Diploma Thesis

A Proof System for CTL^*_μ

Oliver Friedmann

Supervisors

Prof. Dr. Martin Hofmann
Priv.-Doz. Dr. Martin Lange

Submitted to

Chair of Theoretical Computer Science
University of Munich

10 June 2008

Abstract

The logic CTL^*_μ extends the full branching time logic CTL^* with arbitrary least and greatest fixed point operators on sets of paths. This gives it at least the power to express all ω-regular languages on paths. We also consider a syntactical fragment of CTL^*_μ restricting the legal range of fixed point operators that is still as expressive as $ECTL^*$.

First, we present a tableaux-style proof system for validity in the restricted fragment of CTL^*_μ. We prove it sound and complete, and sketch a decision procedure that runs in double exponential time based on this proof system implying that the validity problem for the restricted fragment is 2-EXPTIME-complete. We also show how to obtain counterexamples for non-valid formulas from failing proof attempts.

Second, we present a model-checking tableaux system for full CTL^*_μ. Again, we prove it sound and complete. A decision procedure that runs in exponential time based on the tableaux system is outlined. Last but not least we show that full CTL^*_μ is as expressive as the Modal μ-Calculus.

Acknowledgements

I would like to thank my supervisors Martin Hofmann and Martin Lange for their outstanding support and guidance during the last time. They always helped me to solve my - sometimes very silly - problems with great patience and replied to all my countless email questions straight away providing me with all the help one could hope for.

Finally, I want thank both of them for all their letters of recommendation they wrote for me and for the very enjoyable and inspiring recent terms I had as a student at the University of Munich.

Last but not least, I would like to express my thanks to my parents for their constant support. They were always very kind and helped me to keep a clear head by handling almost all daily-life problems for me.

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 of professional qualification except as specified.

(Oliver Friedmann)

CONTENTS

1. INTRODUCTION

Formal Verification

The testing of software and hardware alone cannot guarantee that the underlying algorithms are correct in the sense that they meet their specification. In general, there are possibly infinitely many cases to be tested which obviously is impossible, thus the developer usually simulates a finite number of crucial scenarios. Nevertheless *testing* can only be used to find faults, not to ensure their absence.

Take for instance the famous FDIV bug of Intel's original Pentium floating point unit. According to Intel, there were five missing entries in the division lookup table due to a failure in a C script transferring some precomputed values into a programmable logic array (PLA). Nobody checked the PLA completely to verify that every entry was copied properly.

The idea of formal verification is to *mathematically prove* that the given algorithm actually meets its specification, mainly using automated methods of mathematics and logics. This leaves the developer with the task to translate the specification as well as the algorithm into the formalism of the verification method. Such a translation often includes an abstraction process in which the developer hides all details being not important for the specification to be verified or not transferable into the verification language.

A widely used family of verification languages are temporal and modal logics.

Temporal and Modal Logics

Temporal and Modal Logics were firstly studied by philosophers of the ancient world, particularly by Aristotle. He proposed a logic similar to the *modal first-order predicate calculus* although not as scientifically and formally founded as it is of today.

There are different kinds of temporal and modal logics: A linear temporal logic for instance features a discrete linear time line enabling the logic to reason about one single future. Branching time logics, however, have the ability to reason about various futures being predestined for contexts that may act unpredictably.

The user of logics is always confronted with the task of choosing the *right* logic for the intended application. Intuitively said, logics which are more expressive than others are far more complicated to handle algorithmically if at all. Therefore the user is advised to choose a logic which is expressive enough to capture the designated problem conveniently and properly on the one hand and on the other hand is not too complex to be processed by formal verification tools.

Games in Logic

Many problems in logic can be seen as a long sequence of alternating quantifications, sometimes even infinitely long. Such scenarios can be modelled by specific logic games consisting of two players, namely *Eliza* and *Albert*, competing each other.

Each player usually corresponds to one quantification type, *Eliza* for instance to existential quantification and *Albert* to universal quantification. The set of playing positions is partitioned into two sets each of them belonging to one of the two players. The game is played by moving a token from one playing position to another connected position while the player possessing the

current playing position chooses which edge leading to another position should be played out. A play in this game is won by a player if the other player gets stuck or if a specific global condition holds.

Now *Eliza* wins a game starting from a playing position if she has a winning strategy: She is able to win each play no matter what moves *Albert* chooses. Hence, the problem of alternating quantifications was reduced to the question whether there is a strategy for *Eliza* so that she wins playing according to her strategy against all strategies of *Albert*.

The Logic CTL^*_μ

The primary concern of this thesis is the investigation of an effective proof system for CTL^*. After only a short time it turned out to be even more interesting to develop a system for a more generalized branching time logic: The logic CTL^*_μ.

This logic extends the full branching time logic CTL^* with arbitrary least and greatest fixed point operators on sets of paths quite similar to those established in the Modal μ-Calculus. The probably best intuition behind fixed points is the concept of recursion: The least fixed point corresponds to finite recursion meaning that the respective process needs to be terminated after a finite amount of time whereas the greatest fixed point allows infinite recursion in the sense that the respective process does not need to terminate.

Being obviously at least as expressive as CTL^* many questions arise. How expressive is this logic in comparison to other modal logics? Is it possible to create tableaux systems for this logic? How complex are the corresponding decision problems, particularly the validity problem and the model-checking problem? Fortunately, we are able to answer almost all of these questions accurately.

Existing Approaches

The central technique used in this thesis - tracking formula threads as a method of capturing global temporal behaviour - was mainly inspired by the Diploma Thesis of Christian Dax [Dax06] and the follow-up paper by Christian Dax, Martin Hofmann and Martin Lange [DHL06] where it was used in the context of the Linear Time μ-Calculus. The author of this thesis also investigated threads once before in his Project Thesis, transferring this technique to the Modal μ-Calculus [Fri06].

There is a full axiomatization of CTL^* by Mark Reynolds using the classic tableaux method of finite natural deduction trees [Rey02]. An effective procedure using this system is difficult to obtain due to the fact that it contains a rule not fulfilling the subformula property as well as a rule quantifying over second order objects.

Another approach for the verification of CTL^* was recently presented by Yonit Kesten and Amir Pnueli, establishing the method of *temporal testers* [KP05]. It basically tries to decompose a formula into smaller subformulas by substituting linear-time subformulas by fresh atomic propositions.

Summary

The second chapter comprises all fundamental definitions and propositions regarding CTL^*_μ in order to provide a theoretical background for all upcoming sections. We also investigate the expressiveness of CTL^*_μ and of some of its fragments.

In the third chapter we inspect and generalize the method of formula threads as a technique to accurately capture the global behaviour of temporal processes. Additionally we construct automata corresponding to the thread identification problems arising in this context. The theoretical background of the associated formal language was put into the appendix in order not to

confuse the reader with all technical details right away.

The fourth chapters employs the technique of threads to create a tableaux system for validity of the restricted fragment of CTL^*_μ. We prove it sound and complete and show how to obtain counterexamples from failing proof attempts. Finally a decision procedure based on proof games corresponding to the tableaux system is sketched.

In the fifth chapter we utilise the technique of threads to develop a tableaux system for the model-checking problem of full CTL^*_μ. Again, we prove it sound and complete and investigate associated model-checking games to finally provide a decision procedure for this problem.

2. THE LOGIC CTL^*_μ

2.1 Definition

Branching time logics are interpreted over labelled transitions systems. A labelled transition system is a directed graph providing a labelling map defining for every node which propositional variables hold in the very spot. These nodes were called *worlds* in the original context of modal logics giving a propositional truth assignment for the *present* and being connected to *future* worlds. Generally, such structures are called *Kripke structures* due to Saul Kripke. For this thesis we will use *pointed* Kripke structures additionally specifying an initial world.

We confine ourselves to a fixed but (countably) infinite set of atomic propositions $\mathcal{P} = \{p, q, ...\}$ which will be used throughout the thesis. Since it will be convenient to particularly consider formulas in positive normal form, i.e. formulas having negations only in front of atoms, one also defines the complement-closure $\mathcal{P}^* := \{q, \neg q \mid q \in \mathcal{P}\}$ containing all *literals*.

Definition 2.1 (Labelled transition system)**.** A *labelled transition system* (LTS) is a quadruple $\mathcal{T} = (\mathcal{S}, \theta, \rightarrow, \lambda)$ with

- $\mathcal{S}$ being a set of states
- $\theta \in \mathcal{S}$ being the initial state
- $\rightarrow \subseteq \mathcal{S} \times \mathcal{S}$ being a binary relation on $\mathcal{S}$ that is assumed to be total
- $\lambda : \mathcal{S} \rightarrow 2^{\mathcal{P}}$ maps every state to a set of propositions

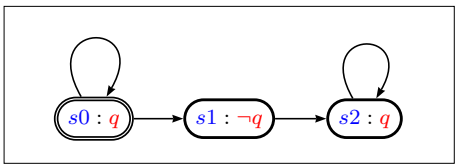
Example 2.2. Consider the following labelled transition system

$$\mathcal{T} = (\{s_0, s_1, s_2\}, s_0, \rightarrow, \lambda)$$

whereas

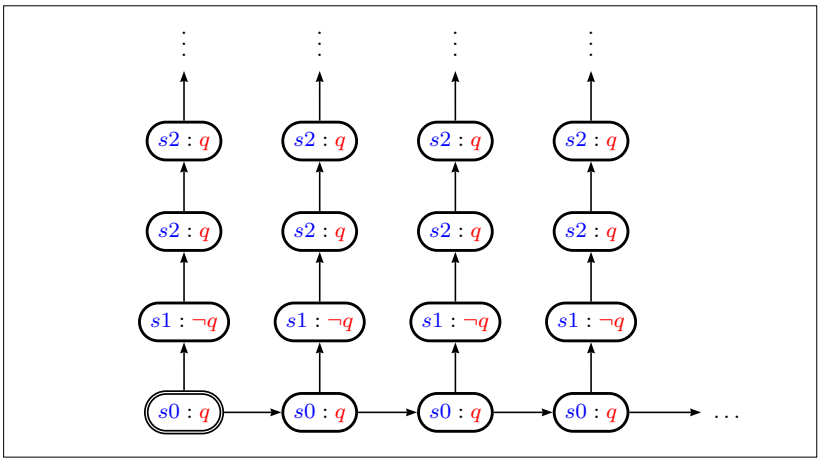
$\mathcal{S}$	s_0	s_1	s_2
$\rightarrow$	s_0, s_1	s_2	s_2
λ	$\{q\}$	$\emptyset$	{q}

A typical *compact* graphical depiction of $\mathcal{T}$ is given as follows:



All states are enclosed by circular arcs while the initial state is denoted by a double circular arc. The image of the labelling map follows right after the colon of respective state. Note that our graphical notation shows $\neg q$ in order to emphasize the difference between state 1 and the other two states.

An *expanded* view of the same transition system can be presented in the following way:



As we will need to choose states in a deterministic way, i.e. to be able to automatically select the same state again under the same circumstances, we require each transition system $\mathcal{T}$ to provide an arbitrary but fixed wellordering $\prec_{\mathcal{T}} \subseteq \mathcal{S} \times \mathcal{S}$ on states. This precondition might seem a little odd but is crucially needed to prove our system sound and complete.

A path, roughly speaking, is an infinitely long connected chain of worlds following the transition relation.

Definition 2.3 (Path)**.** A *path* through an LTS $\mathcal{T}$ is an infinite sequence $\pi = s_0, s_1, ...$ s.t. $s_i \in \mathcal{S}$ and $s_i \to s_{i+1}$ for all i. We say a path π is *rooting in* s_0 when referring to the first state of π.

We will use the following abbreviations to access the single states and suffixes of a path π:

- $\pi(i) := s_i$
- $\pi[i] := s_i, s_{i+1}, ...$

where $i \in \mathbb{N}$.

By $\Pi(\mathcal{T})$ we denote the set of all path in $\mathcal{T}$; $\Pi_{\mathcal{T}}(s)$ denotes the subset of all paths in $\mathcal{T}$ rooting in s. For a set $V \subseteq \mathcal{S}$ of states $\Pi_{\mathcal{T}}[V]$ is defined as the subset of all paths in $\mathcal{T}$ rooting in a state contained in V. The path-projection to the first state, $\pi \mapsto \pi(0)$, is denoted by $\mathcal{S}_{\mathcal{T}} : \Pi(\mathcal{T}) \to \mathcal{S}$.

A *prefix path* is a finite sequence $\xi = s_0, s_1, ..., s_n \in \mathcal{S}$ with $s_i \to s_{i+1}$ for all $i < n$. A path $\pi \in \Pi(\mathcal{T})$ is called a *path completion* of ξ iff $\pi(i) = \xi_i$ for all $i \leq n$.

Since we not only want to be able to choose states deterministically but also and particularly paths, we extend the wellordering $\prec_{\mathcal{T}}$ on states to a wellordering on paths $\prec^s_{\mathcal{T}} \subseteq \Pi_{\mathcal{T}}(s) \times \Pi_{\mathcal{T}}(s)$ w.r.t. a root state $s \in \mathcal{S}$ as follows:

$$\pi_1 \prec^s_{\mathcal{T}} \pi_2 \; : \iff \; \exists i.\, (\pi_1(i) \prec_{\mathcal{T}} \pi_2(i) \wedge \forall j < i.\, \pi_1(j) = \pi_2(j))$$

The language of CTL^*_μ can be seen as a mixture of CTL^* and the Modal μ-Calculus by lumping path quantifications and fixed point operators together. Since formulas in CTL^* are interpreted with respect to paths and not to states

this thesis' fixed points are - in contrast to the Modal μ-Calculus - not interpreted over sets of states but over sets of paths. As before, we confine ourselves to an infinite but fixed set $\mathcal{V} = \{X, Y, ...\}$ of second-order variables.

Definition 2.4 (Syntax)**.** Formulas of CTL^*_μ are given by the following grammar:

$$\psi ::= E\psi \mid A\psi \mid q \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi \mid \bigcirc\psi \mid X \mid \mu X.\psi \mid \nu X.\psi$$

where $q \in \mathcal{P}$ and $X \in \mathcal{V}$.

Formulas built using ψ are also called *path formulas*; the set of all formulas is denoted by CTL^*_μ.

As we will often prove certain properties of CTL^*_μ by induction on the structure of the formula, it is sometimes convenient to abbreviate the syntactical structure by a prefix notation:

- $op(\psi_1, \psi_2)$ for binary operators $op \in \{\wedge, \vee\}$
- $op(\psi)$ for unary operators $op \in \{\neg, \bigcirc, E, A, \sigma X.\}$
- $op()$ for atomic formulas $op \in \mathcal{P} \cup \mathcal{V}$

A tuple $(\psi_1, ..., \psi_n)$ of formulas will sometimes be abbreviated by $\vec{\psi}$, and $|\vec{\psi}| := n$ additionally denotes the length of the vector.

Whenever the type of the fixed point is not important, we will subsume both fixed point binders, μ- and ν-, by σ.

A traditional sublanguage is the set of *state formulas* which is denoted by $sCTL^*_\mu$.

Definition 2.5 (State formula)**.** State formulas are given by the following grammar:

$$\varphi ::= E\psi \mid A\psi \mid q \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

where $q \in \mathcal{P}$ and $\psi \in CTL^*_\mu$.

Every now and then, when proving properties with respect to a certain formula, we will focus on smaller formulas reoccurring in the context of the main formula. To formalise this one defines *subformulas*:

Definition 2.6 (Immediate subformula and set of subformulas)**.** Let ψ be a formula. The set of *immediate subformulas*, $ISub(\psi)$, is defined as follows

$$ISub(\psi) := \begin{cases} \{\psi_1, \psi_2\} & \text{if } \psi \equiv op(\psi_1, \psi_2) \\ \{\psi'\} & \text{if } \psi \equiv op(\psi') \\ \emptyset & \text{otherwise} \end{cases}$$

The set of subformulas *Sub(ψ)* of ψ is the least set that contains ψ and for each formula $\psi' \in Sub(\psi)$ the set of immediate subformulas $ISub(\psi')$ is also contained in $Sub(\psi)$.

One also defines the *size* of a formula ψ. There are various reasonable alternatives measuring the size of the formula ψ directly proportional to the syntactical size of the formula. We will pursue the probably easiest approach to define the size of a formula, namely as the cardinality of the subformula set: $|\psi| := |Sub(\psi)|$.

The semantics of a formula is defined to be the set of paths satisfying the formula in the usual manner when dealing with least and greatest fixed point operators.

Definition 2.7 (Semantics of path formulas)**.** The semantics of a path formula relative to an LTS $\mathcal{T} = (\mathcal{S}, \theta, \rightarrow, \lambda)$ and an *environment* $\rho : \mathcal{V} \rightarrow 2^{\Pi(\mathcal{T})}$ is an inductively defined subset of $\Pi(\mathcal{T})$.

- $[\![q]\!]^{\mathcal{T}}_{\rho} := \{\pi \in \Pi(\mathcal{T}) \mid q \in \lambda(\pi(0))\}$
- $[\![\neg\psi]\!]^{\mathcal{T}}_{\rho} := \Pi(\mathcal{T}) \setminus [\![\psi]\!]^{\mathcal{T}}_{\rho}$
- $[\![E\psi]\!]^{\mathcal{T}}_{\rho} := \{\pi \in \Pi(\mathcal{T}) \mid \exists \pi' \in \Pi(\mathcal{T}).\, \pi(0) = \pi'(0) \wedge \pi' \in [\![\psi]\!]^{\mathcal{T}}_{\rho}\}$
- $[\![A\psi]\!]^{\mathcal{T}}_{\rho} := \{\pi \in \Pi(\mathcal{T}) \mid \forall \pi' \in \Pi(\mathcal{T}).\, \pi(0) = \pi'(0) \Rightarrow \pi' \in [\![\psi]\!]^{\mathcal{T}}_{\rho}\}$
- $[\![\psi_1 \wedge \psi_2]\!]^{\mathcal{T}}_{\rho} := [\![\psi_1]\!]^{\mathcal{T}}_{\rho} \cap [\![\psi_2]\!]^{\mathcal{T}}_{\rho}$
- $[\![\psi_1 \vee \psi_2]\!]^{\mathcal{T}}_{\rho} := [\![\psi_1]\!]^{\mathcal{T}}_{\rho} \cup [\![\psi_2]\!]^{\mathcal{T}}_{\rho}$
- $[\![\bigcirc\psi]\!]^{\mathcal{T}}_{\rho} := \{\pi \in \Pi(\mathcal{T}) \mid \pi[1] \in [\![\psi]\!]^{\mathcal{T}}_{\rho}\}$

- $[\![X]\!]^{\mathcal{T}}_\rho := \rho(X)$
- $[\![\mu X.\psi]\!]^{\mathcal{T}}_\rho := \bigcap\{T \subseteq \Pi(\mathcal{T}) \mid [\![\psi]\!]^{\mathcal{T}}_{\rho[X\mapsto T]} \subseteq T\}$
- $[\![\nu X.\psi]\!]^{\mathcal{T}}_\rho := \bigcup\{T \subseteq \Pi(\mathcal{T}) \mid T \subseteq [\![\psi]\!]^{\mathcal{T}}_{\rho[X\mapsto T]}\}$

where $q \in \mathcal{P}$ and $X \in \mathcal{V}$. The complement set $([\![\psi]\!]^{\mathcal{T}}_\rho)^C$ of $[\![\psi]\!]^{\mathcal{T}}_\rho$ is defined to be $\Pi(\mathcal{T}) \setminus [\![\psi]\!]^{\mathcal{T}}_\rho$. The projection to i-th state is defined as follows:

$$[\![\varphi]\!]^{\mathcal{T}}_\rho(i) := \{\pi(i) \mid \pi \in [\![\varphi]\!]^{\mathcal{T}}_\rho\}$$

Corollary 2.8 (Semantical preservation)**.** *Let $\mathcal{T}$ be an LTS, $op(\vec{\psi})$ be a formula and $\vec{\psi'}$ be a vector of formulas s.t. $|\vec{\psi}| = |\vec{\psi'}|$. If for all i and all ρ holds that $[\![\psi_i]\!]^{\mathcal{T}}_\rho = [\![\psi'_i]\!]^{\mathcal{T}}_\rho$ then $[\![op(\vec{\psi})]\!]^{\mathcal{T}}_\rho = [\![op(\vec{\psi'})]\!]^{\mathcal{T}}_\rho$ for all ρ.*

Since CTL^*_μ is designed to be an extension of CTL^* one needs to show that the classical CTL^*-constructs - such as *Generally* and *Until* - are able to be defined in CTL^*_μ. But this, obviously, is very simple by using greatest and least fixed points of CTL^*_μ.

Definition 2.9 (Abbreviated formulas)**.** The following abbreviations embed the classical CTL^*:

- *True*: $\mathtt{tt} := \nu X. \bigcirc X$
- *False*: $\mathtt{ff} := \mu Y. \bigcirc Y$
- *Finally*: $F\psi := \mu X.(\psi \vee \bigcirc X)$
- *Generally*: $G\psi := \nu X.(\psi \wedge \bigcirc X)$
- *Until*: $\psi_1 U \psi_2 := \mu X.(\psi_2 \vee (\psi_1 \wedge \bigcirc X))$
- *Release*: $\psi_1 R \psi_2 := \nu X.(\psi_1 \wedge (\psi_2 \vee \bigcirc X))$

Now we define the classic relations in logics: The modelling relation, the semantical equivalence of formulas, the validity and satisfiability.

Definition 2.10 (Model, Equivalence, Validity, Satisfiability)**.** A labelled transition system $\mathcal{T}$, relative to an environment ρ, is a *model* of a state formula φ iff $\theta \in [\![\varphi]\!]^{\mathcal{T}}_{\rho}(0)$, i.e.

$$\mathcal{T} \models_{\rho} \varphi \iff \theta \in [\![\varphi]\!]^{\mathcal{T}}_{\rho}(0)$$

Let $\mathcal{T}$ be a labelled transition system, $s \in \mathcal{S}$, ρ an environment and φ a state formula. We also define

$$\mathcal{T}, s \models_{\rho} \varphi \iff s \in [\![\varphi]\!]^{\mathcal{T}}_{\rho}(0)$$

We also extend the modelling relation to path formulas and paths:

$$\mathcal{T}, \pi \models_{\rho} \psi \iff \pi \in [\![\psi]\!]^{\mathcal{T}}_{\rho}$$

Two formulas ψ_1, ψ_2 are *semantically equivalent*, $\psi_1 \models\!\models \psi_2$, iff $[\![\psi_1]\!]^{\mathcal{T}}_{\rho} = [\![\psi_2]\!]^{\mathcal{T}}_{\rho}$ for every transition system $\mathcal{T}$ and every environment ρ.

A state formula φ is *valid*, $\models \varphi$, iff every LTS $\mathcal{T}$ and every environment ρ are models of φ. A state formula φ is *satisfiable* iff it has at least one model.

The model class of φ is denoted by $Mod(\varphi)$ i.e. $Mod(\varphi) := \{\mathcal{T} \mid T \models \varphi\}$.

Example 2.11. The labelled transition system of Example 2.2 falsifies the following formula

$$AFGq \rightarrow AFAGq$$

as each path in $\mathcal{T}$ rooting in s_0 satisfies $AFGq$, but one path falsifies $FAGq$, namely s_0^{ω}, due to the fact that there is always a junction on s_0^{ω} that leads to s_1 falsifying Gq.

The LTS however satisfies the following formula (which is not very suprising since this formula is even valid):

$$EGFq \rightarrow EGEFq$$

As mentioned before, we will need to choose arbitrary paths deterministically in the sense that under the same circumstances we would select exactly the same path. The *circumstances* under which we need to deterministically choose paths are particularly related to the modelling with respect to a certain formula. By using the wellordering on paths the canonical choice function selects the least element fulfilling the desired properties.

Definition 2.12 (Minimal Model)**.** Let $\mathcal{T}$ be an LTS, ρ be an environment, ψ be a formula and $\pi \in \Pi(\mathcal{T})$. The *minimal modelling relation* is defined as follows:

$$\pi \models_{mmin} \psi \ : \iff \ \pi \models \psi \ \wedge \ \forall \pi' \prec_{\mathcal{T}}^{\pi(0)} \pi : \pi' \not\models \psi$$

The *minimal counter modelling relation* is defined likewise by selecting the least path *not* satisfying the formula:

$$\pi \not\models_{cmin} \psi \ : \iff \ \pi \not\models \psi \ \wedge \ \forall \pi' \prec_{\mathcal{T}}^{\pi(0)} \pi : \pi' \models \psi$$

As for every other logic dealing with variables one needs to define some technicalities to specify properly which variables actually depend on their setting in the environment.

Definition 2.13 (Closed formulas, bound and free variables)**.** An occurrence of a variable X in a formula ψ is *bound* iff it is in the scope of a $\sigma X.\psi'$-subformula, otherwise it is *free*. We define the sets of bound variables in ψ as follows

$$Bound(\psi) := \{X \mid \exists \psi' \in Sub(\psi) : \sigma X.\psi' \in Sub(\psi) \wedge X \in Sub(\psi')\}$$

as well as the set of μ- resp. ν-bound variables:

$$Bound_\mu(\psi) := \{X \mid \exists \psi' \in Sub(\psi) : \mu X.\psi' \in Sub(\psi) \wedge X \in Sub(\psi')\}$$

$$Bound_\nu(\psi) := \{X \mid \exists \psi' \in Sub(\psi) : \nu X.\psi' \in Sub(\psi) \wedge X \in Sub(\psi')\}$$

The set of free variables is inductively defined as follows

$$Free(\psi) := \begin{cases} \{X\} & \text{if } \psi \equiv X \text{ and } X \in \mathcal{V} \\ Free(\psi') \setminus \{X\} & \text{if } \psi \equiv \sigma X.\psi' \text{ and } X \in \mathcal{V} \\ \bigcup_{\psi' \in ISub(\psi)} Free(\psi') & \text{otherwise} \end{cases}$$

A formula is said to be *closed* iff it contains no free variables i.e. if $Free(\psi) = \emptyset$. A formula $\psi \in CTL^*_\mu$ fulfils the *binding assumption* iff

$$\forall (\sigma X.\psi' \in Sub(\psi)) : X \in Sub(\psi) \setminus ISub(\psi)$$

and is furthermore *uniquely binding* iff

1. $\forall \sigma X.\psi' \in Sub(\psi) \,:\, X \notin Free(\psi)$
2. $\forall \sigma_1 X.\psi_1, \sigma_2 X.\psi_2 \in Sub(\psi) \,:\, \sigma_1 X.\psi_1 = \sigma_2 X.\psi_2$
3. $\forall \psi' \in Sub(\psi) : (ISub(\psi') \equiv \{\psi_1, \psi_2\} \Rightarrow \nexists \sigma X.\psi^* \in Sub(\psi_1) \cap Sub(\psi_2))$

A formula ψ is called *rectified* iff ψ fulfils the binding assumption and is uniquely binding.

The *binding assumption* roughly states that there are neither fixed point constructs without bound variables nor fixed point constructs whose only body formula is the fixed point variable itself. The *uniquely binding* criteria add the restriction that a variable is not used twice free and bound as well as twice for two different fixed points. Obviously it is not hard to ensure these properties since renaming of bound variables does not affect the semantics of a formula and each trivial fixed point $\nu X.X$ ($\mu X.X$) can be substituted by `tt` (`ff`).

Lemma 2.14 (Rectification)**.** *For every $\psi \in CTL^*_\mu$ there is some rectified $\psi^* \in CTL^*_\mu$ with $\psi^* \models\!\!\models \psi$.*

Clearly, the semantics of a formula only depends on the transition system and the assignment of the free variables: This is known as the *coincidence property.*

Lemma 2.15 (Coincidence lemma)**.** *Let ψ be a formula, $\mathcal{T}$ be a labelled transition system and ρ_1, ρ_2 be two environments. If $\rho_1(X) = \rho_2(X)$ for all $X \in Free(\psi)$ then $[\![\psi]\!]^{\mathcal{T}}_{\rho_1} = [\![\psi]\!]^{\mathcal{T}}_{\rho_2}$.*

As in every logic using variables one defines the *substitution* of free variables by other formulas. In this context, the substitution of a variable is defined as the formula in which all free occurrences of the variable are simultaneously substituted.

Definition 2.16 (Variable substitution)**.** Let ψ, ψ' be formulas and $X \in \mathcal{V}$ be a variable. Then $\psi[\psi'/X]$ is inductively defined as follows:

$$\psi[\psi'/X] := \begin{cases} op(\psi_1[\psi'/X], \psi_2[\psi'/X]) & \text{if } \psi \equiv op(\psi_1, \psi_2) \\ op(\psi_1[\psi'/X]) & \text{if } \psi \equiv op(\psi_1) \text{ and } op \neq \sigma X. \\ \psi' & \text{if } \psi \equiv X \\ \psi & \text{otherwise} \end{cases}$$

The logic CTL^*_μ considers least and greatest fixed points with respect to monotone functions. Thus one needs to ensure that each bound variable X under a fixed point $\sigma X.\psi$ occurs positively in the sense that between X and its binder $\sigma X.\psi$ is an even number of negations.

Definition 2.17 (Well-formed)**.** A formula $\psi \in CTL^*_\mu$ is called *well-formed* iff ψ is rectified and every bound variable is under an even number of negations under its σ-scope.

Next, we define a total ordering on formulas. When dealing with lists of formulas in the proof system, we want to be able to select one of them to proceed with deterministically. Although we do not need this determinism in order to prove our system sound and complete, it aids us to reduce the complexity of corresponding decision problems whenever the decision process needs to select a formula to proceed with. It will turn out that in most cases it does not matter *which* formula is chosen to be subsequently used, hence it is obviously less complex to deterministically select a formula rather than trying each formula out nondeterministically.

Since we want to avoid to demand all variables and propositions to be totally ordered by definition, we will define a total ordering on *subformulas with respect to a fixed formula* ψ in the following sense: The higher a subformula occurs in the parse tree the greater it is and if two subformulas occur in two different subtrees having a common parent node, the formula occurring in the subtree being on the right hand side is greater than the one being on the left hand side.

Definition 2.18 (Total ordering on formulas)**.** Let ψ^* be a well-formed formula. A total order $\prec_{\psi^*} \subseteq Sub(\psi^*) \times Sub(\psi^*)$ on subformulas in ψ^* can be defined as follows:

$$\psi_1 \prec_{\psi^*} \psi_2 : \Longleftrightarrow \begin{cases} \mathtt{tt} & \text{if } \psi_2 = \psi^* \text{ and } \psi_2 \neq \psi_1 \\ \psi_1 \prec_{\psi'} \psi_2 & \text{if } \psi_1, \psi_2 \neq \psi^* \text{ and } \psi^* \equiv op(\psi') \\ \psi_1 \prec_{\psi_3} \psi_2 & \text{if } \psi^* \equiv op(\psi_3, \psi_4) \text{ and } \psi_1, \psi_2 \in Sub(\psi_3) \\ \psi_1 \prec_{\psi_4} \psi_2 & \text{if } \psi^* \equiv op(\psi_3, \psi_4) \text{ and } \psi_1, \psi_2 \notin Sub(\psi_3) \\ \mathtt{tt} & \text{if } \psi^* \equiv op(\psi_3, \psi_4) \text{ and} \\ & \quad \psi_1 \in Sub(\psi_3) \text{ and } \psi_2 \notin Sub(\psi_3) \\ \mathtt{ff} & \text{otherwise} \end{cases}$$

We will also want to select *finite sets* of formulas *deterministically*, therefore we extend a total ordering on elements to a total ordering on finite sets:

Definition 2.19 (Total ordering on finite sets)**.** Let S be a finite set and $<_S$ be a total ordering on S. A total ordering $<_{\mathcal{P}(S)}$ on $\mathcal{P}(S)$ can be defined as follows:

$$S_1 <_{\mathcal{P}(S)} S_2 : \Longleftrightarrow \begin{cases} \mathtt{tt} & \text{if } S_1 = \emptyset \text{ and } S_2 \neq \emptyset \\ \max_{<_S} S_1 <_S \max_{<_S} S_2 & \text{if } S_1, S_2 \neq \emptyset \text{ and} \\ & \max_{<_S} S_1 \neq \max_{<_S} S_2 \\ S_1 \setminus \{x\} <_{\mathcal{P}(S)} S_2 \setminus \{x\} & \text{if } S_1, S_2 \neq \emptyset \text{ and} \\ & x = \max_{<_S} S_1 = \max_{<_S} S_2 \end{cases}$$

The desired total ordering on formula sets is trivially induced as the set of subformulas is finite:

Corollary 2.20 (Total ordering on formula sets)**.** *Let ψ^* be a well-formed formula. Then $\prec_{\mathcal{P}(Sub(\psi^*))}$ is a total ordering on $\mathcal{P}(Sub(\psi^*))$.*

The rectification of a formula ensures for one thing that every bound variable uniquely identifies its fixed point binder. Thus one introduces fixed point maps to resolve variables into their respective binder and fixed point bodies.

Definition 2.21 (Fixed Point Maps)**.** Let $\psi \in CTL^*_\mu$ be rectified. We define two fixed point maps as follows

- $body_\psi : Bound(\psi) \to Sub(\psi)$, $X \mapsto \psi'$
- $fix_\psi : Bound(\psi) \to Sub(\psi)$, $X \mapsto \sigma X.\psi'$

where $\sigma X.\psi' \in Sub(\psi)$. Note that both maps are welldefined as ψ is rectified.

Like many other proof systems dealing with fixed point logics, one basically decomposes a formula into its immediate subformulas and whenever reaching a bound variable the respective fixed point is *unreeled*, meaning that the variable is again substituted by the fixed point body formula. This leads us to the definition of the *extended subformula*.

Definition 2.22 (Extended subformula)**.** Let ψ^* be a well-formed, closed formula and $\psi \in Sub(\psi^*)$. The set of *extended immediate subformulas*, $ESub(\psi)$, is defined as follows

$$ESub(\psi) := \begin{cases} \{X\} & \text{if } \psi \equiv \sigma X.\psi' \\ \{body_{\psi^*}(X)\} & \text{if } \psi \equiv X \\ ISub(\psi) & \text{otherwise} \end{cases}$$

Example 2.23. Consider the formula $\psi \equiv \nu X.(X \wedge p)$ as well as its subformulas w.r.t. $ISub(\star)$, $Sub(\star)$ and $ESub(\star)$:

Formula	$ISub(\star)$	$Sub(\star)$	$ESub(\star)$
$\nu X.(X \wedge p)$	$\{X \wedge p\}$	$\{\nu X.(X \wedge p), X \wedge p, X, p\}$	$\{X\}$
$X \wedge p$	$\{X, p\}$	$\{X \wedge p, X, p\}$	$\{X, p\}$
X	$\emptyset$	$\{X\}$	$\{X \wedge p\}$
p	$\emptyset$	$\{p\}$	$\emptyset$

When observing the unreeling of fixed points one is, roughly speaking, interested in which the *outermost* fixed point is that is unreeled infinitely often. Hence one needs a partial ordering on fixed points arranging them with respect to their "outermostness". We simply implement our total ordering on formulas to get an ordering on fixed points which is even total.

Definition 2.24 (Variable order)**.** Let $\psi \in CTL^*_\mu$ be rectified. The *variable order* $<_\psi$ is defined as follows

$$X <_\psi Y :\iff fix_\psi(X) \prec_\psi fix_\psi(Y)$$

By definition of the total ordering it is obvious that a fixed point variable can only be free in the context of another fixed point iff the later fixed point is a subformula of the former fixed point:

Corollary 2.25 (Free occurrence)**.** Let $\psi \in CTL^*_\mu$ be rectified and $X <_\psi Y$. Then $X \notin Free(body_\psi(Y))$.

The total ordering on bound variables induces index maps assigning each fixed point variable the number of smaller fixed point variables.

Definition 2.26 (Variable-index maps)**.** Let $\psi \in CTL^*_\mu$ be rectified. For both fixed point types we define the variable-index maps, ind^ψ_μ and ind^ψ_ν, as well as a general variable-index map ind^ψ as follows:

$$ind^\psi_\mu : V_\mu \to \{1, ..., |V_\mu|\},\ X \mapsto |\{Y \in V_\mu \mid Y \leq_\psi X\}|$$

$$ind^\psi_\nu : V_\nu \to \{1, ..., |V_\nu|\},\ X \mapsto |\{Y \in V_\nu \mid Y \leq_\psi X\}|$$

$$ind^\psi : \to \{1, ..., |V|\},\ X \mapsto |\{Y \in V \mid Y \leq_\psi X\}|$$

where $V_\sigma = Bound_\sigma(\psi)$ and $V = Bound(\psi)$.

Consider that all three maps are bijective since $<_\psi$ is a strict total order on $Bound(\psi)$. Therefore the inverse maps var^ψ_μ, var^ψ_ν and var^ψ are well-defined.

As we will implement parity games to solve many of the decision problems on the one hand and to derive intrinsic determinism results on the other hand, we need to assign each fixed point variable a priority so that greater fixed points are assigned greater priorities as well as least fixed points are mapped to odd priorities and greatest fixed points are mapped to even priorities.

Definition 2.27 (Priority map)**.** The *priority mapping*, pri^ψ, on bound variables in ψ is defined as follows:

$$pri^\psi : X \mapsto 2 \cdot |\{Y \mid Y <_\psi X\}| + par^\psi(X)$$

where the *parity mapping* par^ψ maps ν-fixed points (μ-fixed points) to 0 (1)

Example 2.28. Consider the following formula

$$\psi \equiv \nu X.\,((\mu Y.X \wedge Y) \vee (\nu Z.X \wedge Z))$$

It induces the total ordering $\prec_\psi$ on subformulas

$$\begin{aligned}\nu X.\,((\mu Y.X \wedge Y) \vee (\nu Z.X \wedge Z)) &\succ_\psi (\mu Y.X \wedge Y) \vee (\nu Z.X \wedge Z)\\ &\succ_\psi \nu Z.X \wedge Z \succ_\psi X \wedge Z \succ_\psi X \succ_\psi Z\\ &\succ_\psi \mu Y.X \wedge Y \succ_\psi X \wedge Y \succ_\psi Z\end{aligned}$$

which induces the total ordering $<_\psi$ on bound variables

$$X >_\psi Z >_\psi Y$$

leading to these mappings:

	X	Y	Z
ind_ψ	3	1	2
pri_ψ	4	1	2

2.2 Negation

We demanded in the previous chapter that well-formed formulas particularly ensure that each bound variable is under an even number of negations under its quantifier. Otherwise we would have to handle non-monotone functions which in general are not even endued with fixed points.

Although all well-formed formulas do not contain any non-monotone fixed point functions, one is interested in also dealing *locally* with monotone functions, e.g. when proving properties with respect to monotonicity by induction on the structure of the formula. That is, if for instance a bound variable is under two negations under its quantifier, say $\mu.(\ldots\neg\ldots\neg\ldots X)$, there is a non-monotone subfunction, namely $\neg\ldots X$; hence we want all negations to be as deep down in the parse tree as possible.

Lemma 2.29 (Negation pushing)**.** *Let ψ, ψ_1 and ψ_2 be formulas. Then:*

1. $\neg \bigcirc \psi \equiv \bigcirc \neg \psi$
2. $\neg\neg\psi \equiv \psi$
3. $\neg(\psi_1 \vee \psi_2) \equiv \neg\psi_1 \wedge \neg\psi_2$
4. $\neg(\psi_1 \wedge \psi_2) \equiv \neg\psi_1 \vee \neg\psi_2$
5. $\neg E\psi \equiv A\neg\psi$
6. $\neg A\psi \equiv E\neg\psi$
7. $\neg\mu X.\psi \equiv \nu X.\neg\psi[\neg X/X]$
8. $\neg\nu X.\psi \equiv \mu X.\neg\psi[\neg X/X]$

The former lemma induces an effective algorithm to transform a well-formed formula into an equivalent one with negations only directly before atomic propositions. This is known as the *positive normal form translation*.

Definition 2.30 (Positive normal form)**.** A well-formed formula ψ is in *positive normal form* iff for all $\psi' \in Sub(\psi)$ where $\psi' \equiv \neg\psi''$ holds that $\psi'' \equiv q$ for $q \in \mathcal{P}$. The set of all well-formed formulas in positive normal form is denoted by $CTL^*_{\mu+}$. The subset of all closed well-formed formulas in positive normal form is denoted by $CTL^*_{c\mu+}$.

We also consider the *negative translation* of a formula ψ which basically is the positive normal form of $\neg\psi$.

When comparing the positive normal form of ψ with its negative translation, the duality of all logic constructs ($\wedge$ versus $\vee$, E versus A, μ versus ν, etc.) becomes apparent. This conformity of *syntactical* and *semantical* negation will be very helpful to transform syntactical proof trees for a formula in positive normal form into proof trees for a formula with negated meaning.

Definition 2.31 (Positive normal form translation)**.** Let ψ be a formula. The *positive normal form translation* of ψ, written as $\psi^{\oplus}$, as well as the *negative translation* of ψ, written as $\psi^{\ominus}$, are inductively defined as follows:

$$\psi^{\oplus} := \begin{cases} \psi_1^{\ominus} & \text{if } \psi \equiv \neg\psi_1 \\ op(\psi_1^{\oplus}, \psi_2^{\oplus}) & \text{if } \psi \equiv op(\psi_1, \psi_2) \\ op(\psi_1^{\oplus}) & \text{if } \psi \equiv op(\psi_1),\ op \not\equiv \neg \\ \psi & \text{if } \psi \in \mathcal{P} \cup \mathcal{V} \end{cases}$$

$$\psi^{\ominus} := \begin{cases} \psi_1^{\oplus} & \text{if } \psi \equiv \neg\psi_1 \\ \psi_1^{\ominus} \wedge \psi_2^{\ominus} & \text{if } \psi \equiv \psi_1 \vee \psi_2 \\ \psi_1^{\ominus} \vee \psi_2^{\ominus} & \text{if } \psi \equiv \psi_1 \wedge \psi_2 \\ \bigcirc\psi_1^{\ominus} & \text{if } \psi \equiv \bigcirc\psi_1 \\ A\psi_1^{\ominus} & \text{if } \psi \equiv E\psi_1 \\ E\psi_1^{\ominus} & \text{if } \psi \equiv A\psi_1 \\ \mu X.(\psi_1[\neg X/X])^{\ominus} & \text{if } \psi \equiv \nu X.\psi_1 \\ \nu X.(\psi_1[\neg X/X])^{\ominus} & \text{if } \psi \equiv \mu X.\psi_1 \\ \neg\psi & \text{if } \psi \in \mathcal{P} \cup \mathcal{V} \end{cases}$$

Example 2.32. Consider the formula

$$\psi \equiv \neg\nu X.(p \wedge E(q \vee \bigcirc X)) \vee \mu Y.(q \vee \neg \bigcirc \neg Y)$$

its positive normal form translation

$$\psi^{\oplus} \equiv \mu X.(\neg p \vee A(\neg q \wedge \bigcirc X)) \vee \mu Y.(q \vee \bigcirc Y)$$

as well as ist negative translation

$$\psi^{\ominus} \equiv \nu X.(p \wedge E(q \vee \bigcirc X)) \wedge \nu Y.(\neg q \wedge \bigcirc Y)$$

Apparently all negations only occur directly before atomic propositions and the negative translation is syntactically (as well as semantically) dual to the positive normal form.

By induction on the structure of a formula and by application of Lemma 2.29 it is not hard to see that the positive normal form translation actually produces equivalent well-formed formulas in positive normal form.

