{A}_1, \mathtt{E}\Sigma_1$.

A sequence of states of length $\kappa_i$ corresponds to a sequence of applications of rules $(\mathtt{AX})$ or $(\mathtt{EX})$ in which the last persisting $\mathtt{E}\Sigma$ was chosen. The labelling of each state is the maximal consistent $\Pi$ that player $\exists$ chose for the corresponding block of configurations. Here, a block is a part of the play in which neither rule $(\mathtt{AX})$ nor $(\mathtt{EX})$ was played.

We show that the transition system of Figure 8.8 is a model for $\mathcal{A}_1 \wedge \mathtt{E}\Sigma_1$. Remember that $\mathtt{E}\Sigma_1$ has the descendant $\mathtt{EX}\Sigma_1'$ which is fulfilled in $T_1$. The same holds for every other $\mathtt{E}\Sigma_i$ that occurs during the play. Therefore, every occurring $\mathtt{E}\Sigma_i$ is fulfilled in $\mathcal{T}$ which also satisfies the corresponding $\mathcal{A}_i$.

Note that we have restricted ourselves to the case of only two $\mathtt{E}\Sigma$ components per configuration. The argument can easily be extended to deal with more than two.

It remains to be seen that the paths from the origin to each $T_i$ satisfy the $\mathcal{A}$ formulas. Remember that the first part of length $\kappa_1$ does not bear a contradiction to $\mathtt{E}\Sigma_1$ or any occurring $\Pi$ in this part, for otherwise player $\forall$ would have won the corresponding play with condition 1. Moreover, the second part of length $\kappa_2$ starting from the root of $T_1$ does not exhibit a contradiction to the next persisting $\mathtt{E}\Sigma_2$ and the corresponding $\Pi$'s. Thus, it satisfies at least one $\Gamma_2^i$. But $\mathcal{A}_2$ is the descendant of $\mathcal{A}_1$. Therefore the paths starting from the root into $T_2$ must satisfy at least one $\Gamma_1^i$, for if this was not the case then the play would not have proceeded this way. Iterating this argument shows that the infinite path connecting all the $T_i$ must satisfy one of the $\Gamma_1^i$. Thus, $\mathcal{A}_1, \mathtt{E}\Sigma_1$ cannot be unsatisfiable.

Conversely, there is at least one $\mathtt{E}\Sigma$ which remains unsatisfiable. But then it is going to persist according to player $\forall$'s strategy of preserving unsatisfiability. ■

**Theorem 171 (Soundness)** *If $\varphi_0$ is unsatisfiable then player $\forall$ wins $\mathcal{G}(\varphi_0)$.*

PROOF We let player $\forall$ use his top-level list strategy as it is described in Definition 167. Furthermore, whenever an application of rule $(\mathtt{AX})$ or $(\mathtt{EX})$ requires him to choose an $\mathtt{E}\Sigma$ he selects the one that preserves unsatisfiability according to Lemma 165.

With the top-level list strategy a terminal configuration is only reached if it contains an atomic contradiction which makes player $\forall$ the winner of the play at hand, or if there are no non-atomic formulas that player $\forall$ can set the focus to. But by preservation of unsatisfiability the resulting configuration must be unsatisfiable, i.e. a win for player $\forall$ with winning condition 1.

According to Lemma 169, there are four possibilities for a configuration

$$\mathtt{A}(\Gamma_1;\ldots;\Gamma_m), \mathtt{E}\Sigma_1, \ldots, \mathtt{E}\Sigma_n, \Pi$$

to be unsatisfiable. Lemma 170 shows that, if no terminal configuration is reached, the $\mathcal{A}$ part remains unsatisfiable possibly in conjunction with the respective propositional parts, or eventually an unsatisfiable $\mathtt{E}\Sigma$ must persist.

Remember that player $\exists$ always chooses $\Pi$ to be satisfiable, and if $\Pi$ is unsatisfiable in conjunction with the $\mathtt{E}\Sigma_i$ then player $\forall$ can immediately win with his winning condition 1 by setting the focus to the appropriate proposition. If the conjunction of the $\mathcal{A}$ part and the $\Pi$ is unsatisfiable then player $\exists$ can only change focus inside $\mathcal{A}$ but player $\forall$ can always play such that the focus reaches a $\overline{q}$ for a $q \in \Pi$. But then a repeat will eventually occur and player $\exists$ has used rule $(\mathrm{FC}_1)$ whereas player $\forall$ has used at most rule $(\mathrm{FC}_2)$ since the focus did not leave $\mathcal{A}$. Hence, he wins with his winning condition 3 unless player $\exists$ has left the focus on a particular $\Gamma$ in which case he wins with condition 1.

Lemma 170 shows that, if this is not the case, then eventually all configurations will contain an $\mathtt{E}\Sigma$ which is either unsatisfiable or whose negation is implied by the $\mathcal{A}$ component. It also shows that eventually one of these must persist.

But unsatisfiability must be given by one or several $\mathtt{U}$ formulas which cannot get fulfilled. If one of them occurs in the persisting $\mathtt{E}\Sigma$ then player $\forall$ will eventually set the focus to it according to Lemma 168. If each $\Gamma$ of the $\mathcal{A}$ part contains one then, again by Lemma 168, player $\forall$ will eventually set the focus to one of them which gets copied into the persisting $\mathtt{E}\Sigma$. Finally, player $\forall$ will change focus to it in $\mathtt{E}\Sigma$ and win with condition 2. ∎

Next we recall Definition 69 and Lemma 70 of Chapter 5. These form the basis for player $\exists$'s winning strategy on a satisfiable formula in the completeness proof for the satisfiability games.

**Definition 172** Take a state $s_0$ of a transition system $\mathcal{T}$ and satisfiable formulas

$$\Gamma_1, \ldots, \Gamma_n$$

Assume that $s_0 \models \mathtt{A}(\Gamma_1 \vee \ldots \vee \Gamma_n)$, i.e. every path $\pi$ starting with $s_0$ satisfies at least one $\Gamma_i$. With each $\Gamma_i$ we associate a set $P'_{\Gamma_i}(s_0)$ of finite prefixes of paths starting with $s_0$ in the following way. Let $\sigma = s_0 \ldots s_k$ be a finite sequence of states s.t. $s_i \rightarrow s_{i+1}$ for all $i = 0, \ldots, k-1$.

$$\sigma \in P'_{\Gamma_i}(s_0) \quad \text{iff} \quad \text{there is a path } \pi = \sigma\pi' \text{ s.t. } \pi \models \Gamma_i$$

Thus, $P'_{\Gamma_i}(s)$ consist of all finite sequences of states starting with $s$ that can be extended to a path satisfying $\Gamma_i$. In the next step we make these sets disjoint. Let $P_{\Gamma_i}(s) \subseteq P'_{\Gamma_i}(s)$ be defined by

$$\sigma \in P_{\Gamma_i}(s) \quad \text{iff} \quad \sigma \in P'_{\Gamma_i}(s) \text{ and for all } j < i : \sigma \notin P'_{\Gamma_j}(s)$$

A path can never occur in a set with a smaller index than any of its prefixes. Thus, $s \in P_{\Gamma_1}(s)$ in any case.

**Lemma 173** *Let $\sigma_1, \sigma_2$ be finite prefixes of a path starting in s, s.t. $\sigma_2 = \sigma_1\sigma$ for some $\sigma$. If $\sigma_1 \in P_{\Gamma_i}(s)$ and $\sigma_2 \in P_{\Gamma_j}(s)$ then $j \geq i$.*

This is the same as Lemma 70 of Chapter 5, simply reformulated to take care of the normal form for CTL$^*$ formulas needed in this chapter.

**Definition 174** An *extended configuration* of a CTL$^*$ satisfiability game $\mathcal{G}(\varphi_0)$ and an LTS $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ is of the form

$$t \ \vdash \ \mathcal{A}, \mathcal{E}, \Pi$$

where $t \in \mathcal{S}$ and $\mathcal{A}, \mathcal{E}, \Pi$ is an ordinary configuration of $\mathcal{G}(\varphi_0)$. It is called *true* if $t \models \mathcal{A}, \mathcal{E}, \Pi$ and *false* otherwise.

An *extended game* is an ordinary satisfiability game with each configuration extended in the following way. If $t$ is the state component of an extended configuration to which rule $(\mathtt{AX})$, $(\mathtt{EX})$ or $(\mathtt{AX}\not{\mathtt{E}})$ is applied and $t'$ is the state component of the successor configuration then $t \rightarrow t'$. Player $\exists$ is allowed to choose such a $t'$. All other rules preserve the state component.

Note the intended similarity to a model checking configuration. It is only the CTL$^*$ normal form that detains us from proving completeness by relating a satisfiability game to one or several model checking games as it is done in Chapter 6.

**Lemma 175** *Player $\exists$ can preserve truth in an extended game, player $\forall$ must preserve truth.*

PROOF There is only one situation in which player $\exists$ performs a genuine choice on a formula: that of a disjunction inside $\mathcal{E}$. All the other rules requiring her to take a choice deal with the position of the focus inside $\mathcal{A}$. Suppose the actual extended configuration is

$$t \ \vdash \ \mathcal{A}, \mathtt{E}(\psi_0 \vee \psi_1, \Sigma), \mathcal{E}, \Pi$$

Player $\exists$ simply chooses the $\psi_i$ that is true. Note that $\mathtt{E}(\psi_0 \vee \psi_1) \equiv \mathtt{E}\psi_0 \vee \mathtt{E}\psi_1$.

Preservation of truth with the rules for the other boolean connectives, the focus moves and changes, and the unfolding of U and R formulas is trivial.

The other interesting case is the one where rule $(\mathtt{AX})$, $(\mathtt{EX})$ or $(\mathtt{AX}\not{\mathtt{E}})$ is played. Suppose

$$s_0 \models \mathtt{A}(\mathtt{X}\Gamma_1;\ldots;\mathtt{X}\Gamma_n),\mathtt{EX}\Sigma_1,\ldots,\mathtt{EX}\Sigma_m,\Pi$$

Then, for every $i = 1,\ldots,m$, there must be a path $\pi_i = s_0\ldots$ s.t.

$$\pi_i \models \mathtt{X}\Sigma_i \wedge \mathtt{X}\Gamma_j$$

for some $j \in \{1,\ldots,n\}$. Let $s_1 := \pi_i^{(1)}$. Player $\exists$ can choose $s_1$ and $\Pi' := L(s_1)$ to make the next extended configuration

$$s_1 \vdash \mathtt{A}(\Gamma_1;\ldots;\Gamma_n),\mathtt{E}\Sigma_i,\Pi'$$

true, too. Note that the position of the focus only determines which rule exactly is played. ■

**Lemma 176** *Consider the set of all sets $\Gamma$ that can occur inside an $\mathcal{A}$ in a configuration of $\mathcal{G}(\varphi_0)$. It can be ordered as $L = L_1,\ldots,L_m$, where $L_i = \Gamma_{i,1},\ldots,\Gamma_{i,n_i}$ for all $i = 1,\ldots,m$ s.t.*

- *if $\Gamma'$ is a descendant of $\Gamma$ but $\Gamma$ is not a descendant of $\Gamma'$, and $\Gamma \in L_i$, $\Gamma' \in L_j$ then $j > i$.*
- *if $\Gamma$ and $\Gamma'$ are descendants of each other, then there is an $i \in \{1,\ldots,m\}$ s.t. $\Gamma,\Gamma' \in L_i$*

PROOF The set of all such configurations together with the immediate descendant relation forms a graph. Each part $L_i$ of the list $L$ represents a strongly connected component of this graph. They can be sorted topologically to obtain $L$. ■

**Theorem 177 (Completeness)** *If $\varphi_0$ is satisfiable then player $\exists$ wins $\mathcal{G}(\varphi_0)$.*

PROOF Suppose $\varphi_0$ is satisfiable. Then there is a transition system $\mathcal{T} = (\mathcal{S}, \to, L)$ with a state $s_0 \in \mathcal{S}$ s.t. $s_0 \models \varphi_0$. Moreover, $s_0 \models \mathtt{A}\varphi_0$ since $\varphi_0$ is a state formula. Consider the extended game for $\mathcal{G}(\varphi_0)$ and $\mathcal{T}$. Its first configuration

$$s_0 \ \vdash \ \mathtt{A}(\left[\varphi_0\right])$$

is true.

Lemma 175 shows that all reached configurations will be true regardless of player $\forall$'s choices. Thus, if a play reaches a terminal configuration she will win it with her winning condition 4.

It remains to be described how player $\exists$ has to set the focus inside an $\mathcal{A}$ in order to win $\mathcal{G}(\varphi_0)$. Since the extended play follows states in $\mathcal{T}$ along the $\to$ relation, there will be a selected finite sequence $\sigma = s_0 s_1 \dots s_k$ of states at any moment in the play. Let

$$L \ = \ L_1, \dots, L_m \ = \ \Gamma_1, \dots, \Gamma_n$$

be the sorted list of all $\Gamma$ that can possibly occur in $\mathcal{G}(\varphi_0)$ according to Lemma 176. Furthermore, let

$$P_{\Gamma_1}(s_0), \ \dots, \ P_{\Gamma_n}(s_0)$$

be the sets of finite sequences of states starting in $s_0$ according to Definition 172. Note that at any moment in a play one of these finite sequences will have been selected.

Whenever the play has outlined the sequence $\sigma$ and player $\forall$ sets the focus to $\left[\mathcal{A}\right]$, then player $\exists$ sets it to the $\Gamma_i$ s.t. $\sigma \in P_{\Gamma_i}(s_0)$. Note that at the beginning an eligible one exists and, similar to the proof of Theorem 71, the rules preserve the following invariant. At any moment in a play that has outlined $\sigma = s_0 \dots s_k$ there is at least one $\Gamma_i$ present in the actual $\mathcal{A}$ s.t. $\sigma \in P_{\Gamma_i}(s_0)$. On the other hand, they are disjoint. Thus, there is always exactly one $P_{\Gamma_i}(s_0)$ that contains $\sigma$.

Suppose the actual extended configuration is

$$t \ \vdash \ \mathtt{A}(\left[\psi_0 \lor \psi_1\right], \Gamma'; \dots), \mathcal{E}, \Pi$$

with $\Gamma := \psi_0 \vee \psi_1, \Gamma'$, and $\Gamma \in L_i$ for some $i \in \{1, \dots, m\}$. Player $\exists$ has to set the focus to one of the two possible descendants. Assume she chooses $\psi_0, \Gamma'$. Then either $\Gamma$ is a descendant of $\psi_0, \Gamma'$ itself, in which case $\psi_0, \Gamma' \in L_i$. Or $\Gamma$ is not reachable from $\psi_0, \Gamma'$ anymore. Then $\psi_0, \Gamma' \in L_j$ for some $j > i$ according to Lemma 176.

Thus, she either remains in the current $L_i$ or increases the index of the actual sublist $L_i$ whenever rule $(\mathtt{A}\vee)$ is played.

Note that there is no need for player $\exists$ to change focus between $\Gamma_i$ and $\Gamma_j$ if both are descendants of each other. Either they never occur in the same configuration in which case a focus change is simply not possible. Or there is another $\Gamma$ s.t. both $\Gamma_i$ and $\Gamma_j$ are descendants of $\Gamma$ and vice versa. If the focus was on $\Gamma$ and is on $\Gamma_i$ later on for example although player $\exists$ would like to change it to $\Gamma_j$ then she could have set the focus accordingly earlier on. If it was not on $\Gamma$ then she can wait until it is on $\Gamma$ and then direct it to $\Gamma_j$. In both cases she does not change focus.

Now consider an application of rule $(\mathtt{AX})$, $(\mathtt{EX})$ or $(\mathtt{AX}\not{\mathtt{E}})$. Remember that after that, player $\exists$ is allowed to change the focus. This might be necessary since the sequence of states $\sigma$ outlined so far is prolonged with another state $t'$.

Suppose the focus was in $\Gamma_i$. Then we know that $\sigma \in P_{\Gamma_i}(s_0)$. But $\sigma t'$ is an extension of $\sigma$ and according to Lemma 173, $\sigma t' \in P_{\Gamma_j}(s_0)$ only if $j \geq i$. If $j = i$ then player $\exists$ can leave the focus in the actual $\Gamma$. Otherwise, the path that player $\forall$ is going to choose by selecting another $\mathtt{E}\Sigma$ does not fulfil $\Gamma_i$. Therefore, player $\exists$ needs to change focus. Lemma 176 shows that it increases the index of the actual sublist $L_i$.