Corollary 2.33 (Positive normal form equivalence)**.** *Let ψ be a well-formed formula s.t. every free variable is under an even number of negations. Then $\psi^{\oplus}$ is well-formed and in positive normal form with $\psi \models\!\models \psi^{\oplus}$.*

Similarly, it is easy to see that the negative translation results in a syntactical and semantical equivalent formula.

Corollary 2.34 (Negative translation properties)**.** *Let ψ be a well-formed formula s.t. every free variable is under an even number of negations. Then $\psi^{\ominus}$ is well-formed and in positive normal form with $\neg\psi \models\!\models \psi^{\ominus}$ and $\psi^{\oplus} = (\psi^{\ominus})^{\ominus}$.*

Since each well-formed formula can be effectively transformed into an equivalent well-formed formula in positive normal form, it is no real restriction to demand formulas to be in positive normal form from now on.

2.3 Fixed points

This chapter recalls some fundamental principles of fixed points and advances to their deployment concerning the least and greatest fixed point operators in CTL^*_μ.

Typical fixed point structures rely on *complete lattices*. Roughly spoken, a complete lattice is a partially ordered set which is closed under arbitrary infimum and supremum operations.

Definition 2.35 (Lattice)**.** Let $(S, \leq)$ be a partially ordered set and $A \subseteq S$ be a subset of S.
An element $s \in S$ is called a *supremum* of A iff

1. $\forall a \in A : a \leq s$
2. $(\exists s' \forall a \in A : a \leq s') \Rightarrow s \leq s'$

Whenever the supremum of A exists, it is denoted by $\sqcup A$.
Dually an element $i \in S$ is called a *infimum* of A iff

1. $\forall a \in A : i \leq a$

2. $(\exists i' \forall a \in A : i' \leq a) \Rightarrow i' \leq i$

Whenever the infimum of A exists, it is denoted by $\sqcap A$.

A pair $(S, \leq)$ is called *lattice* iff $\{\sqcup\{x, y\}, \sqcap\{x, y\}\} \subseteq S$ for every $x, y \in S$. Hence a lattice is a partially ordered set s.t. the infimum and the supremum of every subset are defined.

A lattice $(S, \leq)$ is called *complete* iff $\forall A \subseteq S : \{\sqcup A, \sqcap A\} \subseteq S$.

It might seem to be inaccurately defined regarding infimum and supremum of the empty set, but a complete lattice is intentionally supposed to contain infimum and supremum of any subset of the universe.

Example 2.36 (Some lattices)**.** Consider the set of natural numbers $\mathbb{N}$ with the ordinary ordering $\leq$. $(\mathbb{N}, \leq)$ is a lattice, but not a complete one.

Let A be an arbitrary set. The power set 2^A with the partial ordering $\subseteq$ on subsets of A, $(2^A, \subseteq)$, is a complete lattice.

Particularly the latter example implies that our semantical sets are a complete lattice:

Corollary 2.37. *Let $\mathcal{T}$ be an LTS. $(2^{\Pi(\mathcal{T})}, \subseteq)$ is a complete lattice.*

A *fixed point* with respect to a map $f : S \to S$ is an element $x \in S$ with $f(x) = x$. Additionally we consider *pre-* and *post-fixed points*.

Definition 2.38 (Fixed points)**.** Let $(S, \leq)$ be a lattice and f be a map $f : S \to S$ on S. An element $x \in S$ where

1. $f(x) = x$ is called a *fixed point*

2. $f(x) \leq x$ is called a *pre-fixed point*

3. $f(x) \geq x$ is called a *post-fixed point*

Fixed points in the context of this thesis are only considered with respect to monotone maps.

Definition 2.39 (Monotonicity)**.** Let $(S, \leq)$ be a lattice and f be a map $f : S \to S$ on S. The map f is called *monotone* iff $\forall x, y \in S : x \leq y \Rightarrow f(x) \leq f(y)$.

The following crucial theorem states that least and greatest fixed points of monotone maps always exist uniquely in complete lattices.

Theorem 2.40 (Knaster-Tarski [Tar55])**.** *Let $(S, \leq)$ be a complete lattice and $f : S \to S$ be a monotone map on S. The least fixed point of f, denoted μf, exists uniquely and is the infimum of all pre-fixed points. Dually, the greatest fixed point νf exists uniquely and is the supremum of all post-fixed points.*

$$\mu f := \sqcap\{x \in S \mid f(x) \leq x\}$$

$$\nu f := \sqcup\{x \in S \mid x \leq f(x)\}$$

To determine least and greatest fixed points of a monotone map one usually applies *fixed point iteration*: Starting with the empty respectively the whole set one iterates the map until the respective fixed point is reached.

Definition 2.41 (Fixed point approximants)**.** Let $(S, \leq)$ be a complete lattice and $f : S \to S$ be a monotone map on S. The fixed point approximants are defined for all ordinals as follows

$$\mu^0 f := \sqcap S \qquad \nu^0 f := \sqcup S$$

$$\mu^{\alpha+1} f := f(\mu^\alpha f) \qquad \nu^{\alpha+1} f := f(\nu^\alpha f)$$

$$\mu^\lambda f := \bigsqcup_{\alpha<\lambda} \mu^\alpha f \qquad \nu^\lambda f := \bigsqcap_{\alpha<\lambda} \nu^\alpha f$$

where $\alpha \in \mathbb{O}rd$ and λ being a limit ordinal.

Fixed point iteration is sound in the sense that it is not possible to exceed the respective fixed point.

Lemma 2.42 (Approximant boundary)**.** *Let $(S, \leq)$ be a complete lattice and $f : S \to S$ be a monotone map on S. Then $\mu^\alpha f \leq \mu f$ and $\nu^\alpha f \geq \nu f$ for all $\alpha \in \mathbb{O}rd$.*

Lemma 2.43 (Approximant monotonicity)**.** *Let $(S, \leq)$ be a complete lattice and $f : S \to S$ be a monotone map on S. Then for all $\alpha \leq \beta \in \mathbb{O}rd$ holds that $\mu^\alpha f \leq \mu^\beta f$ and $\nu^\alpha f \geq \nu^\beta f$.*

Finally, fixed point iteration is complete in the sense that there is an ordinal number so that the iteration stops.

Lemma 2.44 (Fixed point approximation)**.** *Let S be a proper set (and not a class), $(S, \leq)$ be a complete lattice and $f : S \to S$ be a monotone map on S. There are $\alpha, \beta \in \mathbb{O}rd$ s.t. $\mu^\alpha f = \mu f$ and $\nu^\beta f = \nu f$.*

To make use of all preceding results we need to show that the fixed point maps occurring in formulas are monotone. This is one of the reasons why we restrict ourselves to well-formed formulas in positive normal form.

Lemma 2.45 (Monotonicity of substitution)**.** *Let $\mathcal{T}$ be a labelled transition system, ρ an environment, ψ a formula in positive normal form and $X \in \mathcal{V}$. Then $f : 2^{\Pi(\mathcal{T})} \to 2^{\Pi(\mathcal{T})}$, $A \mapsto [\![\psi]\!]^{\mathcal{T}}_{\rho[X \mapsto A]}$ is monotone in $(2^{\Pi(\mathcal{T})}, \subseteq)$.*

Since the semantical sets are a complete lattice and fixed point maps occurring in positive well-formed formulas are monotone (former Lemma 2.45) we derive the following corollary, applying Theorem 2.40 (Knaster-Tarski):

Corollary 2.46 (Fixed Point Substitution)**.** *Let ψ be a formula in positive normal form and ρ be an environment. Then the following holds:*

$$[\![\psi[\sigma X.\psi/X]]\!]^{\mathcal{T}}_\rho = [\![\psi]\!]^{\mathcal{T}}_{\rho[X \mapsto [\![\sigma X.\psi]\!]^{\mathcal{T}}_\rho]} = [\![\sigma X.\psi]\!]^{\mathcal{T}}_\rho$$

2.4 Approximants

The *abstract* definition of approximants can be transferred to syntactical as well as semantical definitions. Although limit ordinals lead to formulas of infinite size, this is not a real issue since the *depth* of the formulas remains finite and the utilization of formula approximants will be restricted to theoretical propositions and not used in any of the decision procedures.

The reason why one is interested in formula approximants is, roughly spoken, as follows: The tableaux systems in this thesis basically decompose formulas iteratively and substitute each fixed point variable by its fixed point body. Hence one possibly decomposes certain parts of a formula infinitely often. Now each tableaux has to verify two properties: Whether the decomposition is *locally* correct and whether the decomposition is *globally correct*.

The local correctness can be intrinsically ensured by simply designing sane tableaux rules. For global correctness, one has to check that the infinite behaviour of a formula is alright, meaning that a least fixed point, for instance, is not unfolded infinitely often without termination. To *semantically prove* the correctness of this observation, we annotate all least fixed points in a formula with approximants that cannot be unfolded (and hence decrease) infinitely often due to the wellfoundedness of ordinal numbers.

Definition 2.47 (Approximants)**.** Approximants of a fixed point formula $\sigma X.\psi$ are defined for all ordinals:

$$\mu^0 X.\psi := \mathtt{ff} \qquad \nu^0 X.\psi := \mathtt{tt}$$

$$\mu^{\alpha+1} X.\psi := \psi[\mu^\alpha X.\psi/X] \qquad \nu^{\alpha+1} X.\psi := \psi[\nu^\alpha X.\psi/X]$$

$$\mu^\lambda X.\psi := \bigvee_{\alpha<\lambda} \mu^\alpha X.\psi \qquad \nu^\lambda X.\psi := \bigwedge_{\alpha<\lambda} \nu^\alpha X.\psi$$

where $\alpha \in \mathbb{O}rd$ and λ being a limit ordinal.

The semantical continuation is defined as infinite union or infinite intersection respectively:

$$[\![\bigvee_{\alpha<\lambda} \mu^\alpha X.\psi]\!]^{\mathcal{T}}_\rho := \bigcup_{\alpha<\lambda} [\![\mu^\alpha X.\psi]\!]^{\mathcal{T}}_\rho \qquad [\![\bigwedge_{\alpha<\lambda} \nu^\alpha X.\psi]\!]^{\mathcal{T}}_\rho := \bigcap_{\alpha<\lambda} [\![\nu^\alpha X.\psi]\!]^{\mathcal{T}}_\rho$$

Not very surprisingly the former section yields that fixed point *can* be approximated:

Lemma 2.48 (σ-Approximants)**.** *For every labelled transition system $\mathcal{T}$ and every environment ρ holds:*

$$[\![\mu X.\psi]\!]^{\mathcal{T}}_\rho = \bigcup_\alpha [\![\mu^\alpha X.\psi]\!]^{\mathcal{T}}_\rho$$

$$[\![\nu X.\psi]\!]^{\mathcal{T}}_\rho = \bigcap_\alpha [\![\nu^\alpha X.\psi]\!]^{\mathcal{T}}_\rho$$

Proof. By Lemma 2.45 $A \mapsto [\![\psi]\!]^{\mathcal{T}}_{\rho[X\mapsto A]}$ is monotone.

By Lemma 2.44 there is some β s.t.

$$[\![\mu X.\psi]\!]^{\mathcal{T}}_\rho \subseteq [\![\mu^\beta X.\psi]\!]^{\mathcal{T}}_\rho \quad \text{resp.} \quad [\![\nu X.\psi]\!]^{\mathcal{T}}_\rho \supseteq [\![\nu^\beta X.\psi]\!]^{\mathcal{T}}_\rho$$

By Lemma 2.42 for all β holds that

$$[\![\mu X.\psi]\!]^{\mathcal{T}}_\rho \supseteq [\![\mu^\beta X.\psi]\!]^{\mathcal{T}}_\rho \quad \text{resp.} \quad [\![\nu X.\psi]\!]^{\mathcal{T}}_\rho \subseteq [\![\nu^\beta X.\psi]\!]^{\mathcal{T}}_\rho$$

□

We observe by the former lemma that unfolding of fixed points with approximants decreases the respective ordinal number. This property is necessary to argue that unfolding of fixed points with approximants cannot be done infinitely often.

Corollary 2.49 (Unfolding Fixed Points)**.** *Let $\mathcal{T}$ be an LTS, ρ be an environment, ψ be a formula and $\pi \in \Pi(\mathcal{T})$ be a path. Then:*

1. *$\mathcal{T}, \pi \models_\rho \mu X.\psi$ iff there is an $\alpha \in \mathbb{O}rd$ s.t. $\mathcal{T}, \pi \models_\rho \mu^\alpha X.\psi$*
2. *$\mathcal{T}, \pi \not\models_\rho \nu X.\psi$ iff there is an $\alpha \in \mathbb{O}rd$ s.t. $\mathcal{T}, \pi \not\models_\rho \nu^\alpha X.\psi$*
3. *If $\mathcal{T}, \pi \models_\rho \mu^\beta X.\psi$ then there is a $\gamma < \beta$ s.t. $\mathcal{T}, \pi \models_\rho \psi[\mu^\gamma X.\psi/X]$*
4. *If $\mathcal{T}, \pi \not\models_\rho \nu^\beta X.\psi$ then there is a $\gamma < \beta$ s.t. $\mathcal{T}, \pi \not\models_\rho \psi[\nu^\gamma X.\psi/X]$*

Proof. The first two claims follow directly:

$$\pi \in [\![\mu X.\psi]\!]^{\mathcal{T}}_\rho \overset{2.48}{\Leftrightarrow} \pi \in \bigcup_\alpha [\![\mu^\alpha X.\psi]\!]^{\mathcal{T}}_\rho \Leftrightarrow \exists \alpha \in \mathbb{O}rd : \pi \in [\![\mu^\alpha X.\psi]\!]^{\mathcal{T}}_\rho$$

$$\pi \notin [\![\nu X.\psi]\!]^{\mathcal{T}}_\rho \overset{2.48}{\Leftrightarrow} \pi \notin \bigcap_\alpha [\![\nu^\alpha X.\psi]\!]^{\mathcal{T}}_\rho \Leftrightarrow \exists \alpha \in \mathbb{O}rd : \pi \notin [\![\nu^\alpha X.\psi]\!]^{\mathcal{T}}_\rho$$

To prove the second claim let $\beta \in \mathbb{O}rd$. If β is no limit ordinal, set $\gamma := \beta - 1$ since in this case the claim follows by definition. For β being a limit ordinal the following holds:

$$\pi \in [\![\mu^\beta X.\psi]\!]^{\mathcal{T}}_\rho \Rightarrow \pi \in \bigcup_{\alpha<\beta} [\![\mu^\alpha X.\psi]\!]^{\mathcal{T}}_\rho \Rightarrow \exists \gamma < \beta : \pi \in [\![\mu^{\gamma+1} X.\psi]\!]^{\mathcal{T}}_\rho$$

$$\Rightarrow \exists \gamma < \beta : \pi \in [\![\psi[\mu^\gamma X.\psi/X]]\!]^{\mathcal{T}}_\rho$$

Similarly for the greatest fixed point:

$$\pi \notin [\![\nu^\beta X.\psi]\!]^{\mathcal{T}}_\rho \Rightarrow \pi \notin \bigcap_{\alpha<\beta} [\![\nu^\alpha X.\psi]\!]^{\mathcal{T}}_\rho \Rightarrow \exists \gamma < \beta : \pi \notin [\![\nu^{\gamma+1} X.\psi]\!]^{\mathcal{T}}_\rho$$

$$\Rightarrow \exists \gamma < \beta : \pi \notin [\![\psi[\nu^\gamma X.\psi/X]]\!]^{\mathcal{T}}_\rho$$

□

2.5 Signatures

As outlined in the preceding chapter, we will be annotating fixed points with ordinal numbers to have an upper bound for unfolding the respective fixed points. In general, one has to deal with several nested fixed points with some of them - either all μ- or all ν-fixed points - having an annotation.

To keep track of the unfolding progress of the annotated fixed points, we have to supply the fixed point annotations in a formula with an ordering w.r.t. the occurring fixed points. It is our goal to provide a wellfounded measure that is decreased whenever a "bad" fixed point is unfolded. When unfolding an annotated fixed point X being atop an other annotated fixed point Y, the annotation of X is decreased and the annotation of Y needs to be invalidated. The new annotation of Y will be possibly greater than it was before unfolding X.

Therefore one introduces a lexicographic ordering on annotations to receive a wellfounded measure that is decreased no matter which annotated fixed point is unfolded.

Definition 2.50 (σ-Signatures)**.** Let $\psi \in CTL^*_{\mu+}$. A *μ-signature for ψ* is a tuple $\zeta = (\alpha_1, ..., \alpha_n) \in \mathbb{O}rd^n$ where $n = |Bound_\mu(\psi)|$. Likewise a *ν-signature for ψ* is a tuple $\zeta = (\alpha_1, ..., \alpha_m) \in \mathbb{O}rd^m$ where $m = |Bound_\nu(\psi)|$. We denote the set of μ- resp. ν-signatures by $Sig_\mu(\psi)$ resp. $Sig_\nu(\psi)$.

Signatures induce a lexicographic ordering $(\mathbb{O}rd^k, <)$ as follows (where $k = n, m$):

$$\zeta < \zeta' \iff \exists i.((\zeta_i < \zeta'_i) \wedge \forall j > i.(\zeta_j = \zeta'_j))$$

By $\zeta^{j]}$ we denote the projection to the first j components of a signature ζ, i.e. $\zeta^{j]} := (\zeta(1), ..., \zeta(j))$. By $\zeta[j \mapsto \alpha]$ we denote the substitution of the j-th component in ζ by α.

Clearly, a set of signatures is wellfounded as the signature's length is finite with each component being wellfounded itself.

At next, we extend the definition of syntactical approximation of *one* fixed point to a definition of syntactical approximation of a whole formula possibly containing more than one fixed point. As before, we are only interested in the approximation of one kind of fixed points with respect to a signature and thus we define the *outer approximation* for each fixed point kind.

Definition 2.51 (Outer Approximation)**.** Let $\psi \in CTL^*_{\mu+}$, $n = |Bound(\psi)|$ and $X_i \equiv var^\psi(i)$.
The *outer μ-approximation* of φ, $\uparrow^\psi_\zeta \varphi$, is defined as follows for $\zeta \in Sig_\mu(\psi)$:

$$\uparrow^\psi_\zeta \varphi := (...(\varphi[s_\mu(\zeta, 1)/X_1])...)[s_\mu(\zeta, n)/X_n]$$

where

$$s_\mu(\zeta, i) := \begin{cases} fix_\psi(X_i) & \text{if } X_i \in Bound_\nu(\psi) \\ \mu^{\zeta(ind^\psi_\mu(X_i))} X_i.body_\psi(X_i) & \text{otherwise} \end{cases}$$

The *outer ν-approximation* of φ, $\downarrow^\psi_\zeta \varphi$, is defined likewise for $\zeta \in Sig_\nu(\psi)$:

$$\downarrow^\psi_\zeta \varphi := (...(\varphi[s_\nu(\zeta, 1)/X_1])...)[s_\nu(\zeta, n)/X_n]$$

where

$$s_\nu(\zeta, i) := \begin{cases} fix_\psi(X_i) & \text{if } X_i \in Bound_\mu(\psi) \\ \nu^{\zeta(ind^\psi_\nu(X_i))} X_i.body_\psi(X_i) & \text{otherwise} \end{cases}$$

Additionally we define the *pseudo approximation* of φ, $\updownarrow^{\psi} \varphi$:

$$\updownarrow^{\psi} \varphi := (...(\varphi[s_\sigma(1)/X_1])...)[s_\sigma(n)/X_n]$$

where $s_\sigma(i) := fix_\psi(X_i)$.

The *pseudo approximation* is no real approximation in the sense that the fixed points are annotated with numbers of any kind; each free fixed point variable is substituted by its fixed point binder following the associated ordering ensuring that the resulting formula is closed.

This will be used in meta proofs regarding correctness and completeness of the tableaux systems whenever there are subformulas containing free variables which *semantically* need to be substituted by their respective fixed point binders.

Example 2.52. Consider the following formula

$$\psi \equiv \mu X.\nu Y.(X \vee Y \vee \mu Z.(X \wedge Z))$$

inducing the total ordering $<_\psi$ on bound variables

$$X >_\psi Y >_\psi Z$$

with $ind^{\psi}_{\mu}(X) = 2$ and $ind^{\psi}_{\mu}(Z) = 1$.
The outer μ-approximation of Z w.r.t. $(13, 37)$ is to be computed as follows:

$$\begin{aligned}
\uparrow^{\psi}_{(13,37)} Z &= Z[\mu^{13}Z.(X \wedge Z)/Z][\nu Y.(...)/Y][\mu^{37}X.(...)/X] \\
&= \mu^{13}Z.(X \wedge Z)[\nu Y.(X \vee Y \vee \mu Z.(X \wedge Z))/Y][\mu^{37}X.(...)/X] \\
&= \mu^{13}Z.(X \wedge Z)[\mu^{37}X.\nu Y.(X \vee Y \vee \mu Z.(X \wedge Z))/X] \\
&= \mu^{13}Z.(\mu^{37}X.\nu Y.(X \vee Y \vee \mu Z.(X \wedge Z)) \wedge Z)
\end{aligned}$$

To see that all three kinds of outer approximations indeed result in closed formulas, we show the following lemma:

Lemma 2.53 (Outer approximation of closed formulas is closed)**.** *Let $\psi \in CTL^*_{\mu+}$, $\zeta_\mu \in Sig_\mu(\psi)$, $\zeta_\nu \in Sig_\nu(\psi)$ and $\varphi \in Sub(\psi)$. If ψ is closed then $\uparrow^\psi_{\zeta_\mu} \varphi$, $\downarrow^\psi_{\zeta_\nu} \varphi$ as well as $\updownarrow^\psi \varphi$ are also closed.*

Proof. We prove the following claim by induction on i:

$$\forall j \leq i : X_j \notin Free((...(\varphi[s_\sigma(1)/X_1])...)[s_\sigma(i)/X_i])$$

Case $i = 1$ is obvious. For $i \rightsquigarrow i+1$ let $j \leq i+1$ be arbitrary. If $j = i+1$ the claim follows directly by definition of substitution. For $j \leq i$ by induction hypothesis holds that $X_j \notin Free((...(\varphi[s_\sigma(1)/X_1])...)[s_\sigma(i)/X_i])$. Hence the claim follows by Corollary 2.25. □

The following lemma extends the approximation properties with respect to a single fixed point to signatures.

Lemma 2.54 (Outer approximation properties)**.** *Let $\psi \in CTL^*_{\mu+}$, π be a path and $X \in Bound(\psi)$, $Y \in Bound_\mu(\psi)$, $Z \in Bound_\nu(\psi)$, $\zeta_\mu \in Sig_\mu(\psi)$ and $\zeta_\nu \in Sig_\nu(\psi)$. Then:*

1. $\pi \models \updownarrow^\psi \sigma X.body_\psi(X)$ *iff* $\pi \models \updownarrow^\psi X$ *iff* $\pi \models \updownarrow^\psi body_\psi(X)$
2. $\pi \models \uparrow^\psi_{\zeta_\mu} \nu Z.body_\psi(Z)$ *iff* $\pi \models \uparrow^\psi_{\zeta_\mu} Z$ *iff* $\pi \models \uparrow^\psi_{\zeta_\mu} body_\psi(Z)$
3. $\pi \not\models \downarrow^\psi_{\zeta_\nu} \mu Y.body_\psi(Y)$ *iff* $\pi \not\models \downarrow^\psi_{\zeta_\nu} Y$ *iff* $\pi \not\models \downarrow^\psi_{\zeta_\nu} body_\psi(Y)$
4. *If $\pi \models \uparrow^\psi_{\zeta_\mu} Y$ then exists some $\alpha < \zeta_\mu(k)$ s.t. $\pi \models \uparrow^\psi_{\zeta'} body_\psi(Y)$ where $\zeta' := \zeta_\mu[k \mapsto \alpha]$ and $k := ind^\psi_\mu(Y)$*
5. *If $\pi \not\models \downarrow^\psi_{\zeta_\nu} Z$ then exists some $\alpha < \zeta_\nu(k)$ s.t. $\pi \not\models \downarrow^\psi_{\zeta'} body_\psi(Z)$ where $\zeta' := \zeta_\nu[k \mapsto \alpha]$ and $k := ind^\psi_\nu(Z)$*
6. *If $\pi \models \uparrow^\psi_{\zeta_\mu} \mu Y.body_\psi(Y)$ then exists some $\alpha \in \mathbb{O}rd$ s.t. $\pi \models \uparrow^\psi_{\zeta'} body_\psi(Y)$ where $\zeta' := \zeta_\mu[k \mapsto \alpha]$ and $k := ind^\psi_\mu(Y)$*
7. *If $\pi \not\models \downarrow^\psi_{\zeta_\nu} \nu Z.body_\psi(Z)$ then exists some $\alpha \in \mathbb{O}rd$ s.t. $\pi \not\models \downarrow^\psi_{\zeta'} body_\psi(Z)$ where $\zeta' := \zeta_\nu[k \mapsto \alpha]$ and $k := ind^\psi_\nu(Z)$*

Proof. As follows:

1. $\pi \models\updownarrow^{\psi} \sigma X.body_{\psi}(X)$ iff $\pi \models fix_{\psi}(X)$ iff $\pi \models\updownarrow^{\psi} X$ iff $\pi \models fix_{\psi}(X)$ iff $\pi \models body_{\psi}(X)[fix_{\psi}(X)/X]$ by Corollary 2.46 iff $\pi \models\updownarrow^{\psi} body_{\psi}(X)$.

2. and 3. can be shown the same way.

4. Let $\pi \models\uparrow^{\psi}_{\zeta_\mu} Y$, i.e. $\pi \models\uparrow^{\psi}_{\zeta_\mu} X_k$. We abbreviate $[s_\mu(\zeta, i)/X_i]$ by t^i_ζ. Then:

$$\begin{aligned}
&\pi \models X_k t^1_\zeta ... t^n_\zeta \\
\Rightarrow &\pi \models X_k t^k_\zeta ... t^n_\zeta \\
\Rightarrow &\pi \models \mu^{\zeta(k)} X_k.body_\psi(X_k) t^{k+1}_\zeta ... t^n_\zeta \\
\overset{2.49}{\Rightarrow} &\pi \models body_\psi(X_k) t^{k+1}_\zeta ... t^n_\zeta [\mu^\alpha X_k.body_\psi(X_k) t^{k+1}_\zeta ... t^n_\zeta / X_k] \\
\overset{2.25}{\Rightarrow} &\pi \models body_\psi(X_k)[\mu^\alpha X_k.body_\psi(X_k) t^{k+1}_\zeta ... t^n_\zeta / X_k] t^{k+1}_\zeta ... t^n_\zeta \\
\overset{2.25}{\Rightarrow} &\pi \models body_\psi(X_k)[\mu^\alpha X_k.body_\psi(X_k)/X_k] t^{k+1}_\zeta ... t^n_\zeta \\
\overset{2.25}{\Rightarrow} &\pi \models body_\psi(X_k) t^1_\zeta ... t^{k-1}_\zeta [\mu^\alpha X_k.body_\psi(X_k)/X_k] t^{k+1}_\zeta ... t^n_\zeta
\end{aligned}$$

   Hence $\pi \models\uparrow^{\psi}_{\zeta'} body_\psi(Y)$ whereas $\zeta_\mu[k \mapsto \alpha] =: \zeta' < \zeta_\mu$.

5. This can be shown the same way.

6. Let $\pi \models\uparrow^{\psi}_{\zeta_\mu} \mu Y.body_\psi(Y)$, i.e. $\pi \models\uparrow^{\psi}_{\zeta_\mu} \mu X_k.body_\psi(X_k)$. We abbreviate $[s_\mu(\zeta, i)/X_i]$ by t^i_ζ. Then the following holds:

$$\begin{aligned}
&\pi \models \mu X_k.body_\psi(X_k) t^1_\zeta ... t^n_\zeta \\
\overset{2.25}{\Rightarrow} &\pi \models \mu X_k.body_\psi(X_k) t^{k+1}_\zeta ... t^n_\zeta \\
\overset{2.49}{\Rightarrow} &\pi \models body_\psi(X_k) t^{k+1}_\zeta ... t^n_\zeta [\mu^\alpha X_k.body_\psi(X_k) t^{k+1}_\zeta ... t^n_\zeta / X_k] \\
\overset{2.25}{\Rightarrow} &\pi \models body_\psi(X_k)[\mu^\alpha X_k.body_\psi(X_k) t^{k+1}_\zeta ... t^n_\zeta / X_k] t^{k+1}_\zeta ... t^n_\zeta \\
\overset{2.25}{\Rightarrow} &\pi \models body_\psi(X_k)[\mu^\alpha X_k.body_\psi(X_k)/X_k] t^{k+1}_\zeta ... t^n_\zeta \\
\overset{2.25}{\Rightarrow} &\pi \models body_\psi(X_k) t^1_\zeta ... t^{k-1}_\zeta [\mu^\alpha X_k.body_\psi(X_k)/X_k] t^{k+1}_\zeta ... t^n_\zeta
\end{aligned}$$

   Hence $\pi \models\downarrow^{\psi}_{\zeta'} body_\psi(Y)$ whereas $\zeta' := \zeta_\mu[k \mapsto \alpha]$.

7. This can be shown the same way. □

As before with approximants of single fixed points, one proves that modelling with respect to outer approximation is related to modelling with respect to pseudo approximation.

Lemma 2.55 (Outer approximation preserves models)**.** *Let $\psi \in CTL^*_{\mu+}$, π be a path, $\varphi \in Sub(\psi)$, $\zeta_\mu \in Sig_\mu(\psi)$ and $\zeta_\nu \in Sig_\nu(\psi)$. Then:*

1. *$\pi \not\models \downarrow^\psi_{\zeta_\nu} \varphi$ implies $\pi \not\models \updownarrow^\psi \varphi$*
2. *$\pi \models \uparrow^\psi_{\zeta_\mu} \varphi$ implies $\pi \models \updownarrow^\psi \varphi$*

Proof. Define $\varphi^i_\alpha \equiv \varphi[s_\alpha(1)/X_1]...[s_\alpha(i)/X_i]$ where $\alpha \in \{\sigma, \mu, \nu\}$ and $s_\alpha(k) \equiv s_\alpha(\zeta_\alpha, k)$ for $\alpha = \mu, \nu$. It suffices to show for all $i = 0, ..., n$ that

$$[\![\varphi^i_\mu]\!]^{\mathcal{T}}_\rho \subseteq [\![\varphi^i_\sigma]\!]^{\mathcal{T}}_\rho \subseteq [\![\varphi^i_\nu]\!]^{\mathcal{T}}_\rho \text{ for all } \rho$$

since $\downarrow^\psi_{\zeta_\nu} \varphi \equiv \varphi^n_\nu$, $\updownarrow^\psi \varphi \equiv \varphi^n_\sigma$ and $\uparrow^\psi_{\zeta_\mu} \varphi \equiv \varphi^n_\mu$.

We prove the claim by induction on i (with case $i = 0$ being obvious): Case $i \rightsquigarrow i + 1$: Consider that by Lemma 2.48 the following holds

$$(*)[\![s_\mu(\zeta_\mu, i)]\!]^{\mathcal{T}}_\rho \subseteq [\![s_\sigma(i)]\!]^{\mathcal{T}}_\rho \subseteq [\![s_\nu(\zeta_\nu, i)]\!]^{\mathcal{T}}_\rho \text{ for all } \rho$$

and therefore by monotonicity:

$$\begin{aligned}
[\![\varphi^{i+1}_\mu]\!]^{\mathcal{T}}_\rho &= [\![\varphi^i_\mu[s_\mu(\zeta_\mu, i+1)/X_{i+1}]]\!]^{\mathcal{T}}_\rho \\
&\overset{IH}{\subseteq} [\![\varphi^i_\sigma[s_\mu(\zeta_\mu, i+1)/X_{i+1}]]\!]^{\mathcal{T}}_\rho \\
&\overset{(*)}{\subseteq} [\![\varphi^i_\sigma[s_\sigma(i+1)/X_{i+1}]]\!]^{\mathcal{T}}_\rho \\
&= [\![\varphi^{i+1}_\sigma]\!]^{\mathcal{T}}_\rho \\
&= [\![\varphi^i_\sigma[s_\sigma(i+1)/X_{i+1}]]\!]^{\mathcal{T}}_\rho \\
&\overset{IH}{\subseteq} [\![\varphi^i_\nu[s_\sigma(i+1)/X_{i+1}]]\!]^{\mathcal{T}}_\rho \\
&\overset{(*)}{\subseteq} [\![\varphi^i_\nu[s_\nu(\zeta_\nu, i+1)/X_{i+1}]]\!]^{\mathcal{T}}_\rho \\
&= [\![\varphi^{i+1}_\nu]\!]^{\mathcal{T}}_\rho
\end{aligned}$$

□

The following two corollaries ensure that it is possible to select immediate subformulas in such a way that they are compatible with both pseudo as well as outer approximation. This is important as we will have to pick formulas in tableaux branches long before we are able to annotate them with approximations, but at the same time we already have to make sure that the approximations exist and do not increase by selecting the respective subformula.

Corollary 2.56 (Consistent junctor approximation successors)**.** *Let $\psi \in CTL^*_{\mu+}$ and π be a path in an LTS.*
If $\pi \not\models \updownarrow^{\psi} \psi_1 \wedge \psi_2$ for two formulas ψ_1, ψ_2 then there is an $i \in \mathbb{N}$ s.t. the following holds:

1. $\pi \not\models \updownarrow^{\psi} \psi_i$
2. *If $\zeta \in Sig_\nu(\psi)$ is the least signature s.t. $\pi \not\models \downarrow^{\psi}_{\zeta} \psi_1 \wedge \psi_2$ then $\pi \not\models \downarrow^{\psi}_{\zeta} \psi_i$*

Dually if $\pi \models \updownarrow^{\psi} \psi_1 \vee \psi_2$ for two formulas ψ_1, ψ_2 then there is an $i \in \mathbb{N}$ s.t. the following holds:

1. $\pi \models \updownarrow^{\psi} \psi_i$
2. *If $\zeta \in Sig_\mu(\psi)$ is the least signature s.t. $\pi \models \uparrow^{\psi}_{\zeta} \psi_1 \vee \psi_2$ then $\pi \models \uparrow^{\psi}_{\zeta} \psi_i$*

Proof. Let $\pi \not\models \updownarrow^{\psi} \psi_1 \wedge \psi_2$.
Case 1: There is a least signature $\zeta \in Sig_\nu(\psi)$ s.t. $\pi \not\models \downarrow^{\psi}_{\zeta} \psi_1 \wedge \psi_2$. By definition there is some $i \in \{1, 2\}$ s.t. $\pi \not\models \downarrow^{\psi}_{\zeta} \psi_i$. By Lemma 2.55 we conclude $\pi \not\models \updownarrow^{\psi} \psi_i$.
Case 2: There is no least signature $\zeta \in Sig_\nu(\psi)$ s.t. $\pi \not\models \downarrow^{\psi}_{\zeta} \psi_1 \wedge \psi_2$. By definition there is some $i \in \{1, 2\}$ s.t. $\pi \not\models \updownarrow^{\psi} \psi_i$.
The case $\pi \models \updownarrow^{\psi} \psi_1 \vee \psi_2$ can be shown the same way. □

A similar result can be shown regarding path quantifications. The following corollary additionally states that path witnesses can be selected in a way that they are consistent with pseudo as well as with outer approximation.

Corollary 2.57 (Consistent quantor approximation successors)**.** *Let $\psi \in CTL^*_{\mu+}$ and s be a state in an LTS.*
If $s \not\models\updownarrow^\psi A\psi'$ for some formula ψ' then there is a path π rooting in s s.t. the following holds:

1. $\pi \not\models\updownarrow^\psi \psi'$
2. *If $\zeta \in Sig_\nu(\psi)$ is the least signature s.t. $s \not\models\downarrow^\psi_\zeta A\psi'$ then $\pi \not\models\downarrow^\psi_\zeta \psi'$*

Dually if $s \models\updownarrow^\psi E\psi'$ for some formula ψ' then there is a path π rooting in s s.t. the following holds:

1. $\pi \models\updownarrow^\psi \psi'$
2. *If $\zeta \in Sig_\mu(\psi)$ is the least signature s.t. $s \models\uparrow^\psi_\zeta E\psi'$ then $\pi \models\uparrow^\psi_\zeta \psi$*

Proof. Let $s \not\models\updownarrow^\psi A\psi'$.
Case 1: There is a least signature $\zeta \in Sig_\nu(\psi)$ s.t. $s \not\models\downarrow^\psi_\zeta A\psi'$. By definition there is a path π rooting s s.t. $\pi \not\models\downarrow^\psi_\zeta \psi'$. By Lemma 2.55 we conclude $\pi \not\models\updownarrow^\psi \psi'$.
Case 2: There is no least signature $\zeta \in Sig_\nu(\psi)$ s.t. $s \not\models\downarrow^\psi_\zeta A\psi'$. By definition there is a path π rooting s s.t. $\pi \not\models\updownarrow^\psi \psi'$. The case $s \models\updownarrow^\psi E\psi'$ can be shown the same way. □

2.6 Guarded form

Almost every tableaux system for modal logics based on unreeling formulas and unfolding fixed points has a central rule usually known as *modal rule*. All other rules simply unreel and unfold all non-temporal parts of a formula.

Each node in such a tableaux can be seen as a formula description of a specific world in a certain Kripke structure that is associated with the tableaux. All normal rules can be seen as a semantically equivalent modification of the world description. The modal rule, however, performs a step from one world to connected other worlds in the structure.

The modal rule usually can only be performed iff all formulas of the tableaux node in question are strictly modal in the sense that the highest operator of each formula is a modal operator.

As outlined before, one wants to verify two properties regarding possibly infinite tableaux to derive whether the tableaux ensures a certain property (e.g. "φ is valid"): One needs to check that each branch in a tableaux is locally and globally correct. The local correctness can easily be ensured by designing sound tableaux rules.

For the global correctness, one usually monitors the infinite behaviour of particular processes in a tableaux branch. In the context of this thesis, we are monitoring the unfolding behaviour of each formula occurring in the tableaux branch, basically, to decide whether such a monitored formula is "good" (outermost fixed point operator unfolded infinitely often is a greatest fixed point) or "bad" (outermost fixed point operator unfolded infinitely often is a least fixed point).

To be able to monitor the infinite behaviour of *each* formula occurring in a tableaux branch, one needs to ensure that the branch is *fair*, meaning that each formula finally takes a turn as a principal formula that gets unreeled in a rule application.

The problem with the modal rule is that if there is a formula that can be unreeled infinitely often without enforcing a modal operator, the modal rule cannot be applied. Consequently all strictly modal formulas being aside the formula in question are stalled, hence the whole branch is not fair.

In order to avoid this issue, we want to substitute each subformula which could be unreeled infinitely often without enforcing a modal operator with an equivalent subformula enforcing modal operators in a preprocessing step. The only possibility to unreel a formula infinitely often without seeing a modal operator is a formula containing a fixed point subformula with a bound variable having no modal operators inbetween.

This translation is better known as *guarded transformation*. It remains to be seen whether there is a guarded transformation for full CTL^*_μ; most of the time, however, one does not need a formula to be fully guarded. It usually suffices to be *state-guarded* meaning that between each bound variable and its fixed point binder is at least one modal operator or one path quantifier. Fortunately, each well-formed formula can be effectively translated into a state-

guarded formula.

Example 2.58. The following formula is not guarded due to the fact that the second occurrence of X is not under a $\bigcirc$-modality. It is state-guarded, however, having at least a path quantification inbetween:

$$\nu X.(\bigcirc X \wedge E(p \vee X))$$

The next formula is neither guarded nor state-guarded:

$$\mu X.\nu Y.(Y \vee A((p \wedge X) \vee Y))$$

An equivalent state-guarded formula (that is effectively computable) looks as follows:

$$\mu X.A((p \wedge X) \vee \nu Y.A((p \wedge X) \vee Y))$$

We proceed with the formal definition of state-guardedness demanding that between a fixed point and a bound variable has to be at least one modal or one path operator.

Definition 2.59 (State-guarded)**.** A formula is *state-guarded*, $sguarded(\psi)$, iff $sguarded(\psi, \emptyset)$ holds, where $sguarded(\psi, A)$ is inductively defined as follows:

$$sguarded(\psi, A) := \begin{cases} sguarded(\psi', \emptyset) & \text{if } \psi \equiv \bigcirc\psi', A\psi', E\psi' \\ X \notin A & \text{if } \psi \equiv X,\ X \in \mathcal{V} \\ sguarded(\psi', A \cup \{X\}) & \text{if } \psi \equiv \sigma X.\psi' \\ \forall \psi' \in ISub(\psi).\, sguarded(\psi', A) & \text{otherwise} \end{cases}$$

One substep of the state-guarded translation is the *flattening* of fixed points. The flattening of a least (resp. greatest) fixed point simply substitutes each unguarded variable by `ff` (resp. `tt`).

Definition 2.60 (Flattening)**.** Let ψ be a formula and $X \in \mathcal{V}$ be a variable. The *flattening of* ψ *w.r.t.* X, $fl(\psi, X, \sigma)$, is inductively defined as follows:

$$fl(\psi, X, \sigma) := \begin{cases} op(fl(\psi_1, X, \sigma), fl(\psi_2, X, \sigma)) & \text{if } \psi \equiv op(\psi_1, \psi_2) \\ \texttt{ff} & \text{if } \psi \equiv X \text{ and } \sigma = \mu \\ \texttt{tt} & \text{if } \psi \equiv X \text{ and } \sigma = \nu \\ \psi & \text{otherwise} \end{cases}$$

The following Lemma ensures that the flattening of a formula *semantically* only affects paths that are contained in the free variable.

Lemma 2.61 (Flattening semantics)**.** *Let ψ be a formula, $X \in \mathcal{V}$ be a variable, $\mathcal{T}$ be an LTS, ρ be an environment and $U \subseteq \Pi(\mathcal{T})$. Then:*

$$[\![fl(\psi, X, \nu)]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cap U \subseteq [\![\psi]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \subseteq [\![fl(\psi, X, \mu)]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cup U$$

Proof. By structural induction on ψ. For some ψ' we abbreviate $fl(\psi', X, \sigma)$ by $(\psi')^\sigma$.
Case $\psi \equiv \psi_1 \wedge \psi_2$:

$$\begin{aligned}
[\![\psi^\nu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cap U &= [\![\psi_1^\nu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cap [\![\psi_2^\nu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cap U \\
&\overset{IH}{\subseteq} [\![\psi_1]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cap [\![\psi_2]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \subseteq [\![\psi]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \\
[\![\psi]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} &= [\![\psi_1]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cap [\![\psi_2]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \\
&\overset{IH}{\subseteq} ([\![\psi_1^\mu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cup U) \cap ([\![\psi_2^\mu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cup U) \\
&= ([\![\psi_1^\mu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cap [\![\psi_2^\mu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]}) \cup U = [\![\psi^\mu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cup U
\end{aligned}$$

Case $\psi \equiv \psi_1 \vee \psi_2$:

$$\begin{aligned}
[\![\psi^\nu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cap U &= ([\![\psi_1^\nu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cup [\![\psi_2^\nu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]}) \cap U \\
&\overset{IH}{\subseteq} ([\![\psi_1]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cup [\![\psi_2]\!]^{\mathcal{T}}_{\rho[X\mapsto U]}) \cap U \subseteq [\![\psi]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \\
[\![\psi]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} &= [\![\psi_1]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cup [\![\psi_2]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \\
&\overset{IH}{\subseteq} [\![\psi_1^\mu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cup U \cup [\![\psi_2^\mu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cup U = [\![\psi^\mu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cup U
\end{aligned}$$

Case $\psi \equiv X$:

$$\begin{aligned}
[\![\psi^\nu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cap U &= [\![\mathrm{tt}]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cap U \subseteq U = [\![\psi]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \\
[\![\psi]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} &= U \subseteq [\![\mathrm{ff}]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cup U = [\![\psi^\mu]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} \cup U
\end{aligned}$$

Otherwise:

$$[\![\psi^\sigma]\!]^{\mathcal{T}}_{\rho[X\mapsto U]} = [\![\psi]\!]^{\mathcal{T}}_{\rho[X\mapsto U]}$$

□

We derive that a flattened fixed point preserves its semantics:

Corollary 2.62 (Fixed Point Flattening)**.** *Let ψ be a formula and $X \in \mathcal{V}$ be a variable. Then for all LTS $\mathcal{T}$ and all environments ρ holds:*

$$[\![\sigma X.\psi]\!]^{\mathcal{T}}_{\rho} = [\![\sigma X.fl(\psi, X, \sigma)]\!]^{\mathcal{T}}_{\rho}$$

Proof. Let $\psi' \equiv fl(\psi, X, \sigma)$.
Case $\sigma = \mu$:

- $[\![\mu X.\psi']\!]^{\mathcal{T}}_{\rho} \subseteq [\![\mu X.\psi]\!]^{\mathcal{T}}_{\rho}$: Since fl only replaces unguarded occurrences of X by `ff` this follows by Lemma 2.45.

- $[\![\mu X.\psi]\!]^{\mathcal{T}}_{\rho} \subseteq [\![\mu X.\psi']\!]^{\mathcal{T}}_{\rho}$: Lemma 2.61 implies that following holds:

  $$\{U \subseteq \Pi(\mathcal{T}) \mid [\![\psi']\!]^{\mathcal{T}}_{\rho[X \mapsto U]} \subseteq U\} \subseteq \{U \subseteq \Pi(\mathcal{T}) \mid [\![\psi]\!]^{\mathcal{T}}_{\rho[X \mapsto U]} \subseteq U\}$$

  Since Theorem 2.40 (Knaster-Tarski) the infimum of both sides exist and therefore $[\![\mu X.\psi]\!]^{\mathcal{T}}_{\rho} \subseteq [\![\mu X.\psi']\!]^{\mathcal{T}}_{\rho}$.

Case $\sigma = \nu$: This can be shown the same way. □

The effective translation into state-guarded form implements the following two translation maps that were originally designed for the guarded transformation of the Modal μ-Calculus [Wal00, Mat02]. This readaptation treats path quantification as well as next operator as modal operations in the sense of the original guarded transformation.

Definition 2.63 (State-guarded transformation functions)**.** Let ψ be a well-formed formula. The *state-guarded transformation functions*, $t_i(\psi)$, are inductively defined as follows:

$$t_i(\psi) := \begin{cases} op(t_i(\psi_1), t_i(\psi_2)) & \text{if } \psi \equiv op(\psi_1, \psi_2) \\ op(t_1(\psi')) & \text{if } \psi \equiv op(\psi'),\ op \equiv E, A, \bigcirc \\ \sigma X.fl(t_2(\psi'), X, \sigma) & \text{if } \psi \equiv \sigma X.\psi' \text{ and } i = 1 \\ fl(t_2(\psi'), X, \sigma)[\sigma X.fl(t_2(\psi'), X, \sigma)/X] & \text{if } \psi \equiv \sigma X.\psi' \text{ and } i = 2 \\ \psi & \text{otherwise} \end{cases}$$

where $i = 1, 2$.

Clearly, we need to ensure that these syntactical transformations preserve the original semantics of the formula in question:

Lemma 2.64 (State-guarded transformation semantics)**.** *Let ψ be a well-formed formula. Then for all LTS $\mathcal{T}$ and all environments ρ holds that $[\![t_2(\psi)]\!]^{\mathcal{T}}_{\rho} = [\![t_1(\psi)]\!]^{\mathcal{T}}_{\rho} = [\![\psi]\!]^{\mathcal{T}}_{\rho}$.*

Proof. By structural induction on ψ. Case $\psi \equiv \sigma X.\psi'$.

$$
\begin{aligned}
[\![t_2(\sigma X.\psi')]\!]^{\mathcal{T}}_{\rho} &= fl(t_2(\psi'), X, \sigma)[\sigma X.fl(t_2(\psi'), X, \sigma)/X] \\
&\overset{2.46}{=} \sigma X.fl(t_2(\psi'), X, \sigma) = [\![t_1(\sigma X.\psi')]\!]^{\mathcal{T}}_{\rho} \\
[\![t_1(\sigma X.\psi')]\!]^{\mathcal{T}}_{\rho} &= \sigma X.fl(t_2(\psi'), X, \sigma) \overset{2.62}{=} \sigma X.t_2(\psi') \\
&\overset{IH}{=} \sigma X.\psi'
\end{aligned}
$$

All other cases follow by induction hypothesis, Corollary 2.8 and the definition of t_i. □

It remains to show that the transformation functions indeed lead to state-guarded formulas. Unfortunately these proofs require a bunch of syntactical lemmas. In order to save us the trouble we omit these proofs and show how to syntactically adapt the translation correctness results of the Modal μ-Calculus.

Basically, we decompose the CTL^*_μ-formula in question into subformulas only consisting of linear-time constructs and show that if the transformation maps translate each subformula into (state-)guarded form then also the composed formula is translated into state-guarded form.

Definition 2.65 (Linear-time formula)**.** A *linear-time formula* is a formula ψ s.t. ψ contains no subformula of the form $E\psi'$ or $A\psi'$.

A well-formed formula ψ without any path quantifying subformulas can be seen as a linear-time fragment of CTL^*_μ hence the correctness result of the standard guarded transformation can be transferred.

Theorem 2.66 (Linear-time state-guarded transformation, [Wal00, Mat02])**.** *Let ψ be a well-formed linear-time formula. Then $t_1(\psi)$ is state-guarded.*

As a decomposition tool we define the *greedy substitution* which takes a formula and a formula operation and applies the operation to subformulas of the formula in question top-down:

Definition 2.67 (Greedy substitution)**.** Let ψ be a formula and $\eta : CTL^*_\mu \to CTL^*_\mu$ be a partial map. The *greedy substitution* $\psi \circ \eta$ is defined as follows:

$$\psi \circ \eta := \begin{cases} \eta(\psi) & \text{if } \psi \in dom(\eta) \\ op(\psi' \circ \eta) & \text{if } \psi \equiv op(\psi') \notin dom(\eta) \\ op(\psi_1 \circ \eta, \psi_2 \circ \eta) & \text{if } \psi \equiv op(\psi_1, \psi_2) \notin dom(\eta) \\ \psi & \text{otherwise} \end{cases}$$

After decomposing a formula one wants to be able to reconstruct the original formula when using a bijective operation on the subformulas:

Corollary 2.68 (Bijective greedy substitution)**.** *Let ψ be a formula and $\eta : CTL^*_\mu \to CTL^*_\mu$ be a partial bijection s.t. $\eta[Sub(\psi)] \cap Sub(\psi) = \emptyset$. Then $(\psi \circ \eta) \circ \eta^{-1} = \psi$.*

The *state formula decomposition* finally takes a formula and substitutes each outermost occurrence of a path quantifier by an unused atomic proposition and stores the substituted formula along with the associated proposition in a formula operation.

Definition 2.69 (State formula decomposition)**.** Let ψ be a formula and $L := \{p_1, p_2, ...\} \subseteq \mathcal{P} \setminus Sub(\psi)$ be a fixed but arbitrary set of unused propositions.

The *state formula decomposition* of ψ is an injective partial map $\eta : Sub(\psi) \to L$ with $(\eta, \_) := f(\psi, (\emptyset, 1))$ where $f(\psi, (map, i))$ is inductively defined as follows:

$$f(\psi, (map, i)) := \begin{cases} (map \cup (\psi, p_i), i+1) & \text{if } \psi \equiv E\psi', A\psi' \text{ and } \psi \notin dom(map) \\ f(\psi', (map, i)) & \text{if } \psi \equiv op(\psi') \text{ and } op \not\equiv E, A \\ f(\psi_2, f(\psi_1, (map, i))) & \text{if } \psi \equiv op(\psi_1, \psi_2) \\ (map, i) & \text{if } \psi \equiv op() \end{cases}$$

The next two corollaries establish the basis for the decomposition: First, we notice that if subformulas in a state-guarded formula are substituted by state-guarded formulas, the resulting formula remains state-guarded.

Corollary 2.70 (Guarded decomposition)**.** *Let ψ be a state-guarded formula and $\eta : CTL^*_\mu \to CTL^*_\mu$ be a partial map s.t. all $\psi' \in \eta[Sub(\psi)]$ are state-guarded state formulas. Then $\psi \circ \eta$ is state-guarded.*

Second, we observe that it does not matter in which order one applies t_1 to state subformulas of the formula in question:

Corollary 2.71 (Transformation decomposition)**.** *Let ψ be a formula and $\eta : CTL^*_\mu \to CTL^*_\mu$ be a partial map s.t. all $\psi' \in \eta[Sub(\psi)]$ are state formulas. Then $t_1(\psi \circ \eta) = t_1(\psi) \circ (t_1 \circ \eta)$.*

Finally we are able to derive that t_1 indeed transforms an arbitrary, well-formed formula into state-guarded form:

Lemma 2.72 (State-guarded transformation)**.** *Let ψ be a well-formed formula. Then $t_1(\psi)$ is state-guarded.*

Proof. By structural induction on ψ. If $\psi \equiv Q\psi'$ with $Q \equiv E, A$ the following holds:

$$sguarded(t_1(Q\psi')) \iff sguarded(Q(t_1(\psi'))) \overset{IH}{\iff} sguarded(t_1(\psi'))$$

Otherwise let η be the state formula decomposition of ψ. By definition of η $\psi \circ \eta$ is a linear-time formula, hence $t_1(\psi \circ \eta)$ is state-guarded by Theorem 2.66.

By induction hypothesis all $\varphi \in im(t_1 \circ \eta^{-1})$ are state-guarded, thus by Corollary 2.70 $t_1(\psi \circ \eta) \circ (t_1 \circ \eta^{-1})$ is state-guarded. Finally by applying Corollary 2.71 $t_1((\psi \circ \eta) \circ \eta^{-1})$ is state-guarded, i.e. by Corollary 2.68 $t_1(\psi)$ is state-guarded. □

Unfortunately we were not able to derive an algorithm that effectively transforms a formula into full guarded form.

Definition 2.73 (Guarded form)**.** A formula is in *guarded form* or *guarded*, $guarded(\psi)$, iff every occurrence of a bound variable X is in the scope of a $\bigcirc$-operator under its quantifier σ, i.e. iff $guarded(\psi, \emptyset)$ holds, where $guarded(\psi, A)$ is inductively defined as follows:

$$guarded(\psi, A) := \begin{cases} guarded(\psi', \emptyset) & \text{if } \psi \equiv \bigcirc\psi' \\ X \notin A & \text{if } \psi \equiv X,\ X \in \mathcal{V} \\ guarded(\psi', A \cup \{X\}) & \text{if } \psi \equiv \sigma X.\psi' \\ \forall \psi' \in ISub(\psi).\, guarded(\psi', A) & \text{otherwise} \end{cases}$$

The set of all *guarded* CTL^*_μ formulas is denoted by $CTL^*_{\mu g}$.

Mostly, however, it suffices to have a formula in state-guarded form. If we do need a formula to be in guarded form, we have to forbid that a path quantifier occurs between a bound variable and its fixed point binder. Such a formula is called *restricted*.

Definition 2.74 (Restricted Fixed Points)**.** A well-formed fixed point $\sigma X.\psi$ is *restricted* iff for all $Q\psi' \in Sub(\psi)$ with $Q \in \{E, A\}$ holds that $X \notin Sub(\psi')$. A well-formed formula ψ is *restricted* iff ψ is well-formed and each fixed point $\sigma X.\psi' \in Sub(\psi)$ is restricted. The set of all *restricted* CTL^*_μ formulas is denoted by $CTL^*_{\mu r}$.

If a formula is restricted, there is no path quantifier between a bound variable and its fixed point binder, hence such a formula is state-guarded if and only if it is guarded:

Corollary 2.75 (State-guarded restricted formulas are guarded)**.** *Let ψ be a state-guarded, restricted formula. Then ψ is guarded.*

Obviously the state-guarded transformation does not affect whether a formula is restricted.

Corollary 2.76 (Restricted formulas remain restricted)**.** *Let ψ be a restricted formula. Then $t_1(\psi)$ is still restricted.*

Putting the pieces together, we conclude that we can effectively transform a restricted formula into guarded form.

Corollary 2.77 (Restricted formula guarded transformation)**.** *Let ψ be a restricted formula. Then $t_1(\psi)$ is guarded.*

The main reason why we want a formula to be in guarded form is that each subformula cannot be unreeled infinitely often without seeing a modal operator. To estimate the distance between a subformula and the next occurrence of a modal operator, we introduce the definition of a *progression distance*:

Definition 2.78 (Progression distance)**.** Let $\psi^* \in CTL^*_{c\mu+}$ be guarded. The *progression distance w.r.t.* ψ^*, $pdist_{\psi^*} : Sub(\psi^*) \to \mathbb{N}$, is defined as follows:

$$pdist_{\psi^*} : \psi \mapsto \begin{cases} 0 & \text{if } \psi \in \mathcal{P}^* \text{ or } \psi \equiv \bigcirc\psi' \\ 1 + \max(pdist_{\psi^*}[ESub(\psi)]) & \text{otherwise} \end{cases}$$

Lastly, we need to ensure that $pdist_{\psi^*}$ is actually welldefined.

Lemma 2.79 (Progression distance is welldefined)**.** *Let $\psi^* \in CTL^*_{c\mu+}$ be guarded. Then the progression distance w.r.t. ψ^* is welldefined.*

Proof. First, we define

$$F(\psi) := \begin{cases} \emptyset & \text{if } \psi \equiv \bigcirc\psi' \\ \{X\} & \text{if } \psi \equiv X \\ \{X\} \cup \bigcup_{\psi' \in ISub(\psi')} F(\psi') & \text{if } \psi \equiv \sigma X.\psi' \\ \bigcup_{\psi' \in ISub(\psi')} F(\psi') & \text{otherwise} \end{cases}$$

as well as

$$G(\psi) := \{X \mid \exists Y \in F(\psi) : X \geq_{\psi^*} Y\}$$

and

$$s : \psi \mapsto |\psi| + \sum_{X \in G(\psi)} |body_{\psi^*}(X)|$$

and show that for each $\psi \in Sub(\psi^*)$ holds that $s(\psi) \geq pdist_{\psi^*}(\psi)$ by induction on ψ.

For $\psi \in \mathcal{P}^*, \psi \equiv \bigcirc\psi', \psi \equiv E\psi', \psi \equiv A\psi', \psi \equiv \psi_1 \wedge \psi_2, \psi \equiv \psi_1 \vee \psi_2$ the estimation $s(\psi) \geq pdist_{\psi^*}(\psi)$ directly follows by definition.

For $\psi \equiv \sigma X.\psi'$ consider that $pdist_{\psi^*}(\psi) = 1 + pdist_{\psi^*}(X)$ and $s(\psi) > s(X)$ due to

$$\begin{aligned} s(\psi) &= |\psi| + \sum_{Y \in G(\psi)} |body_{\psi^*}(Y)| \\ &\overset{2.25}{>} |X| + \sum_{Y \geq_{\psi^*} X} |body_{\psi^*}(Y)| \\ &= |X| + \sum_{Y \in G(X)} |body_{\psi^*}(Y)| \\ &= s(X) \end{aligned}$$

For $\psi \equiv X$ consider that $pdist_{\psi^*}(X) = 1 + pdist_{\psi^*}(body_{\psi^*}(X))$ and $s(X) > s(body_{\psi^*}(X))$ due to

$$\begin{aligned} s(X) &= |X| + \sum_{Y \in G(X)} |body_{\psi^*}(Y)| \\ &= |X| + \sum_{Y \geq_{\psi^*} X} |body_{\psi^*}(Y)| \\ &> |body_{\psi^*}(X)| + \sum_{Y >_{\psi^*} X} |body_{\psi^*}(Y)| \\ &\overset{2.25,(*)}{\geq} |body_{\psi^*}(X)| + \sum_{Y \in G(body_{\psi^*}(X))} |body_{\psi^*}(Y)| \\ &= s(body_{\psi^*}(X)) \end{aligned}$$

whereas (*) $X \notin G(body_{\psi^*}(X))$ due to the guardedness of ψ^*. □

2.7 Expressiveness

In this chapter, we want to investigate the expressiveness of CTL^*_μ in comparison to other temporal logics. Particularly we examine the differences in expressiveness concerning restricted and guarded fragments of CTL^*_μ.

Obviously, CTL^* is at most as expressive as the restricted and state-guarded fragment of CTL^*_μ. In fact, it is less expressive. Wolfgang Thomas et. al. [Tho89] for instance showed that CTL^* is strictly less expressive than $ECTL^*$ which is an extension of CTL^* by replacing the fixed points constructs by Büchi automata.

Due to the fact that Büchi automata and the Linear Time μ-Calculus are equally expressive [Var88], we conclude that the restricted fragment of CTL^*_μ is as expressive as $ECTL^*$. As we have seen in the preceding chapter, each restricted formula can be transformed into guarded form, hence the guarded fragment of $CTL^*_{\mu r}$ is as expressive as $CTL^*_{\mu r}$ itself.

It finally turns out that full $sCTL^*_\mu$ is equally expressive to the Modal μ-Calculus. It is reasonably easy to see that the Modal μ-Calculus can be embedded into $sCTL^*_\mu$. For the conversion we basically show that there are alternating parity automata that can be associated with arbitrary state-guarded formulas accepting all these transition system satisfied by the formula. Such automata can be simulated by formulas of the Modal μ-Calculus [Niw97]. We conclude that $sCTL^*_\mu$ and $\mathcal{L}_\mu$ are equally expressive.

Lemma 2.80 (General expressiveness)**.**
$CTL^* < ECTL^* = CTL^*_{\mu r} \leq CTL^*_\mu$

Proof.

- $CTL^* < ECTL^*$: Due to the fact the Büchi automata are strictly more expressive than star-free languages [Tho89].
- $ECTL^* = CTL^*_{\mu r}$: Due to the fact that Büchi automata and the Linear Time μ-Calculus are equivalent [Var88].
- $CTL^*_{\mu r} \leq CTL^*_\mu$: Since $CTL^*_{\mu r}$ is a syntactical fragment of CTL^*_μ.

□

In the remaining part of this section we show that the Modal μ-Calculus can be effectively embedded into CTL^*_μ. Basically the translation creates *path-irrelevant* formulas in the sense that a path models a formula if and only if all paths starting in the same state model the formula.

Definition 2.81 (Translation of Modal μ-Calculus)**.** The translation map $tr : \mathcal{L}_\mu \to CTL^*_\mu$ is defined as follows:

$$tr : \psi \mapsto \begin{cases} l & \text{if } \psi \equiv l \in \mathcal{P}^* \\ tr(\psi_1) \wedge tr(\psi_2) & \text{if } \psi \equiv \psi_1 \wedge \psi_2 \\ tr(\psi_1) \vee tr(\psi_2) & \text{if } \psi \equiv \psi_1 \vee \psi_2 \\ E(\bigcirc(tr(\psi'))) & \text{if } \psi \equiv \Diamond\psi' \\ A(\bigcirc(tr(\psi'))) & \text{if } \psi \equiv \Box\psi' \\ \sigma X.E(tr(\psi')) & \text{if } \psi \equiv \sigma X.\psi' \\ E(X) & \text{if } \psi \equiv X \in \mathcal{V} \end{cases}$$

Example 2.82. The following $\mathcal{L}_\mu$-formula

$$\psi \equiv \mu X.\Diamond(p \vee \nu Y.(q \wedge \Box(X \vee Y)))$$

would be canonically translated into this CTL^*_μ-formula:

$$tr(\psi) \equiv \mu X.E(\bigcirc(p \vee \nu Y.E(q \wedge A(\bigcirc(E(X) \vee E(Y))))))$$

As already mentioned, each CTL^*_μ-formula that is obtained by translating a Modal μ-Calculus formula is path-irrelevant and thus equivalent to a state formula (by prefixing the formula with a quantification).

Lemma 2.83 (Path irrelevance)**.** *Let $\mathcal{T}$ be an LTS, ρ be an environment and $\psi \in \mathcal{L}_\mu$. Then:*

$$\Pi_\mathcal{T}[\mathcal{S}_\mathcal{T}[[\![tr(\psi)]\!]^\mathcal{T}_\rho]] = [\![tr(\psi)]\!]^\mathcal{T}_\rho$$

Proof. By case distinction on ψ.
Case $\psi \equiv \psi_1 \wedge \psi_2$:

$$\begin{aligned} [\![tr(\psi_1 \wedge \psi_2)]\!]^\mathcal{T}_\rho &= [\![tr(\psi_1) \wedge tr(\psi_2)]\!]^\mathcal{T}_\rho \\ &= [\![tr(\psi_1)]\!]^\mathcal{T}_\rho \cap [\![tr(\psi_2)]\!]^\mathcal{T}_\rho \\ &\overset{IH}{=} \Pi_\mathcal{T}[\mathcal{S}_\mathcal{T}[[\![tr(\psi_1)]\!]^\mathcal{T}_\rho]] \cap \Pi_\mathcal{T}[\mathcal{S}_\mathcal{T}[[\![tr(\psi_2)]\!]^\mathcal{T}_\rho]] \\ &= \Pi_\mathcal{T}[\mathcal{S}_\mathcal{T}[[\![tr(\psi_1)]\!]^\mathcal{T}_\rho \cap [\![tr(\psi_2)]\!]^\mathcal{T}_\rho]] \\ &= \Pi_\mathcal{T}[\mathcal{S}_\mathcal{T}[[\![tr(\psi_1) \wedge tr(\psi_2)]\!]^\mathcal{T}_\rho]] = \Pi_\mathcal{T}[\mathcal{S}_\mathcal{T}[[\![tr(\psi_1 \wedge \psi_2)]\!]^\mathcal{T}_\rho]] \end{aligned}$$

Case $\psi \equiv \nu X.\psi'$:

$$\begin{aligned}
[\![tr(\nu X.\psi')]\!]^{\mathcal{T}}_{\rho} &= [\![\nu X.E(tr(\psi'))]\!]^{\mathcal{T}}_{\rho} \\
&= \bigcup\{T \mid T \subseteq \Pi_{\mathcal{S}}[\mathcal{S}_{\mathcal{T}}[[\![tr(\psi')]\!]^{\mathcal{T}}_{\rho[X\mapsto T]}]]\} \\
&= \bigcup\{\Pi_{\mathcal{S}}[\mathcal{S}_{\mathcal{T}}[T]] \mid \Pi_{\mathcal{S}}[\mathcal{S}_{\mathcal{T}}[T]] \subseteq \Pi_{\mathcal{S}}[\mathcal{S}_{\mathcal{T}}[[\![tr(\psi')]\!]^{\mathcal{T}}_{\rho[X\mapsto \Pi_{\mathcal{S}}[\mathcal{S}_{\mathcal{T}}[T]]]}]]\} \\
&= \Pi_{\mathcal{S}}[\mathcal{S}_{\mathcal{T}}[\bigcup\{T \mid T \subseteq \Pi_{\mathcal{S}}[\mathcal{S}_{\mathcal{T}}[[\![tr(\psi')]\!]^{\mathcal{T}}_{\rho[X\mapsto T]}]]\}]] \\
&= \Pi_{\mathcal{S}}[\mathcal{S}_{\mathcal{T}}[[\![tr(\nu X.\psi')]\!]^{\mathcal{T}}_{\rho}]]
\end{aligned}$$

Case $\psi \equiv \mu X.\psi'$: This can be shown the same way.
All other cases are straight-forward. □

This translation is correct in the sense that a state models a $\mathcal{L}_\mu$-formula if and only if a path (actually all paths) rooting in this state models the CTL^*_μ-formula obtained by the canonical translation. We denote the semantical set w.r.t. a $\mathcal{L}_\mu$-formula by $[\![^{\mathcal{L}_\mu}\psi]\!]$.

Lemma 2.84 (Translation correctness)**.** *Let $\mathcal{T}$ be an LTS, ρ_m be an $\mathcal{L}_\mu$-environment and $\psi \in \mathcal{L}_\mu$. Then:*

$$[\![^{\mathcal{L}_\mu}\psi]\!]^{\mathcal{T}}_{\rho_m} = \mathcal{S}_{\mathcal{T}}[[\![tr(\psi)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}] = [\![E(tr(\psi))]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}$$

Proof. By case distinction on ψ.
Case $\psi \equiv l \in \mathcal{P}^*$:

$$\mathcal{S}_{\mathcal{T}}[[\![tr(l)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}] = \mathcal{S}_{\mathcal{T}}[[\![l]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}] = [\![^{\mathcal{L}_\mu}l]\!]^{\mathcal{T}}_{\rho_m}$$

Case $\psi \equiv \psi_1 \wedge \psi_2$:

$$\begin{aligned}
\mathcal{S}_{\mathcal{T}}[[\![tr(\psi_1 \wedge \psi_2)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}] &= \mathcal{S}_{\mathcal{T}}[[\![tr(\psi_1) \wedge tr(\psi_2)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}] \\
&= \mathcal{S}_{\mathcal{T}}[[\![tr(\psi_1)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m} \cap [\![tr(\psi_2)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}] \\
&\overset{2.83}{=} \mathcal{S}_{\mathcal{T}}[\Pi_{\mathcal{T}}[\mathcal{S}_{\mathcal{T}}[[\![tr(\psi_1)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}]] \cap \Pi_{\mathcal{T}}[\mathcal{S}_{\mathcal{T}}[[\![tr(\psi_2)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}]]] \\
&= \mathcal{S}_{\mathcal{T}}[\Pi_{\mathcal{T}}[\mathcal{S}_{\mathcal{T}}[[\![tr(\psi_1)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}] \cap \mathcal{S}_{\mathcal{T}}[[\![tr(\psi_2)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}]]] \\
&= \mathcal{S}_{\mathcal{T}}[[\![tr(\psi_1)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}] \cap \mathcal{S}_{\mathcal{T}}[[\![tr(\psi_2)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}}\circ\rho_m}] \\
&\overset{IH}{=} [\![^{\mathcal{L}_\mu}\psi_1]\!]^{\mathcal{T}}_{\rho_m} \cap [\![^{\mathcal{L}_\mu}\psi_2]\!]^{\mathcal{T}}_{\rho_m} = [\![^{\mathcal{L}_\mu}\psi_1 \wedge \psi_2]\!]^{\mathcal{T}}_{\rho_m}
\end{aligned}$$

Case $\psi \equiv \psi_1 \vee \psi_2$:

$$\begin{aligned}
\mathcal{S}_\mathcal{T}[[\![tr(\psi_1 \vee \psi_2)]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}] &= \mathcal{S}_\mathcal{T}[[\![tr(\psi_1) \vee tr(\psi_2)]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}] \\
&= \mathcal{S}_\mathcal{T}[[\![tr(\psi_1)]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m} \cup [\![tr(\psi_2)]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}] \\
&= \mathcal{S}_\mathcal{T}[[\![tr(\psi_1)]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}] \cup \mathcal{S}_\mathcal{T}[[\![tr(\psi_2)]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}] \\
&\overset{IH}{=} [\![^{\mathcal{L}_\mu}\psi_1]\!]^\mathcal{T}_{\rho_m} \cup [\![^{\mathcal{L}_\mu}\psi_2]\!]^\mathcal{T}_{\rho_m} = [\![^{\mathcal{L}_\mu}\psi_1 \vee \psi_2]\!]^\mathcal{T}_{\rho_m}
\end{aligned}$$

Case $\psi \equiv \Diamond\psi'$:

$$\begin{aligned}
\mathcal{S}_\mathcal{T}[[\![tr(\Diamond\psi')]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}] &= \mathcal{S}_\mathcal{T}[[\![E(\bigcirc(tr(\psi'))]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}] \\
&= \{s \in \mathcal{S} \mid \exists t \in \mathcal{S} :\ s \to t \wedge t \in \mathcal{S}_\mathcal{T}[[\![tr(\psi')]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}]\} \\
&\overset{IH}{=} \{s \in \mathcal{S} \mid \exists t \in \mathcal{S} :\ s \to t \wedge t \in [\![^{\mathcal{L}_\mu}\psi']\!]^\mathcal{T}_{\rho_m}\} \\
&= [\![^{\mathcal{L}_\mu}\Diamond\psi']\!]^\mathcal{T}_{\rho_m}
\end{aligned}$$

Case $\psi \equiv \Box\psi'$:

$$\begin{aligned}
\mathcal{S}_\mathcal{T}[[\![tr(\Box\psi')]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}] &= \mathcal{S}_\mathcal{T}[[\![A(\bigcirc(tr(\psi'))]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}] \\
&= \{s \in \mathcal{S} \mid \forall t \in \mathcal{S} :\ s \to t \Rightarrow t \in \mathcal{S}_\mathcal{T}[[\![tr(\psi')]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}]\} \\
&\overset{IH}{=} \{s \in \mathcal{S} \mid \forall t \in \mathcal{S} :\ s \to t \Rightarrow t \in [\![^{\mathcal{L}_\mu}\psi']\!]^\mathcal{T}_{\rho_m}\} \\
&= [\![^{\mathcal{L}_\mu}\Box\psi']\!]^\mathcal{T}_{\rho_m}
\end{aligned}$$

Case $\psi \equiv \nu X.\psi'$:

$$\begin{aligned}
\mathcal{S}_\mathcal{T}[[\![tr(\nu X.\psi')]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}] &= \mathcal{S}_\mathcal{T}[[\![\nu X.E(tr(\psi'))]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ\rho_m}] \\
&= \mathcal{S}_\mathcal{T}[\bigcup\{T \mid T \subseteq \Pi_\mathcal{S}[\mathcal{S}_\mathcal{T}[[\![tr(\psi')]\!]^\mathcal{T}_{(\Pi_\mathcal{S}\circ\rho_m)[X\mapsto T]}]]\}] \\
&= \mathcal{S}_\mathcal{T}[\bigcup\{T \mid T \subseteq \Pi_\mathcal{S}[\mathcal{S}_\mathcal{T}[[\![tr(\psi')]\!]^\mathcal{T}_{(\Pi_\mathcal{S}\circ\rho_m)[X\mapsto \Pi_\mathcal{S}[\mathcal{S}_\mathcal{T}[T]]}]]\}] \\
&= \mathcal{S}_\mathcal{T}[\bigcup\{T \mid T \subseteq \Pi_\mathcal{S}[\mathcal{S}_\mathcal{T}[[\![tr(\psi')]\!]^\mathcal{T}_{\Pi_\mathcal{S}\circ(\rho_m[X\mapsto \mathcal{S}_\mathcal{T}[T])}]]\}] \\
&\overset{IH}{=} \mathcal{S}_\mathcal{T}[\bigcup\{T \mid T \subseteq \Pi_\mathcal{S}[[\![^{\mathcal{L}_\mu}\psi']\!]^\mathcal{T}_{\rho_m[X\mapsto \mathcal{S}_\mathcal{T}[T]]}]\}] \\
&= \mathcal{S}_\mathcal{T}[\bigcup\{\Pi_\mathcal{S}[U] \mid U \subseteq [\![^{\mathcal{L}_\mu}\psi']\!]^\mathcal{T}_{\rho_m[X\mapsto U]}\}] \\
&= \bigcup\{U \mid U \subseteq [\![^{\mathcal{L}_\mu}\psi']\!]^\mathcal{T}_{\rho_m[X\mapsto U]}\} = [\![^{\mathcal{L}_\mu}\nu X.\psi']\!]^\mathcal{T}_{\rho_m}
\end{aligned}$$

Case $\psi \equiv \mu X.\psi'$: This case can be shown the same way.
Case $\psi \equiv X$:

$$\mathcal{S}_{\mathcal{T}}[[\![tr(X)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}} \circ \rho_m}] = \mathcal{S}_{\mathcal{T}}[[\![E(X)]\!]^{\mathcal{T}}_{\Pi_{\mathcal{S}} \circ \rho_m}] = [\![^{\mathcal{L}_\mu} X]\!]^{\mathcal{T}}_{\rho_m}$$

□

Hence $\mathcal{L}_\mu \leq sCTL^*_\mu$. Although all the techniques to prove the conversion are introduced in the forthcoming chapters, we already note here that the conversion also holds.

Theorem 2.85 (Expressiveness result)**.**
$\mathcal{L}_\mu = sCTL^*_\mu$

Proof. For $\mathcal{L}_\mu \leq sCTL^*_\mu$ consider the former lemma. For the conversion $sCTL^*_\mu \leq \mathcal{L}_\mu$ let φ be a well-formed state formula of CTL^*_μ. By Lemmata 2.72 and 2.64 there is an equivalent well-formed state-guarded formula φ'. We anticipate one major result of this thesis, namely Theorem 5.25, stating that there is an alternating tree automaton $\mathcal{A}$ with $\mathcal{L}(\mathcal{A}) = Mod(\varphi')$. By Theorem B.13 there is a closed formula ψ of the Modal μ-Calculus with $Mod(\psi) = \mathcal{L}(\mathcal{A})$ and thus $Mod(\psi) = Mod(\varphi)$. □

This leaves us with the question whether each formula of CTL^*_μ can be brought into guarded form. It might seem so using the result that $sCTL^*_\mu$ is as expressive as the Modal μ-Calculus. Unfortunately that "trick" actually does not apply, since each formula of the Modal μ-Calculus corresponds to a path-irrelevant formula of CTL^*_μ. This correspondance suffices to be *satisfaction equivalent* but is not helping to solve the general question whether each formula of CTL^*_μ has a *path equivalent* counterpart in guarded form.

Nevertheless we believe that each formula *can* be brought into guarded form although we do not know how to prove this yet.

3. THREADS AND BUNDLES

This chapter introduces the theoretical foundations of CTL^*_μ-tableaux systems, both the proof system and the model-checking tableaux.

As pointed out before the main focus will be put on the global properties of infinite tableaux branches in order to decide whether they are valid or not. The basic technique is to monitor the unreeling behaviour of formulas while infinitely often unfolded greatest fixed points are "good" and least are "bad". Semantically this idea complies with the fixed point annotations of the preceding chapter.

Since we also have to deal with path quantifications, it depends on which formulas we have to focus: For instance, if we are dealing with a universal quantification under which there is a disjunction of formulas, it suffices that there is one formula in the disjunction whose unreeling behaviour is "good". With existential quantifications and conjunctions, each formula needs to provide a "good" unreeling behaviour. As path quantifications can occur in formulas in an alternating way, we rather need to verify some kind of alternating "goodness".

The first section will be providing the formal basis of formula unreeling: A sequence of formulas complying with unreeling is a *thread*. Subsequently we will outline the semantical consequences by tagging threads with *thread annotations*.

Second, we will be focussing on *thread bundles*. A *thread bundle* is a sequence of connected *blocks*, whereas a block is a bunch of single formulas under a path quantification - that is either a disjunction under a universal quantification or a conjunction under an existential quantification. Again, we will outline the semantical consequences.

Finally, we show how to interpret threads and thread bundles as infinite words in order to provide the automata theoretic links for the upcoming decision procedures.

3.1 Threads

This section introduces the *syntactical* formalism of formula unreeling. A sequence of unreeled formulas is called *thread* and is wellformed if a formula either remains unchanged or unreels to an extended sub formula in each step.

Recap that we are particularly interested in the infinite unfolding behaviour of the fixed points in a thread. A thread that remains unchanged after a while is useless due to the simple fact that there is no infinite behaviour that could be analyzed; hence we require threads to be *lively*: Formulas need to be unreeled once in a while.

The reason why we are allowing threads to sometimes remain unchanged at all is that a bunch of threads will be finally put together into thread bundles that usually only affect one thread in a single step. Thus we need to be comfortable with *liveliness*.

Definition 3.1 (Threads)**.** Let $\psi \in CTL^*_{\mu+}$. An infinite sequence $t_0, t_1, ...$ of formulas with $t_i \in Sub(\psi)$ for all $i \in \mathbb{N}$ is called *thread w.r.t.* ψ iff

$$\forall i \in \mathbb{N} : \; t_{i+1} \in ESub(t_i) \cup \{t_i\}$$

Such a thread is called *lively* iff

$$\forall i \in \mathbb{N} \, \exists j > i : \; t_i \neq t_j$$

Unreeling a formula reduces its size unless a fixed point is unfolded. Thus a lively thread unfolds at least one fixed point infinitely often:

Lemma 3.2 (Infinite threads)**.** *Let $\psi \in CTL^*_{\mu+}$ and t be a lively thread w.r.t. ψ. There is a variable $X \in Bound(\psi)$ s.t. $t_i \equiv X$ for infinitely many $i \in \mathbb{N}$.*

Proof. By contradiction. Assume that there is an $i^* \in \mathbb{N}$ s.t. $t_i \not\equiv X$ with $X \in Bound(\psi)$ for all $i \geq i^*$. It suffices to show that

$$|t_{i+1}| \leq |t_i| \text{ for all } i \geq i^*$$

as well as

$$|t_{i+1}| < |t_i| \text{ for infinitely many } i \geq i^*$$

since in this case $|t_{i^*}|, |t_{i^*+1}|, ...$ is an infinitely $<$-decreasing sequence which is impossible with $<$ being well-founded on $\mathbb{N}$.

Since t is lively it suffices to show that for each $i \geq i^*$ with $t_i \neq t_{i+1}$ holds that $|t_{i+1}| < |t_i|$. Hence let $i \geq i^*$ with $t_i \neq t_{i+1}$ be arbitrary. For $t_{i+1} \in ISub(t_i)$ or $t_i \equiv \sigma X.t_{i+1}$ this is obvious; the only case potentially increasing the size of t_i is $t_i \equiv X$ which is not allowed by assumption. □

To keep track of the infinite behaviour of a thread, we are interested in the outermost fixed point that is unfolded infinitely often. All other fixed points unfolded in the thread are either only unfolded finitely often or terminated infinitely often and therefore not relevant for answering the question whether the thread is *globally valid*.

Definition 3.3 (σ-threads)**.** Let $\psi \in CTL^*_{\mu+}$ and t be a lively thread w.r.t. ψ. The *dominating variable* of t, $do_\psi(t)$, is defined as follows:

$$do_\psi(t) := max_{<_\psi}\{X \in Bound(\psi) \mid |\{i \in \mathbb{N} \mid t_i \equiv X\}| = \infty\}$$

Note that $do_\psi(t)$ is well-defined since Lemma 3.2.

A thread t is a *ν-thread* iff $do_\psi(t) \in Bound_\nu(\psi)$; likewise t is a *μ-thread* iff $do_\psi(t) \in Bound_\mu(\psi)$. Again due to Lemma 3.2, each lively thread is obviously either a ν- or a μ-thread.

Particularly we are interested in all occurrences of a path quantification in a thread. For one thing, the occurrence of a path quantification semantically corresponds to a possible change of the current path. Such a path quantifying position in a thread is called *root*.

Definition 3.4 (Thread roots and helper maps)**.** Let $\psi \in CTL^*_{\mu+}$ and t be a thread w.r.t. ψ. The *progress between i and j w.r.t. t* measures the number of unreeled next-operators between i and j:

$$pro_t(i,j) := |\{i \leq k < j \mid t_k \equiv \bigcirc t_{k+1}\}|$$

The set of *roots w.r.t. t* is defined as follows:

$$roots_t := \{i \mid i = 0 \vee t_{i-1} \equiv E(t_i) \vee t_{i-1} \equiv A(t_i)\}$$

The roots without the last root (if existing) of t is denoted by $roots^*_t$ i.e.

$$roots^*_t := \begin{cases} roots_t & \text{if } |roots_t| = \infty \\ roots_t \setminus \{\max(roots_t)\} & \text{otherwise} \end{cases}$$

The root helper functions are moreover defined as follows:

- $root_t : \mathbb{N} \to roots_t,\ i \mapsto \max\{j \in roots_t \mid j \leq i\}$
- $nroot_t : (roots^*_t)_{\leq} \to roots_t,\ r \mapsto \min\{j \in roots_t \mid j > r\}$
- $rsize_t : roots^*_t \to \mathbb{N},\ r \mapsto pro_t(r, nroot_t(r))$

where $A_{\leq} := \{i \in \mathbb{N} \mid \exists a \in A :\ i \leq a\}$.

Again, we introduce the concept of duality. Let t be a thread. The *dual thread of t* is basically the same thread but only for the negative translation of the initial formula. Syntactically one could simply replace each operand and each literal occurring in formulas of t by their dual counterpart.

Definition 3.5 (Dual thread)**.** Let $\psi \in CTL^*_{\mu+}$ and t be a thread w.r.t. ψ. The *dual thread of t*, written as $\overline{t}$, is defined as follows:

$$\overline{t} : \mathbb{N} \to Sub(\psi^{\ominus})\ i \mapsto t_i^{\ominus}$$

All general properties of a thread can obviously be transferred to the corresponding dual thread.

Corollary 3.6 (Dual thread properties)**.** *Let $\psi \in CTL^*_{\mu+}$ and t be a thread w.r.t. ψ. Then the following properties w.r.t. the dual thread $\bar{t}$ hold:*

1. *The dual thread $\bar{t}$ is a thread w.r.t. $\psi^{\ominus}$.*
2. *t is lively iff $\bar{t}$ is lively*
3. *$roots_t = roots_{\bar{t}}$*
4. *t is a μ-thread iff $\bar{t}$ is a ν-thread*
5. *$\bar{\bar{t}} = t$*

3.2 Thread annotations

Thread annotations are a semantical construct enabling us to draw conclusions about the modelling relation by studying the syntactical behaviour of threads. A thread annotation basically is a bunch of data that is either completely falsified or satisfied by a thread. That data, more precisely, needs to be compliant with the thread meaning that whenever a next-operator is unreeled, the associated state in the data pushes along to a connected state and whenever a path-quantification is unreeled, the data changes its current path.

Formally, a thread annotation is a family of paths providing possibly different paths for each root of the thread. Additionally all paths need to be compliant with the unreeling behaviour of the thread in the sense that if the thread unreels k next operators between two path quantifications, then the path associated with the latter path quantification needs to start with the k-th state of the path associated with the former quantification.

Definition 3.7 (Thread annotation)**.** Let $\psi \in CTL^*_{\mu+}$, t be a thread w.r.t. ψ and $\mathcal{T}$ be an LTS. A *thread annotation w.r.t. t* is a family $\pi = (\pi_i)_{i \in roots_t}$ of paths $\pi_i \in \Pi(\mathcal{T})$ s.t. for all $i \in roots_t^*$ holds

$$\pi_i(rsize_t(i)) = \pi_{nroot_t(i)}(0)$$

For each $i \in \mathbb{N}$ we define the *path w.r.t. t, π and i*, $\pi\langle i \rangle$, as follows

$$\pi\langle i \rangle := \pi_{root_t(i)}[pro_t(root_t(i), i)]$$

A thread annotation $\pi = (\pi_i)_{i \in roots_t}$ w.r.t. t is called

- *falsifying* iff $\pi\langle i \rangle \not\models\updownarrow^{\psi} t_i$ for all $i \in \mathbb{N}$
- *satisfying* iff $\pi\langle i \rangle \models\updownarrow^{\psi} t_i$ for all $i \in \mathbb{N}$

As outlined before we want to focus on the unfolding of fixed point formulas in a thread. Technically we are simply using the fixed point annotations to keep track of the unfolding behaviour. If we have a thread and a falsifying thread annotation for instance, we tag all formulas with annotations in order to use their wellfoundedness to prove global fixed point properties of a thread.