If there is a $\left[q\right]$ in the actual $\Gamma$ then player $\forall$ has to change focus since he would inevitably lose at this point. Let $\sigma = s_0 \dots s_k$ be the prefix that has been selected so far. Then $\Pi = L(s_k)$ for the propositional part of the actual configuration. But also $\sigma \in P_{\Gamma_i}(s_0)$ for the $\Gamma_i$ containing the focus and therefore $q \in \Pi$.

If the focus is on an $\left[\mathtt{E}\varphi\right]$ then it is removed from player $\exists$'s control since the $\mathtt{E}\varphi$ gets promoted into the actual $\mathcal{E}$ part and with it the focus. If it is on an $\left[\mathtt{A}\varphi\right]$ the sets of $\Gamma$s get modified. Let $\Gamma_i = \left[\mathtt{A}\varphi\right], \Gamma$ for some $\Gamma$ and $\Gamma_j$ some other set inside $\mathcal{A}$. If $\sigma = s_0 \dots s_k$ is the selected finite sequence for this moment then $s_k \models \mathtt{A}\varphi$. But then $\pi \models \varphi$ for any $\pi$ starting with $s_k$, in particular those paths which have a sequence in $P_{\Gamma_j}(s_0)$. Thus, player $\exists$ can continue with the focus inside the actual $\Gamma$.

All in all, player $\exists$ can change focus s.t. the index of the actual sublist always gets increased. But this means she sets the focus to a $\Gamma$ which cannot regenerate the $\Gamma'$ the focus has been on before. Therefore a repeat is not possible as long as she changes focus appropriately.

Whenever the extended play visits a configuration $s_k \vdash C$ s.t. $s_j \vdash C$ was visited before and $s_k \neq s_j$ then the satisfiability game is restarted at the first occurrence of $C$ with the extended configuration $s_k \vdash C$. This does not influence the choice of the focus position since the $\Gamma$s inside $\mathcal{A}$ of $C$ are the same. Therefore player $\exists$ can choose the position of the focus according to the path $s_0 \ldots s_j \ldots s_k$. If a repeat on an extended configuration is detected such that the states are the same then the satisfiability game is not restarted. Note that in this case one of the winning conditions applies at most 3 steps later according to Lemma 163.

This strategy guarantees player $\exists$ to win $\mathcal{G}(\varphi_0)$. Player $\forall$ cannot win a play with his condition 1 since this would imply an inconsistency in the labelling of a state in the model. He also cannot win with condition 2 because Lemma 173 ensures that player $\exists$ does not change the focus back to a $\Gamma$ where it was before. Moreover, since every selected prefix of a path is guaranteed to be extendable to a path satisfying at least one $\Gamma$, player $\exists$ can "fulfil" every $\varphi \mathtt{U} \psi$ eventually. The satisfiability play cannot perform a repeat before player $\exists$ chooses $\psi$ in its unfolding since this would correspond to a selected prefix $s_0 \ldots s_k \ldots s_k$ such that this cyclic path does not satisfy $\psi$ at any state. ■

As in the cases of LTL, CTL and PDL, the small model property for CTL$^*$ can be derived from its satisfiability games.

**Theorem 178 (Small model property)** *If $\varphi_0 \in$ CTL$^*$ is satisfiable then it has a model of size less than* $2^{|\varphi_0|} \cdot 2^{(2^{|\varphi_0|})}$.

PROOF Suppose $\varphi_0$ is satisfiable. By Theorem 177, player $\exists$ has a winning strategy for the game $\mathcal{G}(\varphi_0)$. A transition system $\mathcal{T} = (\mathcal{S}, \rightarrow, L)$ can be extracted from the game tree in the same way as it is done for CTL in the proof of Theorem 116. States of $\mathcal{T}$ are equivalence classes of configurations, and transitions are given by applications of rules $(\mathtt{AX})$, $(\mathtt{EX})$ and $(\mathtt{AX}\not{E})$. The labelling of the states is taken from player ∃'s choices of the maximal consistent sets $\Pi$.

However, it is much simpler to consider the game tree of the extended game in the proof of Theorem 177. Ignoring the state components of the extended configurations results in a game tree for player ∃ and the game $\mathcal{G}(\varphi_0)$.

On the other hand, the state components alone form a transition system $\mathcal{T}$ with transitions given by applications of rules $(\mathtt{AX})$, $(\mathtt{EX})$ and $(\mathtt{AX}\not{E})$. The fact that they occur as state components in an extended game tree for player ∃ shows that $\mathcal{T}$ is a model for $\varphi_0$. Its size is bounded by $2^{|\varphi_0|} \cdot 2^{(2^{|\varphi_0|})}$ since this is the maximal number of configurations in the (extended) game tree according to Lemma 163. ■

**Corollary 179 (Tree model property)** *CTL$^*$ has the tree model property.*

**Theorem 180 (Winning strategies)**
*a) Player ∃'s winning strategies are history-free.*
*b) Player ∀'s winning strategies are LVR strategies.*

PROOF History-freeness of player ∃'s winning strategies is proved in the same way as it is for the LTL satisfiability games and, hence, for the CTL$^*$ model checking games. First of all, the strategy described in Theorem 177 requires her to choose a model for the underlying formula. This model does not depend on the history of a play. Then she follows the states of the extended configurations in the model and uses the outlined finite paths to put the focus onto a particular $\Gamma$ whenever player ∀ wants the focus to be inside $\mathcal{A}$.

Lemma 67 of Chapter 5 proved for the CTL$^*$ model checking games that player ∀ does not need to remember which $\Gamma$ he set the focus to. Instead, he can recalculate the sets

$$P_{\Gamma_1}(s),\ \ldots,\ P_{\Gamma_n}(s)$$

every time rule $(\mathtt{AX})$ or $(\mathtt{EX})$ is played. It is the fixed index ordering of the $\Gamma_i$s which ensures that he can change focus in such a way that indices only get increased. But the order of the formulas is chosen at the beginning, too, and does not depend on the history of a play. The same holds of course for player ∃ in this case, too.

On one hand, player $\forall$'s winning strategy for $\mathcal{G}(\varphi_0)$ with an unsatisfiable $\varphi_0$ tells him to preserve unsatisfiability. This is history-free since unsatisfiability of a configuration only depends on the configuration itself. On the other hand his strategy tells him how to set the focus. Similar to the satisfiability games for LTL, CTL and PDL from Chapter 6 he can use a list of $\mathtt{U}$ formulas.

The proof of Lemma 168 states that player $\forall$'s top-level list strategy is obtained as the interleaving of several priority list strategies according to Lemma 97 for the LTL satisfiability games for example. By Lemma 118, proved in Section 6.2, an interleaving of two disjoint LVRs is an LVR again. Note that the LVRs in this case consist of all configurations containing $\mathtt{U}$ formulas or, in a simplified version, of all $\mathtt{U}$ formulas only. Each $\mathtt{E}\Sigma$ and $\mathcal{A}$ of $\mathcal{G}(\varphi_0)$ has its own LVR of $\mathtt{U}$ formulas occurring in them. They can easily be made disjoint be marking $\mathtt{U}$ formulas according to which $\mathtt{E}\Sigma$ or $\mathcal{A}$ they come from. Thus, player $\forall$'s winning strategies are LVR strategies. ∎

## Complexity

**Theorem 181 (Complexity)** *Deciding the winner of* $\mathcal{G}(\varphi)$ *is in 2-EXPTIME.*

PROOF An alternating algorithm can be used to determine the winner of the satisfiability game $\mathcal{G}(\varphi)$. It simply needs to store three configurations: the actual one and two (co-)nondeterministically chosen ones to find a repeat on. The size of each configuration is exponential in the size of the input. The size of a set of subformulas of $\varphi$ is linear in the size of $\varphi$, and there can be exponentially many different sets of that kind.

Each time one of the players uses their focus change rule their stored configuration is deleted. Thus, if the play performs a repeat without focus change it is detected by the algorithm. To detect repeats with focus change the algorithm stores a counter to measure the length of a play. According to Lemma 163, its maximal size is

$$\log\left((1+|\varphi|+2^{|\varphi|})\cdot 2^{(2^{|\varphi|})}+5\right) \quad = \quad O(|\varphi|\cdot 2^{|\varphi|})$$

Thus, deciding the winner of $\mathcal{G}(\varphi)$ can be done in alternating EXPSPACE which is the same as 2-EXPTIME, [CKS81]. ∎

## Comparing Automata and Games for CTL$^*$ Satisfiability Checking

So far, automata have been the only successful tool that was used to automatically decide satisfiability of CTL$^*$ formulas. Since CTL$^*$ is a branching time logic, automata over trees are needed.

As with all the other logics, a CTL$^*$ formula $\varphi$ is translated into an automaton $\mathcal{A}_\varphi$, and satisfiability checking is reduced to the non-emptiness test $L(\mathcal{A}_\varphi) \neq \emptyset$. However, in a naive approach the size of the automaton is doubly exponential in the size of $\varphi$, and the non-emptiness test requires time which is double exponential in the size of the automaton.

As outlined in Section 3.2, this has been optimised by inspecting both the structure of CTL$^*$ formulas to obtain smaller automata, and by finding more efficient procedures for the non-emptiness test. They make use of the fact that automata arising from this translation are not arbitrary but have a special structure, too.

The first optimisation mainly deals with the complementation problem for these automata. Complementation is necessary to handle embedded quantifiers. Note that $\overline{\mathtt{E}\varphi} \equiv \mathtt{A}\overline{\varphi}$. Thus, an automaton for an existentially quantified subformula is obtained as the complement of an automaton for a universally quantified subformula.

But complementation generally requires determinisation, i.e. deterministic automata are easy to complement whereas nondeterministic are not. It is possible to complement non-deterministic automata without explicitly transforming them into an equivalent deterministic one. But this can also be seen as an implicit determinisation.

On the other hand, remember the close connection between Büchi automata and LTL formulas as well as the relationship between CTL$^*$ formulas and LTL formulas. Deterministic Büchi automata are strictly weaker than nondeterministic ones. Therefore, a translation into them is not even possible. The complementation problem for Büchi automata generally requires an intricate construction using combinatorial results.

This shows that the real work involved in automata-theoretic satisfiability checking for CTL$^*$ is done on top of the automaton. Consequently, the resulting automata that have undergone an optimised determinisation process bear no syntactic relationship to the original formula anymore.

This distinguishes them from the satisfiability games of this chapter completely. There, the whole complexity of the problem at hand is simply captured by the size of a configuration and, consequently, is expressed in the relatively high number of rules compared to LTL or CTL satisfiability checking games. On the other hand, the algorithm that decides which player has a winning strategy for a $\mathcal{G}(\varphi_0)$ is not more complicated than the ones used for LTL and CTL in Theorems 106 and 120.

As opposed to the automata-theoretic approach the games have the advantage of yielding a structure which does have a close relationship to the input formula. In fact, the game tree is entirely made up of its subformulas. In contrast, the automata for CTL$^*$ formulas need to be regarded as an abstract graph of which a certain reachability property needs to be checked.

# Chapter 9

# Model Checking Games for Fixed Point Logic with Chop

*Mad world! Mad kings!*
*Mad composition!*

—

PHILIP THE BASTARD

## 9.1 Global Model Checking Games for FLC

For a finite transition system $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with $s \in \mathcal{S}$ and an FLC formula $\varphi_0$ the *global model checking game* $\mathcal{G}_\mathcal{T}(s, \varphi_0)$ is played by players $\forall$ and $\exists$ on the game board

$$\mathcal{C} \;=\; \mathcal{S} \times 2^{\mathcal{S}} \times Sub(\varphi_0)$$

The intended meaning of a configuration $s, S \vdash \varphi$ is $s \in [\![\varphi]\!](S)$. Remember that the semantics of $\varphi$ is a function from sets of states to sets of states. This is the reason for the state *set* in a configuration.

Here, a play is a finite sequence $C_0, \ldots, C_n$ of configurations with $C_0 = s, \mathcal{S} \vdash \varphi_0$. This takes into account the definition of the $\models$ relation for FLC from Section 2.5.

The rules for the global model checking games are presented in Figure 9.1. Rules $(\wedge)$ and $(\vee)$ are like the rules for boolean connectives in the PDL model checking games for example. Rule $(\mathtt{FP})$ is the usual unfolding for fixed point formulas. Rule $(\mathtt{VAR})$ expresses the property of a fixed point being equivalent to its unfolding.

The most interesting rule is $(;)$. Here, player $\exists$ chooses a $T$ first. Then player $\forall$ chooses a $t \in T$ and decides whether to follow the left or the right of the lower configurations. The motivation for this is as follows. Let $s, S \vdash \varphi;\psi$ be the actual configuration. Note that its intended meaning is $s \in [\![\varphi]\!]([\![\psi]\!](S))$. To prove this, player $\exists$ has to name a set $T$ s.t. $T \subseteq [\![\psi]\!](S)$. Then, player $\forall$ who wants to show that

$$s \notin [\![\varphi]\!]([\![\psi]\!](S))$$

has two possibilities to do so. Either he shows $s \notin [\![\varphi]\!](T)$. In this case, player $\exists$'s choice of $T$ was not good enough. The best she can do is $T = [\![\psi]\!](S)$.

On the other hand he can decide to refute player $\exists$'s claim that $T \subseteq [\![\psi]\!](S)$ in which case he has to name a $t \in T$ of which he believes $t \notin [\![\psi]\!](S)$. Therefore the play can continue with either $s, T \vdash \varphi$ or $t, S \vdash \psi$.

Note that there are no restrictions on the choice of $T \subseteq \mathcal{S}$. This is the point where the games violate the locality conditions stated in Section 2.8.

Player $\exists$ is allowed to choose $T = \emptyset$. In this case player $\forall$ has no choice with this rule and the next configuration is necessarily $s, \emptyset \vdash \varphi$. This is justified by the fact that trivially $\emptyset \subseteq [\![\psi]\!](S)$ for any $\psi$ and any $S$. Thus, player $\forall$ could not refute this branch anyway.

Player $\forall$ wins the play $C_0, \ldots, C_n$ of the game $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ iff

1. $C_n \ = \ t, T \vdash q$ and $q \notin L(t)$, or
2. $C_n \ = \ t, T \vdash \tau$ and $t \notin T$, or
3. $C_n \ = \ t, T \vdash \langle a \rangle$ and for all $t' \in \mathcal{S}$ : if $t \xrightarrow{a} t'$ then $t \notin T$, or

$$(\wedge)\ \frac{s,S \vdash \varphi_0 \wedge \varphi_1}{s,S \vdash \varphi_i}\ \forall i \qquad (\vee)\ \frac{s,S \vdash \varphi_0 \vee \varphi_1}{s,S \vdash \varphi_i}\ \exists i$$

$$(\mathtt{FP})\ \frac{s,S \vdash \sigma Z.\varphi}{s,S \vdash Z} \qquad (\mathtt{VAR})\ \frac{s,S \vdash Z}{s,S \vdash \varphi}\ \text{if } fp(Z) = \sigma Z.\varphi$$

$$(;)\ \frac{s,S \vdash \varphi;\psi}{s,T \vdash \varphi \quad |_\forall \quad t,S \vdash \psi}\ \exists T \subseteq \mathcal{S}, \forall t \in T$$

Figure 9.1: The rules for the global FLC model checking games.

4. $C_n = t,T \vdash [a]$ and there is $t' \in \mathcal{S}$ s.t. $t \xrightarrow{a} t'$ and $t' \notin T$, or
5. $C_n = t,T' \vdash Y$ s.t. $fp(Y) = \mu Y.\psi$ for some $\psi$, and there is an $i \in \mathbb{N}$, s.t.
   - $C_i = t,T \vdash Y$ with $T' \subseteq T$, and
   - there is no $C_j$ with $i < j < n$ s.t. $C_j = t',S' \vdash Z$ for some $t',S'$ with $fp(Z) = \nu Z.\psi'$ and $Y <_\varphi Z$.

Player $\exists$ wins the play $C_0,\dots,C_n$ of $\mathcal{G}_\mathcal{T}(s,\varphi)$ iff

6. $C_n = t,T \vdash q$ and $q \in L(t)$, or
7. $C_n = t,T \vdash \tau$ and $t \in T$, or
8. $C_n = t,T \vdash \langle a \rangle$ and there is a $t' \in T$ s.t. $t \xrightarrow{a} t'$, or
9. $C_n = t,T \vdash [a]$ and for all $t' \in \mathcal{S}$ : if $t \xrightarrow{a} t'$ then $t \in T$, or
10. $C_n = t,T' \vdash Y$ s.t. $fp(Y) = \nu Y.\psi$ for some $\psi$, and there is an $i \in \mathbb{N}$, s.t.
    - $C_i = t,T \vdash Y$ with $T' \supseteq T$, and
    - there is no $C_j$ with $i < j < n$ s.t. $C_j = t',S' \vdash Z$ for some $t',S'$ with $fp(Z) = \mu Z.\psi'$ and $Y <_\varphi Z$.