Particularly we have to make sure that the annotations are not affected by non-fixed-point unreeling operations. Let $t_i = \psi_1 \wedge \psi_2$ and $t_{i+1} = \psi_1$ for example; it is possible that the minimal signature that falsifies t_i is strictly less than the minimal signature that falsifies t_{i+1} (thus it would be better if $t_{i+1} = \psi_2$. Therefore it does not suffice that an annotation is falsifying: It needs to be *signature compatible falsifying*.

Definition 3.8 (Signature compatible annotations)**.** Let $\psi \in CTL^*_{\mu+}$, t be a thread w.r.t. ψ and $\mathcal{T}$ be an LTS.

A falsifying thread annotation $\pi = (\pi_i)_{i \in roots_t}$ w.r.t. t is called *signature compatible falsifying* iff for all $i \in \mathbb{N}$ with $t_i \neq t_{i+1}$ and $t_i \equiv \psi_1 \wedge \psi_2, A\psi'$ holds that if $\zeta \in Sig_\nu(\psi)$ is the least signature s.t. $\pi\langle i \rangle \not\models\downarrow^{\psi}_{\zeta} t_i$ then also $\pi\langle i+1 \rangle \not\models\downarrow^{\psi}_{\zeta} t_{i+1}$.

A satisfying thread annotation $\pi = (\pi_i)_{i \in roots_t}$ w.r.t. t is called *signature compatible satisfying* iff for all $i \in \mathbb{N}$ with $t_i \neq t_{i+1}$ and $t_i \equiv \psi_1 \vee \psi_2, E\psi'$ holds that if $\zeta \in Sig_\mu(\psi)$ is the least signature s.t. $\pi\langle i \rangle \models\uparrow^{\psi}_{\zeta} t_i$ then also $\pi\langle i+1 \rangle \models\uparrow^{\psi}_{\zeta} t_{i+1}$.

A signature compatible annotation with respect to a thread t corresponds to a *dual annotation* with respect to the dual thread in a natural way.

Corollary 3.9 (Dual thread annotations)**.** *Let $\psi \in CTL^*_{\mu+}$ and t be a thread w.r.t. ψ, $\mathcal{T}$ be an LTS and $\pi = (\pi_i)_{i \in roots_t}$ be a thread annotation w.r.t. t (and $\bar{t}$). Then:*

1. *π is (signature compatible) falsifying w.r.t. t iff π is (signature compatible) satisfying w.r.t. $\bar{t}$*
2. *π is (signature compatible) satisfying w.r.t. t iff π is (signature compatible) falsifying w.r.t. $\bar{t}$*

The following proposition shows a crucial global property of threads building the bridge between syntax and semantics: Lively ν-threads cannot be signature compatible falsified.

By contradiction assume that there is a lively ν-thread t with a signature compatible falsifying annotation. Now annotate each thread formula t_i with a least ν-signature that falsifies t_i (w.r.t. the annotation). Since t is a ν-thread the outermost fixed point which is unfolded infinitely often is a greatest fixed point. Hence the signature annotations are finally infinitely decreasing which is impossible.

Lemma 3.10 (Enforcement of μ-threads)**.** *Let $\psi \in CTL^*_{\mu+}$, t be a lively thread w.r.t. ψ, $\mathcal{T}$ be an LTS and $\pi = (\pi_i)_{i roots_t}$ be a signature compatible falsifying thread annotation w.r.t. t. If t_0 is closed then t is a μ-thread.*

Proof. By contradiction assume that t is a ν-thread (since all threads are either μ- or ν-threads). First, we construct a sequence of outer approximations ζ_i s.t. for all $i \in \mathbb{N}$ the following holds:

1. ζ_i is minimal s.t. $\pi\langle i \rangle \not\models \downarrow^{\psi}_{\zeta_i} t_i$.
2. If $t_i \neq t_{i+1}$ and $t_i \in Bound_\nu(\psi)$ then $\zeta^{k]}_{i+1} < \zeta^{k]}_i$ where $k = ind^{\psi}_{\nu}(t_i)$.
3. If $t_i \neq t_{i+1}$ and $t_i \equiv \nu X.\psi'$ then $\zeta^{k]}_{i+1} \leq \zeta^{k]}_i$ where $k := ind^{\psi}_{\nu}(X) - 1$.
4. If $t_i = t_{i+1}$ or $t_i \not\equiv \nu X.\psi'$ then $\zeta_{i+1} \leq \zeta_i$.

For $i = 0$ let $\zeta := (0, ..., 0)$ as t_0 is closed. Let now $\zeta_0, ..., \zeta_i$ be already constructed. If $t_i = t_{i+1}$ simply set $\zeta_{i+1} := \zeta_i$.

For $t_i \neq t_{i+1}$ consider that it suffices to prove that there is an arbitrary ζ_{i+1} s.t. $\pi\langle i+1\rangle \not\models\downarrow^{\psi}_{\zeta_{i+1}} t_{i+1}$ with 2, 3 and 4:

- *Case* $t_i \equiv \psi_1 \vee \psi_2$ *and* $t_{i+1} \equiv \psi_a$ *with* $a \in \{1,2\}$: Obviously $\pi\langle i\rangle \not\models\downarrow^{\psi}_{\zeta_i} \psi_1 \vee \psi_2$ implies $\pi\langle i\rangle \not\models\downarrow^{\psi}_{\zeta_i} \psi_a$.
- *Case* $t_i \equiv \psi_1 \wedge \psi_2$: By definition of signature compatibility.
- *Case* $t_i \equiv E\psi_1$ *and* $t_{i+1} \equiv \psi_1$: By assumption $\pi\langle i\rangle \not\models\downarrow^{\psi}_{\zeta_i} E\psi_1$ holds. Thus for all $\chi \in \Pi(\mathcal{T})$ with $\chi(0) = s_i$ we get $\chi \not\models\downarrow^{\psi}_{\zeta_i} \psi_1$; since $\pi\langle i+1\rangle$ is such a χ we're done.
- *Case* $t_i \equiv A\psi_1$: By definition of signature compatibility.
- *Case* $t_i \equiv X, \sigma X.\psi_1$: By Lemma 2.54.
- *Case* $t_i \equiv \bigcirc\psi_1$ *and* $t_{i+1} \equiv \psi_1$: By construction we have $\pi\langle i\rangle[1] = \pi\langle i+1\rangle$ and therefore $\pi\langle i\rangle \not\models\downarrow^{\psi}_{\zeta_i} \bigcirc\psi_1$ implies $\pi\langle i+1\rangle \not\models\downarrow^{\psi}_{\zeta_i} \psi_1$.

Let now $X := do_{\psi}(t)$ be the dominating variable in t. We show that there is a j^* s.t. for all $i \geq j^*$ the following holds:

$$(*)\ t_i \not\equiv \sigma Y.body_{\psi}(Y) \text{ for all } Y \geq_{\psi} X$$

Since X is the dominating variable in t we can choose a^* sufficiently large s.t. for all $i \geq a^*$

$$(**)\ t_i \not\equiv Y \text{ for all } Y >_{\psi} X$$

Consequenty we can choose $b^* \geq a^*$ sufficiently large s.t. for all $i \geq b^*$

$$(***)\ t_i \not\equiv \sigma Y.body_{\psi}(Y) \text{ for all } Y >_{\psi} X$$

as otherwise $t_i \equiv \sigma Y.body_{\psi}(Y)$ for some $Y >_{\psi} X$ and infinitely many $i \geq a^*$ which can only be spawned by some $Z \geq_{\psi} Y >_{\psi} X$; but this is impossible since (**).

It remains to show that there is a $j^* \geq b^*$ s.t. for all $i \geq j^*$ holds that $t_i \not\equiv \nu X.body_\psi(X)$. If this holds already for $j^* = b^*$ we are finished; otherwise choose $j^* > b^*$ s.t. $t_i \equiv \nu X.body_\psi(X)$ for some $b^* \leq i < j^*$. As $\nu X.body_\psi(X)$ can only be spawned by some $Y >_\psi X$ - which is impossible since (**) - the claim holds.

Applying this result we finally prove that the following holds:

a. $\zeta^{k]}_{i+1} \leq \zeta^{k]}_i$ for all $i \geq j^*$

b. $\zeta^{k]}_{i+1} < \zeta^{k]}_i$ for infinitely many $i \geq j^*$

where $k := ind^\psi_\nu(X)$.

- To see that a. holds, assume that $\zeta^{k]}_{i+1} > \zeta^{k]}_i$ for some $i \geq j^*$, hence $\zeta_{i+1} > \zeta_i$. Since 3 $\zeta_{i+1} > \zeta_i$ implies that $t_i \equiv \nu Y.body_\psi(Y)$ and $k \geq ind^\psi_\nu(Y)$. But this is impossible since (*), hence $\zeta^{k]}_{i+1} \leq \zeta^{k]}_i$.

- For b. it suffices to show that $t_i = X$ and $t_{i+1} \neq t_i$ for infinitely many $i \geq j^*$ because in this case 2 directly proves the claim.

  As X is the dominating variable in t we have $t_i = X$ for infinitely many i; since t is lively by assumption for each such i there is an $i' \geq i$ s.t. $t_{i'} = X$ and $t_{i'+1} \neq t_{i'}$.

Properties a. and b. imply that $\zeta^{k]}_{j^*}, \zeta^{k]}_{j^*+1}, \ldots$ is an infinitely descending sequence of signatures. Since signatures are wellfounded, no such sequence exists. □

To transfer this result to ν-threads and satisfying annotations, we simply make use of the duality of threads and annotations:

Lemma 3.11 (Enforcement of ν-threads)**.** *Let $\psi \in CTL^*_{\mu+}$, t be a lively thread w.r.t. ψ, $\mathcal{T}$ be an LTS and $\pi = (\pi_i)_{i \in roots_t}$ be a signature compatible satisfying thread annotation w.r.t. t. If t_0 is closed then t is a ν-thread.*

Proof. Consider the dual thread $\bar{t}$. Due to Corollary 3.6 $\bar{t}$ is lively w.r.t. $\psi^\ominus$ and due to Corollary 3.9 π is a signature compatible falsifying thread annotation w.r.t. $\bar{t}$. Since $\bar{t}_0$ is closed Lemma 3.10 implies that $\bar{t}$ is a μ-thread, hence again by Corollary 3.6 t is a ν-thread. □

3.3 Thread bundles

Thread bundles are linear tableaux-style proof-branches whose nodes are labelled with a quantification tag - either being A for universal quantification or E for existential quantification - and a list of CTL^*_μ-formulas. Thread bundles are built according to *bundle rules* specifying one ore more formulas to be altered by the respective rule application.

Most rule applications only affect a single formula and leave all other formulas listed under the quantification tag untouched. There are two rules affecting all formulas under the quantification.

The modal rule for one thing is only applicable when all formulas listed under the path quantification are starting with the next-operator and is to be applied by removing the leading next-operator from all formulas. The other rule affecting all formulas follows a new path quantification: The principal formula of this rule is a path quantifying formula itself. Applying this rule removes all formulas but the principal formula and changes the quantification tag to the quantification of the formula in question.

When investigating thread bundles it will be crucial to know exactly which rule was applied to which principal formula and which formulas were resulting from this operation. Without specifying this information we would possibly have to deal with ambiguous structures that could not be handled properly.

Definition 3.12 (Bundle rules)**.** The *bundle rules* w.r.t. a fixed formula $\psi^* \in CTL^*_{c\mu+}$ are as follows:

$$(BA\vee): \frac{A(\lceil\psi_1\rceil, \lceil\psi_2\rceil, \Lambda)}{A(\lfloor\psi_1 \vee \psi_2\rfloor, \Lambda)} \qquad (BE\vee): \frac{E(\lceil\psi_i\rceil, \Lambda)}{E(\lfloor\psi_1 \vee \psi_2\rfloor, \Lambda)}$$

$$(BA\wedge): \frac{A(\lceil\psi_i\rceil, \Lambda)}{A(\lfloor\psi_1 \wedge \psi_2\rfloor, \Lambda)} \qquad (BE\wedge): \frac{E(\lceil\psi_1\rceil, \lceil\psi_2\rceil, \Lambda)}{E(\lfloor\psi_1 \wedge \psi_2\rfloor, \Lambda)}$$

$$(B\sigma): \frac{Q(\lceil X\rceil, \Lambda)}{Q(\lfloor\sigma X.\psi\rfloor, \Lambda)} \qquad (BV): \frac{Q(\lceil body_{\psi^*}(X)\rceil, \Lambda)}{Q(\lfloor X\rfloor, \Lambda)}$$

$$(BW): \frac{Q(\Lambda)}{Q(\lfloor\varphi\rfloor, \Lambda)} \qquad (BQ): \frac{Q'(\lceil\psi\rceil)}{Q(\lfloor Q'\psi\rfloor, \Lambda)}$$

$$(BX): \frac{Q(\lceil\psi_1\rceil, ..., \lceil\psi_n\rceil)}{Q(\lfloor\bigcirc\psi_1\rfloor, ..., \lfloor\bigcirc\psi_n\rfloor)} \qquad (BS): \frac{Q(\Lambda)}{Q(\Lambda)}$$

whereas $\Lambda \subseteq Sub(\psi^*)$, $\psi_1, \psi_2, ..., \psi_n \in Sub(\psi^*)$, $\varphi \in Sub(\psi^*)$ either being a literal or a path quantified formula, $\sigma X.\psi \in Sub(\psi^*)$, $X \in Bound(\psi^*)$ and $Q, Q' \in \{E, A\}$.

A rule instance moreover specifies a *set of principal formulas* as well as a *set of principal successive formulas*. All principal formulas are encapsulated by $\lfloor$- and $\rfloor$-brackets; all principal successive formulas are encapsulated by $\lceil$- and $\rceil$-brackets.

All expressions of the form $A(\psi_1; ...; \psi_n)$, $A(\Lambda)$, $A(\bigvee\Lambda)$, $E(\psi_1, ..., \psi_n)$, $E(\Lambda)$, $E(\bigwedge\Lambda)$ are called *blocks*. A block simply is a quantification-labelled set of formulas.

More formally a block B is a tuple (Q, Λ) with $Q \in \{E, A\}$ and Λ a set of formulas. To identify both components Q and Λ of a block B one simply defines $B_1 := Q$ and $B_2 := \Lambda$; A block (Q, Λ) is often written as $Q(\Lambda)$.

The set of blocks w.r.t. to a formula ψ, $Blo(\psi)$, consists of all blocks B s.t. each formula occurring in B is a subformula of ψ.

The (BS)-rule might seem to be a bit confusing: What is a rule good for that simply does nothing? We will be integrating the bundle tableaux system into larger tableaux contexts possibly listing more than one bundle block. Due to the fact that not every bundle block is affected by a rule of the larger tableaux context it is useful to already allow thread bundles to be

stalled. In fact it will suffice that a thread bundle is lively, meaning that there are rules applied infinitely often despite the (BS)-rule.

A linear tableaux-style proof built according to the proof rules is formalised as follows. Note that the definition allows thread bundles of finite as well as infinite length:

Definition 3.13 (Thread bundle)**.** Let $\psi \in CTL^*_{c\mu+}$. A *thread bundle w.r.t.* ψ is a tuple $\mathcal{B} = (dom(\mathcal{B}), \mathcal{Q}, \mathcal{L}, \Phi, \Psi, \mathcal{R})$ where

- $dom(\mathcal{B}) = \mathbb{N}$ or $\exists n > 0 : dom(\mathcal{B}) = \{i \in \mathbb{N} \mid i < n\}$
- $idom(\mathcal{B}) := \{i \in \mathbb{N} \mid i + 1 \in dom(\mathcal{B})\}$ ("inner domain")
- $\mathcal{Q} : dom(\mathcal{B}) \to \{E, A\}$ is a labelling map
- $\mathcal{L} = \mathcal{L}_0, \mathcal{L}_1, ...$ is a sequence of non-empty formula sets with $\emptyset \neq \mathcal{L}_i \subseteq Sub(\psi)$ for all $i \in dom(\mathcal{B})$ and $|\mathcal{L}_0| = 1$ with $\mathcal{L}_0 \subseteq CTL^*_{c\mu+}$
- $\Phi : idom(\mathcal{B}) \to \mathcal{P}(Sub(\psi))$ is a principal formulas marker with $\Phi(i) \subseteq \mathcal{L}_i$ for all $i \in D$
- $\Psi : (dom(\mathcal{B}) \setminus \{0\}) \to \mathcal{P}(Sub(\psi))$ is a principal successive formulas marker with $\Psi(i) \subseteq \mathcal{L}_i$ for all $i \in D$
- $\mathcal{R} : idom(\mathcal{B}) \to BundleRules$ is a rule map

s.t. for all $i \in idom(\mathcal{B})$

$$(\mathcal{R}(i)) : \frac{\mathcal{Q}_{i+1}(\lceil \Psi(i+1) \rceil, \mathcal{L}_{i+1})}{\mathcal{Q}_i(\lfloor \Phi(i) \rfloor, \mathcal{L}_i \setminus \Phi(i))}$$

is a valid bundle rule instance.

A thread bundle $\mathcal{B}$ is called *proper thread bundle* iff $dom(\mathcal{B}) = \mathbb{N}$ and *prefix thread bundle* otherwise. The set of all proper thread bundles w.r.t. ψ^* is denoted by $\mathcal{BU}(\psi^*)$.
A proper thread bundle $\mathcal{B}$ is called

- *lively* iff $\forall i \in \mathbb{N}\, \exists j > i : \mathcal{R}(j) \neq (BS)$
- *progressive* iff $\forall i \in \mathbb{N}\, \exists j > i : \mathcal{R}(j) = (BX)$

- *finally preserving* iff $\exists i \in \mathbb{N} \forall j > i : \mathcal{R}(j) \neq (BQ)$

As before, we define the set of *roots*, this time in the context of thread bundles. A position in a thread bundle is a *root* iff it is the very first position or the last rule that was applied before the position in question was a path quantification rule.

Definition 3.14 (Thread bundle roots and helper maps)**.** Let $\psi \in CTL^*_{c\mu+}$ and $\mathcal{B}$ be a thread bundle w.r.t. ψ. The *progress between* $i \in dom(\mathcal{B})$ *and* $j \in dom(\mathcal{B})$ *w.r.t.* $\mathcal{B}$ measures the number of unfolded next-operators between i and j:

$$pro_{\mathcal{B}}(i,j) := |\{i \leq k < j \mid \mathcal{R}_k \equiv (BX)\}|$$

The set of *roots w.r.t.* $\mathcal{B}$ is defined as follows:

$$roots_{\mathcal{B}} := \{i \in dom(\mathcal{B}) \mid i = 0 \vee \mathcal{R}_{i-1} = (BQ)\}$$

The roots without the last root (if existing) of $\mathcal{B}$ is denoted by $roots^*_{\mathcal{B}}$ i.e.

$$roots^*_{\mathcal{B}} := \begin{cases} roots_{\mathcal{B}} & \text{if } |roots_{\mathcal{B}}| = \infty \\ roots_{\mathcal{B}} \setminus \{\max(roots_{\mathcal{B}})\} & \text{otherwise} \end{cases}$$

The root helper functions are moreover defined as follows:

- $root_{\mathcal{B}} : dom(\mathcal{B}) \to roots_{\mathcal{B}}$, $i \mapsto \max\{j \in roots_{\mathcal{B}} \mid j \leq i\}$
- $nroot_{\mathcal{B}} : (roots^*_{\mathcal{B}})_{\leq} \to roots_{\mathcal{B}}$, $r \mapsto \min\{j \in roots_{\mathcal{B}} \mid j > r\}$
- $rsize_{\mathcal{B}} : roots^*_{\mathcal{B}} \to \mathbb{N}$, $r \mapsto pro_{\mathcal{B}}(r, nroot_{\mathcal{B}}(r))$

where $A_{\leq} := \{i \in \mathbb{N} \mid \exists a \in A : i \leq a\}$.

Our formalisation of thread bundles demands the very first node to be containing exactly one formula. The reason for this restriction will become clear in the course of this section. Roughly said, we will use the determinism of parity games to derive knowledge about what kind of threads are contained in a thread bundle (and which kinds are not!) and therefore it will be crucial to have only one starting position - being the single formula in question - in the thread bundle that is considered.

Example 3.15. The prefix of a well-built thread bundle with respect to

$$E(\mu X.((\bigcirc X) \wedge (A(p \vee \bigcirc X)))$$

might run as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{A([(\bigcirc X) \wedge (A(p \vee \bigcirc X)])}\,(BA\wedge)}{A([X])}\,(B\mu)}{A([\bigcirc X])}\,(BX)}{A(\lceil p \rceil; [\bigcirc X])}\,(BW)}{A(\lfloor p \vee \bigcirc X \rfloor)}\,(BA\vee)}{A(\lceil p \vee \bigcirc X \rceil)}\,(BS)}{E(\lceil \bigcirc X \rceil, [A(p \vee \bigcirc X)])}\,(BA)}{E([(\bigcirc X) \wedge (A(p \vee \bigcirc X))])}\,(BE\wedge)}{E([X])}\,(BV)}{E([\mu X.((\bigcirc X) \wedge (A(p \vee \bigcirc X))])}\,(B\mu)$$

A *thread bundle suffix* of a thread bundle is an infinite suffix starting at a root of the thread bundle. We will be considering thread bundles that are built starting with a restricted formula having an important property: They are always finally preserving after a while resulting in a thread bundle suffix with a single root.

Definition 3.16 (Thread bundle suffix)**.** *Let $\psi^* \in CTL^*_{c\mu+}$ and let $\mathcal{B} = (dom(\mathcal{B}), \mathcal{Q}, \mathcal{L}, \Phi, \Psi, \mathcal{R})$ be a thread bundle w.r.t. ψ^*. A thread bundle suffix w.r.t. $\mathcal{B}$ is a thread bundle $\mathcal{B}' = (dom(\mathcal{B}'), \mathcal{Q}', \mathcal{L}', \Phi', \Psi', \mathcal{R}')$ w.r.t. ψ^* s.t. there is an $j \in \mathbb{N}$ with*

- $dom(\mathcal{B}') := \{i \in \mathbb{N} \mid i + j \in dom(\mathcal{B})\}$
- $\mathcal{Q}' : i \mapsto \mathcal{Q}_{i+j}$, $\mathcal{L}' : i \mapsto \mathcal{L}_{i+j}$, $\Phi' : i \mapsto \Phi_{i+j}$, $\Psi' : i \mapsto \Psi_{i+j}$ *and* $\mathcal{R}' : i \mapsto \mathcal{R}_{i+j}$

Such a thread bundle suffix is said to be starting at j and is denoted by $\mathcal{B}[j]$.

All global properties of thread bundle can be transferred to thread bundle suffixes since finite prefixes of a thread bundle have no effect on properties capturing the infinite behaviour.

Corollary 3.17 (Thread bundle suffix properties)**.** *Let $\psi^* \in CTL^*_{c\mu+}$ let $\mathcal{B}$ be a thread bundle w.r.t. ψ^* and let $\mathcal{B}'$ be a thread bundle suffix w.r.t. $\mathcal{B}$. Then $\mathcal{B}$ is proper / lively / progressive / finally preserving iff $\mathcal{B}'$ is.*

If a formula is restricted, there is no fixed point with a bound variable having a path quantification inbetween. Thus a path quantifying subformula is not able to generate any formulas not occurring in the subformula itself, particularly the subformula can not lead to a path quantifying formula that is not occurring in the subformula itself. Hence a path quantification rule following the quantification can only be performed finitely often.

Lemma 3.18 (Restricted thread bundles are finally preserving)**.** *Let $\psi \in CTL^*_{c\mu+}$ be restricted and let $\mathcal{B}$ be a proper thread bundle w.r.t. ψ. Then $\mathcal{B}$ is finally preserving.*

Proof. By contradiction assume that $\mathcal{B}$ is not finally preserving i.e. $roots_{\mathcal{B}}$ is infinite. For each $i \in roots_{\mathcal{B}}$ let $i' := min\{j \in roots_{\mathcal{B}} \mid j > i\}$. Moreover define $\psi(i)$ to be single formula in $\mathcal{B}_i$ for each $i \in roots_{\mathcal{B}}$.

Due to the wellfoundedness of natural numbers it suffices to show that for each $i \in roots_{\mathcal{B}}$ the following holds:

$$|Sub(\psi(i'))| < |Sub(\psi(i))|$$

Hence let $i \in roots_{\mathcal{B}}$ be arbitrary. Due to the fact that ψ is restricted $\psi(i') \in Sub(\psi(i))$ as well as $\psi(i) \notin Sub(\psi(i'))$ holds which implies $|Sub(\psi(i'))| < |Sub(\psi(i))|$. □

Obviously each progressive thread bundle is lively. The conversion holds for thread bundles containing guarded formulas only.

Lemma 3.19 (Lively guarded thread bundles are progressive)**.** *Let $\psi \in CTL^*_{c\mu+}$ be guarded and let $\mathcal{B}$ be a lively thread bundle w.r.t. ψ. Then $\mathcal{B}$ is progressive.*

Proof. Let $\psi \in CTL^*_{c\mu+}$ be guarded and let $\mathcal{B}$ be a lively thread bundle w.r.t. ψ i.e.

$$(1)\ \forall i \in \mathbb{N}\, \exists j > i\ :\ \mathcal{R}(j) \neq (BS)$$

By contradiction assume that $\mathcal{B}$ is not progressive i.e. assume that there is an $i^* \in \mathbb{N}$ s.t.

$$(2)\ \forall j > i^*\ :\ \mathcal{R}(j) \neq (BX)$$

Let $\psi_1, ..., \psi_n$ be all subformulas of ψ s.t. $pdist_\psi(\psi_k) \geq pdist_\psi(\psi_l)$ for all $k \leq l$. We define a linear order $\prec_\psi$ on subsets of $Sub(\psi)$ as follows

$$S \prec_\psi T \iff \exists j \leq n :\ \psi_j \in T \wedge \psi_j \notin S \wedge (\forall k < j :\ \psi_k \in S \iff \psi_k \in T)$$

It suffices to show (by induction on $i > i^*$) that

$$(3)\ \mathcal{L}_{i+1} = \mathcal{L}_i \text{ for all } i > i^* \text{ with } \mathcal{R}(i) = (BS)$$

as well as

$$(4)\ \mathcal{L}_{i+1} \prec_\psi \mathcal{L}_i \text{ for all } i > i^* \text{ with } \mathcal{R}(i) \neq (BS)$$

since in this case $\mathcal{L}_{i^*+1}, \mathcal{L}_{i^*+2}, ...$ is an infinitely $\prec_\psi$-decreasing sequence due to the liveliness (1) of $\mathcal{B}$ which is impossible with $\prec_\psi$ being (obviously) well-founded.

To see that (4) holds let $i > i^*$ be arbitrary s.t. $\mathcal{R}(i) \neq (BS)$ and $\mathcal{R}(i) \neq (BX)$ (due to (2)). For $r_i \equiv (BW)$ (4) holds by definition of $\prec_\psi$ and otherwise (i.e. $r_i \equiv (BA\wedge), (BA\vee), (BE\wedge), (BE\vee), (BQ), (B\sigma), (BV)$) (4) again holds by definition of $\prec_\psi$ and $pdist_\psi$. □

The reason for our interest in the investigation of progressive thread bundles is as follows: A larger tableaux whose nodes are labelled with a list of bundle blocks is likely to provide a modal rule, too. The thing with sound and complete modal rules usually is that one needs to demand that *every* single formula being of potential further use is under a modality.

Hence, to be able to perform the modal rule in a larger tableaux, it will be required that the bundle modal rule is applicable to each bundle block in the tableaux node. Now, if there is a bundle block which only finitely often allows to perform the bundle rule, automatically all other bundle blocks being ready for the modal rule are stalled. Thus, if every thread bundle being possibly contained in the larger tableaux is progressive then all thread bundles contained in the tableaux will be lively due to the fact that all demand the application of the modal rule from time to time affecting *all* thread blocks.

Unfortunately we are not able to design a proof system for full CTL^*_μ for a bunch of reasons. For one thing, we cannot provide a complete guarded transformation. Luckily it suffices for finally preserving thread bundles to be progressive to have state-guarded formulas:

Lemma 3.20 (Lively state-guarded finally preserving thread bundles)**.** *Let $\psi^* \in CTL^*_{c\mu+}$ be state-guarded and let $\mathcal{B}$ be a lively finally preserving thread bundle w.r.t. ψ^*. Then $\mathcal{B}$ is progressive.*

Proof. Let $\psi^* \in CTL^*_{c\mu+}$ be state-guarded and let $\mathcal{B}$ be a lively finally preserving thread bundle w.r.t. ψ^*. Let moreover $r := \max(roots_{\mathcal{B}})$. Then $|\mathcal{L}_r| = 1$, hence $\mathcal{B}' := \mathcal{B}[r]$ is a thread bundle with

$$\forall i \in \mathbb{N} : \ \mathcal{R}'(i) \neq (BQ)$$

Now consider the following replacement function f:

$$f : \psi \mapsto \begin{cases} op(f(\psi_1), f(\psi_2)) & \text{if } \psi \equiv op(\psi_1, \psi_2) \\ op() & \text{if } \psi \equiv op() \\ \bigcirc f(\psi') & \text{if } \psi \equiv E\psi', A\psi' \\ op(f(\psi')) & \text{if } \psi \equiv op(\psi'),\ op \equiv \bigcirc, \neg \end{cases}$$

Apply f to each formula occurring in $\mathcal{B}'$ resulting in $\mathcal{B}''$ and note that

- $\mathcal{B}''$ is a lively thread bundle w.r.t. $f(\psi^*)$
- $f(\psi^*)$ is guarded

Hence Lemma 3.19 implies that $\mathcal{B}''$ is progressive thus so is $\mathcal{B}$. □

Again, we define the *dual thread bundle* which basically simply replaces each information by its canonical counterpart.

Definition 3.21 (Dual thread bundle)**.** Let $\psi^* \in CTL^*_{c\mu+}$ and let $\mathcal{B} = (dom(\mathcal{B}), \mathcal{Q}, \mathcal{L}, \Phi, \Psi, \mathcal{R})$ be a thread bundle w.r.t. ψ^*. The *dual thread bundle* $\overline{\mathcal{B}} = (dom(\mathcal{B}), \overline{\mathcal{Q}}, \overline{\mathcal{L}}, \overline{\Phi}, \overline{\Psi}, \overline{\mathcal{R}})$ is defined as follows:

- $\overline{\mathcal{Q}} : i \mapsto Q_i^C$ with $E^C := A$ and $A^C := E$
- $\overline{\mathcal{L}} : i \mapsto \{\psi^{\ominus} \mid \psi \in \mathcal{L}_i\}$
- $\overline{\Phi} : i \mapsto \{\psi^{\ominus} \mid \psi \in \Phi(i)\}$
- $\overline{\Psi} : i \mapsto \{\psi^{\ominus} \mid \psi \in \Psi(i)\}$
- $\overline{\mathcal{R}} : i \mapsto \mathcal{R}_i^C$ where $(BA\vee)^C := (BE\wedge)$, $(BE\vee)^C := (BA\wedge)$, $(BA\wedge)^C := (BE\vee)$, $(BE\wedge)^C := (BA\vee)$, $(B\mu)^C := (B\nu)^C$, $(B\nu)^C := (B\mu)$, $(BV)^C := (BV)$, $(BW)^C := (BW)$, $(BE)^C := (BA)$, $(BA)^C := (BE)$, $(BX)^C := (BX)$ and $(BS)^C := (BS)$.

All global properties of a thread bundle can clearly be transferred to the dual thread bundle.

Corollary 3.22 (Dual thread bundle properties)**.** *Let $\psi^* \in CTL^*_{c\mu+}$ and let $\mathcal{B}$ be a thread bundle w.r.t. ψ^*. Then:*

1. *The dual thread bundle $\overline{\mathcal{B}}$ is a thread bundle w.r.t. $(\psi^*)^{\ominus}$.*
2. *$\mathcal{B}$ is proper / lively / progressive / finally preserving iff $\overline{\mathcal{B}}$ is.*

Although we are interested in the threads that go through a thread bundle, not each well-built thread that can be matched with formulas occurring in a thread bundle is of use. The threads we are looking for follow the bundle rules, meaning that a formula following another formula in a thread needs to be depending on the preceding formula. Thus either both formulas are equal or the transition of the formulas needs to be an effect of the respective rule application.

Definition 3.23 (Connection relation)**.** Let $\psi \in CTL^*_{c\mu+}$ and let $\mathcal{B}$ be a thread bundle w.r.t. ψ. The *connection relation w.r.t. $\mathcal{B}$ and $i \in idom(\mathcal{B})$* is defined as follows:

$$\psi_1 \rightarrow^i_{\mathcal{B}} \psi_2 \subseteq \mathcal{L}_i \times \mathcal{L}_{i+1}$$
$$\text{iff}$$
$$\text{either } \mathcal{R}(i) = (BX) \text{ and } \psi_1 \equiv \bigcirc\psi_2$$
$$\text{or } \mathcal{R}(i) \neq (BX), (BQ) \text{ and } \psi_1 = \psi_2$$
$$\text{or } \mathcal{R}(i) \neq (BX) \text{ and } \psi_1 \in \Phi(i) \text{ and } \psi_2 \in \Psi(i+1)$$

Each formula occurring in a block of a thread bundle is triggered by a formula occurring in the preceding block of the thread bundle, hence each formula can be connected to at least one preceding formula:

Corollary 3.24 (Connection relation is not empty)**.** *Let $\psi \in CTL^*_{c\mu+}$ and let $\mathcal{B}$ be a thread bundle w.r.t. ψ. Then*

$$\forall i \in idom(\mathcal{B})\, \forall \psi_2 \in \mathcal{L}_{i+1}\, \exists \psi_1 \in \mathcal{L}_i \;:\; \psi_1 \rightarrow^i_{\mathcal{B}} \psi_2$$

The threads in a thread bundle are all these threads having their i-th formula in the i-th block of the bundle following connection relation.

Definition 3.25 (Set of threads)**.** Let $\psi \in CTL^*_{c\mu+}$ and let $\mathcal{B}$ be a proper thread bundle w.r.t. ψ. The *set of threads in $\mathcal{B}$* is defined as follows:

$$t \in Th(\mathcal{B}) : \quad \Longleftrightarrow \quad t \text{ is a thread w.r.t. } \psi \text{ and } t_i \rightarrow^i_{\mathcal{B}} t_{i+1} \text{ for all } i \in \mathbb{N}$$

The subset of ν-threads is denoted by $Th_\nu(\mathcal{B})$; the subset of μ-threads is denoted by $Th_\mu(\mathcal{B})$ accordingly.

A proper thread bundle $\mathcal{B}$ is called *fair* iff each thread $t \in Th(\mathcal{B})$ is lively. The subset of all fair thread bundles w.r.t. ψ^* is denoted by $\mathcal{BUF}(\psi^*)$.

Not very surprisingly, the roots of a thread bundle match the roots of all threads contained in the thread bundle.

Corollary 3.26 (Thread bundle roots are compatible with thread roots)**.** *Let $\psi \in CTL^*_{c\mu+}$ and $\mathcal{B}$ be a proper thread bundle w.r.t. ψ. Let moreover $t \in Th(\mathcal{B})$ be arbitrary. Then all root sets and helper maps w.r.t. t and $\mathcal{B}$ are identical.*

Each thread contained in a thread bundle corresponds to its dual thread contained in the dual thread bundle and vice versa. We conclude that a thread bundle is fair if and only if its dual thread bundle is fair due to Corollary 3.6.

Corollary 3.27 (Dual thread bundle threads)**.** *Let $\psi^* \in CTL^*_{c\mu+}$ and let $\mathcal{B}$ be a proper thread bundle w.r.t. ψ^*. Then*

$$Th(\overline{\mathcal{B}}) = \{\overline{t} \mid t \in Th(\mathcal{B})\}$$

Moreover $\mathcal{B}$ is fair iff $\overline{\mathcal{B}}$ is fair.

Similarly each thread in a thread bundle can be mapped to an infinite thread suffix in a thread bundle suffix having the same infinite properties.

Corollary 3.28 (Suffix thread bundle threads)**.** *Let $\psi^* \in CTL^*_{c\mu+}$, let $\mathcal{B}$ be a proper thread bundle w.r.t. ψ^* and let $\mathcal{B}'$ be a thread bundle suffix w.r.t. $\mathcal{B}$ starting at j. There is a natural identification surjection $th_{(\mathcal{B},\mathcal{B}')} : Th(\mathcal{B}) \to Th(\mathcal{B}')$ defined as follows:*

$$th_{(\mathcal{B},\mathcal{B}')} : t \mapsto \lambda i.t_{i+j}$$

Moreover the following holds:

- *$\mathcal{B}$ is fair iff $\mathcal{B}'$ is fair*
- *$t \in Th_\mu(\mathcal{B})$ iff $th_{(\mathcal{B},\mathcal{B}')}(t) \in Th_\mu(\mathcal{B}')$*
- *$t \in Th_\nu(\mathcal{B})$ iff $th_{(\mathcal{B},\mathcal{B}')}(t) \in Th_\nu(\mathcal{B}')$*

Progressive thread bundles obviously are fair since the modal rule is applied infinitely often which performs an unreeling of each occurring formula every time it is applied.

Corollary 3.29 (Progressive thread bundles are fair)**.** *Let $\psi \in CTL^*_{c\mu+}$ and let $\mathcal{B}$ be a progressive thread bundle w.r.t. ψ. Then $\mathcal{B}$ is fair.*

Proof. Let $t \in Th(\mathcal{B})$ and $i \in \mathbb{N}$ be arbitrary. We need to show that there is a $j > i$ s.t. $t_j \neq t_i$. Since $\mathcal{B}$ is progressive there is a $k > i$ s.t. $\mathcal{R}(k) = (BX)$. If $t_k \neq t_i$ we're done; otherwise note that $t_k = \bigcirc t_{k+1}$ and therefore $t_{k+1} \neq t_k$. □

By Lemma 3.19 and Corollary 3.29 we derive that lively guarded thread bundles are fair.

Corollary 3.30 (Lively guarded thread bundles are fair)**.** *Let $\psi \in CTL^*_{c\mu+}$ be guarded and let $\mathcal{B}$ be a lively thread bundle w.r.t. ψ. Then $\mathcal{B}$ is fair.*

More surprisingly this also holds for lively state-guarded thread bundles: If the thread bundle is finally preserving, it is progressive and thus fair. Otherwise there are path formulas that are infinitely often spawned and followed and hence thread bundles that are not finally preserving are fair.

Corollary 3.31 (Lively state-guarded thread bundles are fair)**.** *Let $\psi \in CTL^*_{c\mu+}$ be state guarded and let $\mathcal{B}$ be a lively thread bundle w.r.t. ψ. Then $\mathcal{B}$ is fair.*

Proof. By Lemma 3.20. If $\mathcal{B}$ is finally preserving then Corollary 3.29 states that $\mathcal{B}$ is fair; otherwise the following holds:

$$\forall i \in \mathbb{N}\, \exists j > i \,:\, \mathcal{R}(j) = (BQ)$$

Since each thread $t \in Th(\mathcal{B})$ is affected by (BQ) rule instances $\mathcal{B}$ is fair. □

Each thread bundle contains at least one thread due to the fact that each formula in a block of the thread bundle is connected to at least one preceding formula.

Lemma 3.32 (Set of threads is not empty)**.** *Let $\psi \in CTL^*_{c\mu+}$ and let $\mathcal{B}$ be a proper thread bundle w.r.t. ψ. Then the set of threads is not empty, i.e. $Th(\mathcal{B}) \neq \emptyset$.*

Proof. Due to Corollary 3.24 and the non-emptiness of all blocks in $\mathcal{B}$, i.e. $\emptyset \neq \mathcal{L}_i$ for all $i \in \mathbb{N}$, the connection relation $\rightarrow^i_{\mathcal{B}}$ induces a finitely branching tree of infinite size, hence Königs Lemma (A.3) comprises a thread $t \in Th(\mathcal{B})$. □

Focussing on single threads in a thread bundle is reasonable if the underlying quantification under the path quantifier is existential i.e. logically a disjunction. In such a case we could simply search for a single ν-thread in a thread bundle to derive that the whole thread bundle is valid. But this scenario is only true for universaly tagged thread bundles.

For existentially tagged thread bundles it is a different story: The underlying quantification is universal, thus logically a conjunction. Hence it does not suffice to find a ν-thread, we rather need to verify that *every* possible thread in the thread bundle is a ν-thread.

In general both cases can occur in a thread bundle in an alternating way. In this general scenario one needs to name one thread part going through each existential segment so that each thread that can be built using the selected existential parts is a ν-thread. Recap that whenever a quantification rule is applied all formulas but the principal formula are removed from the sequent. This particularly implies that all threads going through a thread bundle share the same roots, thus it is a legal way to construct a thread by gluing parts going from one to the next level together.

We formalise the idea of fixing the existential choices of threads by defining an equivalence relation: Two threads are E-equivalent iff they match on all existential segments. Similarly two threads are A-equivalent iff they match on all universal segments.

Definition 3.33 (Equivalent threads)**.** Let $\psi \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a proper thread bundle w.r.t. ψ and $t, s \in Th(\mathcal{L})$. Define $t \equiv^A_{\mathcal{B}} s$ as well as $t \equiv^E_{\mathcal{B}} s$ as follows:

$$t \equiv^A_{\mathcal{B}} s \quad :\Longleftrightarrow \quad \forall i : \mathcal{Q}_i = A \Rightarrow t_i = s_i$$

$$t \equiv^E_{\mathcal{B}} s \quad :\Longleftrightarrow \quad \forall i : \mathcal{Q}_i = E \Rightarrow t_i = s_i$$

Both relations are quite obviously equivalence relations.

Corollary 3.34 (Equivalence relation)**.** *Let $\psi \in CTL^*_{c\mu+}$ and $\mathcal{B}$ be a proper thread bundle w.r.t. ψ. Then $\equiv^A_{\mathcal{B}}$ as well as $\equiv^E_{\mathcal{B}}$ are equivalence relations.*

By duality of threads and thread bundles we derive that the dual versions of E-equivalent threads with respect to a thread bundle B are A-equivalent with respect to the dual thread bundle and vice versa.

Corollary 3.35 (Dual thread equivalence)**.** *Let $\psi \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a proper thread bundle w.r.t. ψ and $s, t \in Th(\mathcal{B})$. Then $s \equiv^E_{\mathcal{B}} t$ iff $\overline{s} \equiv^A_{\overline{\mathcal{B}}} \overline{t}$ as well as $s \equiv^A_{\mathcal{B}} t$ iff $\overline{s} \equiv^E_{\overline{\mathcal{B}}} \overline{t}$.*

An important property of thread bundles and the contained threads is the *inclusion determinism*: There is either an A-equivalence class of ν-threads or an E-equivalence class of μ-threads. In order to prove this we interpret the connected formulas in a thread bundle as game graph of a parity game (see Appendix C).

By determinism of parity games (Theorem C.2) each position in the game graph is either won by the existential player or by the universal player. By definition of thread bundles, all threads in the bundle start with the same formula at the first level and thus the membership of the starting node on a winning set determines which of the two conditions holds.

Lemma 3.36 (Thread equivalence inclusion determinism)**.** *Let $\psi \in CTL^*_{c\mu+}$ and $\mathcal{B}$ be a fair thread bundle w.r.t. ψ. Then*

$$\exists t \in Th(\mathcal{B}) : [t]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B})$$

if not and only if not

$$\exists t \in Th(\mathcal{B}) : [t]_{\equiv^E_{\mathcal{B}}} \subseteq Th_\mu(\mathcal{B})$$

Proof. Let $\psi \in CTL^*_{c\mu+}$ and $\mathcal{B} = (\mathcal{Q}, \mathcal{L}, \Phi, \Psi, \mathcal{R})$ be a thread bundle w.r.t. ψ. Consider the infinite parity game $\mathcal{G} := (V_A, V_E, \delta, \Omega)$ (with $V := V_A \cup V_E$) where

- $V_A := \{(\psi', i) \mid \psi' \in \mathcal{L}_i,\ \mathcal{Q}_i = A,\ \exists t \in Th(\mathcal{B}) : t_i = \psi'\}$
- $V_E := \{(\psi', i) \mid \psi' \in \mathcal{L}_i,\ \mathcal{Q}_i = E,\ \exists t \in Th(\mathcal{B}) : t_i = \psi'\}$
- $\delta : (\psi', i) \in V \mapsto \{(\psi'', i+1) \in V \mid \psi' \rightarrow^i_{\mathcal{B}} \psi''\}$
- $\Omega : (\psi', i) \mapsto \begin{cases} pri^\psi(X) & \text{if } \psi' \equiv X \in \mathcal{V} \\ 0 & \text{otherwise} \end{cases}$

Due to Lemma 3.32 the following two properties (*) hold:

1. For each $i \in \mathbb{N}$ there is a ψ' s.t. $(\psi', i) \in V$.
2. For each $p \in V$ there is a $q \in V$ s.t. $q \in \delta(p)$.

Let α be the singleton formula in $\mathcal{L}_0$. By positional determinism of parity games (Theorem C.2) it suffices to show the following for player A and ν-threads (dually for player E and μ-threads):

$$\exists t \in Th(\mathcal{L}) : [t]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B})$$
$$\Longleftrightarrow$$
$$A \text{ wins on } (\alpha, 0)$$

For the "if"-part let W_A be the winning set of player A and $s_A : W_A \cap V_A \to W_A$ be a winning strategy for player A. Let moreover $s : V_A \cup V_E \to V_A \cup V_E$ be an arbitrary extension of s_A s.t. s is a valid strategy on the whole game (which exists due to (*)).
Define t as follows:

$$t_i := \begin{cases} \alpha & \text{if } i = 0 \\ s(t_{i-1}, i-1) & \text{if } i > 0 \end{cases}$$

It remains to show that $[t]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B})$. Hence let $u \in [t]_{\equiv^A_{\mathcal{B}}}$ be arbitrary; since $(u_i, i)_{i \in \mathbb{N}}$ is a valid play of $\mathcal{G}$ that is played according to the winning strategy s_A starting in a winning position for A the greatest priority p occurring in the play infinitely often is even. Hence the dominating variable in u is a ν-variable.

The "only-if"-part can be easily shown by contraposition. By positional determinism E wins on $(\alpha, 0)$, hence $\exists t \in Th(\mathcal{L}) : [t]_{\equiv^E_{\mathcal{B}}} \subseteq Th_\mu(\mathcal{B})$. Now assume that there is an $u \in Th(\mathcal{L}) : [u]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B})$. Since each thread is either a ν-thread or a μ-thread $[u]_{\equiv^A_{\mathcal{B}}} \cap [t]_{\equiv^E_{\mathcal{B}}} = \emptyset$ holds.

But this is impossible due to the fact that the following thread w is equivalent to both t and u i.e. $w \equiv^E_{\mathcal{L}} t$ as well as $w \equiv^A_{\mathcal{L}} u$:

$$w_i := \begin{cases} t_i & \text{if } \mathcal{Q}_i = E \\ u_i & \text{if } \mathcal{Q}_i = A \end{cases}$$

□

3.4 Thread bundle annotations

Thread bundle annotations are basically the same semantical constructs as thread annotations. Since all threads in a thread bundle share the same roots, the annotation of a thread bundle simply is an annotation for all threads in the bundle. Clearly, it is not possible for a thread bundle annotation to be falsifying w.r.t. each thread or satisfying w.r.t. each thread contained in a bundle in general.

If the bundle annotation is falsifying for instance and the path quantification labelling is universal at some position, then all threads are also falsified at this position as the underlying quantification is a disjunction. With existential labellings, the underlying quantification is conjunction. If a conjunction is falsified there is at least one formula under the conjunction which is falsified itself but not necessarily all of them.

To identify all these threads that are continuously falsified, one inductively defines the *candidate set* for all existential segments of the bundle in question. It sorts all these formulas out that are not falsified or have no preceding formula in the preceding candidate set.

Dual problems arise when dealing with satisfying annotations. In this case the candidate set filters with respect to all universal segments. We simply provide one definition of the candidate set in the following which will be used for both falsifying and satisfying annotations.

Definition 3.37 (Formula candidate set)**.** Let $\psi \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a thread bundle w.r.t. ψ, $\mathcal{T}$ be an LTS and π be a path in $\mathcal{T}$. The *candidate set map w.r.t. $\mathcal{B}$, π and $i \in dom(\mathcal{B})$* is inductively defined as follows:

$$Cand^{\pi}_{\mathcal{B}} : i \mapsto \begin{cases} \mathcal{L}_i & \text{if } i \in roots_{\mathcal{B}} \\ \{\psi' \in \mathcal{L}_i \mid \exists \psi'' \in Cand^{\pi}_{\mathcal{B}}(i-1) : \ pr_{i-1}(\psi'', \psi')\} & \text{otherwise} \end{cases}$$

where $pr_i(\psi_1, \psi_2)$ holds iff one of the following conditions is true:

1. $R(i) \not\equiv (BQ), (BE\wedge), (BA\vee)$ and $\psi_1 \rightarrow^i_{\mathcal{B}} \psi_2$
2. $R(i) \equiv (BE\wedge), (BA\vee)$, $\psi_1 \rightarrow^i_{\mathcal{B}} \psi_2$ and $\psi_1 \notin \Phi(i)$

3. $R(i) \equiv (BE\wedge)$, $\psi_1 \rightarrow^i_{\mathcal{B}} \psi_2$, $\psi_1 \in \Phi(i)$ and if $\zeta \in Sig_\nu(\psi)$ is the least signature s.t. $\pi[r(i)] \not\models\downarrow^\psi_\zeta \psi_1$ then also $\pi[r(i+1)] \not\models\downarrow^\psi_\zeta \psi_2$

4. $R(i) \equiv (BA\vee)$, $\psi_1 \rightarrow^i_{\mathcal{B}} \psi_2$, $\psi_1 \in \Phi(i)$ and if $\zeta \in Sig_\mu(\psi)$ is the least signature s.t. $\pi[r(i)] \models\uparrow^\psi_\zeta \psi_1$ then also $\pi[r(i+1)] \models\uparrow^\psi_\zeta \psi_2$

where $r(i) \equiv pro_{\mathcal{B}}(root_{\mathcal{B}}(i), i)$.

A *thread bundle annotation* now specifies a path for each root of the thread bundle being compliant with the progression: Each path of an annotation starts in the k-th state of the previous path where k denotes the number of unreeled next-operators of the previous segment.

Definition 3.38 (Thread bundle annotation)**.** Let $\psi \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a thread bundle w.r.t. ψ and $\mathcal{T}$ be an LTS. A *thread bundle annotation w.r.t.* $\mathcal{B}$ is a family $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ of paths $\pi_i \in \Pi(\mathcal{T})$ s.t. for all $i \in roots^*_{\mathcal{B}}$ holds

$$\pi_i(rsize_{\mathcal{B}}(i)) = \pi_{nroot_{\mathcal{B}}(i)}(0)$$

For every $i \in dom(\mathcal{B})$ we define the *path w.r.t. t, π and i*, $\pi\langle i\rangle$, as follows

$$\pi\langle i\rangle := \pi_{root_{\mathcal{B}}(i)}[pro_{\mathcal{B}}(root_{\mathcal{B}}(i), i)]$$

A thread bundle annotation $\pi = (\pi_i)_{i roots_{\mathcal{B}}}$ w.r.t. $\mathcal{B}$ is called *falsifying* iff

1. $\pi\langle i\rangle \not\models\updownarrow^\psi E(\bigwedge \mathcal{L}_i)$ for all $i \in dom(\mathcal{B})$ with $\mathcal{Q}_i = E$
2. $\pi\langle i\rangle \not\models\updownarrow^\psi \bigvee \mathcal{L}_i$ for all $i \in dom(\mathcal{B})$ with $\mathcal{Q}_i = A$

A thread bundle annotation $\pi = (\pi_i)_{i roots_{\mathcal{B}}}$ w.r.t. $\mathcal{B}$ is called *satisfying* iff

1. $\pi\langle i\rangle \models\updownarrow^\psi \bigwedge \mathcal{L}_i$ for all $i \in dom(\mathcal{B})$ with $\mathcal{Q}_i = E$
2. $\pi\langle i\rangle \models\updownarrow^\psi A(\bigvee \mathcal{L}_i)$ for all $i \in dom(\mathcal{B})$ with $\mathcal{Q}_i = A$

Again, in order to properly use the bundle annotations for all threads contained in the bundle, we need to demand that the bundle annotation is *signature compatible*.

Definition 3.39 (Signature compatible bundle annotations)**.** Let $\psi \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a thread bundle w.r.t. ψ and $\mathcal{T}$ be an LTS.

A falsifying thread bundle annotation $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ w.r.t. $\mathcal{B}$ is called *signature compatible falsifying* iff for all $i \in idom(\mathcal{B})$ the following properties hold:

1. If $\mathcal{R}_i = (BA\wedge)$, $\Phi(i) = \{\psi_1 \wedge \psi_2\}$, $\Psi(i+1) = \{\psi_k\}$ and $\zeta \in Sig_\nu(\psi)$ is the least signature s.t. $\pi\langle i\rangle \not\models\downarrow^{\psi}_{\zeta} \psi_1 \wedge \psi_2$ then also $\pi\langle i+1\rangle \not\models\downarrow^{\psi}_{\zeta} \psi_k$.
2. If $\mathcal{R}_i = (BQ)$, $\Phi(i) = \{A\psi'\}$, $\Psi(i+1) = \{\psi'\}$ and $\zeta \in Sig_\nu(\psi)$ is the least signature s.t. $\pi\langle i\rangle \not\models\downarrow^{\psi}_{\zeta} A\psi'$ then also $\pi\langle i+1\rangle \not\models\downarrow^{\psi}_{\zeta} \psi'$.
3. If $\mathcal{R}_i \equiv (BQ)$ and $\mathcal{Q}_i = E$ then $\Phi(i) \subseteq Cand_{\mathcal{B}}^{\pi_{root_{\mathcal{B}}(i)}}(i)$
4. If $\mathcal{R}_i \equiv (BW)$ and $\mathcal{Q}_i = E$ then $Cand_{\mathcal{B}}^{\pi_{root_{\mathcal{B}}(i)}}(i+1) \neq \emptyset$

A satisfying thread bundle annotation $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ w.r.t. $\mathcal{B}$ is called *signature compatible satisfying* iff for all $i \in idom(\mathcal{B})$ the following properties hold:

1. If $\mathcal{R}_i = (BE\vee)$, $\Phi(i) = \{\psi_1 \vee \psi_2\}$, $\Psi(i+1) = \{\psi_k\}$ and $\zeta \in Sig_\mu(\psi)$ is the least signature s.t. $\pi\langle i\rangle \models\uparrow^{\psi}_{\zeta} \psi_1 \vee \psi_2$ then also $\pi\langle i+1\rangle \models\uparrow^{\psi}_{\zeta} \psi_k$.
2. If $\mathcal{R}_i = (BQ)$, $\Phi(i) = \{E\psi'\}$, $\Psi(i+1) = \{\psi'\}$ and $\zeta \in Sig_\mu(\psi)$ is the least signature s.t. $\pi\langle i\rangle \models\uparrow^{\psi}_{\zeta} E\psi'$ then also $\pi\langle i+1\rangle \models\uparrow^{\psi}_{\zeta} \psi'$.
3. If $\mathcal{R}_i \equiv (BQ)$ and $\mathcal{Q}_i = A$ then $\Phi(i) \subseteq Cand_{\mathcal{B}}^{\pi_{root_{\mathcal{B}}(i)}}(i)$
4. If $\mathcal{R}_i \equiv (BW)$ and $\mathcal{Q}_i = A$ then $Cand_{\mathcal{B}}^{\pi_{root_{\mathcal{B}}(i)}}(i+1) \neq \emptyset$

By definition of candidate sets and induction on the set index it is not hard to derive some basic facts:

Lemma 3.40 (Candidate set properties)**.** *Let $\psi^* \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a thread bundle w.r.t. ψ^* and $\mathcal{T}$ be an LTS. Let moreover $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ be a signature compatible falsifying (resp. satisfying) thread bundle annotation w.r.t. $\mathcal{B}$. Then the following properties hold for all $i \in idom(\mathcal{B})$ with $\mathcal{Q}_i = E$ (resp. $\mathcal{Q}_i = A$):*

1. $\forall \psi_2 \in Cand_{\mathcal{B}}^{\pi_{root_{\mathcal{B}}(i)}}(i+1)\, \exists \psi_1 \in Cand_{\mathcal{B}}^{\pi_{root_{\mathcal{B}}(i)}}(i) \,:\, \psi_1 \rightarrow^i_{\mathcal{B}} \psi_2$
2. $Cand_{\mathcal{B}}^{\pi_{root_{\mathcal{B}}(i)}}(i) \neq \emptyset$
3. $\pi\langle i \rangle \not\models\!\updownarrow^{\psi} \bigvee Cand_{\mathcal{B}}^{\pi_{root_{\mathcal{B}}(i)}}(i)$ *(resp.* $\pi\langle i \rangle \models\!\updownarrow^{\psi} \bigwedge Cand_{\mathcal{B}}^{\pi_{root_{\mathcal{B}}(i)}}(i)$*)*

The duality of thread bundles complies with the duality of signature compatible thread bundle annotations due to the simple fact that they already comply on the level of threads (Corollary 3.9).

Corollary 3.41 (Dual thread bundle annotations)**.** *Let $\psi \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a thread bundle w.r.t. ψ, $\mathcal{T}$ be an LTS and $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ be a thread bundle annotation w.r.t. $\mathcal{B}$ (and $\overline{\mathcal{B}}$ since $roots_{\mathcal{B}} = roots_{\overline{\mathcal{B}}}$). Then:*

1. *π is (signature compatible) falsifying w.r.t. $\mathcal{B}$ iff π is (signature compatible) satisfying w.r.t. $\overline{\mathcal{B}}$*
2. *π is (signature compatible) satisfying w.r.t. $\mathcal{B}$ iff π is (signature compatible) falsifying w.r.t. $\overline{\mathcal{B}}$*

The following crucial lemma establishes the conditions for transferring signature-compatible falsifying thread bundle annotations to one of its threads: Starting with an arbitrary thread going through the bundle there is at least one A-equivalent thread that is signature-compatible falsified by the bundle annotation.

Lemma 3.42 (Extraction of falsifying threads)**.** *Let $\psi^* \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a fair thread bundle w.r.t. ψ^* and $\mathcal{T}$ be an LTS. Let moreover $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ be a signature compatible falsifying thread bundle annotation w.r.t. $\mathcal{B}$. For each thread $s \in Th(\mathcal{B})$ there is a thread $t \in Th(\mathcal{B})$ with $s \equiv^A_{\mathcal{B}} t$ s.t. $(\pi_i)_{i \in roots_{\mathcal{B}}}$ is a signature compatible falsifying thread annotation w.r.t. t.*

Proof. Let $\psi^* \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a fair thread bundle w.r.t. ψ^* and $\mathcal{T}$ be an LTS. Let moreover $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ be a signature compatible falsifying thread bundle annotation w.r.t. $\mathcal{B}$; let moreover $s \in Th(\mathcal{B})$.
We define C_i for all $i \in \mathbb{N}$ as follows

$$C_i := \begin{cases} \{s_i\} & \text{if } \mathcal{Q}_i = A \\ Cand^{\pi_{root_{\mathcal{B}}(i)}}_{\mathcal{B}}(i) & \text{otherwise} \end{cases}$$

and note that due to Lemma 3.40 the following two properties for all $i \in \mathbb{N}$ hold:

1. $C_i \neq \emptyset$
2. $\forall \psi_2 \in C_{i+1}\, \exists \psi_1 \in C_i \;:\; \psi_1 \rightarrow^i_{\mathcal{B}} \psi_2$

Thus Königs Lemma (A.3) comprises a thread $t \in Th(\mathcal{B})$ with $s \equiv^A_{\mathcal{B}} t$ and

$$t_i \in C_i \text{ for all } i \in \mathbb{N}$$

Then $(\pi_i)_{i \in roots_t}$ is a falsifying thread annotation w.r.t. t since for all $i \in \mathbb{N}$ with $\mathcal{Q}_i = A$ holds that $\pi\langle i \rangle \not\models \updownarrow^{\psi} \bigvee \mathcal{L}_i$ with π being a falsifying thread bundle annotation. For all $i \in \mathbb{N}$ with $\mathcal{Q}_i = E$ Lemma 3.40 states that $\pi\langle i \rangle \not\models \updownarrow^{\psi} \bigvee Cand^{\pi_{root_{\mathcal{B}}(i)}}_{\mathcal{B}}(i)$.

It remains to show that $(\pi_i)_{i \in roots_t}$ is even signature compatible. For principal $t_i \equiv A\psi'$ the signature compatibility directly follows by definition of signature compatible falsifying thread bundle annotations.

For principal $t_i \equiv \psi_1 \wedge \psi_2$ and $\mathcal{Q}_i = A$ the signature compatibility again directly follows by definition; for $\mathcal{Q}_i = E$ the definition of the candidate set ensures the signature compatibility of t. □

To transfer this result to satisfying threads and satisfying annotations, we simply make use of the duality of thread bundles and annotations.

Lemma 3.43 (Extraction of satisfying threads)**.** *Let $\psi^* \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a fair thread bundle w.r.t. ψ^* and $\mathcal{T}$ be an LTS. Let moreover $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ be a signature compatible satisfying thread bundle annotation w.r.t. $\mathcal{B}$. For each thread $s \in Th(\mathcal{B})$ there is a thread $t \in Th(\mathcal{B})$ with $s \equiv^E_{\mathcal{B}} t$ s.t. $(\pi_i)_{i \in roots_{\mathcal{B}}}$ is a signature compatible satisfying thread annotation w.r.t. t.*

Proof. Consider the dual thread bundle $\overline{\mathcal{B}}$. By Corollary 3.27 $\overline{\mathcal{B}}$ is fair and π is a signature compatible falsifying thread bundle annotation due to Corollary 3.41. Let $s \in Th(\mathcal{B})$ i.e. $\overline{s} \in Th(\overline{\mathcal{B}})$ by Corollary 3.27. Hence Lemma 3.42 delivers a thread $t \in Th(\overline{\mathcal{B}})$ with $\overline{s} \equiv^A_{\overline{\mathcal{B}}} t$ i.e. a thread $\overline{t} \in Th(\mathcal{B})$ by Corollary 3.27 s.t. $s \equiv^E_{\mathcal{B}} \overline{t}$ by Corollary 3.35. □

This enables us to draw the central conclusion of this section: If a fair thread bundle is signature compatible falsified then it includes a μ-thread equivalence class. Therefore both the syntactical definitions and the semantical definitions of a "good" (or "bad") thread bundle match.

Lemma 3.44 (Enforcement of μ-threads in falsifying thread bundles)**.** *Let $\psi^* \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a fair thread bundle w.r.t. ψ^* and $\mathcal{T}$ be an LTS. Let moreover $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ be a signature compatible falsifying thread bundle annotation w.r.t. $\mathcal{B}$. Then*

$$\exists t \in Th(\mathcal{B}) : [t]_{\equiv^E_{\mathcal{B}}} \subseteq Th_\mu(\mathcal{B})$$

Proof. Let $\psi^* \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a fair thread bundle w.r.t. ψ^* and $\mathcal{T}$ be an LTS. Let moreover $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ be a signature compatible falsifying thread bundle annotation w.r.t. $\mathcal{B}$.

Assume by contradiction that $\neg\exists t \in Th(\mathcal{B}) : [t]_{\equiv^E_{\mathcal{B}}} \subseteq Th_\mu(\mathcal{B})$ hence by Lemma 3.36 $\exists u \in Th(\mathcal{B}) : [u]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B})$ (*).

Lemma 3.42 comprises a thread $s \in Th(\mathcal{B})$ with $s \equiv^A_{\mathcal{B}} u$ s.t. $(\pi_i)_{i \in roots_{\mathcal{B}}}$ is a signature compatible falsifying thread annotation w.r.t. u. Lemma 3.10 concludes that $s \in Th_\mu(\mathcal{B})$ which is impossible due to (*). □

Again by duality, we derive the dual result regarding ν-threads and thread bundles with signature compatible satisfying annotations.

Lemma 3.45 (Enforcement of ν-threads in satisfying thread bundles)**.** *Let $\psi^* \in CTL^*_{c\mu+}$, $\mathcal{B}$ be a fair thread bundle w.r.t. ψ^* and $\mathcal{T}$ be an LTS. Let moreover $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ be a signature compatible satisfying thread bundle annotation w.r.t. $\mathcal{B}$. Then*

$$\exists t \in Th(\mathcal{B}) : [t]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B})$$

Proof. Consider the dual thread bundle $\overline{\mathcal{B}}$. By Corollary 3.27 $\overline{\mathcal{B}}$ is fair and π is a signature compatible falsifying thread bundle annotation due to Corollary 3.41.

Hence Lemma 3.44 comprises a thread $t \in Th(\overline{\mathcal{B}})$ with $[t]_{\equiv^E_{\overline{\mathcal{B}}}} \subseteq Th_\mu(\overline{\mathcal{B}})$ i.e. there is a thread $\bar{t} \in Th(\mathcal{B})$ with $[\bar{t}]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B})$ by Corollary 3.35 and Corollary 3.6. □

3.5 Induced words

The last section of this chapter addresses the problem of identifying thread bundles including a ν-thread equivalence class. We present an automata-theoretic approach by interpreting thread bundles as infinite words over an alphabet that reflects all information given by the labelling of a thread bundle's node.

For the definition of finite automata and their basic properties consider Appendix B. We will also apply a new construction called *alternating language*. We separate the definition of alternating languages and all subsequent propositions from this section in order to structure the whole construction adequately. Please consider Appendix B.8 for all the details of alternating languages.

Definition 3.46 (Thread bundle induced word)**.** Let $\psi^* \in CTL^*_{c\mu+}$ and let $\mathcal{B} = (dom(\mathcal{B}), \mathcal{Q}, \mathcal{L}, \Phi, \Psi, \mathcal{R})$ be a proper thread bundle w.r.t. ψ^*. Such a thread bundle induces a word $w \in \Sigma^\omega_{\psi^*}$ with

$$\Sigma_{\psi^*} := \{E, A\} \times \mathcal{P}(Sub(\psi^*))^3 \times BundleRules$$

as follows:

$$w : i \mapsto (\mathcal{Q}_i, \mathcal{L}_i, \Phi(i), \Psi(i+1), \mathcal{R}_i)$$

We will not distinguish formally between a proper thread bundle $\mathcal{B}$ and its induced word w.

First, we define a parity automaton that non-deterministically tracks threads and only accepts ν-threads. This can be done by following the connection relation nondeterministically and assigning to all occurring variables their priorities.

Definition 3.47 (Thread tracking automaton)**.** Let $\psi \in CTL^*_{c\mu+}$. The *thread tracking automaton w.r.t.* ψ is a nondeterministic parity automaton $\mathcal{A}_\psi = (Sub(\psi), \Sigma_\psi, \psi, \delta, \Omega)$ with $O(|\psi|)$ states and index $O(|Bound(\psi)|)$ where

$$\delta : (\psi', (\_, \_, \Phi, \Psi, R)) \mapsto \begin{cases} \Psi & \text{if } \psi' \in \Phi \text{ and } R \neq (BX) \\ \{\psi'\} & \text{if } \psi' \notin \Phi \text{ and } R \neq (BQ), (BX) \\ \{\psi''\} & \text{if } \psi' = \bigcirc\psi'' \text{ and } R = (BX) \\ \emptyset & \text{otherwise} \end{cases}$$

$$\Omega : \psi' \mapsto \begin{cases} pri^\psi(X) & \text{if } \psi' \equiv X \in \mathcal{V} \\ 0 & \text{otherwise} \end{cases}$$

It is not hard to see that a run of $\mathcal{A}_\psi$ on a thread bundle corresponds to a thread in the bundle and vice versa. Additionally, the set of accepting runs equals the set of ν-threads.

Corollary 3.48 (Thread tracking automaton tracks threads)**.**
Let $\psi \in CTL^*_{c\mu+}$, let $\mathcal{B}$ be a fair thread bundle w.r.t. ψ and let $\mathcal{A}_\psi$ be the thread tracking NPA. Then the following properties hold

1. $Th(\mathcal{B}) = Runs(\mathcal{A}_\psi, \mathcal{B})$
2. $Th_\nu(\mathcal{B}) = AccRuns(\mathcal{A}_\psi, \mathcal{B})$
3. $Th_\mu(\mathcal{B}) = Runs(\mathcal{A}_\psi, \mathcal{B}) \setminus AccRuns(\mathcal{A}_\psi, \mathcal{B})$

Second, we will construct a deterministic parity automaton accepting all these thread bundles including a ν-thread equivalence class. The alternating

languages of Appendix B.8 are particularly designed to solve this problem: The equivalence relation on threads is generalized to an equivalence relation on runs of a nondeterministic parity automaton.

Corollary 3.48 establishes the correspondence between the runs of $\mathcal{A}_\psi$ and threads in a thread bundle. The *alternating language* w.r.t. $\mathcal{A}_\psi$ accepts all these thread bundles containing a ν-thread equivalence class. For all the technical details refer to the definition of alternating languages (Appendix B.8) and the subsequent propositions.

Corollary 3.49 (Thread bundles and alternating languages)**.** *Let $\psi \in CTL^*_{c\mu+}$ and let $\mathcal{A}_\psi$ be the thread tracking NPA. Then*

$$\ell : \Sigma_\psi \to \{E, A\},\ (\mathcal{Q}, \_, \_, \_, \_) \mapsto \begin{cases} \exists & \text{if } \mathcal{Q} = A \\ \forall & \text{if } \mathcal{Q} = E \end{cases}$$

fulfils the alternating language condition. Moreover the following holds:

$$\mathcal{L}(\mathcal{A}_\psi, \ell) \cap \mathcal{BUF}(\psi) = \{\mathcal{B} \in \mathcal{BUF}(\psi) \mid \exists t \in Th(\mathcal{B}) :\ [t]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B})\}$$

Recall that $\mathcal{BUF}(\psi)$ is defined as the set of all fair thread bundles w.r.t. ψ.

By Corollary 3.49 and Theorem B.11 we conclude that there is a deterministic parity automaton accepting the thread bundle alternating language.

Corollary 3.50 (Thread bundle automaton)**.** *Let $\psi \in CTL^*_{c\mu+}$. There is a DPA $\mathcal{A}^{\mathcal{BU}}_\psi$ with $2^{O(n \cdot m \cdot \log(n \cdot m))}$ states and index $O(n \cdot m)$ s.t.*

$$\mathcal{L}(\mathcal{A}^{\mathcal{BU}}_\psi) \cap \mathcal{BUF}(\psi) = \{\mathcal{B} \in \mathcal{BUF}(\psi) \mid t \in Th(\mathcal{B}) :\ [t]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B})\}$$

where $n = |\psi|$ and $m = |Bound(\psi)|$.

4. PROOF SYSTEM

The Proof System is a tableaux-style system for the validity of the restricted fragment of CTL^*_μ. A proof tableaux is a possibly infinite finitely-branching tree whose nodes are labelled with disjunctions over formula blocks and literals. A valid proof tree for a fixed formula is a tableaux whose root is labelled with the initial formula, built according to the proof rules with all finite branches ending in axioms and all infinite branches fulfilling a global condition induced by the preceding chapter.

There will be either a proof or an unproof for a given formula; an unproof can be seen as a failing proof attempt and corresponds canonically to a falsification witness for the formula in question. A failing proof attempt can be used to extract a counter model in this sense. By reducing the problem of finding a proof to an associated proof parity game, we additionally receive a decision procedure for determining whether a formula is valid or not. If it is not, the respective counter strategy of the game can be used to construct a canonical counter model.

There are basically two reasons why we are not able to extend this proof system in such a way that it could also be used with non-restricted formulas. First, we need formulas to be guarded; this is no problem with restricted formulas due to the fact that in this case state-guardedness and guardedness are the same thing as we have seen. We require the guardedness in order to guarantee a certain fairness to all blocks that are listed in the proof nodes.

Second, we need all thread bundles contained in the proof branches to be finally preserving. For the main results of this chapter, we will be extracting proof branches with signature compatible annotations for all contained thread bundle suffixes being immediately preserving. The problem with never preserving thread bundles is that two blocks of different thread bundle prefixes

on a proof node merging together by a rule application need to get assigned the same annotations which in general is not possible. On the other hand we need to merge equally labelled blocks since otherwise we would get arbitrarily large node descriptions which would result in infinite parity games being not very useful to derive a decision procedure.

4.1 Definition

The labellings of the proof system tableaux are disjunctions of formula blocks and literals. Such a disjunction is called *sequent* which basically is a set of literals and blocks; thus there are only finitely many sequents with respect to a fixed formula. Recall that a block is simply a quantification labelled set of formulas.

Definition 4.1 (Sequent)**.** A *sequent* $\mathcal{R}$ is an expression of the form

$$A(\Delta_1); A(\Delta_2); ...; A(\Delta_a); E(\Gamma_1); E(\Gamma_2)...; E(\Gamma_e); l_1; ...; l_k$$

where all $A(\Delta_i)$ are universal blocks, all $E(\Gamma_i)$ are existential blocks and all l_i are literals.

The set of all blocks in a sequent $\mathcal{R}$ is denoted by $Bl(\mathcal{R})$. The set of all A-blocks (E-blocks) is denoted by $ABl(\mathcal{R})$ ($EBl(\mathcal{R})$). The set of all literals in a sequent $\mathcal{R}$, $\mathcal{R} \cap \mathcal{P}^*$, is denoted by $Lits(\mathcal{R})$.

The set of sequents w.r.t. to a formula ψ, $Seq(\psi)$, consists of all sequents $\mathcal{R}$ s.t. each literal in $\mathcal{R}$ and each formula occurring in a block of $\mathcal{R}$ is a subformula of ψ.

A *proof rule* is a typical tableaux-style rule having sequents as node labellings. As with the bundle rules, we specify principal and principal successive formulas with respect to the affected blocks in the sequent.

Definition 4.2 (Proof rule)**.** A *proof rule* (R) is a formal construct w.r.t. to a fixed formula φ written as follows

$$(R): \frac{\vdash \mathcal{R}_1 \qquad \vdash \mathcal{R}_2 \qquad ... \qquad \vdash \mathcal{R}_n}{\vdash \mathcal{R}}[Q]$$

where all $\mathcal{R}_1, ...\mathcal{R}_n, \mathcal{R}$ are sequent schemes. All sequents atop the line are called *premisses* of (R) while the single sequent below the line is called *conclusion* of (R). A rule (R) is *branching* iff $n > 1$.

A branching rule moreover specifies a *quantification* $Q \equiv \exists, \forall$ where $\exists$-labelled rules are called *existentially branching* and $\forall$-labelled rules are called *universally branching*.

A block of the conclusion can be annotated with $\lfloor$- and $\rfloor$-brackets; such a block is called *principal*. A formula within a principal block can be likewise annotated with $\lfloor$- and $\rfloor$-brackets; such a formula is called *principal* as well.

A block of a premiss can be annotated with $\lceil$- and $\rceil$-brackets; such a block is called *principal successive*. A formula within a principal successive block can be likewise annotated with $\lceil$- and $\rceil$-brackets; such a formula is called *principal successive* as well.

If I is an instance of a sequent scheme $\mathcal{R}$ we write $I \triangleright \mathcal{R}$.

A universally branching / existentially branching rule (R) is *regularly / inversely instantiated* through an instance

$$\frac{\vdash I_1 \qquad \vdash I_2 \qquad ... \qquad \vdash I_n}{\vdash I}$$

iff $I \triangleright \mathcal{R}$ and $I_i \triangleright \mathcal{R}_i$ for all $i \in \mathbb{N}$.

An existentially branching / universally branching (or not at all branching) rule (R) is *regularly / inversely instantiated* through an instance

$$\frac{\vdash I_i}{\vdash I}$$

iff $I \triangleright \mathcal{R}$ and $I_i \triangleright \mathcal{R}_i$ for some $i \in \mathbb{N}$.

A rule (R) is *branchingly instantiated* iff (R) is universally branching and regularly instantiated or existentially branching and inversely instantiated.

The *proof rules* of this tableaux system correspond to the *bundle rules* by simply unreeling formulas and unfolding fixed points in the blocks. Whenever

a conjunction needs to be unreeled at the outermost region - that is either a quantification formula in an existential block or a conjunction in a universal block - the respective proof is universally branching in order to be locally sound and complete.

The *modal rules* of this tableaux require *each* block to only contain formulas starting with the next-operator. If there is more than one universal block one needs to *choose* one of them to proceed with in the successive sequent.

Definition 4.3 (Proof rules)**.** W.r.t. to a fixed formula φ, the proof rules are as follows:

$$(PE\vee): \frac{\vdash \lceil E(\lceil \psi_1 \rceil, \Delta)\rceil; \lceil E(\lceil \psi_2 \rceil, \Delta)\rceil; \mathcal{R}}{\vdash \lfloor E(\lfloor \psi_1 \vee \psi_2 \rfloor, \Delta)\rfloor; \mathcal{R}} \qquad (PA\vee): \frac{\vdash \lceil A(\lceil \psi_1 \rceil; \lceil \psi_2 \rceil; \Gamma)\rceil; \mathcal{R}}{\vdash \lfloor A(\lfloor \psi_1 \vee \psi_2 \rfloor; \Gamma)\rfloor; \mathcal{R}}$$

$$(PA\wedge): \frac{\vdash \lceil A(\lceil \psi_1 \rceil; \Gamma)\rceil; \mathcal{R} \qquad \vdash \lceil A(\lceil \psi_2 \rceil; \Gamma)\rceil; \mathcal{R}}{\vdash \lfloor A(\lfloor \psi_1 \wedge \psi_2 \rfloor; \Gamma)\rfloor; \mathcal{R}}[\forall] \qquad (PE\wedge): \frac{\vdash \lceil E(\lceil \psi_1 \rceil, \lceil \psi_2 \rceil, \Delta)\rceil; \mathcal{R}}{\vdash \lfloor E(\lfloor \psi_1 \wedge \psi_2 \rfloor, \Delta)\rfloor; \mathcal{R}}$$

$$(PEQ): \frac{\vdash \lceil Q(\lceil \psi \rceil)\rceil; \mathcal{R} \qquad \vdash \lceil E(\Delta)\rceil; \mathcal{R}}{\vdash \lfloor E(\lfloor Q\psi \rfloor, \Delta)\rfloor; \mathcal{R}}[\forall] \qquad (PAQ): \frac{\vdash \lceil A(\Gamma)\rceil; \lceil Q(\lceil \psi \rceil)\rceil; \mathcal{R}}{\vdash \lfloor A(\lfloor Q\psi \rfloor; \Gamma)\rfloor; \mathcal{R}}$$

$$(PQV): \frac{\vdash \lceil Q(\lceil body_\varphi(X)\rceil, \Lambda)\rceil; \mathcal{R}}{\vdash \lfloor E(\lfloor X \rfloor, \Lambda)\rfloor; \mathcal{R}} \qquad (PQ\sigma): \frac{\vdash \lceil Q(\lceil X \rceil, \Lambda\rceil; \mathcal{R}}{\vdash \lfloor Q(\lfloor \sigma X.\psi \rfloor, \Lambda\rfloor); \mathcal{R}}$$

$$(PEl): \frac{\vdash l; \mathcal{R} \qquad \vdash \lceil E(\Delta)\rceil; \mathcal{R}}{\vdash \lfloor E(\lfloor l \rfloor, \Delta)\rfloor; \mathcal{R}}[\forall] \qquad (PAl): \frac{\vdash \lceil A(\Gamma)\rceil; l; \mathcal{R}}{\vdash \lfloor A(\lfloor l \rfloor; \Gamma)\rfloor; \mathcal{R}}$$

$$(PAff): \frac{\vdash \mathcal{R}}{\vdash \lfloor A(\emptyset)\rfloor; \mathcal{R}}$$

where $\mathcal{R}$ a sequent, $\psi, \psi_1, \psi_2 \in Sub(\varphi)$, $\Gamma, \Delta, \Lambda \subseteq Sub(\varphi)$, $Q \in \{E, A\}$ and $l \in \mathcal{P}^*$. The modal rules finally are as follows:

$$(PX_0): \frac{\vdash \lceil E\lceil \Delta_1 \rceil\rceil; ...; \lceil E\lceil \Delta_n \rceil\rceil}{\vdash \lfloor E\lfloor \bigcirc \Delta_1 \rfloor\rfloor; ...; \lfloor E\lfloor \bigcirc \Delta_n \rfloor\rfloor; L}$$

$$(PX_1): \frac{\vdash \lceil A\lceil \Gamma_1 \rceil\rceil; \lceil E\lceil \Delta_1 \rceil\rceil; ...; \lceil E\lceil \Delta_n \rceil\rceil \quad ... \quad \vdash \lceil A\lceil \Gamma_k \rceil\rceil; \lceil E\lceil \Delta_1 \rceil\rceil; ...; \lceil E\lceil \Delta_n \rceil\rceil}{\vdash \lfloor A\lfloor \bigcirc \Gamma_1 \rfloor\rfloor; ...; \lfloor A\lfloor \bigcirc \Gamma_k \rfloor\rfloor; \lfloor E\lfloor \bigcirc \Delta_1 \rfloor\rfloor; ...; \lfloor E\lfloor \bigcirc \Delta_n \rfloor\rfloor; L}[\exists]$$

where $\Delta_1, ..., \Delta_n \subseteq Sub(\varphi)$, $\Gamma_1, ..., \Gamma_k \subseteq Sub(\varphi)$, $L \subseteq \mathcal{P}^*$ whereas for all $q \in L \cap \mathcal{P}$ holds that $\neg q \notin L$.

The *axioms* of the proof system comprehend all these sequences that can be directly verified to be correct: If the tertium-non-datur w.r.t. literals or the totality requirement is contained in the sequent then the whole sequent is valid.

Definition 4.4 (Proof axioms)**.** With respect to a fixed formula φ the proof axioms are as follows:

$$(PP): \frac{}{\vdash q; \neg q; \mathcal{R}} \qquad (PEtt): \frac{}{\vdash E(\emptyset); \mathcal{R}}$$

where $\mathcal{R}$ is a sequent and $q \in \mathcal{P} \cap Sub(\varphi)$

The *counteraxioms* contain all these sequences that can be directly verified to be incorrect: A sequent without blocks or the tertium-non-datur are clearly not valid.

Definition 4.5 (Counteraxioms)**.** With respect to a fixed formula φ the counteraxioms are as follows:

$$(PFF): \frac{}{\vdash q_1; ...; q_n; \neg p_1; ...; \neg p_m}$$

where $q_1, ..., q_n, p_1, ..., p_m \in \mathcal{P}$ and $q_i \neq p_j$ for all i, j.

A *pre-proof* for a guarded formula ψ is basically a tableaux-style tree built according to the proof rules with its root being labelled with $E\psi$. Additionally, all finite branches need to end in axioms. Therefore a pre-proof is a *locally sound and complete* tableaux.

Please refer to Appendix A for the abstract definition of trees.

Definition 4.6 (Pre-Proof)**.** Let $\varphi \in sCTL^*_{c\mu+}$ be guarded. A *pre-proof* for φ is a possibly infinite finitely-branching tree P where

- each inner node is labelled with a sequent and a proof rule name
- the root is labelled with $\vdash E(\varphi)$ and a proof rule name
- each node along with its direct subnodes is a regularly instantiated proof rule
- all finite branches end in proof axioms

More formally, P is tuple $(dom(P), \mathcal{S}, \mathcal{R}, \mathcal{U}, \Phi, \mathcal{V}, \Psi)$ where

- $dom(P)$ is a tree; for two nodes $t, s \in dom(P)$ the relation $t \rightarrow s$ holds iff s is an immediate successor of t.

- $\mathcal{S} : dom(P) \to Seq(\varphi)$ maps every node in $dom(P)$ to a sequent s.t. $\mathcal{S}(\varepsilon) \equiv E(\varphi)$
- $\mathcal{R}$ maps every node in $dom(P)$ having a successor to a rule identifier and every leaf to an axiom identifier
- $\mathcal{U}$ maps every node t in $dom(P)$ to the set of principal blocks in $Bl(\mathcal{S}(t))$
- Φ maps every node t in $dom(P)$ and every principal block B in $\mathcal{U}(t)$ to the set of principal formulas in B
- $\mathcal{V}$ maps every node $t \neq \varepsilon$ in $dom(P)$ to the set of principal successive blocks in $Bl(\mathcal{S}(t))$
- Ψ maps every node $t \neq \varepsilon$ in $dom(P)$ and every principal successive block B in $\mathcal{V}(t)$ to the set of principal successive formulas in B

s.t. for all nodes $t \in dom(P)$ with successors $s_1, ..., s_n$

$$(\mathcal{R}(t)) : \frac{\vdash \mathcal{S}(s_1)\lceil\Psi(s_1)\rceil \qquad \vdash \mathcal{S}(s_2)\lceil\Psi(s_2)\rceil \qquad ... \qquad \vdash \mathcal{S}(s_n)\lceil\Psi(s_n)\rceil}{\vdash \mathcal{S}(t)\lfloor\Phi(t)\rfloor}$$

is a valid regular proof rule (or proof axiom) instance.

A *pre-unproof* is basically the dual version of a pre-proof thus we can do without a formal definition.

Definition 4.7 (Pre-Unproof)**.** A *pre-unproof* is defined likewise whereas each rule is instantiated inversely and each finite branch ends in a counteraxiom.

The *block connection* connects two successive blocks if the second block is triggered by the first block in the sense that if the first block is not principal then both blocks need to be equal. Otherwise, if the first block is actually principal, then the second block needs to be principal successive and somehow "related" to its predecessor.

Definition 4.8 (Block connection)**.** Let $\varphi \in sCTL^*_{c\mu+}$ be guarded, P be a pre-(un)proof for φ and $t, s \in dom(P)$ s.t. $t \to s$. The *block connection relation* $BCon_P(t,s) \subseteq Bl(\mathcal{S}(t)) \times Bl(\mathcal{S}(s))$ is defined as follows:

$$BCon_P(t,s) := BCon_P(\mathcal{S}(t), \mathcal{U}(t), \mathcal{V}(s), \mathcal{R}(t))$$

where

$$\begin{gathered}(B_1, B_2) \in BCon_P(S, U, V, R) \\ \text{iff} \\ B_1 \in S \text{ and} \\ \text{either } B_1 = B_2 \text{ and } B_1 \notin U \\ \text{or } B_1 \in U \text{ and } B_2 \in V \text{ and } R \neq (PX_*) \\ \text{or } (B_1)_1 = (B_2)_1 \text{ and } (B_1)_2 \equiv \bigcirc (B_2)_2 \text{ and } R = (PX_*)\end{gathered}$$

Such a pair $(B_1, B_2) \in BCon_P(t,s)$ is called *blockspawning* iff $\mathcal{V}(s, B_2)$ is not empty and $\mathcal{R}(t) \in \{(PEQ), (PAQ)\}$. Otherwise (B_1, B_2) is called *blockpreserving*.