**Example 182** Let $\mathcal{T}$ be the transition system consisting of states $\{s,t\}$ with transitions $s \xrightarrow{a} t$, $t \xrightarrow{a} t$ and $t \xrightarrow{b} s$. Consider the FLC formula

$$\varphi \;=\; \nu Y.[b]\mathtt{ff} \wedge [a](\nu Z.[b] \wedge [a](Z;Z));(([a]\mathtt{ff} \wedge [b]\mathtt{ff}) \vee Y)$$

from Example 22. $\varphi$ requires the number of $b$'s that have been seen on every path never to exceed the number of $a$s that have been seen so far. State $s$ of $\mathcal{T}$ satisfies $\varphi$. The full game tree for player $\exists$ is shown in Figure 9.2. Let $\psi := \nu Z.[b] \wedge [a](Z;Z)$ and $\delta := ([a]\mathtt{ff} \wedge [b]\mathtt{ff}) \vee Y$.

She wins all the plays which end on a $[a]$ or $[b]$ with her winning condition 9. She wins the other plays with condition 10 since the only fixed point formulas occurring in $\varphi$ are greatest ones.

## Correctness

**Fact 183** *Rule* (VAR) *is the only rule that increases the size of the formula in the actual configuration. All other rules decrease it.*

**Lemma 184** *Every play of* $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ *has a uniquely determined winner.*

PROOF Suppose the play at hand reaches an atomic formula. Then no further game rule applies. The winner is uniquely determined since winning conditions 1 – 4 and 6 – 9 cover these cases and are mutually exclusive.

Now take a play that does not reach an atomic formula. According to Fact 183, all the game rules apart from (VAR) decrease the size of the formula component in the actual configuration. Therefore, there must be at least one variable $Z$ which gets replaced by its defining fixed point formula each time it occurs in the actual configuration. Since the underlying transition system is finite a configuration must eventually be reached such that the first part of condition 5 or 10 is fulfilled. The second part will also be fulfilled eventually since for every variable $Z$ that does not regenerate itself there must be another variable $Y$ such that $Z <_\varphi Y$. But there are only finitely many variables in a formula. Thus, one must be outermost. Its fixed point type determines whether condition 5 or 10 applies. ∎

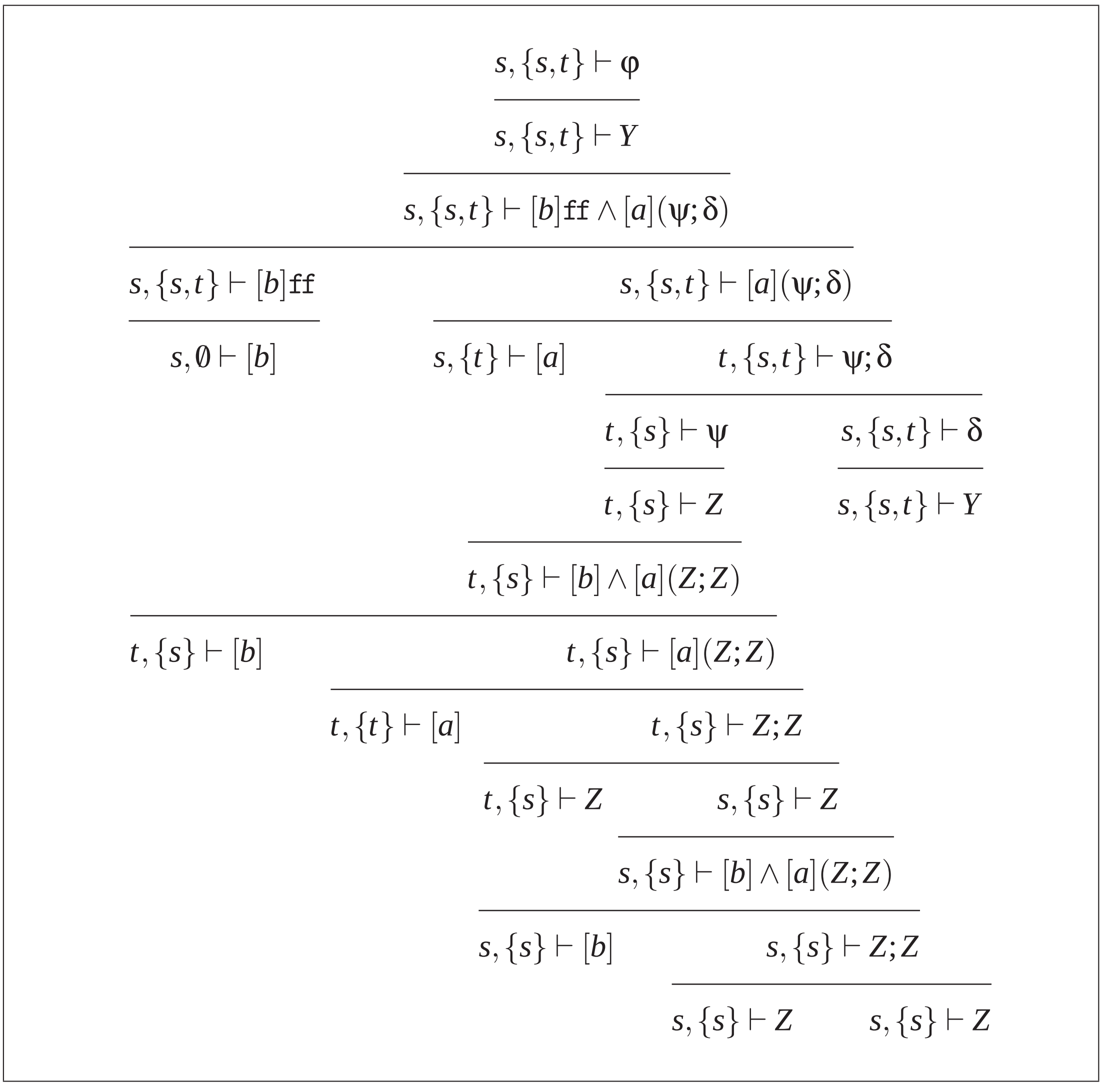


Figure 9.2: The game tree for player ∃ from Example 182.

**Lemma 185** *Every play of* $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ *has length at most* $(|\mathcal{S}| \cdot 2^{|\mathcal{S}|} \cdot |\varphi|)^{ad(\varphi)+1}$.

PROOF This upper bound on the length of a play is proved by induction on the fixed point depth of $\varphi$. Suppose $ad(\varphi) = 0$. Then the requirement of being outermost in winning conditions 5 and 10 becomes void. In a configuration $t,T \vdash \psi$ there are $|\mathcal{S}|$ many possibilities for $t$ and $|\varphi|$ many for $\psi$. There are $2^{|\mathcal{S}|}$ many possibilities to choose subsets $T_1,\ldots,T_n$ of $\mathcal{S}$ s.t. that $T_i \not\subseteq T_j$, resp. $T_i \not\supseteq T_j$ for all $1 \leq i < j \leq n$.

Suppose now $k := ad(\varphi) > 0$. Let $Z$ be the outermost variable with $fp(Z) = \sigma Z.\psi$. Then $Z$ can be unfolded at most $|\mathcal{S}| \cdot 2^{|\mathcal{S}|} \cdot |Sub(\varphi)|$ times. Each unfolding can result in an embedded subplay starting with $\psi$ which has fixed point depth $k-1$. ■

The next result follows from Lemmas 184, 185 and Theorem 37.

**Corollary 186 (Determinacy)** *Player* $\forall$ *wins* $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ *iff player* $\exists$ *does not win* $\mathcal{G}_{\mathcal{T}}(s,\varphi)$.

Generally, correctness proofs for model checking games split up into two parts: soundness and completeness. According to Theorem 39 of Section 2.6, one part can easily be used to prove the other if

- the logic is closed under negation, i.e. for every $\varphi$ there is a $\overline{\varphi}$ s.t. $s \models \varphi$ iff $s \not\models \overline{\varphi}$, and
- the rules and winning conditions of the games are dual in the sense that player $p$ can use $\overline{p}$'s winning strategy from $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ to win $\mathcal{G}_{\mathcal{T}}(s,\overline{\varphi})$.

This is given for the PDL model checking games in Chapter 4 and the CTL$^*$ model checking games in Section 5.2.

For the FLC model checking games we have to prove both soundness and completeness explicitly. The reason is the requirement of being closed under negation. Remember that the $\models$ relation for FLC formulas is defined indirectly via the semantics $[\![\cdot]\!]$. Therefore the right complement formula $\overline{\varphi}$ satisfies

$$[\![\overline{\varphi}]\!](S) \quad = \quad \mathcal{S} - [\![\varphi]\!](S) \quad \text{for all } S \subseteq \mathcal{S}$$

But according to this definition of complementation, FLC is not negation closed. A simple counterexample is the formula $\tau$ which has no complement. This follows from Lemma 19 of Section 2.5 which states that the semantics of any FLC formula is a monotone state transformer. But

$$[\![\overline{\tau}]\!] \;=\; \lambda X.\mathcal{S} - X \tag{9.1}$$

is not monotone. Also, it is not obviously clear how to express $\overline{\varphi;\psi}$ with the operators of FLC.

One solution to this problem would be to extend the syntax and semantics of FLC. In fact, it would suffice to add $\overline{\tau}$ as a new primitive. Then the complement of a formula $\varphi$ can be defined as $\overline{\varphi} := \overline{\tau};\varphi$. But functions $\lambda f.[\![\varphi(X)]\!]_{[X\mapsto f]}$ need to be monotone for fixed points over these functions to exist. A simple criterion like the one used for $\mathcal{L}_\mu$ formulas where variables are required to occur in the scope of an even number of negations does not seem to exist for FLC. Note that because of (9.1) the following holds.

$$\overline{\varphi;\psi} \quad\equiv\quad \overline{\varphi};\psi$$

Thus, an occurrence of a variable in $\psi$ on the right side of this has to take the negation of $\varphi$ into account as well. This means the *scope* of a negation symbol is not its subtree in the formula's syntax tree anymore.

The second way to repair negation closure of FLC is to consider complementation with respect to weak equivalence. This means any $\psi$ is a candidate for $\overline{\varphi}$ if for every state $s$ of every transition system $s \models \varphi$ iff $s \not\models \psi$ holds. But there must be an effective way to construct $\psi$ from $\varphi$ to meet the second requirement above. One possibility is to use deMorgan's laws, the duality of least and greatest fixed points, the complementation closure of atomic propositions and the duality of the modal operators to eliminate negation from formulas. But then $\tau$ has to be eliminated too, which can destroy the original structure of the formula at hand. This will result in games on $\varphi$ and $\overline{\varphi}$ that are not obviously dual to each other anymore.

**Definition 187** A configuration $t, T \vdash \psi$ in the game $\mathcal{G}_{\mathcal{T}}(s,\varphi_0)$ is called *true* if $t \in [\![\psi]\!](T)$ and *false* otherwise.

**Lemma 188** *Player $\exists$ preserves falsity and can preserve truth with her choices. Player $\forall$ preserves truth and can preserve falsity with his choices.*

PROOF First consider rule $(\vee)$. Take a configuration

$$C \;=\; t,T \vdash \varphi_0 \vee \varphi_1$$

Suppose $C$ is false, i.e. $t \not\in [\![\varphi_0]\!](T)$ and $t \not\in [\![\varphi_1]\!](T)$. Regardless of which $i$ player $\exists$ chooses, the configuration $t,T \vdash \varphi_i$ will be false. On the other hand, suppose $C$ is true. Then $t \in [\![\varphi_0]\!](T)$ or $t \in [\![\varphi_1]\!](T)$, and player $\exists$ can preserve truth by choosing $i$ accordingly. The proof for rule $(\wedge)$ where player $\forall$ makes a choice is dual.

Consider now a configuration

$$C \;=\; t,T \vdash \varphi;\psi$$

Suppose $C$ is true, i.e. $t \in [\![\varphi]\!]([\![\psi]\!](T))$. Then player $\exists$ can choose $T' = [\![\psi]\!](T)$ and the configuration $t,T' \vdash \varphi$ will be true. Moreover, for every $t' \in T'$ the configuration $t',T \vdash \psi$ will be true, too. Therefore, player $\forall$ must preserve truth with his choice in rule $(;)$.

Suppose on the other hand that $C$ is false, i.e.

$$t \;\not\in\; [\![\varphi]\!]([\![\psi]\!](T))$$

In the application of rule $(;)$, player $\exists$ chooses $T'$ first. Assume she chooses $T' = [\![\psi]\!](T)$. Then player $\forall$ can continue with the false configuration $t,T' \vdash \varphi$. Assume therefore, player $\exists$ chooses any other $T'$ s.t. $t \in [\![\varphi]\!](T')$. Then there must be a $t' \in T'$ s.t. $t' \not\in [\![\psi]\!](T)$ and player $\forall$ can continue with the false configuration $t',T \vdash \psi$.

To prove this last claim assume that $T' \subseteq [\![\psi]\!](T)$. By monotonicity of $[\![\varphi]\!]$ we have

$$[\![\varphi]\!](T') \;\subseteq\; [\![\varphi]\!]([\![\psi]\!](T))$$

But then $t \in [\![\varphi]\!](T')$ contradicts the assumption that $t \not\in [\![\varphi]\!]([\![\psi]\!](T))$.

Note that both truth and falsity are preserved by application of the deterministic rules $(\texttt{FP})$ and $(\texttt{VAR})$ if variables are interpreted by their approximants. ■

For the correctness proofs it is helpful to consider a more flexible definition of a game. In order to do so we will, overloading notation, denote by $s, S \vdash \varphi$ the game starting with the configuration $s, S \vdash \varphi$. An underlying transition system is implicitly assumed so that player $\exists$ knows which set of states to choose from. The model checking game $\mathcal{G}_{\mathcal{T}}(s, \varphi_0)$ in the original sense is simply the same as the game for $s, \mathcal{S} \vdash \varphi_0$.

**Theorem 189 (Soundness)** *Player $\forall$ wins the game for $s, S \vdash \varphi_0$ if $s \notin [\![\varphi_0]\!](S)$.*

PROOF Suppose $s \notin [\![\varphi_0]\!](S)$, i.e. the configuration $s, S \vdash \varphi_0$ is false. Preserving falsity in the sense of Lemma 188, we will construct a game tree for player $\forall$. Whenever rules $(\vee)$, $(\wedge)$ or $(;)$ need to be played, continue with the false configurations as in the proof of Lemma 188. Rules $(\mathtt{FP})$ and $(\mathtt{VAR})$ are applied deterministically.

This way, the game tree cannot contain a play which is won by player $\exists$ with her winning condition 6,7,8 or 9. These conditions require the last configuration of the play to be true which is excluded by the preservation of falsity.

It remains to be shown that player $\exists$ cannot win a play with her winning condition 10 either. In order to do so we interpret $\nu$-variables by their approximants. Suppose the construction of the game tree reaches a configuration

$$C \;=\; t, T \vdash \nu Z.\psi$$

By preservation of falsity $C$ is false as well as the following configuration $t, T \vdash Z$. There we interpret $Z$ as the least approximant that makes it false, i.e. as the $Z^k$ s.t.

$$t \;\notin\; [\![Z^k]\!](T) \qquad \text{but} \qquad t \;\in\; [\![Z^{k-1}]\!](T)$$

By the definition of approximants $k = 0$ is impossible. $k \in \mathbb{N}$ since the underlying transition system is assumed to be finite.

Note that the game rules follow the syntactic structure of formulas and that $Z^k$ is defined as $\psi[Z^{k-1}/Z]$. This means that the next time a configuration

$$C' \;=\; s', S' \vdash Z$$

is reached, $Z$ can be interpreted as $Z^{k-1}$ to make $C'$ false. This does not hold if the computation of $\nu Z.\psi$ has been restarted in the meantime, i.e. a least fixed point variable $Y$ has been visited in between, s.t. $Z \leq_{\varphi_0} Y$.

Suppose now the construction of the game tree reaches a configuration $C' = t, T' \vdash Z$, s.t. $C$ and $C'$ fulfill the requirements of winning condition 10. Then there must have been at least one unfolding of $Z$ with rule $(\mathtt{VAR})$ between $C$ and $C'$, and there is no $\mu$-variable $Y$ on this path such that $Z <_{\varphi_0} Y$. Therefore, in the false configuration $C'$, $Z$ will be interpreted as $Z^m$ with $m < k$. But if $t \notin [\![Z^m]\!](T')$ and $T' \supseteq T$ then $t \notin [\![Z^m]\!](T)$ by monotonicity.