By definition, each block of a sequent is connected to at least one other block in the preceding block.

Corollary 4.9 (Block connection is not empty)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be guarded, P be a pre-(un)proof for φ, $t, s \in dom(P)$ s.t. $t \to s$ and $B \in Bl(\mathcal{S}(s))$. There is a block $B' \in Bl(\mathcal{S}(t))$ with $(B', B) \in BCon_P(t,s)$.*

An *infinite branch* in a tableaux is formalised as a map specifying an infinite path in the tableaux tree.

Definition 4.10 (Infinite branch)**.** An *infinite branch* Ξ in a pre-(un)proof P for a formula φ is a map $\Xi : \mathbb{N} \to dom(P)$ with $\Xi(0) = \varepsilon$ and $\Xi(i) \to \Xi(i+1)$ for all $i \in \mathbb{N}$. An infinite branch Ξ is *progressive* iff $\mathcal{R}(\Xi(i)) = (PX_*)$ for infinitely many $i \in \mathbb{N}$.

Each infinite branch contains a bunch of thread bundles, namely all these block sequences induced by the block connection relation. It is not hard to see that each sequence which is obtained in this manner is actually a thread bundle.

Lemma 4.11 (Thread bundles in infinite branches)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be guarded, $P = (dom(P), \mathcal{S}, \mathcal{R}', \mathcal{U}, \Phi', \mathcal{V}, \Psi')$ be a pre-(un)proof for φ and Ξ be an infinite branch in P.*

An infinite sequence $B_0, B_1, ...$ with $(B_i, B_{i+1}) \in BCon_P(\Xi(i), \Xi(i+1))$ for all $i \in \mathbb{N}$ induces a thread bundle $\mathcal{B} = (\mathcal{Q}, \mathcal{L}, \Phi, \Psi, \mathcal{R})$ as follows

- $\mathcal{Q}_i(\mathcal{L}_i) := B_i$ *for all* $i \in \mathbb{N}$
- $\Phi : i \mapsto \Phi'(\Xi(i), B_i)$ *for all* $i \in \mathbb{N}$
- $\Psi : i \mapsto \Psi'(\Xi(i), B_i)$ *for all* $i \in \mathbb{N}$
- $\mathcal{R} : i \mapsto prule(\mathcal{R}'(\Xi(i)), B_i, \Psi(i+1))$ *for all* $i \in \mathbb{N}$

$$prule : (R, B, D) \mapsto \begin{cases} (BS) & \text{if } B \notin \mathcal{U}(t) \\ (BW) & \text{if } B \in \mathcal{U}(t),\ R \equiv (PEl), (PAl) \\ (BW) & \text{if } B \in \mathcal{U}(t),\ R \equiv (PEQ), (PAQ),\ D = \emptyset \\ (BQ) & \text{if } B \in \mathcal{U}(t),\ R \equiv (PEQ), (PAQ),\ D \neq \emptyset \\ (B\mathcal{X}) & \text{otherwise with } R \equiv (P\mathcal{X}) \end{cases}$$

The set of all thread bundles in Ξ w.r.t. P is denoted by $ThB(P, \Xi)$. Moreover define

- $ThB_\nu(P, \Xi) := \{B \in ThB(P, \Xi) \mid \exists t \in Th(B) :\ [t]_{\equiv^A_B} \subseteq Th_\nu(B)\}$
- $ThB_\mu(P, \Xi) := \{B \in ThB(P, \Xi) \mid \exists t \in Th(B) :\ [t]_{\equiv^E_B} \subseteq Th_\mu(B)\}$

We require each thread bundle to be lively since otherwise we would not be able to apply all semantical results of the preceding chapter. The idea of the proof runs as follows: We firstly show that each infinite branch needs to contain at least one thread bundle that is lively. By the preceding chapter we derive the progressivity of the thread bundle, directly implying the progressivity of the infinite branch. Thus, all thread bundles in the branch need to be lively.

For the extraction of a lively thread bundle we use a technical tree Lemma (Appendix A.4) based on Königs Lemma (Appendix A.3). Please consider the appendix for all the details.

Lemma 4.12 (Infinite branches are progressive)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be guarded, P be a pre-(un)proof for φ, Ξ be an infinite branch in P. Then Ξ is progressive.*

Proof. Let $\varphi \in sCTL^*_{c\mu+}$ be guarded, P be a pre-(un)proof for φ and Ξ be an infinite branch in P.
Define $S := \{(i, B) \mid B \in Bl(\Xi(i))\}$ with

$$(i, B) \to (i+1, B') \; : \iff \; (B, B') \in BCon_P(\Xi(i), \Xi(i+1))$$

and consider that the induced tree (see Definition A.2) $T := T(S, (0, \Xi(0)), \to)$ is infinite due to Corollary 4.9 and obviously finitely branching.

Moreover define $A_i := \bigcup_{B \in \mathcal{U}(\Xi(i))} Lift_T(i, B)$ for all $i \in \mathbb{N}$ and consider that $\emptyset \neq A_i \subseteq Lev_T(i)$ for all $i \in \mathbb{N}$; verify by case distinction on $R(\Xi(i))$ that $|Succ_t(p)| > 1$ implies $p \in A_{|p|}$ for all $p \in T$.

Hence Lemma A.4 comprises a path $t \in Path^\infty_T$ s.t. $t_i \in A_i$ for infinitely many $i \in \mathbb{N}$. Set $B_i := Head_T(t_i)$ for all $i \in \mathbb{N}$ and consider that $(B_i, B_{i+1}) \in BCon_P(\Xi(i), \Xi(i+1))$ for all $i \in \mathbb{N}$ and $B_i \in \mathcal{U}(\Xi(i))$ for infinitely many i i.e. the induced thread bundle $\mathcal{C} \in ThB(P, \Xi)$ is lively. Lemma 3.19 thus claims that $\mathcal{C}$ is even progressive, hence $R(\Xi(i)) = (PX_*)$ for infinitely many $i \in \mathbb{N}$. □

It is crucial that the lively thread bundle we extract is progressive since otherwise it is possible that there is only one lively thread bundle in the infinite branch with all other thread bundles being stalled. This is one of the reasons why we require the guardedness (and not the state-guardedness) of the formula in question.

Corollary 4.13 (Thread bundles are lively)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be guarded, P be a pre-(un)proof for φ, Ξ be an infinite branch in P and $\mathcal{B} \in ThB(P, \Xi)$ be a thread bundle in Ξ w.r.t. P. Then $\mathcal{B}$ is lively.*

Proof. Let $\varphi \in sCTL^*_{c\mu+}$ be guarded, P be a pre-(un)proof for φ and Ξ be an infinite branch in P. By Lemma 4.12, Ξ is progressive. Thus each $B \in ThB(P, \Xi)$ is progressive and hence lively. □

By Corollary 4.13 and Corollary 3.30 we conclude that all occurring thread bundles in a pre-(un)proof tableaux are fair.

Corollary 4.14 (Thread bundles are fair)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be guarded, P be a pre-(un)proof for φ, Ξ be an infinite branch in P and $\mathcal{B} \in ThB(P, \Xi)$ be a thread bundle in Ξ w.r.t. P. Then $\mathcal{B}$ is fair.*

A *proof* extends the definition of a pre-proof by *global conditions* that need to be fulfilled for being a witness for the validity of the formula in question. All finite branches obviously need to end in an axiom. Since a sequent is a big disjunction over blocks, an infinite branch has to contain at least one valid thread bundle in order to be valid itself.

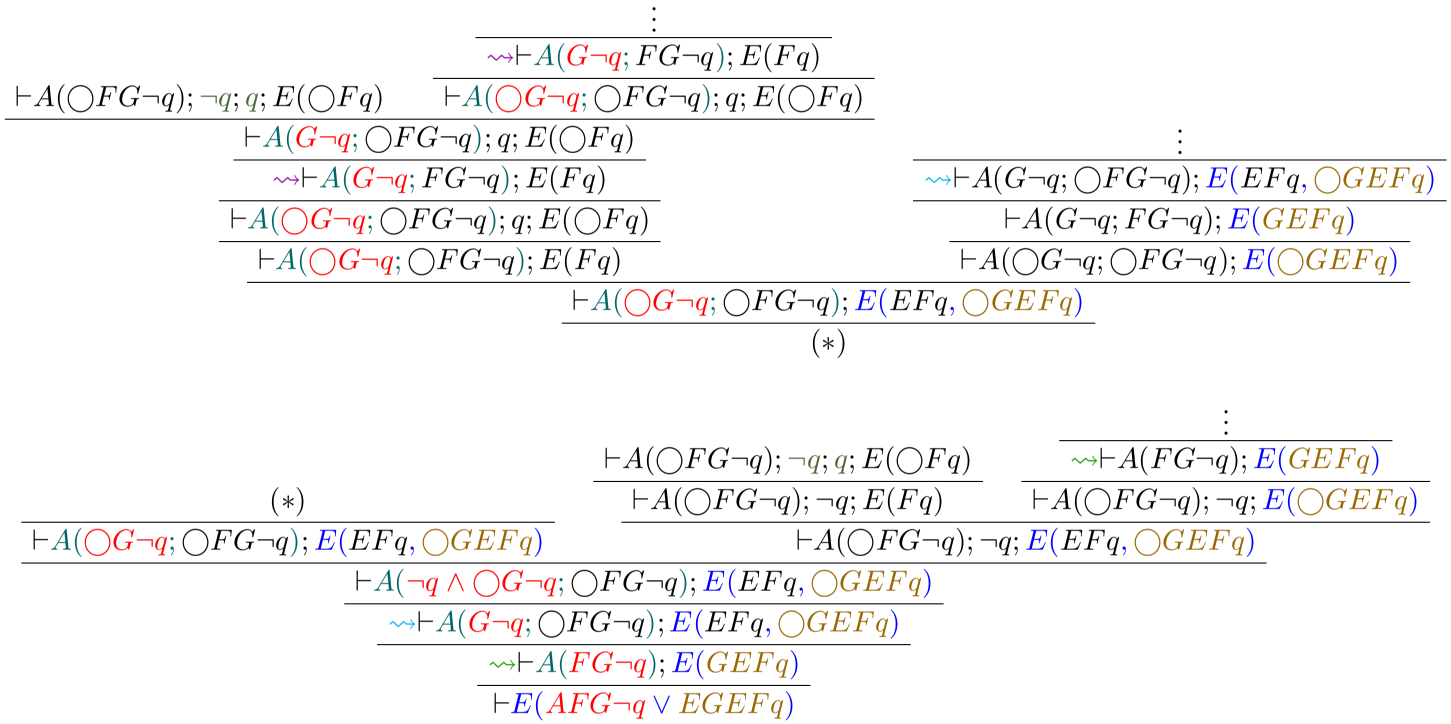
Definition 4.15 (Proof)**.** A *proof* for a guarded $\varphi \in sCTL^*_{c\mu+}$ is a pre-proof P s.t. all finite branches end in proof axioms and for all infinite branches Ξ there exists a thread bundle $\mathcal{B} \in ThB(P, \Xi)$ and a thread $t \in Th(\mathcal{B})$ s.t. $[t]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B})$.
We also write $\vdash \varphi$ to indicate that there is a proof for φ.

Example 4.16. Consider the following formula

$$EGFq \rightarrow EGEFq$$

stating that if there is a path in an LTS with infinitely many states satisfying q then there is also a path (in a semantical proof one would use the path of the assumption here) leading to (possibly other) paths each of them finally satisfying q (in a semantical proof one would not choose real other paths but the path itself).

A proof tree can be built as follows. For better readability we instantiate customized fixed point unfolding rules for G and F and do without some trivial tableaux steps.



where $EGFq \rightarrow EGEFq \equiv AFG\neg q \vee EGEFq$.

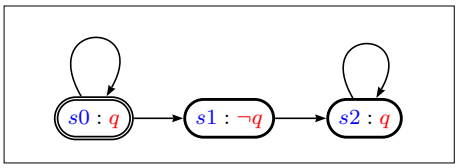
An *unproof* is similarly defined: Instead of ending in axioms, an unproof ends in *counteraxioms* and instead of including a thread bundle with a ν-equivalence class, each bundle in an infinite branch of an unproof has to contain a μ-equivalence class.

Definition 4.17 (Unproof)**.** An *unproof* for a guarded $\varphi \in sCTL^*_{c\mu+}$ is a pre-unproof P s.t. all finite branches end in proof counteraxioms and for all infinite branches Ξ and all thread bundles $\mathcal{B} \in ThB(P, \Xi)$ there exists a thread $t \in Th(\mathcal{B})$ s.t. $[t]_{\equiv^E_{\mathcal{B}}} \subseteq Th_\mu(\mathcal{B})$.
We also write $\nvdash \varphi$ to indicate that there is an unproof for φ.

Example 4.18. We give an unproof tableaux for the formula

$$AFGq \rightarrow AFAGq$$

The formula, stating that if on each path finally holds q on the rest of the path then on each path finally holds q for the whole reachable subgraph, is not valid: A model satisfying the assumption but not the conclusion is the LTS of Example 2.2:



An unproof for the formula runs as follows:

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{\vdots}{\rightsquigarrow \nvdash E(F\neg q, \bigcirc GF\neg q)}
}{\nvdash E(F\neg q, GF\neg q)}
}{\nvdash \neg q;\, E(\bigcirc F\neg q, \bigcirc GF\neg q)}\,(\overline{X_1})
}{\nvdash E(\neg q, \bigcirc GF\neg q);\, E(\bigcirc F\neg q, \bigcirc GF\neg q)}
}{\nvdash E(\neg q \vee \bigcirc F\neg q, \bigcirc GF\neg q)}
}{\rightsquigarrow \nvdash E(F\neg q, \bigcirc GF\neg q)}
}{\nvdash E(GF\neg q);\, E(F\neg q, GF\neg q)}
}{\nvdash q;\, E(\bigcirc GF\neg q);\, E(\bigcirc F\neg q, \bigcirc GF\neg q)}\,(\overline{X_1})
}{\nvdash E(\bigcirc GF\neg q);\, E(\bigcirc F\neg q, \bigcirc GF\neg q);\, A(q \wedge \bigcirc Gq)}
}{\nvdash E(\neg q, \bigcirc GF\neg q);\, E(\bigcirc F\neg q, \bigcirc GF\neg q);\, A(q \wedge \bigcirc Gq)}
}{\nvdash E(F\neg q, \bigcirc GF\neg q);\, A(q \wedge \bigcirc Gq)}
\qquad
\dfrac{
\dfrac{\vdots}{\rightsquigarrow \nvdash E(F\neg q, \bigcirc GF\neg q);\, A(AGq; \bigcirc FAGq)}
}{\nvdash E(F\neg q, GF\neg q);\, A(FAGq)}
}{\nvdash E(F\neg q, GF\neg q);\, A(Gq)}\,(\overline{X_1})
}{\nvdash \neg q;\, E(\bigcirc F\neg q, \bigcirc GF\neg q);\, A(\bigcirc Gq);\, A(\bigcirc FAGq)}
}{\nvdash \neg q;\, E(\bigcirc F\neg q, \bigcirc GF\neg q);\, A(q \wedge \bigcirc Gq);\, A(\bigcirc FAGq)}
}{\nvdash E(\neg q, \bigcirc GF\neg q);\, E(\bigcirc F\neg q, \bigcirc GF\neg q);\, A(Gq);\, A(\bigcirc FAGq)}
}{\rightsquigarrow \nvdash E(F\neg q, \bigcirc GF\neg q);\, A(AGq; \bigcirc FAGq)}
}{\nvdash E(GF\neg q);\, A(FAGq)}
}{\nvdash E(EGF\neg q \vee AFAGq)}
$$

where $AFGq \rightarrow AFAGq \equiv EGF\neg \vee AFAGq$.

4.2 Soundness

In order to prove our proof system sound, we need to show that if an LTS falsifies a formula also having a proof tree, one can extract an infinite branch with each thread bundle having a signature compatible falsifying annotation. We will thereafter use that result to show that such a scenario is impossible which brings us to the conclusion that our system is sound.

To be more specific, we will show that each thread bundle in the extracted infinite branch has a thread bundle suffix with a signature compatible falsifying annotation. These will be totally block-preserving thread bundle suffixes which always exist due to the restrictedness of the formula in question. This is also the reason why we demand formulas to be restricted: If they are not, there are thread bundles in general with no block-preserving suffix.

Basically, we construct a sequence of states corresponding to the sequence of nodes in the infinite branch that is extracted. Two successive states are connected by the transition relation if the corresponding rule application is a modal rule instance, otherwise they remain equal. The states obtained in this manner suffice to falsify all existential blocks. For each universal block we store a falsifying path compatible with the extracted states. Since two universal blocks accidentally can merge together, we need to make sure that both will be assigned the same path even before. This can be done be always choosing the *least* path which falsifies the respective block. Whenever the principal formula is a falsified state formula, we choose to proceed with the branch following it, to finally reach block-preserving suffixes.

The problem with extending this construction to non-restricted formulas is as follows: Take for instance a never block-preserving thread bundle with infinitely many existential segments. We need to construct a signature compatible annotation for the whole thread bundle, thus particularly for each existential segment. Now imagine that we are at a position at which the rule application is a path quantification rule instance affecting this very thread bundle. We need to decide on whether we proceed with the block-preserving or with the block-spawning part of the bundle. This depends on whether the candidate set contains the principal state formula or not in the sense that

if there is a path completion so that the respective candidate set contains the principal state formula, then we are choosing the block-spawning branch. Unfortunately, if there are two thread bundle prefixes merging together both candidate sets do not necessarily need to match, thus we are in general not able to adequately choose a direction being sound for both thread bundles.

Lemma 4.19 (Extraction of a falsifying branch)**.** *Let $\mathcal{T}$ be a transition system, $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded, P be a proof for φ and $\mathcal{T} \not\models \varphi$. There is an infinite branch Ξ in P s.t. for each $\mathcal{B} \in ThB(P, \Xi)$ there is a thread bundle suffix $\mathcal{B}'$ w.r.t. $\mathcal{B}$ as well as a signature compatible falsifying thread bundle annotation $\pi = (\pi_i)_{i \in roots_{\mathcal{B}'}}$.*

Proof. Let $\mathcal{T} = (S, \theta, \rightarrow, \lambda)$ be a transition system, $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded, $P = (dom(P), \mathcal{S}, \mathcal{R}, \mathcal{U}, \Phi, \mathcal{V}, \Psi)$ be a proof for φ and $\mathcal{T} \not\models \varphi$. We will inductively construct

- a sequence of nodes $z_0, z_1, z_2, ... \in dom(P)$ s.t. $z_0 = \varepsilon$ and z_{i+1} is a successor of z_i for all $i \in \mathbb{N}$
- a sequence of states $s_0, s_1, s_2, ... \in S$ with $s_0 = \theta$
- a sequence of path maps $\chi_0, \chi_1, \chi_2, ...$ with $\chi_i : ABl(\mathcal{S}(z_i)) \rightarrow \Pi(\mathcal{T})$ for all $i \in \mathbb{N}$

s.t. the following properties hold:

1. State construction
   (a) $s_i = s_{i+1}$ for all $i \in \mathbb{N}$ with $\mathcal{R}(z_i) \neq (PX_*)$
   (b) $s_i \rightarrow s_{i+1}$ for all $i \in \mathbb{N}$ with $\mathcal{R}(z_i) = (PX_*)$
2. Path construction
   (a) $\chi_i(B)(0) = s_i$ for all $i \in \mathbb{N}$ and all $B \in ABl(\mathcal{S}(z_i))$
   (b) $\chi_i(B) = \chi_{i+1}(B')$ for all $i \in \mathbb{N}$ with $\mathcal{R}(z_i) \neq (PX_*)$, $B \in ABl(\mathcal{S}(z_i))$, $B' \in ABl(\mathcal{S}(z_{i+1}))$ with $(B, B') \in BCon_P(z_i, z_{i+1})$

(c) $\chi_i(B)[1] = \chi_{i+1}(B')$ for all $i \in \mathbb{N}$ with $\mathcal{R}(z_i) = (PX_*)$, $B \in ABl(\mathcal{S}(z_i))$, $B' \in ABl(\mathcal{S}(z_{i+1}))$ with $(B, B') \in BCon_P(z_i, z_{i+1})$

3. Falsification conditions

(a) $s_i \not\models \bigvee Lits(\mathcal{S}(z_i))$

(b) $s_i \not\models\!\updownarrow^{\varphi} E(\bigwedge \Lambda)$ for all $i \in \mathbb{N}$ and all $(E, \Lambda) \in Bl(\mathcal{S}(z_i))$

(c) $\chi_i(\Lambda) \not\models_{cmin}\!\updownarrow^{\varphi} \bigvee \Lambda$ for all $i \in \mathbb{N}$ and all $(A, \Lambda) \in Bl(\mathcal{S}(z_i))$

4. Branching policies

(a) For all $i \in \mathbb{N}$ with $\mathcal{R}(z_i) = (PA\wedge)$, $B \in \mathcal{U}(z_i)$, $B' \in \mathcal{V}(z_{i+1})$ with principal formula $\psi_1 \wedge \psi_2$ and principal successive ψ_k holds: If $\zeta \in Sig_\nu(\varphi)$ is the least signature s.t. $\chi_i(B) \not\models\!\downarrow^{\varphi}_{\zeta} \psi_1 \wedge \psi_2$ then $\chi_{i+1}(B') \not\models\!\downarrow^{\varphi}_{\zeta} \psi_k$

(b) For all $i \in \mathbb{N}$ with $\mathcal{R}(z_i) = (PEE), (PEA), (PEl)$, $B \in \mathcal{U}(z_i)$ with principal formula φ' holds: If $s_i \not\models\!\updownarrow^{\varphi} \varphi'$ then there is a $B' \in \mathcal{V}(z_{i+1})$ with $(B, B') \in BCon_P(z_i, z_{i+1})$ blockspawning

Consider that we are allowed to use the model-relation without environments due to Lemma 2.53.

Set $z_0 := \varepsilon$, $s_0 := \theta$ and $\xi_0 := \emptyset$. For better readability we abbreviate $R_i \equiv \mathcal{R}(z_i)$; for $i \rightsquigarrow i+1$ we apply a case distinction w.r.t. R_i; note that R_i can neither be a counteraxiom by definition of proofs nor a proof axiom by construction. Moreover let z_{i+1} automatically be defined as the only successor of z_i iff R_i is not branching; similarly let $s_{i+1} := s_i$ for all $i \in \mathbb{N}$ with $R_i \neq (PX_*)$.

Whenever $R_i \neq (PX_*), (PQA)$ we define ξ_{i+1} as follows: Set $\xi_{i+1} : B_2 \mapsto \xi(B_1)$ for all $((A, B_1), (A, B_2)) \in BCon_P(z_i, z_{i+1})$. The welldefinition of ξ_{i+1} can be easily seen since the modelling relation is minimal and all paths are preserved by rules $R_i \neq (PX_*), (PQA)$.

Now the case distinction w.r.t. R_i:

- *Cases* $R_i \equiv (P\sigma), (PV)$: By Lemma 2.54
- *Cases* $R_i \equiv (PQ\vee), (PQl), (PAff), (PE\wedge), (PQE)$: Obvious.
- *Cases* $R_i \equiv (PX_*)$: If $R_i \equiv (X_0)$ choose $s_{i+1} \in \mathcal{S}$ arbitrarily s.t. $s_i \rightarrow s_{i+1}$; such s_{i+1} exists since $\mathcal{T}$ is total. Moreover set $\xi_{i+1} = \emptyset$. If otherwise $R_i \equiv (X_1)$ let $ABl(\mathcal{S}(z_{i+1})) \equiv \{B'\}$ and $B \in ABl(\mathcal{S}(z_i))$ s.t. $(B, B') \in BCon_P(z_i, z_{i+1})$. Set $s_{i+1} := \xi_i(B)(1)$ and $\xi_{i+1} : B' \mapsto \xi_i(B)[1]$.
- *Case* $R_i \equiv (PA\wedge)$: By Corollary 2.56 and considering that the minimality requirement is not violated.
- *Case* $R_i \equiv (PAA)$: Let $A\psi$ be the principal formula in the principal block B^*. Set $\xi_{i+1} : B \mapsto \xi_i(B)$ for all $B \in dom(\xi_i) \setminus \{B^*\}$. If $B^* \setminus \{A\psi\} \notin dom(\xi_i)$ set $\xi_{i+1} : (B^* \setminus \{A\psi\}) \mapsto \xi_i(B^*)$.

  If the principal successive block merges together with another block, i.e. $A(\psi) = B^* \setminus \{A\psi\}$ or $A(\psi) \in dom(\xi_i)$, then ξ_{i+1} is fully defined fulfilling all properties due to the minimality of the chosen paths. Otherwise let $\pi \in \Pi_{\mathcal{T}}(s_{i+1})$ s.t. $\pi \not\models_{cmin}\updownarrow^{\varphi} \psi$ and set $\xi_{i+1} : (A(\psi)) \mapsto \pi$.
- *Case* $R_i \equiv (PEA)$: Choose z_{i+1} according to the respective branching policy; ξ_{i+1} can be constructed as in the case above.

This construction induces an infinite branch $\Xi : i \mapsto z_i$.

It remains to show that for each $\mathcal{B} \in ThB(P, \Xi)$ there is a thread bundle suffix $\mathcal{B}'$ w.r.t. $\mathcal{B}$ as well as a signature compatible falsifying thread bundle annotation $\pi = (\pi_i)_{i \in roots_{\mathcal{B}'}}$.

Hence let $\mathcal{B} \in ThB(P, \Xi)$ be arbitrary. By Lemma 3.18 $r := \max(roots_B)$ is welldefined; moreover note that $\mathcal{B}_r$ is a singleton of a closed formula (due to the restrictedness of φ). Let $\mathcal{B}'$ be the thread bundle suffix w.r.t. $\mathcal{B}$ starting in r.

A falsifying thread bundle annotation w.r.t. $\mathcal{B}'$ consists of only one path π_0 since $|roots_{\mathcal{B}'}| = 1$; such a path π_0 is constructed as follows:

$$\pi_0 : i \mapsto s_{\min\{k \geq r \mid i = pro_{\mathcal{B}}(r,k)\}}$$

Consider that π_0 is welldefined due to the fact that $\mathcal{B}$ is progressive (Corollary 4.13 in combination with Lemma 3.19). Note that the given annotation is even signature compatible falsifying by construction. □

Now it is not hard to show that our proof system is sound. We simply assume by contradiction that an LTS falsifies a formula on the one hand and that we have a proof tree on the other hand for the formula in question. Then, by the former lemma, we conclude that there is a branch in the proof with each thread bundle having a signature compatible falsified suffix.

By definition of proofs there has to be one thread bundle suffix including a ν-thread equivalence class which is impossible by the semantical result stating - since it is signature compatible falsified - that it has to contain a μ-equivalence class.

Theorem 4.20 (Soundness)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded. If $\vdash\varphi$ then $\models \varphi$.*

Proof. Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded and let $\vdash\varphi$ i.e. there is a proof P for φ. By contradiction assume that $\not\models \varphi$ i.e. there is a transition system $\mathcal{T}$ with $\mathcal{T} \not\models \varphi$, hence Lemma 4.19 claims that there is an infinite branch Ξ in P s.t. the following holds (*):

For each $\mathcal{B} \in ThB(P, \Xi)$ there is a thread bundle suffix $\mathcal{B}'$ w.r.t. $\mathcal{B}$ as well as a signature compatible falsifying thread bundle annotation $\pi = (\pi_i)_{i \in roots_{\mathcal{B}'}}$.

By definition of proofs there is a thread bundle $\mathcal{B} \in ThB(P, \Xi)$ and a thread $t \in Th(\mathcal{B})$ s.t.

$$(**)\ [t]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B})$$

By (*) there is a thread bundle suffix $\mathcal{B}'$ w.r.t. $\mathcal{B}$ as well as a signature compatible falsifying thread bundle annotation $\pi = (\pi_i)_{i \in roots_{\mathcal{B}'}}$; due to Corollary 3.28 (**) implies that

$$(**)'\ [th_{(\mathcal{B},\mathcal{B}')}(t)]_{\equiv^A_{\mathcal{B}'}} \subseteq Th_\nu(\mathcal{B}')$$

holds.

Due to Corollary 4.14 and Corollary 3.17 thread bundle $\mathcal{B}'$ is fair, therefore Lemma 3.44 comprises a thread $s \in Th(\mathcal{B}')$ s.t.

$$(***)\ [s]_{\equiv^E_{\mathcal{B}'}} \subseteq Th_\mu(\mathcal{B}')$$

But due to Lemma 3.36 (**)' and (***) cannot hold at the same time. □

4.3 Completeness

An unproof is more than a syntactical object: It can be canonically mapped to a transition system falsifying the formula in question. While a *proof tree* basically resembles a systematic case distinction comprising all possible transition systems, an *unproof tree* can be seen as a *witness* for the falsifiability.

Given an unproof, a counter model can be extracted quite easily: All maximal consecutive sequence of nodes in the unproof tree not containing an application of a modal rule are identified with states in the transition system whereas the transition relation follows the structure of the unproof tree. The labelling map is configured to falsify the respective node sequence locally in the sense that it falsifies all literals listed in the respective sequent.

Definition 4.21 (Canonical Countermodel)**.** Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded and P be an unproof for φ. The *canonical countermodel w.r.t.* φ *and* P, $\mathcal{T}_P \equiv (\mathcal{S}_P, \theta_P, \rightarrow_P, \lambda_P)$, is defined as follows:

- $\mathcal{S}_P := up_P[dom(P)]$
- $\theta_P := up_P(\epsilon)$
- $\rightarrow_P := \{(t, up_P(s)) \in \mathcal{S}_P \times \mathcal{S}_P \mid t \rightarrow s\}$
- $\lambda_P : t \mapsto \{\neg l \mid l \in Lits(seq_P(t))\}$

where

$$up_P : t \mapsto \begin{cases} t & \text{if } rule_P(t) \equiv (PX_*) \vee \nexists s \in dom(P) : t \rightarrow s \\ up_P(s) & \text{otherwise with } t \rightarrow s \end{cases}$$

Due to the fact that only the (X_*)-rule is possibly branching and each infinite branch in P is progressive (Lemma 4.12), up_P is well-defined.

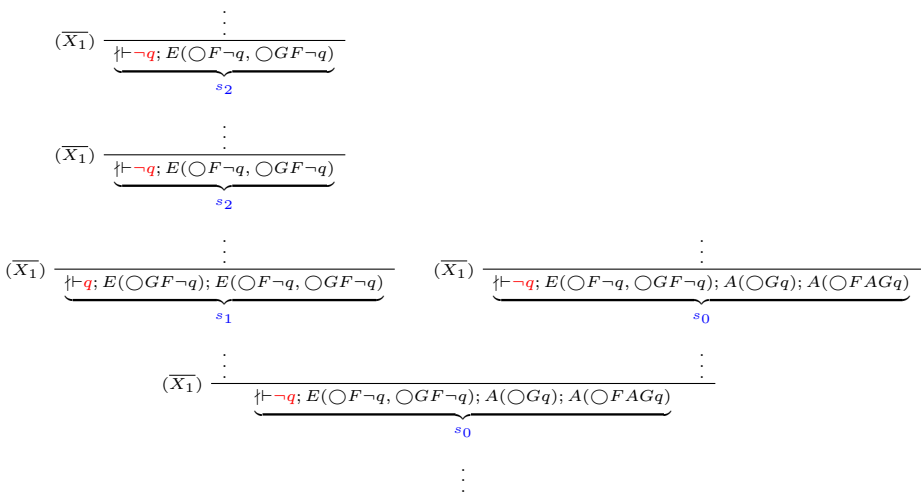
Example 4.22. Again we consider the formula

$$AFGq \to AFAGq$$

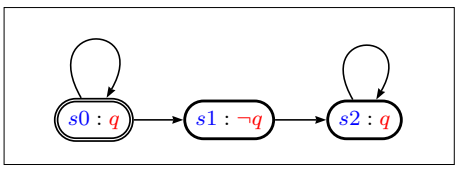
as well as the corresponding unproof of Example 4.18. The associated countermodel can be extracted as follows.

All successive nodes in a branch between two applications of a model rule are grouped together and used as nodes in the countermodel; the transition relation corresponds to the tree structure of the unproof. The labelling map simply returns all these propositions for a node which are not positively listed in the respective sequent.

By omitting all nodes between two modal rule applications the unproof looks as follows



and induces the LTS of Example 2.2:



The semantical completeness result corresponds to its soundness counterpart following the same principle; not very surprisingly, dual problems arise with non-restricted formulas as discussed in the preceding section.

Lemma 4.23 (Extraction of a satisfying branch)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded, P be an unproof for φ and $\mathcal{T}_P$ be the canonical countermodel w.r.t. P with $\mathcal{T}_P \models \varphi$.*

There is an infinite branch Ξ in P and a thread bundle $\mathcal{B} \in ThB(P, \Xi)$ s.t. there is a thread bundle suffix $\mathcal{B}'$ w.r.t. $\mathcal{B}$ as well as a signature compatible satisfying thread bundle annotation $\pi = (\pi_i)_{i \in roots_{\mathcal{B}'}}$.

Proof. Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded, P be an unproof for φ with $P = (dom(P), \mathcal{S}, \mathcal{R}, \mathcal{U}, \Phi, \mathcal{V}, \Psi)$ and $\mathcal{T}_P = (\mathcal{S}_P, \theta_P, \rightarrow_P, \lambda_P)$ be the canonical countermodel w.r.t. P with $\mathcal{T}_P \models \varphi$.

We will inductively construct

- a sequence of nodes $z_0, z_1, z_2, ... \in dom(P)$ s.t. $z_0 = \varepsilon$ and z_{i+1} is a successor of z_i for all $i \in \mathbb{N}$
- a sequence of states $s_0, s_1, s_2, ...$ induced by $s_i := up_P(z_i)$ (which implies particularly $s_0 = \theta$, $s_i = s_{i+1}$ iff $\mathcal{R}(z_i) \not\equiv (PX_*)$ and $s_i \rightarrow s_{i+1}$ iff $\mathcal{R}(z_i) \equiv (PX_*)$ for all $i \in \mathbb{N}$)
- a sequence of blocks $(Q_0, B_0), (Q_1, B_1), ...$ s.t. $(Q_i, B_i) \in Bl(\mathcal{S}(z_i))$ as well as $((Q_i, B_i), (Q_{i+1}, B_{i+1})) \in BCon_P(z_i, z_{i+1})$ for all $i \in \mathbb{N}$
- a sequence of paths ξ_i for all $i \in \mathbb{N}$ with $Q_i = E$

s.t. the following properties hold:

1. Path construction
   (a) $\xi_i(0) = s_i$ for all $i \in \mathbb{N}$ with $Q_i = E$
   (b) $\xi_i = \xi_{i+1}$ for all $i \in \mathbb{N}$ with $Q_i = Q_{i+1} = E$ and $R_i \neq (PX_*), (PEE)$
   (c) $\xi_i[1] = \xi_{i+1}$ for all $i \in \mathbb{N}$ with $Q_i = Q_{i+1} = E$ and $R_i = (PX_*)$
2. Satisfaction properties
   (a) $Q_i = E$ implies $\xi_i \models^{\varphi}_{\updownarrow} \bigwedge B_i$
   (b) $Q_i = A$ implies $s_i \models^{\varphi}_{\updownarrow} A(\bigvee B_i)$

3. Selection policies

   (a) For all $i \in \mathbb{N}$ with $\mathcal{R}(z_i) = (PE\vee)$, $B_i \in \mathcal{U}(z_i)$, $B_{i+1} \in \mathcal{V}(z_{i+1})$ with principal formula $\psi_1 \vee \psi_2$ and principal successive ψ_k holds: If $\zeta \in Sig_\mu(\varphi)$ is the least signature s.t. $\xi_i \models\uparrow_\zeta^\varphi \psi_1 \vee \psi_2$ then $\xi_{i+1} \models\uparrow_\zeta^\varphi \psi_k$

   (b) For all $i \in \mathbb{N}$ with $\mathcal{R}(z_i) = (PAQ)$ and B_i principal with principal formula φ holds: If $s_i \models\updownarrow^\varphi \varphi$ then (B_i, B_{i+1}) is blockspawning

Consider that we are allowed to use the model-relation without environments due to Lemma 2.53.

For the basis of the construction set $z_0 := \varepsilon$, $B_0 := (E, \{\psi\})$ and let ξ_0 be an arbitrary path rooting in s_0 s.t. $\xi_0 \models\updownarrow^\psi \psi$. For better readability we abbreviate $R_i \equiv \mathcal{R}(z_i)$; for $i \rightsquigarrow i+1$ we apply a case distinction w.r.t. R_i; note that R_i can neither be a proof axiom by definition of unproofs nor a counteraxiom by construction. Moreover let z_{i+1} automatically be defined as the only successor of z_i iff R_i is not branching; similarly let $s_{i+1} := s_i$ for all $i \in \mathbb{N}$ with $R_i \neq (PX_*)$.

Now the case distinction w.r.t. R_i; assume w.l.o.g. that B_i is principal:

- *Cases* $R_i \equiv (P\sigma), (PV)$: By Lemma 2.54
- *Cases* $R_i \equiv (PA\vee), (PEl), (PAff), (PQ\wedge), (PEQ)$: Obvious.
- *Case* $R_i \equiv (PE\vee)$: By Corollary 2.56.
- *Case* $R_i \equiv (PAl)$: By construction of the canonical countermodel which implying that $s_i \not\models l$ if l principal; hence R_i will be instantiated block-preserving.
- *Case* $R_i \equiv (PAQ)$: Choose the successive block B_{i+1} s.t. the respective branching policy is fulfilled.
- *Cases* $R_i \equiv (PX_*)$: If $Q_i = E$ choose z_{i+1} s.t. $z_i \rightarrow z_{i+1}$ as well as $\xi_{i+1} = up_P(z_{i+1})$ (this is possible by construction of the canonical countermodel); if otherwise $Q_i = A$ choose z_{i+1} with $z_i \rightarrow z_{i+1}$ s.t. there is a principal successive block w.r.t. B_i in $\mathcal{S}(z_{i+1})$.

This construction induces an infinite branch $\Xi : i \mapsto z_i$ as well as a thread bundle $\mathcal{B} : i \mapsto B_i$ in Ξ. By Lemma 3.18 $r := \max(roots_{\mathcal{B}})$ is welldefined; moreover note that B_r is a singleton of a closed formula (due to the restrictedness of φ). Let $\mathcal{B}'$ be the thread bundle suffix w.r.t. $\mathcal{B}$ starting in r.

A satisfying thread bundle annotation w.r.t. $\mathcal{B}'$ consists of only one path π_0 since $|roots_{\mathcal{B}'}| = 1$; such a path π_0 is constructed as follows:

$$\pi_0 : i \mapsto s_{\min\{k \geq r | i = pro_{\mathcal{B}}(r,k)\}}$$

Consider that π_0 is welldefined due to the fact that $\mathcal{B}$ is progressive (Corollary 4.13 in combination with Lemma 3.19). Note that the given annotation is even signature compatible satisfying by construction. □

Now it is not hard to see that our proof system is complete. We assume by contradiction that the canonical countermodel satisfies a formula on the one hand and that we have an unproof tree on the other hand for the formula in question. Then, by the former lemma, we conclude that there is a branch in the proof containing a signature compatible satisfied thread bundle suffix.

By definition of unproofs all thread bundle suffixes include a μ-thread equivalence class which is impossible by the semantical result stating - since it is signature compatible satisfied - that the extracted thread bundle suffix has to contain a ν-equivalence class.

Theorem 4.24 (Counter model property)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded, P be an unproof for φ and $\mathcal{T}_P$ be the canonical countermodel w.r.t. P. Then $\mathcal{T}_P \not\models \varphi$.*

Proof. Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded, P be an unproof for φ and $\mathcal{T}_P$ be the canonical countermodel w.r.t. P. By contradiction assume that $\mathcal{T}_P \models \varphi$, hence Lemma 4.23 claims that there is an infinite branch Ξ in P and a thread bundle $\mathcal{B} \in ThB(P, \Xi)$ s.t. there is a thread bundle suffix $\mathcal{B}'$ w.r.t. $\mathcal{B}$ as well as a signature compatible satisfying thread bundle annotation $\pi = (\pi_i)_{i \in roots_{\mathcal{B}'}}$.
By definition of unproofs there is a thread $t \in Th(\mathcal{B})$ s.t.

$$(*)\ [t]_{\equiv^E_{\mathcal{B}}} \subseteq Th_\mu(\mathcal{B})$$

Due to Corollary 3.28 (*) implies that

$$(*)' \ [th_{(\mathcal{B},\mathcal{B}')}(t)]_{\equiv^E_{\mathcal{B}'}} \subseteq Th_\mu(\mathcal{B}')$$

holds. Due to Corollary 4.14 and Corollary 3.17 thread bundle $\mathcal{B}'$ is fair, therefore Lemma 3.45 comprises a thread $s \in Th(\mathcal{B}')$ s.t.

$$(**) \ [s]_{\equiv^A_{\mathcal{B}'}} \subseteq Th_\nu(\mathcal{B}')$$

But due to Theorem 3.36 (*)' and (**) cannot hold at the same time. □

As a Corollary we derive the *completeness* of the proof system: If $\mathcal{T}_P$ falsifies a formula φ with an unproof P then φ is particularly not valid.

Theorem 4.25 (Completeness)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded. If $\nvdash \varphi$ then $\not\models \varphi$.*

Proof. Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded and let $\nvdash \varphi$ i.e. there is an unproof P for φ. By Theorem 4.24 $\mathcal{T}_P \not\models \varphi$ and thus particularly $\not\models \varphi$. □

It remains to show that the proof system fulfils the tertium-non-datur: Given a formula there has to be (either) a proof or an unproof.

4.4 Effectiveness

We handle two problems in one go in this chapter: We still need to show that there is always (either) a proof or an unproof. The determinism result of parity games can be easily employed to solve this problem by simply building a parity game whose nodes correspond to nodes in the proof tableaux and whose winning strategies correspond to proofs or unproofs respectively. Since parity games can be effectively solved, we get a decision procedure for $sCTL^*_{\mu r}$-validity for free.

In order to construct automata running over infinite branches, one needs to identify an infinite branch with a word over an alphabet comprising the same information as the infinite branch does: The sequent, the next rule application, the principal blocks, the principal successive blocks and two maps identifying the principal respectively principal successive formulas with respect to a principal respectively principal successive block.

Definition 4.26 (Infinite branch induced word)**.** Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded and $P = (dom(P), \mathcal{S}, \mathcal{R}, \mathcal{U}, \Phi, \mathcal{V}, \Psi)$ be a pre-(un)proof for φ. An infinite branch Ξ in P induces a word $w \in (\Sigma^{Br}_\varphi)^\omega$ with

$$\Sigma^{Br}_\varphi := Seq(\varphi) \times ProofRules \times \mathcal{P}(Blo(\varphi))^2 \times (Blo(\varphi) \to \mathcal{P}(Sub(\varphi)))^2$$

where $Blo(\varphi) = \{E, A\} \times \mathcal{P}(Sub(\varphi))$ as follows:

$$w : i \mapsto (\mathcal{S}(\Xi(i)), \mathcal{R}(\Xi(i)), \mathcal{U}(\Xi(i)), \mathcal{V}(\Xi(i)), \Phi(\Xi(i)), \Psi((\Xi(i+1)))$$

The *bundle tracking automaton* nondeterministically follows a thread bundle contained in the given infinite branch while simulating the deterministic thread bundle automaton of Corollary 3.50 at the same time in order to obtain an automaton accepting infinite branches containing a ν-equivalence class containing thread bundle.

Definition 4.27 (Bundle tracking automaton)**.** Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded and $P = (dom(P), \mathcal{S}, \mathcal{R}, \mathcal{U}, \Phi, \mathcal{V}, \Psi)$ be a pre- (un)proof for φ. Let $\mathcal{A}'$ moreover be the deterministic thread bundle automaton of Corollary 3.50.

The *bundle tracking automaton w.r.t.* φ is a nondeterministic parity automaton $\mathcal{A}^{\mathcal{BUT}}_\varphi = (Blo(\varphi) \times Q_{\mathcal{A}'}, \Sigma^{Br}_\varphi, ((E, \{\varphi\}), q_0^{\mathcal{A}'}), \delta, \Omega)$ with index $O(n \cdot m)$ and $2^{O(n \cdot m \cdot \log(n \cdot m))}$ states (with $n = |\psi|$ and $m = |Bound(\psi)|$) where

$$\Omega : (B, q) \mapsto \Omega_{\mathcal{A}'}(q)$$

and δ maps $(B, q^{\mathcal{A}'}), (S, R, U, V, \Phi, \Psi)$ to

$$\{(B', \delta_{\mathcal{A}'}(q^{\mathcal{A}'}, (B, \Phi(B), \Psi(B'), h(B')))) \mid (B, B') \in BCon_P(S, U, V, R)\}$$

where $h(B') := prule(R, B, \Psi(B'))$.

Due to the fact that the bundle tracking automaton nondeterministically searches for a thread bundle in an infinite branch containing a ν-thread equivalence class (by Corollary 3.50), an accepting run of the bundle tracking automaton corresponds to such a thread bundle.

Corollary 4.28 (Bundle tracking automaton tracks bundles)**.**
Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded, P be a pre-(un)proof for φ, let Ξ be an infinite branch in P and let $\mathcal{A}^{\mathcal{BUT}}_{\varphi}$ be the bundle tracking NPA. Then the following properties hold

1. $ThB(P,\Xi) = pr_1[Runs(\mathcal{A}^{\mathcal{BUT}}_{\varphi},\Xi)]$
2. $ThB_\nu(P,\Xi) = pr_1[AccRuns(\mathcal{A}^{\mathcal{BUT}}_{\varphi},\Xi)]$
3. $ThB_\mu(P,\Xi) = pr_1[Runs(\mathcal{A}^{\mathcal{BUT}}_{\varphi},\Xi) \setminus AccRuns(\mathcal{A}^{\mathcal{BUT}}_{\varphi},\Xi)]$

where $pr_1 : (\lambda i \mapsto (a(i), b(i))) \mapsto (\lambda i \mapsto a(i))$.

The determinization (by Corollary B.6) of the bundle tracking automaton leads to a deterministic parity automaton accepting all these infinite branches containing a ν-thread equivalence class.

Corollary 4.29 (Proof branch automaton)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded. There is a deterministic parity automaton $\mathcal{A}^{\mathcal{PB}}_{\varphi}$ with index $2^{O(n\cdot m\cdot\log(n\cdot m))}$ and $2^{\left(2^{O(n\cdot m\cdot\log(n\cdot m))}\right)}$ states s.t. for all pre-(un)proof P and all infinite branches Ξ in P holds*

$$\Xi \text{ is accepted by } \mathcal{A}^{\mathcal{PB}}_{\varphi} \text{ iff } ThB_\nu(P,\Xi) \neq \emptyset$$

where $n = |\psi|$ and $m = |Bound(\psi)|$.

In order to make the induced parity game less complex, we reduce the degree of freedom in choosing which proof rule should by applied next at a certain position in a tableaux. This can easily be done due to the fact that most rules are *confluent* w.r.t. a specific equivalence-preserving equivalence relation. That said as a motivation for introducing a deterministic rule selection, we do without the details of confluence of proof rules here.

Definition 4.30 (Deterministic rule selection)**.** Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded.
First, we induce a total ordering $<_R$ on *ProofRules* by $(PP) >_R (PEtt) >_R (PFF) >_R (PAff) >_R (PAl) >_R (PEl) >_R (PA\vee) >_R (PE\wedge) >_R (PQ\nu) >_R (PQ\mu) >_R (PQV) >_R (PE\vee) >_R (PA\wedge) >_R (PAA) >_R (PAE) >_R (PEA) >_R (PEE) >_R (PX_0) >_R (PX_1)$.
Second, we define a total ordering $<_B$ on $B := \{E, A\} \times \mathcal{P}(Sub(\varphi))^2$ as the lexicographic ordering induced by $E < A$ and twice $\prec_{\mathcal{P}(Sub(\varphi))}$. Subsequently a total ordering on $\mathcal{P}(\{E, A\} \times \mathcal{P}(Sub(\varphi))^2)$ is induced by $<_{\mathcal{P}(B)}$.
Third, we define a total ordering $<$ on

$$ProofRules \times \mathcal{P}(\{E, A\} \times \mathcal{P}(Sub(\varphi))) \times \mathcal{P}(\{E, A\} \times \mathcal{P}(Sub(\varphi))^2)$$

as the lexicographic ordering induced by $<_R$ and $<_{\mathcal{P}(B)}$.
Finally we define a rule instantiation relation $\rightsquigarrow$ as follows:

$$S \rightsquigarrow (U, \Phi, R, \{(S_1, V_1, \Psi_1), ..., (S_n, V_n, \Psi_n)\})$$

iff

$$(R): \frac{\vdash S_1\lceil\Psi_1\rceil \qquad \vdash S_2\lceil\Psi_2\rceil \qquad ... \qquad \vdash S_n\lceil\Psi_n\rceil}{\vdash S\lfloor\Phi\rfloor}$$

is a branching instantiation of R.

Note that $\{(S_1, V_1, \Psi_1), ..., (S_n, V_n, \Psi_n)\}$ is uniquely determined by S, U, Φ and R, hence $F(S) := \{((R, U, \Phi), M) \mid S \rightsquigarrow (U, \Phi, R, M)\}$ is a function.

The *deterministic rule selection function* $detR_\varphi$ finally is defined as follows:

$$detR_\varphi : S \mapsto (max_< dom(F(S)), F(S)(max_< dom(F(S))))$$

The *proof game* is a parity game associated with a fixed formula. Its states are pairs consisting of a sequent and a state of the deterministic proof branch automaton. The transition relation of the game corresponds to the tableaux building options; the parity assignment function simply uses the parity assignment function of the proof branch automaton.

Definition 4.31 (Proof Game)**.** Let $\varphi \in CTL^*_{c\mu+}$ be restricted and state-guarded. Let $\mathcal{A}^{\mathcal{PB}}_{\varphi}$ be the proof branch automaton of Corollary 4.29.

The *induced proof game w.r.t.* φ, $G_\varphi = (Pos_\exists, Pos_\forall, \delta, \Omega)$, is defined as follows:

- $Pos_G := Seq(\varphi) \times Q^{\mathcal{PB}}$ with
  - $Pos_\exists := \{(S,q) \in Pos_G \mid detR_\varphi(S) \text{ is existential}\}$ whereas *existential* is defined as existentially branching, not branching or being a counteraxiom
  - $Pos_\forall := \{(S,q) \in Pos_G \mid detR_\varphi(S) \text{ is universal}\}$ whereas *universal* is defined as universally branching or being an axiom

  i.e. , G_φ has totally $2^{\left(2^{O(n\cdot m\cdot \log(n\cdot m))}\right)}$ states (where $n = |\varphi|$ and $m = |Bound(\varphi)|$)
- $\delta : (S,q) \mapsto h(S,q,\mathcal{U},\Phi,R)[H]$ where $(R,\mathcal{U},\Phi,H) = detR_\varphi(S)$ and

$$h : (S,q,\mathcal{U},\Phi,R,S',\mathcal{V},\Psi) \mapsto (S', \delta^{\mathcal{BU}}(q,(S,R,\mathcal{U},\mathcal{V},\Phi,\Psi)))$$

- $\Omega : (S,q) \mapsto \Omega^{\mathcal{PB}}(q)$

It is not hard to see that a proof tree corresponds to a winning strategy for $\exists$ in the proof game while an unproof corresponds to a winning strategy for $\forall$.

Theorem 4.32 (Proof Game Property)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded and G_φ be the induced proof game. Then:*

$$\vdash\varphi \quad \textit{if} \quad (E(\varphi), q_0^{\mathcal{PB}}) \textit{ is won by player } \exists$$
$$\textit{and}$$
$$\nvdash\varphi \quad \textit{if} \quad (E(\varphi), q_0^{\mathcal{PB}}) \textit{ is won by player } \forall$$

Proof. Let $(E(\varphi), q_0^{\mathcal{PB}})$ be won by player $\exists$ (the other case can be shown the same way) and let $s_\exists : W_\exists \cap Pos_\exists \to W_\exists$ be a positional winning strategy for player $\exists$. A proof $(dom(P), \mathcal{S}, \mathcal{R}, \mathcal{U}, \Phi, \mathcal{V}, \Psi)$ is induced as follows:

- $dom(P) := T(Pos_G, (E(\varphi), q_0^{\mathcal{PB}}), \rightarrow)$ where

$$p_1 \rightarrow p_2 \ : \Longleftrightarrow \ \begin{cases} p_2 \in \delta(p_1) & \text{if } p_1 \in Pos_\forall \\ p_2 = s_\exists(p_1) & \text{if } p_1 \in Pos_\exists \end{cases}$$

- $\mathcal{S} : z \mapsto S$ where $(S, q) = Head(z)$
- $\mathcal{R} : z \mapsto R$, $\mathcal{U} : z \mapsto U$ and $\Phi : z \mapsto P$ where $(S, q) = Head(z)$ and $(R, U, \Phi, H) = detR_\varphi(S)$
- $\mathcal{V} : z' \mapsto V$ and $\Psi : z' \mapsto X$ where $(S', q') = hd(z')$, z is the immediate predecessor of z', $(S, q) = Head(z)$, $(R, U, P, H) = detR_\varphi(S)$ and $(S', V, X) \in H$ with $\delta^{\mathcal{PB}}(q, (S, R, U, V, P, X)) = q'$

Consider that $\mathcal{P}$ is actually a pre-proof; $\mathcal{P}$ moreover is a proof since all finite branches end in axioms by definition of $detR_\varphi$ and all infinite branches fulfil the proof condition because each infinite branch resembles an accepting run of the proof branch automaton of Corollary 4.29 and each branch is a fair (by Corollary 4.14) thread bundle (since $\mathcal{PM}$ is a pre-proof). □

In the preceding sections we have shown that

- Theorem 4.20: If there is a proof tree then the formula is valid
- Theorem 4.25: If there is an unproof then the formula is falsifiable

Due the positional determinism of parity games (Theorem C.2) and Theorem 4.32 we know that there is either a proof or an unproof.

Combining these results we finally derive that the proof system is sound and complete.

Corollary 4.33 (Proof system is sound and complete)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded. Then:*

$$\vdash \varphi \text{ if not and only if not } \nvdash \varphi$$

In other words:

$$\vdash \varphi \text{ iff } \models \varphi$$

By definition of the proof game and the application of a parity game solving decision procedure, say Theorem C.3, we derive the following decision complexity:

Theorem 4.34 (Decision complexity)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded. Deciding whether $\models \varphi$ holds or not using the induced proof game works in space*

$$2^{\left(2^{O(n \cdot m \cdot \log(n \cdot m))}\right)}$$

and its running time is

$$2^{\left(2^{O(n \cdot m \cdot \log(n \cdot m))}\right)}$$

where $n = |\varphi|$ and $m = |Bound(\varphi)|$.

We conclude that the validity decision problem for restricted CTL^*_{μ} is *2-EXPTIME-complete* with CTL^* being *2-EXPTIME-hard* [VS85] and less expressive as $CTL^*_{\mu r}$.

Theorem 4.35 (2-EXPTIME-Completeness)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be restricted and state-guarded. Deciding whether $\models \varphi$ is* 2-EXPTIME-complete*.*

Proof. By Theorem 4.34 the decision complexity is in *2-EXPTIME*. For the hardness result note that the validity problem for CTL^* has been shown to be *2-EXPTIME-hard* [VS85] and since $CTL^* \leq CTL^*_{\mu r}$ we finally conclude that the decision problem indeed is *2-EXPTIME-complete*. □

5. MODEL-CHECKING

The Model-Checking Tableaux System is a tableaux-style system for the model-checking problem of CTL^*_μ. A model-checking tableaux is a possibly infinite, possibly infinitely-branching tree whose nodes are labelled with a state and a formula block. A valid model-checking proof tree for a fixed formula and a fixed transition system is a tableaux whose root is labelled with the initial formula and the initial state, built according to the model-checking rules with all finite branches ending in model-checking axioms and all infinite branches fulfilling a global condition.

As with the proof system, there will be either a model-checking proof or a model-checking unproof again by a determinism result derived from the determinism of parity games. Fortunately, our model-checking tableaux work with full $sCTL^*_\mu$ basically due to the fact that the labellings of the tableaux do not comprise a set of blocks but only one block.

5.1 Definition

The labellings of the model-checking tableaux basically combine a state in the transition system with a formula block. A model-checking rule in general can be infinitely branching, for instance if a state in the transition system is connected to infinitely many successive states.

Definition 5.1 (Model-Checking Rule)**.** A *model-checking rule* (R) is a formal construct w.r.t. a transition system $\mathcal{T} = (\mathcal{S}, \theta, \rightarrow, \lambda)$ and a fixed formula φ written as follows

$$(R): \frac{s_1 \vdash Q_1(\Lambda_1) \qquad s_2 \vdash Q_2(\Lambda_2) \qquad \ldots}{s \vdash Q'(\Lambda)}[Q]$$

where all $\Lambda_1, \Lambda_2, ..., \Lambda \subseteq Sub(\varphi)$ are formula sets, all $Q_1, Q_2, ..., Q' \in \{E, A\}$ are quantor labels and all $s_1, s_2, ..., s \in \mathcal{S}$ are states.

All other definitions with respect to rules and tableaux systems in general are defined likewise as in the preceding chapter.

The rules of the model-checking tableaux are given as follows. We basically follow the bundle rules, branch whenever it is necessary and handle all occurring literals by evaluating the labelling map of the respective transition system.

Definition 5.2 (Model-Checking Rules)**.** The *model-checking rules* w.r.t. to a transition system $\mathcal{T} = (\mathcal{S}, \theta, \rightarrow, \lambda)$ and a fixed formula $\varphi \in sCTL^*_{c\mu+}$ are defined as follows:

$$(MA\wedge): \frac{s\vdash A(\lceil\psi_1\rceil, \Lambda) \qquad s\vdash A(\lceil\psi_2\rceil, \Lambda)}{s\vdash A(\lfloor\psi_1 \wedge \psi_2\rfloor, \Lambda)}[\forall] \qquad (MA\vee): \frac{s\vdash A(\lceil\psi_1\rceil, \lceil\psi_2\rceil, \Lambda)}{s\vdash A(\lfloor\psi_1 \vee \psi_2\rfloor, \Lambda)}$$

$$(ME\vee): \frac{s\vdash E(\lceil\psi_1\rceil, \Lambda) \qquad s\vdash E(\lceil\psi_2\rceil, \Lambda)}{s\vdash E(\lfloor\psi_1 \vee \psi_2\rfloor, \Lambda)}[\exists] \qquad (ME\wedge): \frac{s\vdash E(\lceil\psi_1\rceil, \lceil\psi_2\rceil, \Lambda)}{s\vdash E(\lfloor\psi_1 \wedge \psi_2\rfloor, \Lambda)}$$

$$(M\sigma): \frac{s\vdash Q(\lceil X\rceil, \Lambda)}{s\vdash Q(\lfloor\sigma X.\psi\rfloor, \Lambda)} \qquad (MV): \frac{s\vdash Q(\lceil body_\varphi(X)\rceil, \Lambda)}{s\vdash Q(\lfloor X\rfloor, \Lambda)}$$

$$(MAQ): \frac{s\vdash Q(\lceil\psi\rceil) \qquad s\vdash A(\Lambda)}{s\vdash A(\lfloor Q\psi\rfloor, \Lambda)}[\exists] \qquad (MAl): \frac{s\vdash A(\Lambda)}{s\vdash A(\lfloor l\rfloor, \Lambda)} l \notin \lambda(s)$$

$$(MEQ): \frac{s\vdash Q(\lceil\psi\rceil) \qquad s\vdash E(\Lambda)}{s\vdash E(\lfloor Q\psi\rfloor, \Lambda)}[\forall] \qquad (MEl): \frac{s\vdash E(\Lambda)}{s\vdash E(\lfloor l\rfloor, \Lambda)} l \in \lambda(s)$$

$$(MEX): \frac{s_1\vdash E(\lceil\psi_1\rceil, ..., \lceil\psi_n\rceil) \qquad s_2\vdash E(\lceil\psi_1\rceil, ..., \lceil\psi_n\rceil) \qquad ...}{s\vdash E(\lfloor\bigcirc\psi_1\rfloor, ..., \lfloor\bigcirc\psi_n\rfloor)}[\exists] s \rightarrow s_k$$

$$(MAX): \frac{s_1\vdash A(\lceil\psi_1\rceil, ..., \lceil\psi_n\rceil) \qquad s_2\vdash A(\lceil\psi_1\rceil, ..., \lceil\psi_n\rceil) \qquad ...}{s\vdash A(\lfloor\bigcirc\psi_1\rfloor, ..., \lfloor\bigcirc\psi_n\rfloor)}[\forall] s \rightarrow s_k$$

where $\Lambda \subseteq Sub(\varphi)$, $\psi_1, \psi_2, ..., \psi_n \in Sub(\varphi)$, $l \in \mathcal{P}^*$ state formula, $\sigma X.\psi \in Sub(\varphi)$ $X \in Bound(\varphi)$, $Q \in \{E, A\}$ and $s, s_1, ..., s_m \in \mathcal{S}$.

The modal rule condition $s \rightarrow s_k$ specifies that for each s_k with $s \rightarrow s_k$ there is a premiss labelled with s_k.

The *axioms* of the model-checking tableaux system comprehend all these tableaux nodes that can be directly verified to be correct.

Definition 5.3 (Model-Checking Axioms)**.** The *model-checking axioms* w.r.t. a fixed formula $\varphi \in sCTL^*_{c\mu+}$ and a transition system $\mathcal{T} = (\mathcal{S}, \theta, \rightarrow, \lambda)$ are as follows:

$$(MAl^*): \frac{}{s \vdash A(\lfloor l \rfloor, \Lambda)} l \in \lambda(s) \qquad (MEtt): \frac{}{s \vdash E(\emptyset)}$$

where $\Lambda \subseteq Sub(\varphi)$ and $s \in \mathcal{S}$.

Whereas the counteraxioms contain all these tableaux nodes that can be directly verified to be incorrect.

Definition 5.4 (Model-Checking Counteraxioms)**.** The *model-checking counteraxioms* w.r.t. a fixed formula $\varphi \in sCTL^*_{c\mu+}$ and a transition system $\mathcal{T} = (\mathcal{S}, \theta, \rightarrow, \lambda)$ are as follows:

$$(MEl^*): \frac{}{s \vdash E(\lfloor l \rfloor, \Lambda)} l \notin \lambda(s) \qquad (MAff): \frac{}{s \vdash A(\emptyset)}$$

where $\Lambda \subseteq Sub(\varphi)$ and $s \in \mathcal{S}$.

A *pre-model-checking-proof* for a formula φ and a transitions system $\mathcal{T}$ is basically a tableaux-style tree built according to the model-checking rules with its root being labelled with $\theta \vdash E\varphi$. Additionally, all finite branches need to end in model-checking axioms. Therefore a pre-model-checking-proof is a *locally sound and complete* tableaux.

Please refer to Appendix A for the abstract definition of trees.

Definition 5.5 (Pre-Model-Checking-Proof)**.** Let $\varphi \in sCTL^*_{c\mu+}$ be a state-guarded state formula and $\mathcal{T} = (\mathcal{S}, \theta, \rightarrow, \lambda)$ be an LTS. A *pre-model-checking-proof* for φ w.r.t. $\mathcal{T}$ is a possibly infinite, possibly infinitely branching tree M where

- each inner node is labelled with a state, a block and a model-checking rule name
- the root is labelled with $\theta \vdash E(\varphi)$ and a model-checking rule name
- each node along with its direct subnodes is a regularly instantiated model-checking rule

- all finite branches end in model-checking axioms

More formally, M is tuple $(dom(M), \mathcal{Q}, \mathcal{L}, \varsigma, \mathcal{R}, \Phi, \Psi)$ where

- $dom(M)$ is a tree; for two nodes $t, s \in dom(M)$ the relation $t \rightarrow s$ holds iff s is an immediate successor of t.
- $\mathcal{Q} : dom(M) \rightarrow \{E, A\}$ is a labelling map
- $\mathcal{L} : dom(M) \rightarrow \mathcal{P}(Sub(\varphi))$ maps every node in M to a formula block
- $\varsigma : dom(M) \rightarrow \mathcal{S}$ maps every node in M to a state in $\mathcal{S}$ with $\varsigma(\varepsilon) = \theta$
- $\Phi : dom(M) \rightarrow \mathcal{P}(Sub(\varphi))$ is a principal formulas marker with $\Phi(t) \subseteq \mathcal{L}(t)$ for all $t \in dom(M)$
- $\Psi : dom(M) \rightarrow \mathcal{P}(Sub(\varphi))$ is a principal successive formulas marker with $\Psi(t) \subseteq \mathcal{L}(t)$ for all $t \in dom(M)$
- $\mathcal{R}$ maps every node in $dom(M)$ having a successor to a rule identifier and every leaf to an axiom identifier

s.t. for all nodes $t \in dom(M)$ with successors $t_1, ...$

$$(\mathcal{R}(t)) : \frac{\varsigma(t_1) \vdash \mathcal{Q}(t_1)(\lceil \Psi(t_1) \rceil, \mathcal{L}(t_1)) \qquad \varsigma(t_2) \vdash \mathcal{Q}(t_2)(\lceil \Psi(t_2) \rceil, \mathcal{L}(t_2)) \qquad ...}{\varsigma(t) \vdash \mathcal{Q}(t)(\lfloor \Phi(t) \rfloor, \mathcal{L}(t) \setminus \Phi(t))}$$

is a valid regular model-checking rule (or model-checking axiom) instance. Note that if the proof tree is infinitely branching then $\mathcal{T}$ is infinitely branching, too.

The *pre-model-checking-unproof* is similarly defined, thus we do without a too formal definition:

Definition 5.6 (Pre-Model-Checking-Unproof)**.** A *pre-model-checking-unproof* is defined likewise whereas each rule is instantiated inversely and each finite branch ends in a counter model-checking axiom.

A *proof branch* is a finite or infinite connected part of a pre-model-checking-(un)proof branch starting with the root.

Definition 5.7 (Proof branch)**.** A *proof branch* Ξ in a pre-model-checking-(un)proof M for a formula ψ w.r.t. $\mathcal{T}$ is a map $\Xi : dom(\Xi) \to dom(M)$ with $\Xi(0) = \varepsilon$ and $\Xi(i) \to \Xi(i+1)$ for all $i \in idom(\Xi)$ where $dom(\Xi) = \mathbb{N}$ or $dom(\Xi) = \{i \in \mathbb{N} \mid i < n\}$ for some $n > 0$ and $idom(\Xi) := \{i \in \mathbb{N} \mid i+1 \in dom(\Xi)\}$. Such a proof branch Ξ is called *infinite branch* iff $dom(\Xi) = \mathbb{N}$ and *prefix branch* otherwise.

Every infinite branch induces a thread bundle in a natural way as model-checking branches simply consist of a sequence of states and a thread bundle.

Lemma 5.8 (Branches are thread bundles)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be state-guarded, $M = (dom(M), \mathcal{Q}', \mathcal{L}', \varsigma, \mathcal{R}', \Phi', \Psi')$ be a pre-model-checking-(un)proof for φ w.r.t. $\mathcal{T}$ and Ξ be a proof branch in M.*
Then Ξ induces a thread bundle $\mathcal{B}_M(\Xi) := (\mathcal{Q}, \mathcal{L}, \Phi, \Psi, \mathcal{R})$ as follows

- $\mathcal{Q}_i := \mathcal{Q}'(\Xi(i))$ *for all* $i \in dom(\Xi)$
- $\mathcal{L}_i := \mathcal{L}'(\Xi(i))$ *for all* $i \in dom(\Xi)$
- $\Phi : i \mapsto \Phi'(\Xi(i))$ *for all* $i \in idom(\Xi)$
- $\Psi : i \mapsto \Psi'(\Xi(i))$ *for all* $i \in dom(\Xi)$
- $\mathcal{R} : i \mapsto mrule(\mathcal{R}'(\Xi(i)), \Psi(i+1))$ *for all* $i \in idom(\Xi)$

where

$$mrule : (R, D) \mapsto \begin{cases} (BW) & \text{if } R \in \{(MEl), (MAl)\} \\ (BW) & \text{if } R \in \{(MEQ), (MAQ)\},\ D = \emptyset \\ (BQ) & \text{if } R \in \{(MEQ), (MAQ)\},\ D \neq \emptyset \\ (B\mathcal{X}) & \text{otherwise with } R \equiv (M\mathcal{X}) \end{cases}$$

The contained thread bundles in a pre-model-checking-(un)-proof are cleary lively due to the fact that the (BS)-bundle rule is never applied.

Corollary 5.9 (Induced thread bundles are lively)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be state-guarded, M be a pre-model-checking-(un)proof for φ w.r.t. $\mathcal{T}$ and Ξ be an infinite branch in M. Then $\mathcal{B}_M(\Xi)$ is lively.*

By Corollary 5.9 and Corollary 3.31 we derive that all induced thread bundles in model-checking tableaux are fair.

Corollary 5.10 (Induced thread bundles are fair)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be state-guarded, M be a pre-model-checking-(un)proof for φ w.r.t. $\mathcal{T}$ and Ξ be an infinite branch in M. Then $\mathcal{B}_M(\Xi)$ is fair.*

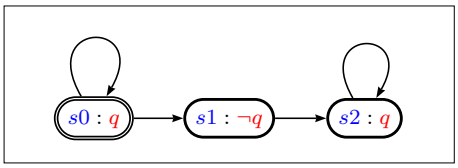
A *model-checking-proof* extends the definition of a pre-model-checking-proof by *global conditions* that need to be fulfilled for being a witness for the validity of the modelling relation w.r.t. the formula and the transition system in question. All finite branches obviously need to end in an axiom whereas all infinite branches need to fulfil the standard bundle conditions.

Definition 5.11 (Model-Checking-Proof)**.** Let $\mathcal{T}$ be a transition system. A *model-checking-proof* M for a state-guarded $\varphi \in sCTL^*_{c\mu+}$ w.r.t. $\mathcal{T}$ is a pre-model-checking-proof M s.t. all finite branches end in model-checking axioms and for all infinite branches Ξ exists a thread $t \in \mathcal{B}_M(\Xi)$ s.t. $[t]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B}_M(\Xi))$.
We also write $\mathcal{T} \vdash \varphi$ to indicate that there is a model-checking-proof for φ w.r.t. $\mathcal{T}$.

Example 5.12. We give a model-checking proof for the rather simple formula

$$AFGq$$

and the transition system of Example 2.2:



A model-checking proof runs as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{s_2 \vdash A(\bigcirc FGq; q) \qquad \dfrac{\dfrac{\dfrac{\vdots}{\rightsquigarrow s_2 \vdash A(\bigcirc FGq; Gq)}}{s_2 \vdash A(FGq; Gq)}}{s_2 \vdash A(\bigcirc FGq; \bigcirc Gq)}}{s_2 \vdash A(\bigcirc FGq; q \wedge \bigcirc Gq)}}{\rightsquigarrow s_2 \vdash A(\bigcirc FGq; Gq)}}{s_2 \vdash A(FGq)}}{s_1 \vdash A(\bigcirc FGq)}}{s_1 \vdash A(\bigcirc FGq; q)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{s_2 \vdash A(\bigcirc FGq; q) \qquad \dfrac{\dfrac{\dfrac{\vdots}{\rightsquigarrow s_2 \vdash A(\bigcirc FGq; Gq)}}{s_2 \vdash A(FGq; Gq)}}{s_2 \vdash A(\bigcirc FGq; \bigcirc Gq)}}{s_2 \vdash A(\bigcirc FGq; q \wedge \bigcirc Gq)}}{\rightsquigarrow s_2 \vdash A(\bigcirc FGq; Gq)}}{s_2 \vdash A(FGq; Gq)}}{s_1 \vdash A(\bigcirc FGq; \bigcirc Gq)}}{\dfrac{s_1 \vdash A(\bigcirc FGq; q \wedge \bigcirc Gq)}{\dfrac{s_1 \vdash A(FGq; Gq)}{(*)}}}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\rightsquigarrow s_0 \vdash A(\bigcirc FGq; Gq)}}{s_0 \vdash A(FGq; Gq)} \qquad (*)}{s_0 \vdash A(\bigcirc FGq; \bigcirc Gq)} \qquad s_0 \vdash A(\bigcirc FGq; q)}{s_0 \vdash A(\bigcirc FGq; q \wedge \bigcirc Gq)}}{\rightsquigarrow s_0 \vdash A(\bigcirc FGq; Gq)}}{s_0 \vdash A(FG\neg q)}$$

Definition 5.13 (Model-Checking-Unproof)**.** Let $\mathcal{T}$ be a transition system. A *model-checking-unproof* M for a state-guarded $\varphi \in sCTL^*_{c\mu+}$ w.r.t. $\mathcal{T}$ is a pre-model-checking-unproof M s.t. all finite branches end in model-checking counteraxioms and for all infinite branches Ξ exists a thread $t \in \mathcal{B}_M(\Xi)$ s.t. $[t]_{\equiv^E_{\mathcal{B}}} \subseteq Th_\mu(\mathcal{B}_M(\Xi))$.
We also write $\mathcal{T} \nvdash \varphi$ to indicate that there is a model-checking-unproof for φ w.r.t. $\mathcal{T}$.

An interesting property of these model-checking tableaux is that the subsequent (meta-)proofs of soundness and completeness basically are the same thing due to the fact a model-checking proof for a formula is canonically related to a model-checking unproof for the dual formula. To formalise this we define the *dual* version of a model-checking proof.

Definition 5.14 (Dual Model-Checking Proof)**.** Let $\mathcal{T}$ be a transition system, $M = (dom(M), \mathcal{Q}, \mathcal{L}, \varsigma, \mathcal{R}, \Phi, \Psi)$ be a model-checking-(un)proof for state-guarded $\varphi \in sCTL^*_{c\mu+}$ w.r.t. $\mathcal{T}$.

The *dual model-checking proof* $\overline{M} = (dom(M), \overline{\mathcal{Q}}, \overline{\mathcal{L}}, \varsigma, \overline{\mathcal{R}}, \overline{\Phi}, \overline{\Psi})$ is defined as follows:

- $\overline{\mathcal{Q}} : z \mapsto Q(z)^C$ with $E^C := A$ and $A^C := E$
- $\overline{\mathcal{L}} : z \mapsto \{\psi^\ominus \mid \psi \in \mathcal{L}(z)\}$
- $\overline{\Phi} : z \mapsto \{\psi^\ominus \mid \psi \in \Phi(z)\}$
- $\overline{\Psi} : z \mapsto \{\psi^\ominus \mid \psi \in \Psi(z)\}$
- $\overline{\mathcal{R}} : z \mapsto \mathcal{R}(z)^C$ where $(MA\vee)^C := (ME\wedge)$, $(ME\vee)^C := (MA\wedge)$, $(MA\wedge)^C := (ME\vee)$, $(ME\wedge)^C := (MA\vee)$, $(M\mu)^C := (M\nu)^C$, $(M\nu)^C := (M\mu)$, $(MV)^C := (MV)$, $(MQ_1Q_2)^C := (M\overline{Q_1Q_2})$, $(MAl)^C := (MEl)$, $(MEl)^C := (MAl)$, $(MEX)^C := (MAX)$ and $(MAX)^C := (MEX)$.

Clearly, the dual model-checking proof simply substitutes each formula, each path quantification label and each rule by their corresponding counterpart. Therefore it is not hard to see that the dual model-checking-proof for a formula is a model-checking-unproof for the dual formula.

Lemma 5.15 (Dual Model-Checking Proof)**.** *Let $\mathcal{T}$ be a transition system, M be a model-checking-proof (unproof) for state-guarded $\varphi \in sCTL^*_{c\mu+}$ w.r.t. $\mathcal{T}$. Then $\overline{M}$ is a model-checking-unproof (proof) for $\varphi^{\ominus}$.*

5.2 Soundness

In order to prove our model-checking tableaux system sound - and also complete by duality - we need to show that if an LTS falsifies a formula which also has model-checking proof, one could extract a thread bundle with a signature compatible falsifying annotation. We will thereafter use that result to show that such a scenario is impossible which brings us to the conclusion that our system is sound.

In principle, we construct the thread bundle along with the falsifying annotation iteratively. Whenever a new A-block is spawned and followed, we directly select a corresponding path falsifying the respective root formula signature compatible. Whenever a new E-block is spawned, we simply select connected states when handling the next rule; it is not possible to immediately choose a whole path due to the fact that the falsification of each formula in an E-block depends on the chosen path and since we do not know at that time which formula will be subsequently followed, we specify the corresponding path afterwards.

Lemma 5.16 (Extraction of a falsifying thread bundle)**.** *Let $\mathcal{T}$ be a transition system, $\varphi \in sCTL^*_{c\mu+}$ state-guarded, M be a model-checking-proof for φ w.r.t. $\mathcal{T}$ and $\mathcal{T} \not\models \varphi$. There is an infinite branch Ξ in M as well as a signature compatible falsifying thread bundle annotation $\pi = (\pi_i)_{i \in roots_{\mathcal{B}_M(\Xi)}}$ w.r.t. $\mathcal{B}_M(\Xi)$.*

Proof. Let $\mathcal{T}$ be a transition system, $M = (dom(M), \mathcal{Q}, \mathcal{L}, \varsigma, \mathcal{R}, \Phi, \Psi)$ be a model-checking-proof for state-guarded $\varphi \in CTL^*_{c\mu+}$ w.r.t. $\mathcal{T}$ and $\mathcal{T} \not\models \varphi$.
We will inductively construct

- a sequence of nodes $z_0, z_1, z_2, ... \in dom(M)$ s.t. $z_0 = \varepsilon$ and z_{i+1} is a successor of z_i for all $i \in \mathbb{N}$ (and use the abbreviations $s_i := \varsigma(z_i)$, $Q_i := \mathcal{Q}(z_i)$, $\Lambda_i := \mathcal{L}(z_i)$, $R_i := \mathcal{R}(z_i)$ for all $i \in \mathbb{N}$)

- an ascending sequence of induced thread bundles $\mathcal{B}_0, \mathcal{B}_1, ...$ where $\mathcal{B}_i := \mathcal{B}_M(\Xi_i)$ with $\Xi_i : \{0, ..., i\} \to dom(M)$, $j \mapsto z_j$
- a sequence of paths χ_i for all $i > 0$ with $R_{i-1} = (MEQ)$ and $\Psi(z_i) \neq \emptyset$ (these will be path completions of all finite existential segments)
- a sequence of paths ξ_i for all $i \in \mathbb{N}$ with $Q_i = A$

s.t. the following properties hold:

1. Falsification conditions
   (a) $s_i \not\models\!\updownarrow^{\varphi} E(\bigwedge \Lambda_i)$ for all $i \in \mathbb{N}$ with $Q_i = E$
   (b) $\xi_i \not\models\!\updownarrow^{\varphi} \bigvee \Lambda_i$ for all $i \in \mathbb{N}$ with $Q_i = A$
2. Path construction
   (a) $\xi_i(0) = s_i$ for all $i \in \mathbb{N}$ with $Q_i = A$
   (b) $\xi_i = \xi_{i+1}$ for all $i \in \mathbb{N}$ with $Q_i = Q_{i+1} = A$ and $R_i \neq (MAX), (MAA)$
   (c) $\xi_i[1] = \xi_{i+1}$ for all $i \in \mathbb{N}$ with $Q_i = Q_{i+1} = A$ and $R_i = (MAX)$
3. Branching policies
   (a) If $R_i = (MEE), (MEA)$ with principal ψ then $\Psi(z_{i+1}) \neq \emptyset$ iff there is a path completion χ of $APath_M(i)$ s.t. $\psi \in Cand^{\chi}_{\mathcal{B}_i}(i)$ - in this case additionally set $\chi_{i+1} := \chi$
   (b) If $R_i = (MEA), (MAA)$ with principal successive $\psi \in \Psi(z_{i+1})$ and $\zeta \in Sig_{\nu}(\varphi)$ is the least signature s.t. $s_i \not\models\!\downarrow^{\varphi}_{\zeta} A\psi$ then also $\xi_{i+1} \not\models\!\downarrow^{\varphi}_{\zeta} \psi$
   (c) If $R_i = (MA\wedge)$ and $\zeta \in Sig_{\nu}(\varphi)$ is the least signature s.t. $\xi_i \not\models\!\downarrow^{\varphi}_{\zeta} \bigvee \Phi(i)$ then also $\xi_{i+1} \not\models\!\downarrow^{\varphi}_{\zeta} \bigvee \Psi(i+1)$

where $APath_M(i)$ denotes the prefix-path of length $pro_{\mathcal{B}_i}(root_{\mathcal{B}_i}(i), i)$ where

$$APath_M(i) : j \mapsto s_{\min\{k \geq i | j = pro_{\mathcal{B}_i}(root_{\mathcal{B}_i}(i), k)\}}$$

Consider that we are allowed to use the model-relation without environments due to Lemma 2.53.

For $i = 0$ simply set $z_0 := \varepsilon$. For $i \rightsquigarrow i+1$ we apply a case distinction w.r.t. R_i; note that R_i can neither be a model-checking counteraxiom by definition of model-checking proofs nor a model-checking axiom by construction. Moreover let z_{i+1} automatically be defined as the only successor of z_i iff R_i is not branching.

- *Cases* $R_i \equiv (MQ\vee), (ME\wedge), (MQl), (MEX), (MAE)$: Obvious.
- *Cases* $R_i \equiv (M\sigma), (MV)$: By Lemma 2.54.
- *Case* $R_i \equiv (MA\wedge)$: By Corollary 2.56.
- *Case* $R_i \equiv (MAX)$: Select z_{i+1} s.t. $\varsigma(z_{i+1}) = \xi_i(1)$ and set $\xi_{i+1} := \xi_i[1]$.
- *Case* $R_i \equiv (MAA)$: By Corollary 2.57.
- *Case* $R_i \equiv (MEQ)$: Let $\psi \in \Phi(i)$ be the principal formula. If there is a path completion χ of $APath_M(i)$ s.t. $\psi \in Cand^{\chi}_{\mathcal{B}_i}(i)$ set $\chi_{i+1} := \chi$ and choose z_{i+1} s.t. $\Psi(z_{i+1}) \neq \emptyset$; otherwise choose z_{i+1} s.t. $\Psi(z_{i+1}) = \emptyset$. If $R_i \equiv (MEA)$ finally apply Corollary 2.57.

This construction induces an infinite branch $\Xi : i \mapsto z_i$, a thread bundle $\mathcal{B} := \mathcal{B}_M(\Xi)$ as well as a falsifying thread bundle annotation $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ by setting

$$\pi_i := \begin{cases} \xi_i & \text{if } Q_i = A \\ \chi_{i+1} & \text{if } Q_i = E \wedge \exists j \in roots_{\mathcal{B}} : j > i \\ \xi^* & \text{if } Q_i = E \wedge \forall j \in roots_{\mathcal{B}} : j \leq i \end{cases}$$

where ξ^* is constructed as follows if there is $r \in roots_{\mathcal{B}}$ s.t. there is no $m \in roots_{\mathcal{B}}$ with $m > r$:
If $\{i \mid R(i) = (MQX)\}$ is an infinite set define

$$\xi^* : j \mapsto s_{\min\{k \geq r \mid j = pro_{\mathcal{B}}(root_{\mathcal{B}}(r), k)\}}$$

otherwise let $u := \min\{i \mid \forall j \geq i : R(j) \neq (MQX)\}$, $d := pro_{\mathcal{B}}(r, u)$ and define ξ^* to be an arbitrary path-completion of

$$\lambda.j \leq d \mapsto s_{\min\{k \geq r \mid j = pro_{\mathcal{B}}(root_{\mathcal{B}}(r), k)\}}$$

Note that $(\pi_i)_{i \in roots_{\mathcal{B}_M(\Xi)}}$ is even signature compatible falsifying by construction. □

Now it is not hard to show that our model-checking tableaux system is sound. We simply assume by contradiction that an LTS falsifies a formula on the one hand and that we have a model-checking proof on the other hand for the formula and the LTS in question. Then, by the former lemma, we conclude that there is a thread bundle in the tableau with a signature compatible falsifying annotation.

Hence the bundle needs to include a μ-thread equivalence class which is impossible due to the definition of model-checking proofs demanding to contain a ν-thread equivalence class.

Theorem 5.17 (Soundness)**.** *Let $\mathcal{T}$ be a transition system and $\varphi \in sCTL^*_{c\mu+}$ state-guarded. If $\mathcal{T} \vdash \varphi$ then $\mathcal{T} \models \varphi$.*

Proof. Let $\mathcal{T}$ be a transition system, $\varphi \in sCTL^*_{c\mu+}$ state-guarded and $\mathcal{T} \vdash \varphi$. By definition there is a model-checking-proof M for φ w.r.t. $\mathcal{T}$. By contradiction assume that $\mathcal{T} \not\models \varphi$, hence Lemma 5.16 claims that there is an infinite branch Ξ in M as well as a signature compatible falsifying thread bundle annotation $\pi = (\pi_i)_{i \in roots_{\mathcal{B}}}$ w.r.t. $\mathcal{B}$ with $\mathcal{B} := \mathcal{B}_M(\Xi)$.
By definition of model-checking-proofs there is a thread $s \in Th(\mathcal{B})$ s.t.

$$(*)\ [s]_{\equiv^A_{\mathcal{B}}} \subseteq Th_\nu(\mathcal{B})$$

Due to Corollary 5.10 thread bundle $\mathcal{B}$ is fair, therefore Lemma 3.44 comprises a thread $t \in Th(\mathcal{B})$ s.t.

$$(**)\ [t]_{\equiv^E_{\mathcal{B}}} \subseteq Th_\mu(\mathcal{B})$$

But due to Lemma 3.36 (*) and (**) cannot hold at the same time. □

5.3 Completeness

The completeness of the model-checking tableaux can be derived by the duality of model-checking proofs and the negative translation: Since each model-checking unproof for φ is canonically related to a model-checking proof for $\varphi^{\ominus}$

(and vice versa), the *correctness* result of the preceding chapter leads to the conclusion that a model-checking unproof indeed is a falsification witness.