From this we conclude that there is no least $k$ that makes $t, T \vdash Z^k$ false. By Theorem 30 of Section 2.5

$$t, T \vdash \nu Z.\psi$$

could not have been false either without contradicting the assumption. Thus, player $\exists$ cannot win a single play in the game tree constructed in this way and, by Corollary 186, player $\forall$ wins the game for $s, S \vdash \varphi_0$. ∎

**Theorem 190 (Completeness)** *Player* $\exists$ *wins the game for* $s, S \vdash \varphi_0$ *if* $s \in [\![\varphi_0]\!](S)$.

PROOF Suppose $s \in [\![\varphi_0]\!](S)$, i.e. the configuration $s, S \vdash \varphi_0$ is true. In a similar way to the proof of Theorem 189, we will construct a game tree for player $\exists$ preserving truth. Starting with configuration $s, S \vdash \varphi_0$, whenever rules $(\vee)$, $(\wedge)$ or $(;)$ need to be played, continue with the true successor configurations as described in the proof of Lemma 188. Again, rules $(\mathtt{FP})$ and $(\mathtt{VAR})$ are applied deterministically.

This way, the game tree cannot contain a play which is won by player $\forall$ with his winning condition 1, 2 or 3. These conditions require the last configuration of the play to be false which is impossible by the preservation of truth.

It remains to be shown that player $\forall$ cannot win a play with her winning condition 4 either. This time we interpret $\mu$-variables by their approximants. Suppose the construction of the game tree reaches a configuration

$$C \;=\; t, T \vdash \mu Y.\psi$$

By preservation of truth $C$ is true as well as the following configuration $t, T \vdash Y$ where $Y$ is interpreted as the least approximant that makes it true, i.e. as the $Y^k$ s.t.

$$t \;\in\; [\![Y^k]\!](T) \qquad \text{but} \qquad t \;\notin\; [\![Y^{k-1}]\!](T)$$

Again, by the definition of the approximants $k = 0$ is impossible.

With the same argument as used in the proof of Theorem 189, the next time a configuration $C' = s', S' \vdash Y$ is reached, $Y$ can be interpreted as $Y^{k-1}$ to make $C'$ true. This holds of course only if no greatest fixed point variable $Z$ has been visited in between, s.t. $Y \leq_{\varphi_0} Z$.

Suppose now the construction of the game tree reaches a configuration $C' = t, T' \vdash Y$, s.t. $C$ and $C'$ fulfill the requirements of winning condition 4. Then there must have been at least one unfolding of $Y$ with rule $(\mathtt{VAR})$ between $C$ and $C'$, and $Y$ is the outermost variable on this path. Therefore, in the true configuration $C'$, $Y$ will be interpreted as $Y^m$ with $m < k$. But if $t \in [\![Y^m]\!](T')$ and $T' \subseteq T$ then $t \in [\![Y^m]\!](T)$ by monotonicity.

From this we conclude that there is no least $k$ that makes $t, T \vdash Y^k$ true. By Theorem 30, $t, T \vdash \nu Y.\psi$ could not have been true either without contradicting the assumption. Thus, player $\forall$ cannot win a single play in the game tree constructed in this way and, by Corollary 186, player $\exists$ wins the game for $s, S \vdash \varphi_0$. ■

Remember that preservation of truth and falsity plays an important role in the winning strategies of the PDL model checking games. Here, the situation is similar. However, unlike the least fixed point formulas in FLC, their PDL counterparts exhibit a very simple structure. There is only one path through their syntax trees that leads from a least fixed point construct $\langle\alpha^*\rangle\psi$ via the subformula relation back to itself. This is the reason why it is sufficient in the PDL case to choose the smaller formula of two that both preserve truth. The same holds for greatest fixed point constructs and falsity of course.

The syntax trees of FLC formulas however can have several different loops. Moreover, the explicit use of propositional variables rules out the possibility of simply choosing the smaller of two options since variables are smallest formulas. The following formula is equivalent to the PDL property $\langle a^*\rangle\langle a\rangle\mathtt{tt}$ used in Example 47.

$$\mu Y.\langle a\rangle;Y \vee \langle a\rangle;\mathtt{tt};Y$$

Note that the $Y$ in the right disjunct has no effect and could be left out without changing the semantics of the formula. However, it shows that even a criterion which considers the occurrences of variables in disjuncts, resp. conjuncts, does not suffice. It might, however, work for $\mathcal{L}_\mu$ formulas. Instead, it is necessary for both players to use approximants as done in the proofs of Theorems 189 and 190.

**Theorem 191 (Winning strategies)** *The winning strategies for the global FLC model checking games are history-free.*

PROOF Player $\forall$'s winning strategies for these games consist of preserving falsity with his choices and annotating variables with their respective approximant indices. Then, he cannot postpone showing falsity of a greatest fixed point formula infinitely often since his task simply is to avoid a formula $Z^0$ in a play if $fp(Z) = \nu Z.\psi$ for some $\psi$.

But falsity and approximant indices only depend on the actual configuration, in particular on the state components of the actual configuration and not on the history of a play.

The case for player $\exists$ who preserves truth and attempts to avoid a $Y^0$ if $fp(Z) = \mu Z.\psi$ for some $\psi$, is dual. Thus, her winning strategies are history-free, too. ∎

## Complexity

**Theorem 192 (Complexity)** *Deciding the winner of a global FLC model checking game is in EXPTIME.*

PROOF Let $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ be the game at hand. An alternating algorithm can determine the winner using polynomial space only. As in the proofs of Theorems 120 and 134, we let the algorithm store the actual configuration, one for each player to recognise a repeat in the sense of winning conditions 5 and 10, and a counter to stop a play that has not found a repeat. The size of each configuration is polynomial in the size of the input, and so is the space needed for the counter according to Lemma 184.

Furthermore, the algorithm needs to store a flag $f \in \{\mu, \nu\}$ to indicate the fixed point type of the greatest variable w.r.t. $<_\varphi$ that occurred after a configuration was stored. This flag is also used to indicate whether a stored configuration can be overwritten if the variable in it cannot be outermost in the play at hand anymore.

Again, alternating PSPACE is the same as EXPTIME, [CKS81]. ∎

**Corollary 193 (Complexity)** *Deciding the winner of a global FLC$^k$ model checking game is in PSPACE for every $k \in \mathbb{N}$.*

PROOF The same algorithm as in the proof of Theorem 192 can be used in this case. However, here we analyse the time the alternating algorithm needs. This is proportional to the length of a play. Lemma 184 shows that the maximal length of a play is polynomial in the size of the input if the alternation depth of the input formula is fixed. But APTIME = PSPACE according to [CKS81]. ■

Global model checking games for FLC over infinite transition systems would result in game trees with infinite out-degree even if the underlying transition system has finite out-degree. The reason for this is the unrestricted choice player $\exists$ has in rule $(;)$. Therefore we resist the urge to amend the definition of the global games to capture infinite-state transition systems as well.

## 9.2 Local Model Checking Games for FLC

The *local model checking game* $\mathcal{G}_{\mathcal{T}}(s, \varphi_0)$ is played on an LTS $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with $s \in \mathcal{S}$ and an FLC formula $\varphi_0$. Here we do not restrict ourselves to finite transition systems only. Player $\exists$ tries to establish that $s$ satisfies $\varphi_0$, whereas $\forall$ tries to show that $s \not\models \varphi_0$.

A play is a (possibly infinite) sequence $C_0, C_1, \ldots$ of configurations, and a configuration is an element of

$$\mathcal{C} \quad = \quad \mathcal{S} \times Sub(\varphi)^* \times Sub(\varphi)$$

It is written $s, \delta \vdash \psi$ where $\delta$ is interpreted as a stack of subformulas with its top on the left. The empty stack is denoted by $\varepsilon$. With a stack $\delta = \varphi_0 \ldots \varphi_k$ we associate the eponymous formula $\delta := \varphi_0; \ldots; \varphi_k$ while $\varepsilon$ is associated with the formula $\tau$.

The intended meaning of a configuration $t, \delta \vdash \psi$ is: $t \in [\![\psi]\!]([\![\delta]\!](\mathcal{S}))$. Thus, the stack $\delta$ plays the role of the state set component in a global FLC model checking game. Note that this condition is equivalent to $t \in [\![\psi; \delta]\!](\mathcal{S})$.

$$(\vee)\ \frac{s,\delta \vdash \varphi_0 \vee \varphi_1}{s,\delta \vdash \varphi_i}\ \exists i \qquad (\wedge)\ \frac{s,\delta \vdash \varphi_0 \wedge \varphi_1}{s,\delta \vdash \varphi_i}\ \forall i$$

$$(\mathtt{FP})\ \frac{s,\delta \vdash \sigma Z.\varphi}{s,\delta \vdash Z} \qquad (\mathtt{VAR})\ \frac{s,\delta \vdash Z}{s,\delta \vdash \varphi}\ \text{ if } fp(Z) = \sigma Z.\varphi$$

$$(;)\ \frac{s,\delta \vdash \varphi_0;\varphi_1}{s,\varphi_1\delta \vdash \varphi_0} \qquad (\tau)\ \frac{s,\psi\delta \vdash \tau}{s,\delta \vdash \psi}$$

$$(\langle a\rangle)\ \frac{s,\psi\delta \vdash \langle a\rangle}{t,\delta \vdash \psi}\ \exists s \xrightarrow{a} t \qquad ([a])\ \frac{s,\psi\delta \vdash [a]}{t,\delta \vdash \psi}\ \forall s \xrightarrow{a} t$$

Figure 9.3: The rules for the local FLC model checking games.

Each play of $\mathcal{G}_{\mathcal{T}}(s_0,\varphi_0)$ begins with

$$C_0 \;=\; s_0,\varepsilon \vdash \varphi_0$$

A play proceeds according to the rules given in Figure 9.3. Rules $(\vee)$ and $(\wedge)$ are straightforward. Rules $(\mathtt{VAR})$ and $(\mathtt{FP})$ are justified by the unfolding characterisations of fixed points: $\sigma Z.\varphi \equiv \varphi[\sigma Z.\varphi/Z]$. If a formula $\varphi;\psi$ is encountered $\psi$ is stored on the stack with rule $(;)$ to be dealt with later on while the players try to prove resp. refute $\varphi$. Modalities cause either of the players to choose a successor state. After that, rules $(\langle a\rangle)$ and $([a])$ pop the top formula from the stack into the right side of the actual configuration. Rule $(\tau)$ does the same without a choice by one of the players. In both cases the last formula on the right-hand side has been proved and the next thing to do is to prove, resp. refute, those formulas that have been collected on the stack.

**Definition 194** Recall the tail $tl_Z$ of a variable $Z$ from Definition 18 of Section 2.5. A variable $Z$ is called *stack-increasing* in a play $C_0, C_1, \ldots$ if there are infinitely many configurations $C_{i_0}, C_{i_1}, \ldots$, s.t.

- $i_j < i_{j+1}$ for all $j \in \mathbb{N}$
- $C_{i_j} = s_j, \delta_j \vdash Z$ for some $s_j$ and $\delta_j$,
- for all $j \in \mathbb{N}$ exists $\gamma \in tl_Z$ s.t. $\delta_{j+1} = \gamma\delta_j$, where $\delta = \tau\delta$ for example.

Player $\forall$ wins the play $C_0, C_1, \ldots$ of $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ iff

1. there is an $n \in \mathbb{N}$ s.t. $C_n \;=\; t, \delta \vdash q$ and $q \notin L(t)$, or
2. there is an $n \in \mathbb{N}$ s.t. $C_n \;=\; t, \delta \vdash \langle a \rangle$ and $t \stackrel{a}{\not\to}$, or
3. the play is infinite, and there is a $Y$ that is the greatest, w.r.t. $<_\varphi$, stack-increasing variable and $fp(Y) = \mu Y.\psi$ for some $\psi$.

Player $\exists$ wins the play $C_0, C_1, \ldots$ of $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ iff

4. there is an $n \in \mathbb{N}$ s.t. $C_n \;=\; t, \delta \vdash q$ and $q \in L(t)$, or
5. there is an $n \in \mathbb{N}$ s.t. $C_n \;=\; t, \varepsilon \vdash \tau$, or
6. there is an $n \in \mathbb{N}$ s.t. $C_n \;=\; t, \varepsilon \vdash \langle a \rangle$ and there is a $t \in \mathcal{S}$ with $t \xrightarrow{a} t'$, or
7. there is an $n \in \mathbb{N}$ s.t. $C_n \;=\; t, \delta \vdash [a]$, and $\delta = \varepsilon$ or $t \stackrel{a}{\not\to}$, or
8. the play is infinite, and there is a $Z$ that is the greatest, w.r.t. $<_\varphi$, stack-increasing variable and $fp(Z) = \nu Z.\psi$ for some $\psi$.

Winning conditions 1 and 4 suggest that game rule $(;)$ can be refined. Whenever the formula to be put on the stack is a $q \in Prop$ then the existing stack can be discarded.

$$\frac{s, \delta \vdash \varphi; q}{s, q \vdash \varphi}$$

This does not effect the worst-case complexities, therefore we merely mention this optimisation.

The following example illustrates the importance of being stack-increasing. Note that in a $\mathcal{L}_\mu$ model checking game the winner is determined by the outermost variable

$$
\begin{array}{c}
s, \varepsilon \vdash \mu Y.\langle b \rangle \vee \langle a \rangle \nu Z.Y;Z;Y \\ \hline
s, \varepsilon \vdash Y \\ \hline
s, \varepsilon \vdash \langle b \rangle \vee \langle a \rangle \nu Z.Y;Z;Y \\ \hline
s, \varepsilon \vdash \langle a \rangle \nu Z.Y;Z;Y \\ \hline
s, \nu Z.Y;Z;Y \vdash \langle a \rangle \\ \hline
t, \varepsilon \vdash \nu Z.Y;Z;Y \\ \hline
t, \varepsilon \vdash Z \\ \hline
t, \varepsilon \vdash Y;Z;Y \\ \hline
t, Z;Y \vdash Y \\ \hline
t, Z;Y \vdash \langle b \rangle \vee \langle a \rangle \nu Z.Y;Z;Y \\ \hline
t, Z;Y \vdash \langle b \rangle \\ \hline
t, Y \vdash Z \\ \hline
t, Y \vdash Y;Z;Y \\ \hline
t, Z;Y;Y \vdash Y \\ \hline
\vdots
\end{array}
$$

Figure 9.4: Player $\exists$'s winning play of Example 195.

that occurs infinitely often. There, if two variables $Y$ and $Z$ occur infinitely often and $Y <_\varphi Z$ for example, then $fp(Y)$ occurs infinitely often, too. Thus, two occurrences of $Y$ cannot be related to each other in terms of their approximants. FLC only has this property for stack-increasing variables. But note also that according to Definition 194 every variable of a $\mathcal{L}_\mu$ formula that gets unfolded infinitely often in a play is stack-increasing.

**Example 195** Take the formula

$$\varphi \quad = \quad \mu Y.\langle b \rangle \vee \langle a \rangle \nu Z.Y;Z;Y$$

$ad(\varphi) = 1$ and $sd(\varphi) = 2$. Let $\mathcal{T}$ be the transition system consisting of states $\{s,t\}$ and transitions $s \xrightarrow{a} t$ and $t \xrightarrow{b} t$. $s \models \varphi$. The game tree for player $\exists$ is shown in Figure 9.4.

Since $\varphi$ does not contain any $\wedge$, $[a]$ or $[b]$, player $\forall$ does not make any choices and the tree is in fact a single play.

Both $Y$ and $Z$ occur infinitely often in the play. However, neither $fp(Y)$ nor $fp(Z)$ does. Note that $Z <_\varphi Y$. $Y$ gets "fulfilled" each time it is replaced by its defining fixed point formula, but reproduced by $Z$. On the other hand, $Y$ does not start a new computation of $fp(Z)$ each time it is reproduced. But $Y$ is not stack-increasing whereas $Z$ is. And $Z$ denotes a greatest fixed point, therefore player $\exists$ wins this play.

## Correctness

Before we can prove soundness and completeness of the games we need a few technical lemmas. Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$, $s \in \mathcal{S}$, $\varphi \in$ FLC, and $C = s, \delta \vdash \psi$ be a configuration in a game for $s$ and $\varphi$. As usual, $C$ is called *true* if $s \in [\![\varphi]\!]([\![\delta]\!](\mathcal{S}))$, and *false* otherwise.

**Lemma 196** *Player $\exists$ preserves falsity and can preserve truth with her choices. Player $\forall$ preserves truth and can preserve falsity with his choices.*

PROOF The cases of disjunctions and conjunctions are similar to those of Lemma 188. Consider a configuration

$$C \;=\; s, \psi\delta \vdash \langle a \rangle$$

If $C$ is true then there is a $t$ s.t. $s \xrightarrow{a} t$ and $t \in [\![\psi;\delta]\!](\mathcal{S})$. By choosing this $t$, player $\exists$ can make the next configuration $t, \delta \vdash \psi$ true. If $C$ is false then there is no such $t$ and regardless of which transition $\exists$ chooses the following configuration will be false, too.

The proofs of the other cases are dual or similar to preservation of truth and falsity for the global model checking games in Lemma 188.

Note that the rules which do not require a player to make a choice preserve both truth and falsity if variables are interpreted via their approximants. ∎

**Lemma 197** *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$, $s \in \mathcal{S}$, $\varphi \in FLC$. In an infinite play $C_0, C_1, \ldots$ for $s$ and $\varphi$ there is a unique greatest, with respect to $<_\varphi$, stack-increasing variable $Z$.*

PROOF Note that a finite play trivially cannot have a stack-increasing variable. Let the play at hand $C_0, C_1, \ldots$ be infinite. Suppose first there are two stack-increasing variables $Z$ and $Y$. Then there must be two configurations

$$C_i = s, \delta \vdash Z \qquad \text{and} \qquad C_j = t, \delta' \vdash Y$$

with $i < j$. Either $Y$ has been generated from the unfolding of $Z$ in which case one of them is greater than the other. The reason is that the stack only contains elements of $tl_V$ for some variable $V$ up to a fixed part at its bottom which is never popped. But $Y \in tl_Z$ implies either $Y$ is free in $fp(Z)$ or $fp(Y) \in Sub(fp(Z))$. Therefore they must be comparable.

Suppose $\delta = \delta_0 Y \delta_1$. But then $Z$ has either been generated from the unfolding of $Y$ and they are comparable or $\delta' = \delta'_0 Z \delta'_1$. At every configuration the stack can only hold a finite number of variables. Therefore, in such an infinite play it is not possible that neither of the variables generates the other one infinitely often, and they must be comparable.

It remains to be shown that at least one variable is stack-increasing. There must be a variable $Z$ that occurs infinitely often. Moreover, this $Z$ must generate itself infinitely often. Let $fp(Z) = \sigma Z.\varphi$. This means that for every occurrence of $Z$ in a $C_i = s, \delta \vdash Z$, when $Z$ is replaced by $\varphi$, the play must follow the syntactical structure of $\varphi$ to one occurrence of $Z$ in $\varphi$. In order to pop an element from $\delta$ an atomic formula in $\varphi$ must have been reached, and $Z$ in $C_i$ did not regenerate itself. Suppose it did and the stack has been increased. Since rule (VAR) replaces a variable $Z$ with its defining fixed point formula $\varphi$ the additional part of the stack must consist of subformulas of $\varphi$ only. Moreover, every subformula that occurs "before" $Z$ in $\varphi$ must have been removed from the stack before $Z$ can be reached again. Therefore, the extension of the stack must be an element of $tl_Z$. ■

One important property of an outermost stack-increasing variable is: If its occurrence in a configuration $s, \delta \vdash Z$ is interpreted as the approximant $Z^\alpha$ then in its next occurrence $Z$ will denote $Z^{\alpha-1}$. This is because $Z$ is outermost in the play at hand and the second occurrence stems from the first, i.e. the play has followed the syntactical structure of $fp(Z)$ between these occurrences. Thus the computation of $fp(Z)$ does not get restarted.

**Fact 198** *Rules* $(\vee)$, $(\wedge)$, $(\mathtt{FP})$, $(\tau)$, $(\langle a\rangle)$ *and* $([a])$ *decrease the size of the actual configuration. Rule* $(\mathtt{VAR})$ *increases it. Rule* $(;)$ *maintains its size.*

**Lemma 199** *Every play of* $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ *has a uniquely determined winner.*

PROOF Suppose the play reaches a configuration to which no rule can be applied. This is either because a proposition has been reached. But then either player $\forall$ wins with winning condition 1 or player $\exists$ wins with condition 4.

The other possibility to get stuck is to reach a configuration $t,\delta \vdash \langle a\rangle$ or $t,\delta \vdash [a]$ with $t \stackrel{a}{\not\rightarrow}$. In the first case player $\forall$ wins with condition 2. In the second case player $\exists$ wins with condition 7.

Finally, the stack can become empty and the last formula on the right side is atomic. If it is a $\tau$ then player $\exists$ wins with condition 5, with condition 6 if it is a $\langle a\rangle$ and with condition 7 if it is a $[a]$.

If it never reaches such a configuration then it must be of infinite length. According to Lemma 197, there is a unique outermost stack-increasing variable that determines the winner with condition 3 or 8. ∎

Again, in order to prove soundness and completeness we generalise the notion of an FLC model checking game. Overloading notation we let $s,\delta \vdash \varphi$ also denote the game that starts with this configuration. Then, $\mathcal{G}_{\mathcal{T}}(s,\varphi)$ is equivalent to $s,\varepsilon \vdash \varphi$ where $s$ is a state of $\mathcal{T}$.

**Theorem 200 (Soundness)** *Let* $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ *with* $s \in \mathcal{S}$ *and* $\varphi, \delta_0 \in FLC$. *If* $s \notin [\![\varphi;\delta_0]\!](\mathcal{S})$ *then player* $\forall$ *wins* $s,\delta_0 \vdash \varphi$.

PROOF Suppose $s \notin [\![\varphi]\!]([\![\delta_0]\!](\mathcal{S}))$. We construct a (possibly infinite) game tree for $\forall$ starting with $s,\delta_0 \vdash \varphi$. If $\varphi = \varphi_0 \wedge \varphi_1$, $\forall$ chooses the $\varphi_i$ that makes $s,\delta \vdash \varphi_i$ false. If $\varphi = \varphi_0 \vee \varphi_1$ then the game tree is extended with both false configurations $s,\delta \vdash \varphi_i$. Similar arguments hold for the applications of rules $(\langle a\rangle)$, $([a])$, and $(\tau)$. Since falsity is preserved no finite path can be won by player $\exists$ since a false leaf implies that $\forall$ is the winner of that particular play.

The game tree can be constructed such that player $\exists$ cannot win an infinite play either. Let its construction reach a configuration

$$t,\delta \;\vdash\; \nu Z.\psi$$

s.t. $Z$ is the unique stack-increasing variable according to Lemma 197. In the following configuration $t,\delta \vdash Z$, $Z$ is interpreted as the least approximant $Z^\alpha$ s.t.

$$t \;\notin\; [\![Z^\alpha]\!]([\![\delta]\!](\mathcal{S})) \qquad \text{but} \qquad t \;\in\; [\![Z^{\alpha-1}]\!]([\![\delta]\!](\mathcal{S}))$$

Note that $\alpha$ cannot be a limit ordinal $\lambda$ since $t \notin [\![\bigwedge_{\beta<\lambda} Z^\beta]\!](S)$ for any $S \subseteq \mathcal{S}$ implies $t \notin [\![Z^\beta]\!](S)$ for some $\beta < \lambda$. The next time a configuration $t',\delta' \vdash Z$ is reached $Z$ is consequently interpreted as $Z^{\alpha-1}$. Again, if $\alpha - 1$ is a limit ordinal $\lambda$, then there must be a $\beta < \lambda$ such that

$$t' \;\notin\; [\![Z^\beta]\!]([\![\delta']\!](\mathcal{S}))$$

But ordinals are well-founded, i.e. the play must eventually reach a false configuration $t'',\delta'' \vdash Z$ in which $Z$ is interpreted as $Z^0$. But $Z^0 \equiv \mathtt{tt}$ and $t'' \notin [\![\mathtt{tt}]\!](S)$ is not possible for any $S \subseteq \mathcal{S}$. We conclude that there is no least $\alpha$ that makes $t,\delta \vdash Z^\alpha$ false and, by Theorem 30, that therefore $t,\delta \vdash \nu Z.\psi$ could not have been false either.

Since player $\exists$ cannot win any play in the game tree that is constructed in the described way player $\forall$ must win the game on $s,\delta_0 \vdash \varphi$. ■

**Theorem 201 (Completeness)** *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with $s \in \mathcal{S}$ and $\varphi, \delta_0 \in$ FLC. If $s \in [\![\varphi;\delta_0]\!](\mathcal{S})$ then player $\exists$ wins $s,\delta_0 \vdash \varphi$.*

PROOF This is dual to the proof of Theorem 200. Assuming $s \in [\![\varphi]\!]([\![\delta_0]\!](\mathcal{S}))$ we build a game tree for player $\exists$ starting with the true configuration $s,\delta_0 \vdash \varphi$ whilst preserving truth. If the construction of the game tree reaches a leaf the corresponding play must be won by $\exists$ since only she wins a finite play that ends in a true configuration.

Again, we show that player $\forall$ cannot win an infinite play either. Suppose there is a configuration $t,\delta \vdash \mu Y.\psi$ with $Y$ being stack-increasing and outermost according to

Lemma 197. In the next step, $Y$ is interpreted as the least approximant $Y^\alpha$ s.t.

$$t \in [\![Y^\alpha]\!]([\![\delta]\!](\mathcal{S})) \qquad \text{but} \qquad t \notin [\![Y^{\alpha-1}]\!]([\![\delta]\!](\mathcal{S}))$$

Again, $\alpha$ cannot be a limit ordinal. The next time a configuration $t', \delta' \vdash Y$ is reached it becomes true if $Y$ is interpreted as $Y^{\alpha-1}$. If $\alpha - 1$ is a limit ordinal then there is a smaller one that makes the configuration true.

Because of well-foundedness of the ordinals every infinite play must reach a configuration $t'', \delta'' \vdash Y$ in which $Y$ is interpreted as $Y^0$. But $Y^0 \equiv \mathtt{ff}$ and therefore $t'', \delta'' \vdash Y$ cannot be true. Thus, $t, \delta \vdash \mu Y.\psi$ could not have been true either.

Since player $\forall$ cannot win any play of the game tree that is constructed in the described way player $\exists$ must win the game starting with $s, \delta_0 \vdash \varphi$. ■

From Theorems 200 and 201 follows that the model checking problem for FLC can be rephrased as: $s \models \varphi$ iff player $\exists$ wins $s, \varepsilon \vdash \varphi$.

**Corollary 202 (Determinacy)** *Player $\forall$ wins $\mathcal{G}_\mathcal{T}(s, \varphi)$ iff player $\exists$ does not win $\mathcal{G}_\mathcal{T}(s, \varphi)$.*

The next theorem is proved in the same way as the history-freeness of winning strategies for the global model checking games, see Theorem 191. Note that, again, winning strategies consist of preserving truth, resp. falsity, and using approximant indices.

**Theorem 203 (Winning strategies)** *The winning strategies for the local FLC model checking games are history-free.*

## Complexity

**Theorem 204 (Complexity)** *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ be finite with $s \in \mathcal{S}$ and $\varphi, \delta \in FLC^{k,n}$. Deciding the winner of $s, \delta \vdash \varphi$ is in PSPACE for all $k, n \in \mathbb{N}$.*

PROOF We can assume $\delta = \varepsilon$ since the game for $s, \delta \vdash \varphi$ is equivalent to the game for $s, \varepsilon \vdash \varphi; \delta$. Note that $\varphi; \delta$ has fixed alternation and sequential depth, too.

If the underlying transition system is finite then the least approximants used in the proofs of Theorems 200 and 201 are bounded by $|\mathbb{S}|$ according to Lemma 30. An algorithm deciding the winner of $s, \varepsilon \vdash \varphi$ can index variables occurring in a play as the corresponding approximant. This means, rules $(\mathtt{FP})$ and $(\mathtt{VAR})$ are used as

$$\frac{s,\delta \vdash \sigma Z.\varphi}{s,\delta \vdash Z^{|\mathbb{S}|}} \qquad \text{and} \qquad \frac{s,\delta \vdash Z^k}{s,\delta \vdash \varphi[Z^{k-1}/Z]} \quad \text{if } fp(Z) = \sigma Z.\varphi$$

Then, configurations of the form $t, \delta \vdash Z^0$ with $fp(Z) = \sigma Z.\psi$ for some $\psi, \delta$ and $t$ are winning for player $\exists$ if $\sigma = \nu$ and winning for player $\forall$ if $\sigma = \mu$. Infinite plays are ruled out.

Next we analyse the maximal length of a play of $s, \varepsilon \vdash \varphi$. Suppose $ad(\varphi) = 0$. At most $|\mathbb{S}| \cdot |\varphi|$ steps are possible before a terminal configuration with a $Z^0$ must be reached, if the sequential depth of $\varphi$ is 1. However, if it is greater than 0 then at the beginning a $Z^{|\mathbb{S}|}$ can be pushed onto the stack where it remains while another $Z^k$ gets unfolded at most $|\mathbb{S}|$ times before it might disappear. Then the $Z^{|\mathbb{S}|}$ from the stack can be popped and create the same situation by unfolding to more than one $Z^{|\mathbb{S}|-1}$ of which one remains on the stack again. Generally, $(|\mathbb{S}| \cdot |\varphi|)^{sd(\varphi)}$ is the maximal length of a play in this situation.

Let now $ad(\varphi) = k > 0$. Take the outermost variable $Z$ that occurs in the play at hand. With each unfolding it can start a subplay on a formula with alternation depth $k-1$. Therefore the overall maximum length of the play is

$$((|\mathbb{S}| \cdot |\varphi|)^{sd(\varphi)})^{ad(\varphi)+1} \;=\; (|\mathbb{S}| \cdot |\varphi|)^{O(sd(\varphi) \cdot ad(\varphi))}$$

An alternating algorithm can decide the winner of $s, \varepsilon \vdash \varphi$ by simply playing the game for it. For input formulas $\delta, \varphi \in \mathrm{FLC}^{k,n}$ the alternation depth and sequential depth are bounded. Thus, the time needed is polynomial in the size of the formula and the size of the transition system. According to [CKS81] there is a deterministic procedure that needs space which is polynomial in the size of the formula and in the size of the transition system. ■

This argument, applied to formulas of arbitrary alternation or sequential depth, yields an EXPSPACE procedure. This follows from the fact that the alternating algorithm needs time exponential in the alternation and sequential depth of the input formula, and AEXPTIME = EXPSPACE. To show that game-based model checking for FLC can be done in EXPTIME an alternating algorithm must not use more than polynomial space. Equally, a single play must be playable using polynomially bounded space.

We will leave it as an open question where there exists a local model checking procedure for FLC which runs in exponential time. However, we illustrate the problem of finding an EXPTIME procedure. First we consider a slightly different way of proving soundness and completeness of the games which only applies if the underlying transition system is finite. Remember that in the proofs of Theorems 200 and 201 variables are interpreted as approximants, and contradictions arise at configurations $t, \delta' \vdash Z^0$. Suppose $fp(Z) = \mu Z.\psi$ and the game tree is constructed preserving truth. Then at its first occurrence $Z$ is interpreted as the least $Z^k$ which makes the configuration, say, $t, \delta' \vdash Z^k$ true. However, if later another true configuration $t, \delta'' \vdash Z^j$ is seen and $[\![\delta']\!](\mathcal{S}) \subseteq [\![\delta'']\!](\mathcal{S})$ then this already contradicts the fact that $k$ was chosen least. Compare this to the winning conditions of the global FLC model checking games.

This occurs trivially after $|\mathcal{S}| \cdot 2^{|\mathcal{S}|} \cdot |\varphi|$ steps since there are only $|\mathcal{S}|$ many different states and $2^{|\mathcal{S}|}$ many different sets of them. In most cases this situation will occur in a stack of polynomial size already. However, there are cases in which the stack can grow super-polynomially. This means there are $m$ configurations $s_i, \delta_i \vdash Z$ s.t.

$$[\![\delta_i]\!](\mathcal{S}) \quad \not\subseteq \quad [\![\delta_j]\!](\mathcal{S})$$

for $j < i \leq m$ and $m$ is not polynomially bounded by the input size.

**Example 205** Let $a, b \in \mathcal{A}$. Take $n$ pairwise different prime numbers $p_1, \ldots, p_n$. Let $P_0 = 0$ and $P_i = \sum_{j=1}^{i} p_j$ be the sum of the first $i$ prime numbers for $i = 1, \ldots, n-1$. We construct a transition system $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with $\mathcal{S} = \{0, \ldots, P_n - 1\}$. Transitions in $\mathcal{T}$ are given by

$$j \xrightarrow{a} (j+1) \quad \text{for all } j < P_n, \text{ s.t. } j \neq P_i - 1 \text{ for all } i \in \{1, \ldots, n\}$$

and

$$(P_i - 1) \xrightarrow{a} P_{i-1} \quad \text{for all } i \in \{1, \ldots, n\}$$

Finally, $i \xrightarrow{b} j$ iff $j \xrightarrow{a} i$. $\mathcal{T}$ consists of $n$ cycles of length $p_1, \ldots, p_n$ which can be traversed along $a$-transitions, say, clockwise and through $b$-transitions counterclockwise. Feel free to add as many $c$-transitions if $c \neq a$ and $c \neq b$ to make $\mathcal{T}$ connected. Finally, we use one proposition $q$ which holds on one state of each cycle only.

$$q \in L(j) \quad \text{iff} \quad j = P_i \text{ for some } i \in \{0, \ldots, n-1\}$$

The formula under examination is

$$\varphi \;\; := \;\; (\nu Z.\tau \wedge \langle a \rangle Z \langle b \rangle); q$$

It says that there is an infinite $a$-path s.t. after every sequence of $n$ $a$'s another $n$ $b$'s can be made to a state which satisfies $q$. $0 \models \varphi$ which can also be seen using the games of this section. Player $\forall$ can never choose $\tau$ since $0 \models q$ and every sequence of $m$ $a$-transitions away from 0 leads to a state that can do $m$ $b$-transitions back to 0. But then player $\exists$ wins because the play repeats on a $\nu$-variable. Her game tree is shown in Figure 9.5.

If approximants are used explicitly as suggested in the proof of Theorem 204, the stack cannot grow larger than $P_n$. This is not surprising since $\varphi \in \mathrm{FLC}^{0,1}$. However, let

$$S_q \;\; := \;\; \{P_i \mid i \in \{0, \ldots, n-1\}\}$$

be the set of all states satisfying $q$. We claim that

$$[\![\underbrace{\langle b \rangle \ldots \langle b \rangle}_{i \text{ times}}]\!](S_q) \;\; \neq \;\; [\![\underbrace{\langle b \rangle \ldots \langle b \rangle}_{j \text{ times}}]\!](S_q) \quad \text{for } i, j < \prod_{i=1}^{n} p_i, \;\; i \neq j$$

and even

$$[\![\underbrace{\langle b \rangle \ldots \langle b \rangle}_{i \text{ times}}]\!](S_q) \;\; \not\subseteq \;\; [\![\underbrace{\langle b \rangle \ldots \langle b \rangle}_{j \text{ times}}]\!](S_q) \quad \text{for } i, j < \prod_{i=1}^{n} p_i, \;\; i \neq j$$

because

$$|[\![\underbrace{\langle b \rangle \ldots \langle b \rangle}_{i \text{ times}}]\!](S_q)| \;\; = \;\; n \quad \text{for all } i < \prod_{i=1}^{n} p_i$$

Take a state in the $k$-th cycle. It belongs to

$$[\![\underbrace{\langle b\rangle\ldots\langle b\rangle}_{i \text{ times}}]\!](S_q)$$

iff it is the $(i \bmod p_k)$-th $b$-predecessor of $P_{k-1}$. In other words, the sets

$$[\![\underbrace{\langle b\rangle\ldots\langle b\rangle}_{i \text{ times}}]\!](S_q)$$

can be defined by moving markers along $a$-transitions in each cycle starting with $S_q$. Since the lengths of the cycles are pairwise different prime numbers the same set is only marked after $\prod_{i=1}^{n} p_i$ steps.

This means that the stacks

$$\underbrace{\langle b\rangle\ldots\langle b\rangle}_{i \text{ times}};q$$

with $i+1$ elements, $1 \leq i < (\prod_{j=1}^{n} p_j)$, define pairwise incomparable sets of states. Note that $\prod_{j=1}^{n} p_j \notin O(n^k)$ for any $k \in \mathbb{N}$.

**Corollary 206 (Complexity)** *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ be finite with $s \in \mathcal{S}$ and $\varphi, \delta \in FLC$. Deciding the winner of $s, \delta \vdash \varphi$ is in EXPSPACE.*

The next theorem analyses the complexity of the games if applied to $\mathcal{L}_\mu$ formulas. In this case it is helpful to start the game with an empty stack.

**Theorem 207 (Complexity)** *Let $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ be finite with $s \in \mathcal{S}$ and $\varphi \in FLC^-$. Deciding the winner of $s, \varepsilon \vdash \varphi$ is in NP∩co-NP.*

PROOF The stack can never grow larger than $\varphi$ and will be empty each time a variable is reached. The resulting games are essentially the same as the model checking games for $\mathcal{L}_\mu$ from [Sti95]. It is known from [EJS01] for example that the winner of those games can be decided in NP∩co-NP. The same technique applies here.

The game graph for $s, \varepsilon \vdash \varphi$ is finite and of size polynomial in the input. To decide whether player $\exists$ wins $s, \varepsilon \vdash \varphi$ a nondeterministic algorithm can guess annotations $(k_1, \ldots, k_n)$ for each $\mu$-variable $Y$. The meaning of such an annotation is: $Y$ has to be

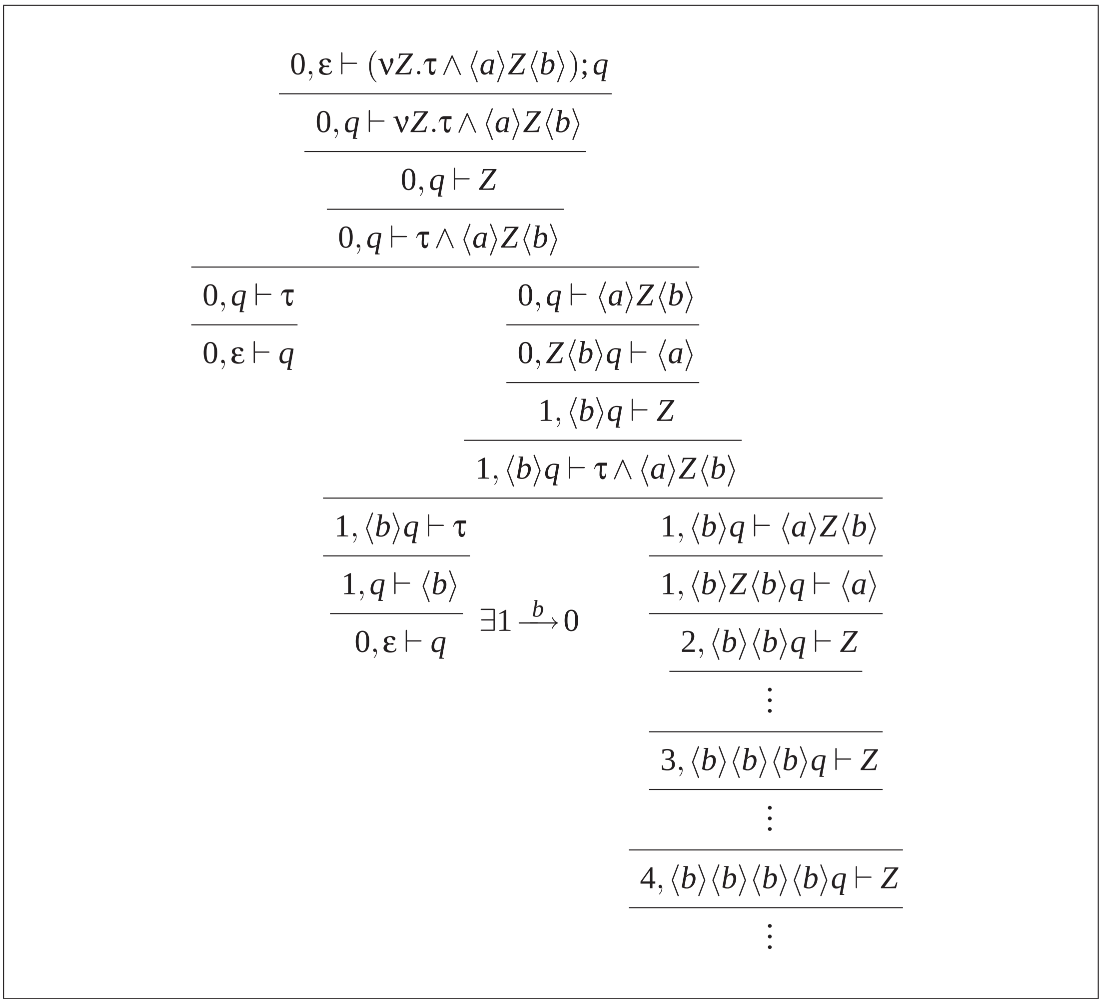

Figure 9.5: Player ∃'s game tree of Example 205.

unfolded $k_n$ times at this moment and there are outer variables $Z_1, \dots, Z_{n-1}$ of type $\nu$ which have to be unfolded $k_1, \dots, k_{n-1}$ times. The maximal size of such an annotation is $O(ad(\varphi) \cdot log|\mathbb{S}|)$.

Finally, the algorithm has to verify that the order of the annotations is well-founded, i.e. for every $\mu$-variable $Y$: if there is a path from $s, \delta \vdash Y$ with annotation $\overline{k} = (k_1, \dots, k_n)$ to $t, \delta' \vdash Y$ with annotation $\overline{k}' = (k'_1, \dots, k'_n)$ then $\overline{k}'$ is lexicographically smaller than $\overline{k}$.

This proves that deciding the winner of $s, \varepsilon \vdash \varphi$ is in NP. Inclusion in co-NP follows from the fact that the same argument applies to player $\forall$ and $\nu$-variables to decide whether he wins $s, \varepsilon \vdash \varphi$. ∎

This is not a contradiction to the PSPACE-hardness of FLC model checking proved in [LS02a]. There, reductions from the validity problem for QBF and from the universal acceptance problem for NFAs are presented. The latter is courtesy of Müller-Olm. In both cases the constructed formulas are not in $\mathrm{FLC}^-$.

Even if the starting stack in the game of Theorem 207 is non-empty, the semantics of approximants will always be evaluated on the same set of states. However, if the stack is $\delta = \psi\delta'$ and deciding the winner of $t, \delta' \vdash \psi$ is in the complexity class $\mathcal{C}$ for any $t \in \mathcal{S}$, then deciding the winner of $s, \delta \vdash \varphi$ is in (NP$\cap$co-NP)$\cup\mathcal{C}$.

Theorem 207 becomes interesting if applied to formulas in $\mathrm{FLC}^-$ that are not a translation of a $\mathcal{L}_\mu$ formula but are equivalent to a formula in $\mathcal{L}_\mu$. One example is

$$\nu Z.(\langle a_0\rangle \wedge \langle b_0\rangle);\ldots;(\langle a_0\rangle \wedge \langle b_0\rangle);Z$$

which is exponentially more succinct than its equivalent in $\mathcal{L}_\mu$, see [LS02a].

# Chapter 10

# Further Research

*Smokey my friend, you're*
*entering a world of pain.*

—

WALTER SOBCHAK

## Extensions of PDL

We have shown how to extend the PDL model checking games in order to handle variations like PDL with the repeat construct or converse modalities. Another variant of PDL that has attracted some attention because of its relationship to description logics is PDL *with intersection*, PDL-$\cap$, [Dan84]. There, programs can contain an operator $\alpha \cap \beta$ with the following semantics.

$$s \xrightarrow{\alpha\cap\beta} t \quad \text{iff} \quad s \xrightarrow{\alpha} t \ \text{ and } \ s \xrightarrow{\beta} t$$

It is not obvious how to extend the PDL model checking games in order to handle this operator, too.

PDL with intersection is known to be decidable. However, a direct decision procedure has not been given yet. It remains to be seen whether focus games in the style of Chapter 6 can be used to decide this logic. It also remains to be seen whether such focus games yield an axiomatisation of PDL with intersection.

One of the problems that comes with this logic is the loss of bisimulation invariance. PDL-$\cap$ can distinguish bisimilar models.

**Example 208** Let $\mathcal{T}_1$ be the transition system consisting of states $s, t_1, t_2$ with transitions $s \xrightarrow{a} t_1$ and $s \xrightarrow{b} t_2$. Let $\mathcal{T}_2$ arise from $\mathcal{T}_1$ by collapsing states $t_1$ and $t_2$. Clearly, $\mathcal{T}_1$ and $\mathcal{T}_2$ are bisimilar. However, take the PDL-$\cap$ formula

$$\varphi \;=\; \langle a \cap b \rangle \mathtt{tt}$$

$\mathcal{T}_1, s \not\models \varphi$ whereas $\mathcal{T}_2, s \models \varphi$.

This also comes with a loss of the tree model property depending on whether $\mathcal{T}_2$ is still considered to be a tree. The satisfiability games of Chapters 6 and 8 seem to work well because of the tree model properties the considered logics have. This is why a model for a satisfiable formula can easily be extracted from a game tree for player $\exists$. In the case of PDL with intersection the game structure might have to be a graph rather than a tree.

## Logics with Past Operators

Another extension of PDL that the focus approach might be applicable to is PDL with converse operators. In general, it remains to be seen whether focus games can decide the satisfiability problem of modal and temporal logics with past operators and yield complete axiomatisations for them.

In this setting it makes sense to distinguish LTL with Past from CTL and PDL with their respective past or converse operators. In the linear time framework, forwards and backwards operators cancel each other out, i.e.

$$\mathtt{XY}\varphi \;\equiv\; \mathtt{YX}\varphi \;\equiv\; \varphi \tag{10.1}$$

where $\mathtt{Y}$ is the *previous* operator which behaves like $\mathtt{X}$ for the past. Its semantics is defined as

$$\pi^i \models \mathtt{Y}\psi \qquad \text{iff} \qquad \pi^{i-1} \models \psi$$

In a satisfiability game with past operators one cannot simply discard formulas that speak about the present moment and make a step towards the next future moment with a rule like $(\mathtt{X})$. They need to be preserved since formulas speaking about the future can contain formulas speaking about the future's past which can also be the present's past.

However, because of (10.1) it is not possible for a satisfiability play to create arbitrarily large formulas that alternatingly speak about the past and the future. Another way of seeing this is LTL with Past's *separation theorem*: every formula can be transformed into a boolean conjunction of three formulas, each speaking about the past, the present and the future respectively, [Gab89].

In the branching time setting, arbitrarily large formulas can indeed be created. This is reflected in general inequivalences of the form

$$\langle a \rangle \langle \overline{a} \rangle \varphi \;\not\equiv\; \varphi$$

However, some formulas can speak about the past and influence the present, like the validity

$$\models \langle \overline{a} \rangle [a] \varphi \rightarrow \varphi$$

Thus, satisfiability games for these logics need to carry much more information around than the games for LTL with Past. This can potentially result in an infinite set of configurations.

### Logics without Until Operators

Instead of extending logics and asking whether the focus idea is still applicable, it is also possible to restrict logics and consider syntactic fragments in order to obtain complete axiomatisations for them. One example is LTL or CTL without $\mathtt{U}$ and $\mathtt{R}$ but $\mathtt{F}$ and $\mathtt{G}$ instead. Clearly, the satisfiability games from Sections 6.1 and 6.2 still work for these fragments. However, the axiomatisations in Sections 7.1 and 7.3 heavily depend on the presence of an $\mathtt{U}$.

It remains to be seen whether there are different forms of Lemma 136 and 144 that allow player $\exists$ to strengthen $\mathtt{F}$ formulas s.t. a repeat on such a formula is only possible if the input formula is unsatisfiable. Note that the strengthening of $\mathtt{F}\varphi$ with a $\psi$, considered as an abbreviation of an $\mathtt{U}$, becomes

$$\neg\psi\mathtt{U}(\varphi \wedge \neg\psi)$$

which is not expressible in LTL without $\mathtt{U}$, [Kam68].

## First-Order Temporal Logics

*First-Order Temporal Logics* feature constructs of both temporal logics and predicate calculi. There, the propositional part of a temporal formula is replaced by a fragment of First-Order logic, see [Eme90] for an introduction. In general, these logics are undecidable, but depending on which fragment of First-Order Logic is used, the resulting logic might be decidable.

As with their propositional counterparts one distinguishes linear and branching time logics. For an overview over decidable fragments in general see [HWZ00] and [HWZ02] for branching time logics in particular.

It would be interesting to see whether the elegance of the focus approach for propositional temporal logics carries over to decidable predicate temporal logics as well. Moreover, if it does it also remains to be seen whether this yields complete axiomatisations in a relatively simple way, too.

## A Complete Axiomatisation for CTL$^*$

The most obvious piece of further work based on this thesis is the extraction of a complete axiomatisation from the CTL$^*$ satisfiability games in Chapter 8. The existence of a complete axiom system for CTL$^*$ had been an open question for approximately 15 years until recently. However, the completeness proof in [Rey01] is rather intricate and long. There is reason to believe that the focus games for satisfiability of CTL$^*$ formulas would yield a much shorter proof of CTL$^*$'s completeness. It remains to be seen what the right strengthening lemma for CTL$^*$ along the lines of Lemmas 136, 144 and 151 would be. Furthermore, parts of the soundness proof of the games (Theorem 171) need to be formalised as CTL$^*$ axioms, in particular Lemma 170.

## Other modal logics

The *Linear Time $\mu$-Calculus* $\mu$-LIN is $\mathcal{L}_\mu$'s counterpart interpreted over linear structures only, [Sti92]. A model checking procedure for $\mu$-LIN was given in [BEM96] for example. Similar to LTL, the fact that the underlying transition system is only built state-by-state requires the use of sets of formulas. Since $\mu$-LIN features explicit fixed point operators it is reasonable to ask whether focus games can provide an elegant characterisation of $\mu$-LIN's model checking problem.

It also remains to be seen whether satisfiability games for $\mu$-LIN can be defined along the same lines as Section 6.1 and whether a complete axiomatisation can easily be extracted from them.

Another way of getting beyond the restricted expressive power of $\mathcal{L}_\mu$ is by using fixed point constructs other than just least and greatest. [DGK01] defined MIC, the *Modal Iteration Calculus*, which extends multi-modal logic with simultaneous inflationary fixed points. There, the semantics of a formula $\varphi(X)$ is not required to be monotone in $X$ anymore. Hence, it can feature negation. Approximants for inflationary fixed points are defined as

$$X^0 := \emptyset\,, \qquad X^{\alpha+1} := X^\alpha \cup [\![\varphi(X)]\!]_{[X \mapsto X^\alpha]}\,, \qquad X^\lambda := \bigcup_{\alpha<\lambda} X^\alpha$$

Thus, the chain of approximants is always increasing. The inflationary fixed point is found when this chain becomes stationary.

Since a variable $X$ is allowed to occur negatively in a $\varphi(X)$, formulas of MIC are even less easy to understand than $\mathcal{L}_\mu$ formulas. Therefore, a game-based account of MIC's model checking problem could help to make MIC more usable.

There is no point in trying to define satisfiability games for MIC since it is undecidable as shown in [DGK01].

### Satisfiability Games for $\mathcal{L}_\mu$

Another interesting issue this thesis has not touched at all is a game-based account of the satisfiability problem for $\mathcal{L}_\mu$. A tableau-based decision procedure that uses games to determine successful branches has been given in [NW97]. An axiomatisation came already with the introduction of $\mathcal{L}_\mu$ in [Koz83] and was finally proved to be complete in a series of papers, [Wal93, Wal95, Wal96], using these tableaux.

Comparable to the situation for CTL$^*$ it might be desirable to have a simpler and more intuitive proof of $\mathcal{L}_\mu$'s completeness. Moreover, it would be interesting to see whether the focus game approach works for $\mathcal{L}_\mu$ as well and how it might have to be tailored to this specific logic.

One of the problems with focus games for $\mathcal{L}_\mu$ is the fact that variables can occur more than once in a fixed point formula. Thus, there are several different ways through the syntax tree of a formula that lead to a variable. We will illustrate this with an example.

**Example 209** Let $\mathcal{A} = \{a,b\}$. Take the $\mathcal{L}_\mu$ formula

$$\varphi \;=\; \mu Y.\nu Z.(\langle a\rangle Z \vee \langle b\rangle Z) \wedge [a]Y \wedge [b]Y$$

It stipulates the existence of an infinite path labelled with *a*s or *b*s, such that from any state on this path onwards only finitely many *a*s and *b*s are possible. Clearly, $\varphi$ is unsatisfiable. Suppose satisfiability games for $\mathcal{L}_\mu$ are defined along the lines of Chapter 6, i.e. configurations are sets of formulas which are interpreted conjunctively, and player $\forall$ controls a focus. Then player $\exists$ can win $\mathcal{G}(\varphi)$ in the way that is shown in Figure 10.1.

Her strategy is the following: if player $\forall$ sets the focus to $[a]Y$ then choose the disjunct $\langle b\rangle Z$ and vice versa. Note that player $\forall$ has to put the focus onto one of the $[-]$-formulas because only they contain a least fixed point variable. This way, player $\exists$ forces him to change focus in order to keep the play going once it reaches a configuration in which all formulas begin with a modal operator.

Thus, if a repeat occurs player $\forall$ will have changed focus and therefore should lose. But none of the least fixed point formulas became fulfilled. Hence, player $\exists$ should indeed have lost the game.

$$\begin{array}{c}
\left[\mu Y.\nu Z.(\langle a\rangle Z \vee \langle b\rangle Z) \wedge [a]Y \wedge [b]Y\right] \\ \hline
\left[Y\right] \\ \hline
\left[\nu Z.(\langle a\rangle Z \vee \langle b\rangle Z) \wedge [a]Y \wedge [b]Y\right] \\ \hline
\left[Z\right] \\ \hline
\left[(\langle a\rangle Z \vee \langle b\rangle Z) \wedge [a]Y \wedge [b]Y\right]
\end{array}$$

$$\begin{array}{cc}
\langle a\rangle Z \vee \langle b\rangle Z, \left[[a]Y\right], [b]Y & \langle a\rangle Z \vee \langle b\rangle Z, [a]Y, \left[[b]Y\right] \\ \hline
\langle b\rangle Z, \left[[a]Y\right], [b]Y & \langle a\rangle Z, [a]Y, \left[[b]Y\right] \\ \hline
\langle b\rangle Z, [a]Y, \left[[b]Y\right] & \langle a\rangle Z, \left[[a]Y\right], [b]Y \\ \hline
Z, \left[Y\right] & Z, \left[Y\right] \\ \hline
Z, \left[\nu Z.(\langle a\rangle Z \vee \langle b\rangle Z) \wedge [a]Y \wedge [b]Y\right] & Z, \left[\nu Z.(\langle a\rangle Z \vee \langle b\rangle Z) \wedge [a]Y \wedge [b]Y\right] \\ \hline
\left[Z\right] & \left[Z\right] \\ \hline
\left[(\langle a\rangle Z \vee \langle b\rangle Z) \wedge [a]Y \wedge [b]Y\right] & \left[(\langle a\rangle Z \vee \langle b\rangle Z) \wedge [a]Y \wedge [b]Y\right] \\ \hline
\langle a\rangle Z \vee \langle b\rangle Z, \left[[a]Y\right], [b]Y & \langle a\rangle Z \vee \langle b\rangle Z, [a]Y, \left[[b]Y\right] \\ \hline
\langle b\rangle Z, \left[[a]Y\right], [b]Y & \langle a\rangle Z, [a]Y, \left[[b]Y\right] \\ \hline
\langle b\rangle Z, [a]Y, \left[[b]Y\right] & \langle a\rangle Z, \left[[a]Y\right], [b]Y \\ \hline
Z, \left[Y\right] & Z, \left[Y\right]
\end{array}$$

Figure 10.1: A sketch of player $\exists$'s game tree for Example 209.

# Index

# Bibliography


[Abr97] S. Abramsky. Game semantics for programming languages (abstract). In I. Prívara and P. Ruzicka, editors, *Proc. 22nd Symp. on Math. Foundations of Computer Science, MFCS'97*, volume 1295 of *LNCS*, pages 3–4, Bratislava, Slovakia, August 1997. Springer.

[AI00] N. Alechina and N. Immerman. Reachability logic: An efficient fragment of transitive closure logic. *Logic Journal of the IGPL*, 8(3):325–338, May 2000.

[AI01] M. Adler and N. Immerman. An $n!$ lower bound on formula size. In *Proc. 16th Symp. on Logic in Computer Science, LICS'01*, pages 197–208, Boston, MA, USA, June 2001. IEEE.

[And94] H. R. Andersen. Model checking and Boolean graphs. *TCS*, 126(1):3–30, April 1994.

[AvBN98] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[AVV97] S. Abiteboul, M. Y. Vardi, and V. Vianu. Fixpoint logics, relational machines, and computational complexity. *Journal of the ACM*, 44(1):30–56, January 1997.

[BAPM83] M. Ben-Ari, A. Pnueli, and Z. Manna. The temporal logic of branching time. *Acta Informatica*, 20(3):207–226, December 1983.

[BC96] G. Bhat and R. Cleaveland. Efficient local model-checking for fragments of the modal $\mu$-calculus. In T. Margaria and B. Steffen, editors, *Proc. 2nd Int. Workshop on Tools and Algorithms for Construction and Analysis of Systems, TACAS'96*, volume 1055 of *LNCS*, pages 107–126. Springer, March 1996.

[BCG95] G. Bhat, R. Cleaveland, and O. Grumberg. Efficient on-the-fly model checking for CTL$^*$. In *Proc. 10th Symp. on Logic in Computer Science, LICS'95*, pages 388–397, San Diego, CA, USA, June 1995. IEEE.

[BCM$^+$92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: $10^{20}$ states and beyond. *Information and Computation*, 98(2):142–170, June 1992.

[BEM96] J. Bradfield, J. Esparza, and A. Mader. An effective tableau system for the linear time $\mu$-calculus. *LNCS*, 1099:98–109, 1996.

[Bet55] E. W. Beth. Semantic entailment and formal derivability. *Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde, N.R.*, 18(13):309–342, 1955.

[Bry86] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, August 1986.

[BS01] J. Bradfield and C. Stirling. Modal logics and $\mu$-calculi: an introduction. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*. Elsevier, 2001.

[Büc62] J. R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Congress on Logic, Method, and Philosophy of Science*, pages 1–12, Stanford, CA, USA, 1962. Stanford University Press.

[BVW94] O. Bernholtz, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In D. L. Dill, editor, *Proc. 6th Conf. on Computer Aided Verification, CAV'94*, volume 818 of *LNCS*, pages 142–155, Stanford, June 1994. Springer.

[CE81] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In D. Kozen, editor, *Proc. Workshop on Logics of Programs*, volume 131 of *LNCS*, pages 52–71, Yorktown Heights, New York, May 1981. Springer.

[CES83] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications. In *Proc. 10th Symp. on Principles of Programming Languages, POPL'83*, pages 117–126. ACM, January 1983.

[CGL93] E. M. Clarke, O. Grumberg, and D. Long. Verification tools for finite-state concurrent systems. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Proceedings REX School/Symp. A Decade of Concurrency*, Noordwijkerhout, The Netherlands, June 1993, volume 803 of *LNCS*, pages 124–175. Springer, 1993.

[CKS81] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, January 1981.

[Cle90] R. Cleaveland. Tableau-based model checking in the propositional $\mu$-calculus. *Acta Informatica*, 27(8):725–748, 1990.

[CS92] R. Cleaveland and B. Steffen. A linear–time model–checking algorithm for the alternation–free modal $\mu$–calculus. In K. G. Larsen and A. Skou, editors, *Proc. 3rd Int. Conf. on Computer Aided Verification, CAV'91*, volume 575 of *LNCS*, pages 48–58, Berlin, Germany, July 1992. Springer.

[Cur52] H. B. Curry. The elimination theorem when modality is present. *Journal of Symbolic Logic*, 17(4):249–265, 1952.

[CVWY91] C. Courcoubetis, M. Y. Vardi, P. Wolper, and M. Yannakakis. Memory efficient algorithms for the verification of temporal properties. In E. M. Clarke and R. P. Kurshan, editors, *Proc. 2nd Conf. on Computer Aided Verification, CAV'90*, volume 531 of *LNCS*, pages 233–242, Berlin, Germany, June 1991. Springer.

[Dam94] M. Dam. CTL$^*$ and ECTL$^*$ as fragments of the modal $\mu$-calculus. *TCS*, 126(1):77–96, April 1994.

[Dan84] S. Danecki. Nondeterministic propositional dynamic logic with intersection is decidable. In A. Skowron, editor, *Proc. 5th Symp. on Computation Theory*, volume 208 of *LNCS*, pages 34–53, Zaborów, Poland, December 1984. Springer.

[Dem01] E. D. Demaine. Playing games with algorithms: algorithmic combinatorial game theory. In J. Sgall, A. Pultr, and P. Kolman, editors, *Proc. 22nd Symp. on Math. Foundations of Computer Science, MFCS'01*, volume 2136 of *LNCS*, pages 18–32. Springer, 2001.

[DGK01] A. Dawar, E. Grädel, and S. Kreutzer. Inflationary fixed points in modal logic. In L. Fribourg, editor, *Proc. 15th Workshop on Computer Science Logic, CSL'01*, LNCS, pages 277–291, Paris, France, September 2001. Springer.

[EC80] E. A. Emerson and E. M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In J. W. de Bakker and J. van Leeuwen, editors, *Proc. 7th Int. Coll. on Automata, Languages and Programming, ICALP'80*, volume 85 of *LNCS*, pages 169–181, Noordweijkerhout, NL, July 1980. Springer.

[EC82] E. A. Emerson and E. M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2(3):241–266, December 1982.

[EF95] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Perspectives in Math. Logic. Springer, Berlin, 1995.

[EFT94] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Undergraduate Texts in Mathematics. Springer, Berlin, 2nd edition, 1994.

[EH85] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30:1–24, 1985.

[EH86] E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, January 1986.

[Ehr61] A. Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fund. Math.*, 49:129–141, 1961.

[EJ91] E. A. Emerson and C. S. Jutla. Tree automata, $\mu$-calculus and determinacy. In *Proc. 32nd Symp. on Foundations of Computer Science*, pages 368–377, San Juan, Puerto Rico, October 1991. IEEE.

[EJ00] E. A. Emerson and C. S. Jutla. The complexity of tree automata and logics of programs. *SIAM Journal on Computing*, 29(1):132–158, February 2000.

[EJS01] E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model checking for the $\mu$-calculus and its fragments. *TCS*, 258(1–3):491–522, 2001.

[EL87] E. A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 8(3):275–306, 1987.

[Eme85] E. A. Emerson. Automata, tableaux and temporal logics. In R. Parikh, editor, *Proc. Conf. on Logic of Programs*, volume 193 of *LNCS*, pages 79–87, Brooklyn, NY, June 1985. Springer.

[Eme87] E. A. Emerson. Uniform inevitability is tree automaton ineffable. *Information Processing Letters*, 24(2):77–79, January 1987.

[Eme90] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 16, pages 996–1072. Elsevier and MIT Press, New York, USA, 1990.

[Eme96] E. A. Emerson. *Automated Temporal Reasoning about Reactive Systems*, volume 1043 of *LNCS*, pages 41–101. Springer, New York, NY, USA, 1996.

[Eme97] E. A. Emerson. Model checking and the $\mu$-calculus. In N. Immerman and P. G. Kolaitis, editors, *Descriptive Complexity and Finite Models*, volume 31 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, chapter 6. AMS, 1997.

[ES84] E. A. Emerson and A. P. Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, June 1984.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *Complexity and Computation*, 7:43–73, 1974.

[Fis91] M. Fisher. A resolution method for temporal logic. In J. Mylopoulos and R. Reiter, editors, *Proc. 12th Joint Conf. on Artificial Intelligence*, pages 99–104, Sydney, Australia, August 1991. Morgan Kaufmann.

[Fit83] M. C. Fitting. *Proof Methods for Modal and Intutionistic Logics*. Reidel, Dordrecht, 1983.

[FL77] M. J. Fischer and R. E. Ladner. Propositional modal logic of programs (extended abstract). In *Proc. 9th Symp. on Theory of Computing, STOC'77*, pages 286–294, Boulder, Colorado, May 1977. ACM.

[FL79] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, April 1979.

[Fra54] R. Fraïssé. Sur quelques classifications des systèmes de relations. *Publ. Sci. Univ. Alger. Sér. A*, 1:35–182, 1954.

[Fri76] E. P. Friedman. The inclusion problem for simple languages. *TCS*, 1(4):297–316, April 1976.

[Gab89] D. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Proc. Conf. on Temporal Logic in Specification*, volume 398 of *LNCS*, pages 409–448, Berlin, April 1989. Springer.

[Gen35] G. Gentzen. Untersuchungen über das logische Schliessen. *Math. Zeitschrift*, 39:176–210,405–431, 1935.

[GH82] Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proc. 14th Symp. on Theory of Computing, STOC'82*, pages 60–65, San Francisco, California, May 1982. ACM.

[GH94] J. F. Groote and H. Hüttel. Undecidable equivalences for basic process algebra. *Information and Computation*, 115(2):354–371, December 1994.

[Gor99] R. Goré. Tableau methods for modal and temporal logics. In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*. Kluwer, Dordrecht, 1999.

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. The temporal analysis of fairness. In *Proc. 7th Symp. on Principles of Programming Languages, POPL'80*, pages 163–173. ACM, January 1980.

[GPVW95] R. Gerth, D. Peled, M. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification Testing and Verification*, pages 3–18, Warsaw, Poland, 1995. Chapman & Hall.

[GS53] D. Gale and F. M. Stewart. Infinite games of perfect information. *Ann. Math. Studies*, 28:245–266, 1953.

[GW99] E. Grädel and I. Walukiewicz. Guarded fixed point logic. In *Proc. 14th Symp. on Logic in Computer Science, LICS'99*, pages 45–55. IEEE, July 1999.

[HHWT97] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: A Model Checker for Hybrid Systems. In O. Grumberg, editor, *Proc. 9th Conf.*

*on Computer Aided Verification, CAV'97*, volume 1254 of *LNCS*, pages 460–463. Springer, 1997.

[Hin69] J. Hintikka. *Models for Modalities*. D. Reidel, Dordrecht, 1969.

[HM96] Y. Hirshfeld and F. Moller. *Decidability Results in Automata and Process Theory*, volume 1043 of *LNCS*, pages 102–148. Springer, New York, NY, USA, 1996.

[Hoa78a] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, August 1978. See corrigendum [Hoa78b].

[Hoa78b] C. A. R. Hoare. Corrigendum: "Communicating Sequential Processes". *Communications of the ACM*, 21(11):958–958, November 1978.

[Hol97] G. J. Holzmann. The Spin model checker. *IEEE Transactions on Software Engineering*, 23(5):279–95, May 1997.

[HT87] T. Hafer and W. Thomas. Computation tree logic $CTL^*$ and path quantifiers in the monadic theory of the binary tree. In T. Ottmann, editor, *Proc. 14th Coll. on Automata, Languages and Programming, ICALP'87*, volume 267 of *LNCS*, pages 269–279, Karlsruhe, Germany, July 1987. Springer.

[HWZ00] I. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable fragments of first-order temporal logics. *Annals of Pure and Applied Logic*, 106(1–3):85–134, 2000.

[HWZ02] I. Hodkinson, F. Wolter, and M. Zakharyaschev. Decidable and undecidable fragments of first-order branching temporal logics. In *Proc. 17th Symp. on Logic in Computer Science, LICS'02*, pages 393–402. IEEE, July 2002.

[Imm82] N. Immerman. Upper and lower bounds for first order expressibility. *Journal of Computer and System Sciences*, 25(1):76–98, August 1982.

[Imm86] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68(1–3):86–104, 1986.

[Imm89] N. Immerman. Descriptive and computational complexity. In J. Hartmanis, editor, *Computational Complexity Theory, Proc. Symp. Applied Math.*, volume 38, pages 75–91. AMS, 1989.

[JW96] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional $\mu$-calculus with respect to monadic second order logic. In U. Montanari and V. Sassone, editors, *Proc. 7th Conf. on Concurrency Theory, CONCUR'96*, volume 1119 of *LNCS*, pages 263–277, Pisa, Italy, August 1996. Springer.

[Kam68] H. W. Kamp. *On tense logic and the theory of order*. PhD thesis, Univ. of California, 1968.

[Kan57] S. Kanger. *Provability in Logic*, volume 1 of *Stockholm Studies in Philosophy*. Almqvist & Wiksell, Stockholm, 1957.

[KG96] O. Kupferman and O. Grumberg. Branching-time temporal logic and tree automata. *Information and Computation*, 125(1):62–69, February 1996.

[Koz83] D. Kozen. Results on the propositional $\mu$-calculus. *TCS*, 27:333–354, December 1983.

[Kri59] S. A. Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24(1):1–14, 1959.

[KT90] D. Kozen and J. Tiuryn. Logics of programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 14, pages 789–840. Elsevier and MIT Press, New York, USA, 1990.

[KVW00] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, March 2000.

[Lam80] L. Lamport. ”Sometime” is sometimes ”not never”: on the temporal logic of programs. In *Proc. 7th Symp. on Principles of Programming Languages, POPL'80*, pages 174–185, New York, USA, January 1980. ACM.

[Lan00] M. Lange. A game based approach to CTL$^*$ model checking. In *Proc. summer school MOVEP'2k*, Nantes, France, June 2000.

[Lan02a] M. Lange. Alternating context-free languages and linear time $\mu$-calculus with sequential composition. In P. Panangaden and U. Nestmann, editors, *Proc. 9th Workshop on Expressiveness in Concurrency, EXPRESS'02*, volume 68.2 of *ENTCS*, pages 71–87, Brno, Czech Republic, August 2002. Elsevier.

[Lan02b] M. Lange. Local model checking games for fixed point logic with chop. In L. Brim, P. Jančar, M. Křetínský, and A. Kučera, editors, *Proc. 13th Conf. on Concurrency Theory, CONCUR'02*, volume 2421 of *LNCS*, pages 240–254, Brno, Czech Republic, August 2002. Springer.

[LLNT99] M. Lange, M. Leucker, T. Noll, and S. Tobies. TRUTH – a verification platform for concurrent systems. In *Tool Support for System Specification, Development, and Verification*, Advances in Computing Science. Springer, 1999.

[LMS01] F. Laroussinie, N. Markey, and P. Schnoebelen. Model checking $CTL^+$ and $FCTL$ is hard. In *Proc. 4th Conf. Foundations of Software Science and Computation Structures, FOSSACS'01*, volume 2030 of *LNCS*, pages 318–331, Genova, Italy, April 2001. Springer.

[LP85] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc. 12th Symp. on Principles of Programming Languages, POPL'85*, pages 97–107, New York, January 1985. ACM.

[LP00] O. Lichtenstein and A. Pnueli. Propositional temporal logics: Decidability and completeness. *Logic Journal of the IGPL*, 8(1):55–85, 2000.

[LS00] M. Lange and C. Stirling. Model checking games for CTL$^*$. In *Proc. Conf. on Temporal Logic, ICTL'00*, pages 115–125, Leipzig, Germany, October 2000.

[LS01] M. Lange and C. Stirling. Focus games for satisfiability and completeness of temporal logic. In *Proc. 16th Symp. on Logic in Computer Science, LICS'01*, Boston, MA, USA, June 2001. IEEE.

[LS02a] M. Lange and C. Stirling. Model checking fixed point logic with chop. In M. Nielsen and U. H. Engberg, editors, *Proc. 5th Conf. on Foundations of Software Science and Computation Structures, FOSSACS'02*, volume 2303 of *LNCS*, pages 250–263, Grenoble, France, April 2002. Springer.

[LS02b] M. Lange and C. Stirling. Model checking games for branching time logics. *Journal of Logic and Computation*, 12(4):623–639, 2002.

[Mar75] D. A. Martin. Borel determinacy. *Ann. Math.*, 102:363–371, 1975.

[May00] R. Mayr. Process rewrite systems. *Information and Computation*, 156:264–286, 2000.

[McM93] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Norwell Massachusetts, 1993.

[McN66] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9(5):521–530, October 1966.

[McN93] R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, December 1993.

[Mil80] R. Milner. A calculus of communicating systems. *LNCS*, 92, 1980.

[Mil89] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.

[MO99] M. Müller-Olm. A modal fixpoint logic with chop. In C. Meinel and S. Tison, editors, *Proc. 16th Symp. on Theoretical Aspects of Computer Science, STACS'99*, volume 1563 of *LNCS*, pages 510–520, Trier, Germany, 1999. Springer.

[Mol92] F. Moller. *The Edinburgh Concurrency Workbench (Version 6.1)*. Department of Computer Science, University of Edinburgh, October 1992.

[MP92] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems Specification*. Springer, 1992.

[MPW92] R. Milner, J. Parrow, and J. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–40,41–77, September 1992.

[MR99] F. Moller and A. Rabinovich. On the expressive power of CTL$^*$. In *Proc. 14th Symp. on Logic in Computer Science, LICS'99*, pages 360–369. IEEE, July 1999.

[MS95] D. E. Muller and P. E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. *TCS*, 141(1–2):69–107, April 1995.

[MSS88] D. E. Muller, A. Saoudi, and P. E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *Proc. 3rd Symp. on Logic in Computer Science, LICS'88*, pages 422–427, Edinburgh, Scotland, July 1988. IEEE.

[MSS92] D. E. Muller, A. Saoudi, and P. E. Schupp. Alternating automata, the weak monadic theory of trees and its complexity. *TCS*, 97(2):233–244, April 1992.

[NW97] D. Niwiński and I. Walukiewicz. Games for the $\mu$-calulus. *TCS*, 163:99–116, 1997.

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, New York, 1994.

[Pet62] C. A. Petri. Fundamentals of a theory of asynchronous information flow. In C. M. Popplewell, editor, *Proc. IFIP Congress*, Information Processing, pages 386–391. North-Holland, August 1962.

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark, September 1981.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th Symp. on Foundations of Computer Science, FOCS'77*, pages 46–57, Providence, RI, USA, October 1977. IEEE.

[Pra79] V. R. Pratt. Models of program logics. In *Proc. 20th Symp. on Foundations of Computer Science, FOCS'79*, pages 115 – 122. IEEE, 1979.

[Pra80] V. R. Pratt. A near optimal method for reasoning about action. *Journal of Computer and System Sciences*, 2:231–254, April 1980.

[Pre29] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. *Sprawozdanie z I Kongresu Matematikow Krajow Slowcanskich Warszawa*, 395:92–101, 1929.

[QS82] J. P. Queille and J. Sifakis. Specification and verification of concurrent systems in CESAR. In *Proc. 5th Symp. on Programming*, volume 137 of *LNCS*, pages 337–371. Springer, 1982.

[Rab69] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. of Amer. Math. Soc.*, 141:1–35, 1969.

[Rau79] W. Rautenberg. *Klassische und nichtklassische Aussagenlogik*. Vieweg, Braunschweig/Wiesbaden, 1979.

[Rei85] W. Reisig. *Petri Nets (an Introduction)*. Number 4 in EATCS Monographs on Theoretical Computer Science. Springer, 1985.

[Rey01] M. Reynolds. An axiomatization of full computation tree logic. *Journal of Symbolic Logic*, 66(3):1011–1057, September 2001.

[Saf88] S. Safra. On the complexity of ω-automata. In *Proc. 29th Symp. on Foundations of Computer Science, FOCS'88*, pages 319–327. IEEE, October 1988.

[Sav69] W. J. Savitch. Deterministic simulation of nondeterministic Turing Machines. In *Symp. on Theory of Computing, STOC'69*, pages 247–248, New York, May 1969. ACM.

[SC85] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the Association for Computing Machinery*, 32(3):733–749, July 1985.

[SE84] R. S. Streett and E. A. Emerson. The propositional $\mu$-calculus is elementary. In J. Paredaens, editor, *Proc. 11th Coll. on Automata, Languages, and Programming, ICALP'84*, volume 172 of *LNCS*, pages 465–472. Springer, Berlin, 1984.

[Seg77] K. Segerberg. A completeness theorem in the modal logic of programs. *Notices of the AMS*, 24(6):A–552, October 1977.

[SGL97] P. H. Schmitt and J. Goubault-Larrecq. A tableau system for linear time temporal logic. In *Proc. 3rd Workshop on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'97*, volume 1217 of *LNCS*, pages 130–144. Springer, Enschede, Netherlands, April 1997.

[Smu95] R. M. Smullyan. *First-Order Logic*. Dover Publications, New York, second corrected edition, 1995.

[Sti89] C. Stirling. Comparing linear and branching time temporal logics. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Proc. Conf. on Temporal Logic in Specification*, volume 398 of *LNCS*, pages 1–20, Berlin, April 1989. Springer.

[Sti92] C. Stirling. Modal and temporal logics. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2 (Background: Computational Structures), pages 477–563. Clarendon Press, Oxford, 1992.

[Sti95] C. Stirling. Local model checking games. In I. Lee and S. A. Smolka, editors, *Proc. 6th Conf. on Concurrency Theory, CONCUR'95*, volume 962 of *LNCS*, pages 1–11, Berlin, Germany, August 1995. Springer.

[Sti96a] C. Stirling. Games and modal $\mu$-calculus. In T. Margaria and B. Steffen, editors, *Proc. 2nd Int. Workshop on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'96*, volume 1055 of *LNCS*, pages 298–312. Springer, 1996.

[Sti96b] C. Stirling. *Modal and Temporal Logics for Processes*, volume 1043 of *LNCS*, pages 149–237. Springer, NY, USA, 1996.

[Sti01] C. Stirling. *Modal and Temporal Properties of Processes*. Texts in Computer Science. Springer, 2001.

[Sto76] L. J. Stockmeyer. The polynomial-time hierarchy. *TCS*, 3(1):1–22, October 1976.

[Str81] R. S. Streett. Propositional dynamic logic of looping and converse. In *Proc. 13th Symp. on Theory of Computation, STOC'81*, pages 375–383, Milwaukee, Wisconsin, May 1981. ACM.

[Str85] R. S. Streett. Fixpoints and program looping: Reductions from the propositional $\mu$-calculus into propositional dynamic logics of looping. In R. Parikh, editor, *Proc. Conf. on Logic of Programs*, volume 193 of *LNCS*, pages 359–372, Brooklyn, NY, June 1985. Springer.

[SVW83] A. P. Sistla, M. Y. Vardi, and P. Wolper. Reasoning about infinite computation paths. In *Proc. 24th Symp. on Foundations of Computer Science, FOCS'83*, pages 185–194, Los Alamitos, Ca., USA, November 1983. IEEE.

[SW91] C. Stirling and D. Walker. Local model checking in the modal $\mu$–calculus. *TCS*, 89(1):161–177, 1991.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its application. *Pacific J.Math.*, 5:285–309, 1955.

[Tar72] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Computing*, 1:146–160, 1972.

[Tho79] W. Thomas. Star-free regular sets of ω-sequences. *Information and Control*, 42(2):148–156, August 1979.

[Tho95] W. Thomas. On the synthesis of strategies in infinite games. In *Proc. 12th Symp. on Theoretical Aspects of Computer Science, STACS'95*, volume 900 of *LNCS*, pages 1–13, Munich, Germany, March 1995. Springer.

[Tho99] W. Thomas. Complementation of Büchi automata revisited. In J. Karhumäki et al., editor, *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 109–122. Springer, 1999.

[Var82] M. Y. Vardi. The complexity of relational query languages (extended abstract). In *Proc. 14th Symp. on Theory of Computing, STOC'82*, pages 137–146, San Francisco, CA, USA, May 1982. ACM.

[Var96] M. Y. Vardi. *An Automata-Theoretic Approach to Linear Temporal Logic*, volume 1043 of *LNCS*, pages 238–266. Springer, New York, NY, USA, 1996.

[Var01] M. Y. Vardi. Branching vs. linear time: Final showdown. In T. Margaria and W. Yi, editors, *Proc. 7th Int. Conf. on Tools and Algorithms for the*

*Construction and Analysis of Systems, TACAS'01*, volume 2031 of *LNCS*, pages 1–22. Springer, April 2001.

[vB96] J. van Benthem. *Exploring Logical Dynamics*. CSLI Publications, Stanford, California, 1996.

[VB00] W. Visser and H. Barringer. Practical CTL$^*$ model checking: Should SPIN be extended? *Int. J. on Software Tools for Technology Transfer*, 2(4):350–365, 2000.

[VS85] M. Y. Vardi and L. Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *Proc. 17th Symp. on Theory of Computing, STOC'85*, pages 240–251, Baltimore, USA, May 1985. ACM.

[VW86a] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proc. 1st Symp. on Logic in Computer Science, LICS'86*, pages 332–344. IEEE, Washington, DC, 1986.

[VW86b] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logic of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.

[VW94] M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.

[Wal93] I. Walukiewicz. On completeness of the $\mu$-calculus. In *Proc. 8th Symp. on Logic in Computer Science, LICS'93*, pages 136–146. IEEE, Los Alamitos, CA, 1993.

[Wal95] I. Walukiewicz. Completeness of Kozen's axiomatization of the propositional $\mu$-calculus. In *Proc. 10th Symp. on Logic in Computer Science, LICS'95*, pages 14–24, Los Alamitos, CA, 1995. IEEE.

[Wal96] I. Walukiewicz. A note on the completeness of Kozen's axiomatization of the propositional $\mu$-calculus. *Bulletin of Symbolic Logic*, 2(3):349–366, 1996.

[Wil99] T. Wilke. $CTL^+$ is exponentially more succinct than CTL. In *Proc. 19th Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'99*, volume 1738 of *LNCS*, pages 110–121. Springer, 1999.

[Win93] Glynn Winskel. *The Formal Semantics of Programming Languages: An Introduction.* Foundations of Computing series. MIT Press, February 1993.

[Zem73] J. J. Zeman. *Modal Logic / the Lewis-Modal System*. Oxford University Press, Oxford, 1 edition, 1973.

[Zer13] E. Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In *Proc. 5th Int. Congress of Mathematicians*, volume II, pages 501–504. Cambridge University Press, 1913.