Theorem 5.18 (Completeness)**.** *Let $\mathcal{T}$ be a transition system and $\varphi \in sCTL^*_{c\mu+}$ state-guarded. If $\mathcal{T} \nvdash \varphi$ then $\mathcal{T} \not\models \varphi$.*

Proof. Let $\mathcal{T}$ be a transition system, $\varphi \in sCTL^*_{c\mu+}$ state-guarded and $\mathcal{T} \nvdash \varphi$. By definition there is a model-checking-unproof M for φ w.r.t. $\mathcal{T}$. By Lemma 5.15 $\overline{M}$ is a model-checking-proof for $\varphi^{\ominus}$ w.r.t. $\mathcal{T}$.
Thus $\mathcal{T} \models \varphi^{\ominus}$ by Theorem 5.17, hence $\mathcal{T} \models \neg\varphi$ by Corollary 2.34 i.e. $\mathcal{T} \not\models \varphi$. □

It remains to show that model-checking tableaux fulfil the tertium-non-datur: Given a formula and a transition system there has to be (either) a model-checking proof or a model-checking unproof.

5.4 Effectiveness

Again, we solve two problems at once: We show that there is always (either) a model-checking proof or a model-checking unproof by the determinism result of parity games. The nodes of the model-checking parity game correspond to nodes in the model-checking tableaux and the winning strategies correspond to model-checking proofs or unproofs, respectively. Since parity games can be effectively solved, we get a decision procedure for CTL^*_{μ}-model-checking for free.

As with the deterministic rule selection of the proof system, we apply a similar reduction of the degree of freedom here.

Definition 5.19 (Deterministic rule selection)**.** Let $\varphi \in sCTL^*_{c\mu+}$ be state-guarded and $\mathcal{T} = (\mathcal{S}, \theta, \rightarrow, \lambda)$ be a transition system.
First, we induce a total ordering $<_R$ on *BundleRules* by $(MAl^*) >_R (MEtt) >_R (MEl^*) >_R (MAff) >_R (MAl) >_R (MEl) >_R (MA\vee) >_R (ME\wedge) >_R (M\nu) >_R (M\mu) >_R (MV) >_R (MA\wedge) >_R (ME\vee) >_R (MAA) >_R (MAE) >_R (MEA) >_R (MEE) >_R (MAX) >_R (MEX)$.

Second, we induce a total ordering $<$ on $BundleRules \times \mathcal{P}(Sub(\varphi))$ by

$$(R_1, M_1) < (R_2, M_2) : \Longleftrightarrow \quad R_1 <_R R_2 \vee (R_1 = R_2 \wedge M_1 \prec_{\mathcal{P}(Sub(\varphi))} M_2)$$

Third, we define a rule instantiation relation $\rightsquigarrow$ as follows:

$$(s, Q, \Lambda) \rightsquigarrow (\Phi, R, \{(s_1, Q_1, \Lambda_1, \Psi_1), ..., (s_n, Q_n, \Lambda_n, \Psi_n)\})$$

iff

$$(R): \frac{s_1 \vdash Q_1(\lceil \Psi_1 \rceil, \Lambda_1) \qquad s_2 \vdash \mathcal{Q}_2(\lceil \Psi_2 \rceil, \Lambda_2) \qquad ...}{s \vdash Q(\lfloor \Phi \rfloor, \Lambda \setminus \Phi)}$$

is a branching instantiation of R.

Note that $\{(s_1, Q_1, \Lambda_1, \Psi_1), ..., (s_n, Q_n, \Lambda_n, \Psi_n)\}$ is uniquely determined by s, Q, Λ, Φ and R, hence $F(s, Q, \Lambda) := \{((R, \Phi), M) \mid (s, Q, \Lambda) \rightsquigarrow (\Phi, R, M)\}$ is a function.

The *deterministic rule selection function* $detR^{\mathcal{T}}_{\varphi}$ finally is defined as follows:

$$detR^{\mathcal{T}}_{\varphi} : x \mapsto (max_{<} dom(F(x)), F(x)(max_{<} dom(F(x))))$$

The *model-checking game* is a parity game associated with a fixed formula and a fixed transition system. Its states are triples consisting of a state in the transition system, a formula block and a state of the deterministic thread bundle automaton. The transition relation of the game corresponds to the tableaux building options; the parity assignment function simply uses the parity assignment function of the thread bundle automaton.

Definition 5.20 (Model-Checking Game)**.** Let $\varphi \in sCTL^*_{c\mu+}$ be state-guarded and $\mathcal{T} = (\mathcal{S}, \theta, \rightarrow, \lambda)$ be a transition system. Let $\mathcal{A}^{\mathcal{BU}}_{\varphi}$ moreover be the thread bundle automaton of Corollary 3.50.

The *induced model-checking game w.r.t.* φ *and* $\mathcal{T}$, $G^{\mathcal{T}}_{\varphi} = (Pos_{\exists}, Pos_{\forall}, \delta, \Omega)$, is defined as follows:

- $Pos_G := \mathcal{S} \times Blo(\varphi) \times Q^{\mathcal{BU}}$ with
  - $Pos_{\exists} := \{(s, l, F, q) \in Pos_G \mid detR^{\mathcal{T}}_{\varphi}(s, l, F)$ is existential$\}$ whereas *existential* is defined as existentially branching, not branching or being a model-checking counteraxiom

  - $Pos_\forall := \{(s,l,F,q) \in Pos_G \mid detR^\mathcal{T}_\varphi(s,l,F) \text{ is universal}\}$ whereas *universal* is defined as universally branching or being a model-checking axiom

  i.e. iff $\mathcal{S}$ is finite, $G^\mathcal{T}_\varphi$ has $2^{O(n\cdot m\cdot\log(n\cdot m\cdot s))}$ states (where $n = |\varphi|$, $m = |Bound(\varphi)|$ and $s = |\mathcal{S}|$)

- $\delta : (s,l,F,q) \mapsto h(l,F,q,\Phi,R)[H]$ where $(\Phi,R,H) = detR^\mathcal{T}_\varphi(s,l,F)$ and

  $h : (l,F,q,\Phi,R,s',l',F',\Psi) \mapsto (s',l',F',\delta^{\mathcal{BU}}(q,(l,F,\Phi,\Psi,mrule(R,\Psi))))$

- $\Omega : (s,l,F,q) \mapsto \Omega^{\mathcal{BU}}(q)$

It is not hard to see that a model-checking proof corresponds to a winning strategy for $\exists$ in the model-checking game while a model-checking unproof corresponds to a winning strategy for $\forall$.

Theorem 5.21 (Model-Checking Game Property)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be state-guarded, $\mathcal{T}$ be a transition system and $G^\mathcal{T}_\varphi$ be the induced model-checking game. Then:*

$$\mathcal{T} \vdash \varphi \quad \text{if} \quad (\theta, E, \{\varphi\}, q_0^{\mathcal{BU}}) \text{ is won by player } \exists$$
$$\text{and}$$
$$\mathcal{T} \nvdash \varphi \quad \text{if} \quad (\theta, E, \{\varphi\}, q_0^{\mathcal{BU}}) \text{ is won by player } \forall$$

Proof. Let $(\theta, E, \{\varphi\}, q_0^{\mathcal{BU}})$ be won by player $\exists$ (the other case can be shown the same way) and let $s_\exists : W_\exists \cap Pos_\exists \to W_\exists$ be a positional winning strategy for player $\exists$. A model-checking-proof $M = (dom(M), \mathcal{Q}, \mathcal{L}, \varsigma, \mathcal{R}, \Phi, \Psi)$ is induced as follows:

- $dom(M) := T(Pos_G, (\theta, E, \{\varphi\}, q_0^{\mathcal{BU}}), \to)$ where

$$p_1 \to p_2 \;:\iff\; \begin{cases} p_2 \in \delta(p_1) & \text{if } p_1 \in Pos_\forall \\ p_2 = s_\exists(p_1) & \text{if } p_1 \in Pos_\exists \end{cases}$$

- $\mathcal{Q} : z \mapsto l$, $\mathcal{L} : z \mapsto F$ and $\varsigma : z \mapsto s$ where $(s,l,F,q) = Head(z)$

- $\mathcal{R} : z \mapsto R$ and $\Phi : z \mapsto P$ where $(s, l, F, q) = Head(z)$ and $(P, R, H) = detR^{\mathcal{T}}_{\varphi}(s, l, F)$
- $\Psi : z' \mapsto X$ where $(s', l', F', q') = Head(z')$, z is the immediate predecessor of z', $(s, l, F, q) = Head(z)$, $(P, R, H) = detR^{\mathcal{T}}_{\varphi}(s, l, F)$ and $(s', l', F', X) \in H$ with $\delta^{\mathcal{BU}}(q, (l, F, P, X, R)) = q'$

Consider that $\mathcal{M}$ is actually a pre-model-checking-proof; $\mathcal{M}$ moreover is a model-checking-proof since all finite branches end in model-checking axioms by definition of $detR^{\mathcal{T}}_{\varphi}$ and all infinite branches fulfil the model-checking-proof condition because each infinite branch resembles an accepting run of the thread bundle automaton of Corollary 3.50 and each branch is a fair (by Corollary 5.10) thread bundle (since $\mathcal{M}$ is a pre-model-checking-proof). □

In the preceding sections we have shown that

- Theorem 5.17: If there is a model-checking proof then the modelling relation holds
- Theorem 5.18: If there is a model-checking unproof then the modelling relation does not hold

Due the positional determinism of parity games (Theorem C.2) and Theorem 5.21, we know that there is either a model-checking proof or a model-checking unproof.

Combining these results we finally derive that the model-checking tableaux are sound and complete.

Corollary 5.22 (Model-Checking is sound and complete)**.** *Let $\varphi \in sCTL^{*}_{c\mu+}$ be state-guarded and $\mathcal{T}$ be a transition system. Then:*

$$\mathcal{T} \vdash \varphi \text{ if not and only if not } \mathcal{T} \nvdash \varphi$$

In other words:

$$\mathcal{T} \vdash \varphi \text{ iff } \mathcal{T} \models \varphi$$

By definition of the model-checking game and the application of a parity game solving decision procedure, say Theorem C.3, we derive the following decision complexity:

Theorem 5.23 (Decision complexity)**.** *Let $\varphi \in sCTL^*_{c\mu+}$ be state-guarded and $\mathcal{T}$ be a finite state transition system. Deciding whether $\mathcal{T} \models \varphi$ holds or not using the induced model-checking game works in space*

$$2^{O(nm \cdot \log(nms) \cdot \log(nm))}$$

and its running time is

$$2^{O\left(\log(nms) \cdot (n^2m^2 + nm \cdot \log(s) \cdot \log(nm))\right)}$$

where $n = |\varphi|$, $m = |Bound(\varphi)|$ and $s = |\mathcal{T}|$.

Therefore model-checking of CTL^*_μ is in $EXPTIME$. If the induced decision procedure would have been in $PSPACE$, we could directly derive its $PSPACE$-completeness due to the $PSPACE$-completeness of CTL^*-model-checking [SC82]. Unfortunately, we are neither able to find a decision procedure that runs in $PSPACE$ nor to prove that model-checking for CTL^*_μ is $EXPTIME$-complete.

5.5 Model language

We still need to prove that the Modal μ-Calculus is at least as expressive as $sCTL^*_\mu$: We show how to obtain an alternating tree automaton for a given state formula that accepts all these transition system satisfying the formula. Since alternating tree automata can be simulated by formulas of the Modal μ-Calculus (see Theorem B.13), we conclude that each state formula of CTL^*_μ can be translated into an equivalent formula of the Modal μ-Calculus. Please refer to Appendix B.12 for the definition of alternating tree automata.

The basic idea of the *model language automaton* we are going to construct in the following is to emulate a model-checking tableaux for the formula in question. The states of the automaton store the current block as well as the state of the corresponding bundle automaton that is performed on every branch in the tableaux in order to verify the global condition of each tableaux branch. The transition relation nondeterministically chooses an applicable model-checking rule and applies it to the current block and the associated bundle state.

Definition 5.24 (Model language automaton)**.** Let $\varphi \in sCTL^*_{c\mu+}$ be state-guarded and let $\mathcal{A}^{\mathcal{BU}}_\varphi$ be the thread bundle automaton of Corollary 3.50.
The *model language automaton w.r.t.* φ is an alternating tree automaton $\mathcal{A}l_\varphi = (Blo(\varphi) \times Q^{\mathcal{BU}}, (E, \{\varphi\}, q_0^{\mathcal{BU}}), \delta, \Omega : (\_, \_, q) \mapsto \Omega^{\mathcal{BU}}(q))$ where the transition function δ is defined as follows:

$$\delta(E, \emptyset, q) := tt \qquad \delta(A, \emptyset, q) := ff$$

For $\Lambda \equiv \{\bigcirc\psi_1, ..., \bigcirc\psi_n\}$ and $\Lambda' \equiv \{\psi_1, ..., \psi_n\}$:

$$\delta(E, \Lambda, q) := \Diamond(E, \Lambda', \delta^{\mathcal{BU}}(q, (E, \Lambda, \Lambda, \Lambda', (BX))))$$

$$\delta(A, \Lambda, q) := \Box(A, \Lambda', \delta^{\mathcal{BU}}(q, (A, \Lambda, \Lambda, \Lambda', (BX))))$$

For Λ with $\exists \psi \in \Lambda : \psi \not\equiv \bigcirc \psi'$:

$$\delta(Q, \Lambda, q) := \bigvee_{\psi \in \Lambda, \psi \not\equiv \bigcirc \psi'} \delta'(Q, \psi, \Lambda \setminus \{\psi\}, q)$$

with

$$\begin{aligned}
\delta'(E, l, \Lambda, q) :=& l \wedge (E, \Lambda, \delta^{\mathcal{BU}}(q, E, \Lambda, l, \emptyset, (BW))) \\
\delta'(A, l, \Lambda, q) :=& l \vee (A, \Lambda, \delta^{\mathcal{BU}}(q, A, \Lambda, l, \emptyset, (BW))) \\
\delta'(E, Q\psi, \Lambda, q) :=& (Q, \{\psi\}, \delta^{\mathcal{BU}}(q, (E, \Lambda \cup \{Q\psi\}, \{Q\psi\}, \{\psi\}, (BQ))) \wedge \\
& (E, \Lambda, \delta^{\mathcal{BU}}(q, E, \Lambda, Q\psi, \emptyset, (BW))) \\
\delta'(A, Q\psi, \Lambda, q) :=& (Q, \{\psi\}, \delta^{\mathcal{BU}}(q, (A, \Lambda \cup \{Q\psi\}, \{Q\psi\}, \{\psi\}, (BQ))) \vee \\
& (A, \Lambda, \delta^{\mathcal{BU}}(q, A, \Lambda, Q\psi, \emptyset, (BW))) \\
\delta'(Q, \sigma X.\psi, \Lambda, q) :=& (Q, \Lambda \cup \{X\}, \delta^{\mathcal{BU}}(q, Q, \Lambda, \sigma X.\psi, \{X\}, (B\sigma))) \\
\delta'(Q, X, \Lambda, q) :=& (Q, \Lambda \cup \{body_\varphi(X)\}, \\
& \delta^{\mathcal{BU}}(q, Q, \Lambda, X, \{body_\varphi(X)\}, (BV))) \\
\delta'(E, \psi_1 \wedge \psi_2, \Lambda, q) :=& (E, \Lambda \cup \{\psi_1, \psi_2\}, \delta^{\mathcal{BU}}(q, E, \Lambda, \psi_1 \wedge \psi_2, \{\psi_1, \psi_2\}, (BE\wedge))) \\
\delta'(A, \psi_1 \vee \psi_2, \Lambda, q) :=& (A, \Lambda \cup \{\psi_1, \psi_2\}, \delta^{\mathcal{BU}}(q, A, \Lambda, \psi_1 \vee \psi_2, \{\psi_1, \psi_2\}, (BA\vee))) \\
\delta'(E, \psi_1 \vee \psi_2, \Lambda, q) :=& (E, \Lambda \cup \{\psi_1\}, \delta^{\mathcal{BU}}(q, E, \Lambda, \psi_1, \{\psi_1\}, (BE\vee)))) \vee \\
& (E, \Lambda \cup \{\psi_2\}, \delta^{\mathcal{BU}}(q, E, \Lambda, \psi_2, \{\psi_2\}, (BE\vee))) \\
\delta'(A, \psi_1 \wedge \psi_2, \Lambda, q) :=& (A, \Lambda \cup \{\psi_1\}, \delta^{\mathcal{BU}}(q, A, \Lambda, \psi_1, \{\psi_1\}, (BA\wedge)))) \wedge \\
& (A, \Lambda \cup \{\psi_2\}, \delta^{\mathcal{BU}}(q, A, \Lambda, \psi_2, \{\psi_2\}, (BA\wedge)))
\end{aligned}$$

We finally show that the automaton accepts all these transition systems modelling the formula in question. Due to the fact that a run of the automaton corresponds to a pre-model-checking-proof and each accepting run to a model-checking-proof (and vice versa), it is easy to see that the automaton's language matches the modelling set.

Theorem 5.25 (Model language automaton accepts models)**.**
Let $\varphi \in sCTL^*_{c\mu+}$ be state-guarded and $\mathcal{A}l_\varphi$ be the model language automaton. Then:

$$Mod(\varphi) = \mathcal{L}(\mathcal{A}l_\varphi)$$

By soundness and completeness of model-checking-proofs in other words: For each LTS $\mathcal{T}$ holds that

there is an accepting run R of $\mathcal{A}l_\varphi$ on $\mathcal{T}$
iff
there is a model-checking-proof M for φ w.r.t. $\mathcal{T}$

Proof. Let $\varphi \in sCTL^*_{c\mu+}$ be state-guarded, $\mathcal{A}l_\varphi$ be the model language automaton and $\mathcal{T}$ be a transition system. Let $G^{\mathcal{T}}_\varphi$ moreover be the induced model-checking game.

It suffices to show that $(\theta, E, \{\varphi\}, q_0^{\mathcal{BU}})$ is won by player $\exists$ iff there is an accepting minimal run R of $\mathcal{A}l_\varphi$ on $\mathcal{T}$.
"$\Rightarrow$": Let $s_\exists : W_\exists \cap Pos_\exists \to W_\exists$ be a positional winning strategy for player $\exists$. An accepting run R of $\mathcal{A}l_\varphi$ on $\mathcal{T}$ is induced by

$$dom(R) := T(Pos_G, (\theta, E, \{\varphi\}, q_0^{\mathcal{BU}}), s_\exists \cup (\to \cap (W_\forall \times Pos_G)))$$

where $R : z \mapsto Head_{dom(R)}(z)$.
"$\Leftarrow$": For the conversion let R be a minimally accepting run of $\mathcal{A}l_\varphi$ on $\mathcal{T}$. A positional winning strategy $s_\exists : W_\exists \cap Pos_\exists \cap dom(R) \to W_\exists$ for player $\exists$ is induced as follows:

$$s_\exists : z \in W_\exists \cap Pos_\exists \cap dom(R) \mapsto R[Succ_{dom(R)}(z)]$$

which is welldefined by definition of δ and the minimality of R. □

6. CONCLUSION

We have introduced the logic CTL^*_μ which extends the full branching time logic CTL^* with arbitrary least and greatest fixed point operators on sets of paths. We have also considered a syntactical fragment of CTL^*_μ restricting the range of fixed point operators to path properties.

In order to guarantee some sort of fixed point fairness we have also taken a look at the guarded transformation of arbitrary CTL^*_μ-formulas. Unfortunately, we have only been able to provide a guarded transformation for the restricted fragment; nevertheless we assume that it should be possible to develop an effective guarded transformation for full CTL^*_μ. Additionally an accurate in-depth analysis of the guarded transformation's formula size blow-up remains to be done.

We have shown that the restricted fragment of CTL^*_μ is as expressive as $ECTL$ and therefore strictly more expressive than CTL^*. Regarding full CTL^*_μ, we have shown that its state formula fragment is as expressive as the Modal μ-Calculus. The succinctness of CTL^*_μ in comparison to the Modal μ-Calculus as well as with respect to CTL^* and $ECTL^*$ remains to be investigated in detail.

We have generalized the technique of using formula threads as analysis method for global temporal properties and investigated the abstract automata theoretic counterpart.

In this thesis, we have established a tableaux-style proof system for the validity of state formulas of the restricted fragment of CTL^*_μ. We have proved it sound and complete and have shown how to obtain counterexamples for non-valid formulas from failing proof attempts. Additionally we have sketched a decision procedure running in double exponential time based on the induced proof game. Last but not least we have concluded that this decision problem

is $2-EXPTIME$-complete.

Unfortunately, we have not been able to extend the proof system to full CTL^*_μ which should be possible assuming that a full guarded transformation is at hand: One needs to show that if a thread bundle at a position in an infinite branch has more than one prefix one can assume that one of these prefixes is always the best choice.

The model-checking tableaux system we have developed, however, is sound and complete for full CTL^*_μ. A decision procedure that runs in exponential time based on the induced model-checking game was outlined. It remains an open question whether the model-checking problem for CTL^*_μ is even $EXPTIME$-complete or if there is a refined decision procedure being in $PSPACE$ for instance.

A practical implementation of the decision procedures also remains to be done. This could be easily accomplished by generating the induced parity games and evaluate them with an optimized parity game solver such as the PGSolver Collection [FL08].

APPENDIX

A. TREES

There are many approaches to define trees, each of them having different advantages. For the purpose of this thesis, we define a tree to be the set of its nodes. A node itself is defined as the word of labellings on the path leading to the node in question. To formalise this, we consider a tree with respect to a set M providing the labellings. Hence a tree is a subset of M^* which needs to be closed under prefixes.

Definition A.1 (Trees)**.** A *tree T over M* is a prefix-closed subset of M^* i.e. for all $qi \in T$ with $q \in M^*$ and $i \in M$ holds that $q \in T$. Particularly the *root* ε is in T.

Define the following access maps w.r.t. a tree T over M:

- $Succ_T : T \to \mathcal{P}(T)$, $q \mapsto \{qi \in T \mid i \in M\}$
- $Lev_T : \mathbb{N} \to \mathcal{P}(T)$, $i \mapsto \{q \in T \mid |q| = i\}$
- $Head^r_T : T \to M \cup \{r\}$, $p \mapsto \begin{cases} r & \text{if } p = \varepsilon \\ s & \text{if } p = xs,\ s \in M \end{cases}$
- $Lift^r_T : M \cup \{r\} \times \mathbb{N} \to \mathcal{P}(T)$, $(s, i) \mapsto \{x \in Lev_T(i) \mid Head^r_T(x) = s\}$

A tree T is called *infinite* iff $|T| = \infty$ and *finitely branching* iff for all $q \in T$ holds that $|Succ_T(q)| < \infty$.

An *infinite path t in T* is a map $t : \mathbb{N} \to T$ with $|t_i| = i$ and $t_{i+1} \in Succ(t_i)$ for all $i \in \mathbb{N}$. The set of infinite paths w.r.t. T is denoted by $Path^\infty_T$.

Moreover define a partial order $\leq_T$ on T as follows

$$p \leq_T q \ :\iff \ \exists r \in M^* : pr = q$$

A set S along with a binary relation $\rightarrow$ on S induces trees in a natural way: Starting with a root $r \in S$ one connects r with all $s \in S$ satisfying $r \rightarrow s$ resulting in a tree. Proceeding with the leaves, this process is infinitely iterated and finally resulting in an infinite tree iff $\rightarrow$ is total w.r.t. r.

Definition A.2 (Induced trees)**.** Let S be a set, $r \in S$ and $\rightarrow \subseteq S \times S$. The *induced tree* $T(S, r, \rightarrow)$ *rooting in* r is defined as follows:

$$T(S, r, \rightarrow) := \bigcup_{i \in \mathbb{N}} T(S, r, \rightarrow)_i$$

with $T(S, r, \rightarrow)_0 := \{\varepsilon\}$, $T(S, r, \rightarrow)_1 := \{s \in S \mid r \rightarrow s\}$ and

$$T(S, r, \rightarrow)_{i+2} := \{xus \in S^{i+2} \mid x \in S^i,\, xu \in S^{i+1},\, u \rightarrow s\}$$

Note that $T(S, r, \rightarrow)$ is actually a tree.

The Lemma of König certainly is one of the most common propositions when handling infinite trees: If a tree has infinitely many nodes but is only finitely branching then there has to be an infinite path.

Lemma A.3 (Königs Lemma)**.** *Every finitely branching tree is infinite iff it has an infinite path.*

The next lemma again is combinatorial. It states that if a finitely branching infinite tree has infinitely many levels with marked nodes and additionally each properly branching node is marked, then there is an infinite path in the tree that is infinitely often marked.

Lemma A.4 (Tree level combinatorics)**.** *Let T be a finitely branching infinite tree and let $(A_i)_{i \in \mathbb{N}}$ be a family of sets with $A_i \subseteq Lev_T(i)$ for all $i \in \mathbb{N}$. There is an infinite path $t \in Path_T^\infty$ with $t_i \in A_i$ for infinitely many $i \in \mathbb{N}$ if the following two properties hold:*

1. *$A_i \neq \emptyset$ for infinitely many $i \in \mathbb{N}$*
2. *$|Succ_T(p)| > 1$ implies $p \in A_{|p|}$ for all $p \in T$*

Proof. Let T be a finitely branching infinite tree over M and let $(A_i)_{i\in\mathbb{N}}$ be a family of sets with $A_i \subseteq Lev_T(i)$ for all $i \in \mathbb{N}$. Define $D := \{p \in T \mid \exists r \in M^* : pr \in A_{|pr|}\}$ and note that D is a finitely branching infinite tree by condition *1* with $Path_D^\infty \subseteq Path_T^\infty$.

Königs Lemma (A.3) comprises a path $t \in Path_D^\infty$ with

$$\forall i \in \mathbb{N}\, \exists j > i \,:\, t_j \in A_j$$

which can be easily shown:

Let $i \in \mathbb{N}$ be arbitrary. By definition of D there is a path suffix r with $t_i r \in A_{|t_i r|}$; if $t_{i+|r|} = t_i r$ simply choose $j := i + |r|$. Otherwise let $j := i + \max\{h \mid \forall u \leq h : t_{i+u} = t_i r_0 r_1 ... r_{u-1}\}$ and note that $i \leq j < l$. Since t_j is branching we conclude by condition *2* that $t_j \in A_j$. □

B. AUTOMATA

We present three kinds of automata in this section. First, we introduce the classic finite-state infinite-word automata and present transformation as well as determinization results. Second, we define a specific language class, the *alternating languages*, providing a suitable abstraction for the solution of the word problem of thread bundles. Finally we shortly present alternating tree automata and compare their expressiveness to the Modal μ-Calculus.

Definition B.1 (Nondeterministic parity automaton)**.** A *nondeterministic parity automaton (NPA)* is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ where

1. Q is a finite set of states with $q_0 \in Q$ being the *initial state*
2. Σ is a finite alphabet set
3. $\delta : Q \times \Sigma \to 2^Q$ is the transition function
4. $\Omega : Q \to \mathbb{N}$ is the priority function

The *index* of $\mathcal{A}$ is the largest priority of $\mathcal{A}$, i.e. $\max(im(\Omega))$.
A *run* r of $\mathcal{A}$ on an infinite word $w \in \Sigma^\omega$ is a map $r : \mathbb{N} \to Q$ with $r(0) = q_0$ and $r(i+1) \in \delta(r(i), w_i)$ for all $i \in \mathbb{N}$. The set of runs of $\mathcal{A}$ on w is denoted by $Runs(\mathcal{A}, w)$.
A run r is called *accepting* iff $\max(\Omega[Inf(r)])$ is even, where

$$Inf(f : \mathbb{N} \to M) := \{m \in M \mid \forall i \, \exists j \geq i : \; f(j) = m\}$$

The subset of accepting runs of $\mathcal{A}$ on w is denoted by $AccRuns(\mathcal{A}, w)$.
The language that the NPA $\mathcal{A}$ *accepts*, $\mathcal{L}(\mathcal{A})$, is the set of all Σ^ω-words that have an accepting run in $\mathcal{A}$, i.e.

$$\mathcal{L}(\mathcal{A}) := \{w \in \Sigma^\omega \mid AccRuns(\mathcal{A}, w) \neq \emptyset\}$$

A *co-NPA* $\mathcal{A}$ has exactly the same signature as an NPA but the language that a co-NPA accepts is defined differently, namely as the set of all Σ^ω-words that only have accepting runs in $\mathcal{A}$, i.e.

$$\mathcal{L}(\mathcal{A}) := \{w \in \Sigma^\omega \mid AccRuns(\mathcal{A}, w) = Runs(\mathcal{A}, w)\}$$

An NPA $\mathcal{A}$ is called *deterministic (DPA)* iff

$$\forall q \in Q. \forall a \in \Sigma. |\delta(q, a)| \leq 1$$

The *transition image w.r.t.* $B \subseteq Q$ *and* $s \in \Sigma$, $\delta(B, s)$, is defined as follows:

$$\delta(B, s) := \bigcup_{q \in B} \delta(q, s)$$

We proceed with an obvious corollary stating that it is legal to decompose a run into infinitely many finite parts and apply the parity condition to the greatest priorities occurring in the finite segments.

Corollary B.2 (Covering Corollary)**.** *Let* $\mathcal{A}$ *be an NPA,* w *be a word,* r *be a run of* $\mathcal{A}$ *on* w *and let* $f : \mathbb{N} \to \mathbb{N}$ *be a map with* $f(i) < f(i+1)$ *for all* $i \in \mathbb{N}$ *and set* $p_i := \max\{\Omega(r(f(i))), ..., \Omega(r(f(i+1)-1))\}$*. Then* r *is accepting iff* $\max(Inf(\lambda i \mapsto p_i))$ *is even.*

A simplified version of the parity automaton using only two priorities is the *Büchi automaton*: These automata define which states are "good" in the sense that a run is accepting iff there is a good state occurring infinitely often.

Definition B.3 (Büchi automata)**.** A Büchi automaton is a simplified parity automaton that only uses two priorities: A *nondeterministic büchi automaton (NBA)* is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ where

1. Q, Σ, q_0 and δ as in Definition B.1
2. $F \subseteq Q$ is the set of final states

A run of $\mathcal{A}$ is called accepting iff there is a state $q \in F$ occurring infinitely often in the run; i.e. the priority function of the equivalent NPA is induced by:

$$\Omega(q) := \begin{cases} 2 & \text{if } q \in F \\ 1 & \text{if } q \notin F \end{cases}$$

Each parity automaton can be transformed into an equivalent Büchi automaton by a small polynomial blow-up. The Büchi automaton simply guesses the greatest priority occurring infinitely often and verifies that the priority indeed occurs infinitely often as well as no greater priority also does.

Lemma B.4 (NPA to NBA)**.** *Let $\mathcal{A}$ be an NPA with n states and index m. There is an equivalent NBA $\mathcal{B}$ with at most $n \cdot \lceil \frac{m}{2} \rceil$ states.*

Each nondeterministic Büchi automaton can be transformed into an equivalent *deterministic* parity automaton. The best approach is the determinization due to Piterman [Pit06].

Theorem B.5 (NBA to DPA [Pit06])**.** *Let $\mathcal{A}$ be an NBA with n states. There is an equivalent DPA $\mathcal{B}$ with at most n^{2n+2} states and at most index $2n-1$.*

Composing the Piterman determinization with the former construction we conclude that parity automata can be determinized:

Corollary B.6 (NPA to DPA)**.** *Let $\mathcal{A}$ be an NPA with n states and index m. There is an equivalent DPA $\mathcal{B}$ with at most $(n \cdot \lceil \frac{m}{2} \rceil)^{n\cdot(m+1)+2}$ states and at most index $n \cdot (m+1)$.*

The Piterman determinization can also be used to transform a co-parity automaton into a deterministic parity automaton.

Lemma B.7 (co-NPA to DPA)**.** *Let $\mathcal{A}$ be a co-NPA with n states and index m. There is an equivalent DPA $\mathcal{B}$ with at most $(n \cdot \lceil \frac{m+1}{2} \rceil)^{n\cdot(m+2)+2}$ states and at most index $n \cdot (m+2)+1$.*

The idea of an *alternating language* is motivated by the following observation: Consider a nondeterministic parity word automaton $\mathcal{A}$ and an infinite word w. All runs of $\mathcal{A}$ on w can be summarized in a memoryless run-tree; by *memoryless* we mean that two nodes on the same level having the same description bear the same sub tree.

Additionally assume that each symbol of the alphabet can be labelled by either $\exists$ or $\forall$ and that whenever the labelling changes at a certain position in

the word, either all or all but one branches in the corresponding run tree end. This condition is called *alternating language condition*.

The associated alternating language now accepts the word w iff there is a memoryless tree t *contained* in the full run-tree fulfilling the parity condition on each branch with the following additional property: The tree contains a single branch segment of the run-tree for all $\exists$-labelled segments of w and all branch segments of the run-tree for all $\forall$-labelled segments. The parity condition aside, it is not hard to see that every infinite run-tree contains at least one such memoryless tree due to the alternating language condition.

Such a tree contained in the full run-tree is a witness for some kind of alternation with respect to the runs of the given parity automaton on w: No matter which transitions are selected on $\forall$-labelled segments, there is a strategy to select transitions on $\exists$-labelled segments so that the resulting run is accepting.

The reason why we demand the alternating language condition is as follows: This language class was designed in order to accept thread bundles obviously fulfilling this condition by the path quantification rule. Without this restricting condition, we would need to implement full alternating parity automata which would finally result in exponentially greater deterministic parity automata.

Definition B.8 (Alternating language)**.** Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ be a parity automaton and $\ell : \Sigma \to \{\exists, \forall\}$ be a labelling map s.t. the following holds:

$$\forall s, t \in \Sigma : \ (\ell(s) \neq \ell(t) \Rightarrow |\delta(\delta(Q, s), t)| \leq 1)$$

This condition is called *alternating language condition*.
The *alternating language w.r.t.* $\mathcal{A}$ *and* ℓ, $\mathcal{L}(\mathcal{A}, \ell)$, is defined as follows:

$$\mathcal{L}(\mathcal{A}, \ell) := \{w \in \Sigma^\omega \mid \exists r \in \mathit{Runs}(\mathcal{A}, w) : \ [r]_{\equiv^w_{(\mathcal{A},\ell)}} \subseteq \mathit{AccRuns}(\mathcal{A}, w)\}$$

where $\equiv^w_{(\mathcal{A},\ell)}$ is an equivalence relation defined as follows:

$$r \equiv^w_{(\mathcal{A},\ell)} t \ : \iff \ \forall i \in \mathbb{N} : (\ell(w_i) = \exists \Rightarrow r_{i+1} = t_{i+1})$$

A word $w \in \mathcal{L}(\mathcal{A}, \ell)$ is

- *truly alternating* iff $\forall i \in \mathbb{N}\, \exists j > i : \ell(w_i) \neq \ell(w_j)$
- *finally existential* iff $\exists i \in \mathbb{N}\, \forall j \geq i : \ell(w_j) = \exists$
- *finally universal* iff $\exists i \in \mathbb{N}\, \forall j \geq i : \ell(w_j) = \forall$

The subset of all truly alternating / finally existential / finally universal words w.r.t. $\mathcal{L}(\mathcal{A}, \ell)$ is denoted by $\mathcal{L}(\mathcal{A}, \ell)_\infty$ / $\mathcal{L}(\mathcal{A}, \ell)_\exists$ / $\mathcal{L}(\mathcal{A}, \ell)_\forall$.
Note that

$$\mathcal{L}(\mathcal{A}, \ell) = \mathcal{L}(\mathcal{A}, \ell)_\infty \dot\cup \mathcal{L}(\mathcal{A}, \ell)_\exists \dot\cup \mathcal{L}(\mathcal{A}, \ell)_\forall$$

A word $w \in \Sigma^\omega$ moreover is called *finally stationary* iff w is either finally existential or finally universal.

It is our goal to construct a deterministic parity automaton accepting alternating languages. Observe that there are basically three kinds of words that can be accepted: Truly alternating words, finally existential words and finally universal words. All three kinds require different methods to determine whether a word is to be accepted or rejected.

The construction follows a three step procedure: The first lemma creates a deterministic parity automaton accepting finally stationary words, the second lemma constructs a DPA accepting truly alternating words and finally the last proposition puts both DPA together in order to receive a deterministic parity automaton accepting alternating languages.

A DPA accepting finally stationary words can be constructed as follows with respect to a given NPA $\mathcal{A}$: First, build two DPAs one being the determinization of $\mathcal{A}$ and the other being the co-determinization of $\mathcal{A}$. The first automaton obtained in this manner accepts finally existential words and the second one accepts finally universal words.

Our construction thus runs as follows: Both automata are simulated simultaneously the whole time. Whenever the labelling is existential, the priority map of the existential automaton is simulated and whenever it is universal the universal priority map is used. To reject truly alternating words we pick an

odd priority that is greater than all other priorities occurring and return it whenever the labelling in a word changes.

Lemma B.9 (Finally stationary alternating language to DPA)**.** *Let $\mathcal{L}(\mathcal{A}, \ell)$ be an alternating language s.t. $\mathcal{A}$ is an NPA with n states and index m. There is a DPA $\mathcal{A}'$ with at most $4 \cdot (n \cdot \lceil \frac{m+1}{2} \rceil)^{n \cdot (2 \cdot m+3)+4}$ states and at most index $n \cdot (m+2)+3$ s.t. $\mathcal{L}(\mathcal{A}, \ell)_\exists \dot\cup \mathcal{L}(\mathcal{A}, \ell)_\forall = \mathcal{L}(\mathcal{A}')$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ be an NPA with n states and index m and $\mathcal{L}(\mathcal{A}, \ell)$ be an alternating language.

Let $\mathcal{A}_\exists$ be a total NPA determinization of $\mathcal{A}$ (Corollary B.6) with $n_\exists$ states and index $m_\exists$. Let moreover $\mathcal{A}_\forall$ be a total co-NPA determinization of $\mathcal{A}$ (Lemma B.7) with $n_\forall$ states and index $m_\forall$.
The DPA $\mathcal{A}' = (Q', \Sigma, q_0', \delta', \Omega')$ is now defined as follows:

- $Q' = \{\exists, \forall\} \times \{0, 1\} \times Q_\exists \times Q_\forall$ i.e. Q' has $n' := 4 \cdot n_\exists \cdot n_\forall$ states
- $q_0' := (\exists, 1, q_0^\exists, q_0^\forall)$
- $\delta' : (l, k, q^\exists, q^\forall), s \mapsto (\ell(s), \mathbb{1}_{\ell(s) \neq l}, \delta^\exists(q^\exists), \delta^\forall(q^\forall))$
- The priority assignment function Ω' with index $m' := m^* + 1$ where $m^* := 2 \cdot \lfloor \frac{\max(m_\exists, m_\forall)}{2} \rfloor$ is defined as follows:

$$\Omega' : (l, k, q^\exists, q^\forall) \mapsto \begin{cases} m' & \text{if } k = 1 \\ \Omega_\exists(q^\exists) & \text{if } k = 0 \text{ and } l = \exists \\ \Omega_\forall(q^\forall) & \text{if } k = 0 \text{ and } l = \forall \end{cases}$$

Note that if a word w is truly alternating then the highest priority occurring infinitely often in the run of $\mathcal{A}'$ on w is m' which is odd; otherwise, if w is not alternating, the respective automaton $\mathcal{A}_\exists$ / $\mathcal{A}_\forall$ is continuously simulated after a finite point at which the alternation stops. □

The following lemma is the key part of our construction: A deterministic parity automaton accepting all truly alternating words of an alternating language. The basic idea is to simulate the full run-tree and decompose it into segments with the same labelling.

Each segment contains *best* and *worst subruns*. A *subrun* is a path going through the segment in question which is not terminated neither at any point in the segment nor at the end of the segment. All other finite paths are not of any interest. One subrun is better than the other iff its greatest priority gives a greater *reward* than the greatest priority of the other subrun where the *reward order* is defined as follows: $p_1 \preceq p_2 \ :\Longleftrightarrow \ \ rew(p_1) \leq rew(p_2)$ with $rew(p) = p$ for all even p and $rew(p) = -p$ for all odd p.

Now, the priority assignment function only outputs a "real" priority at the end of each segment with the following associated meaning: If the labelling of the segment was existential, the assigned priority is the greatest priority occurring in a best subrun. If the labelling of the segment was universal, the assigned priority similarly is the greatest priority occurring in a worst subrun.

By assigning priorities in this manner, the automaton ensures to accept all these truly alternating words that are able to provide a "good" decision in all existential segments and to cope with all "bad" paths through the universal segments.

Technically, we simulate each *state level* in the memoryless run-tree and assign each state in a level a priority that is updated by the transition relation as follows: A state q occurring in the next level gets its priority assigned by combining the priority p of the state in question and all assigned priorities $p_1, ..., p_k$ of all of its predecessors. If the segment is existentially labelled the best priority p^* (w.r.t. the reward order) out of $p_1, ..., p_k$ is selected; otherwise, if the segment is universally labelled, the worst priority p^* out of $p_1, ..., p_k$ is selected. The assigned priority for the state in question finally is $\max(p, p^*)$.

Lemma B.10 (Truly alternating language to DPA)**.** *Let $\mathcal{L}(\mathcal{A}, \ell)$ be an alternating language with $\mathcal{A}$ being an NPA with n states and index m. There is a DPA $\mathcal{A}'$ with at most $4 \cdot (m+1)^n$ states and at most index $m+2$ s.t. $\mathcal{L}(\mathcal{A}, \ell)_\infty = \mathcal{L}(\mathcal{A}')$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ be an NPA with n states and index m and $\mathcal{L}(\mathcal{A}, \ell)$ be an alternating language.
The DPA $\mathcal{A}' = (Q', \Sigma, q_0', \delta', \Omega')$ is constructed as follows:

- $Q' := \{\exists, \forall\} \times \{0, 1\} \times F$ with $F := (im(\Omega) \cup \{\bot\})^Q$ i.e. Q' has $4 \cdot (m+1)^n$ states; for $f \in F$ we define the domain $dom'(f) := \{q \in Q \mid f(q) \neq \bot\}$ as well as the range $ran'(f) := im(f) \cap im(\Omega)$.
  We moreover identify each partial $f : Q \to im(\Omega)$ with the total closure mapping each $q \notin dom(f)$ to $\bot$.

- $q_0' := (\exists, 1, \{(q_0, \Omega(q_0))\})$

- The deterministic transition function δ' is a bit complicated due to the fact that merging states of $\mathcal{A}$ have to be treated in a smart way:

  $$\delta' : (l, k, f), s \mapsto (\ell(s), \mathbb{1}_{\ell(s) \neq l}, g(s, k, f))$$

  where

  $$g : (s, k, f) \mapsto \begin{cases} \lambda q \in \delta(dom'(f), s).\, \Omega(q) & \text{if } k = 1 \\ \lambda q.\, h(s, \{\max(\Omega(q), p) \mid p \in T(f, s, q)\}) & \text{otherwise} \end{cases}$$

  $$T : (f, s, q) \mapsto \{f(q') \in ran'(f) \mid \exists q' \in dom'(f) :\ q \in \delta(q', s)\}$$

  and

  $$h : (s, P) \mapsto \begin{cases} \bot & \text{if } P = \emptyset \\ \max_{\preceq} P & \text{if } \ell(s) = \exists \\ \min_{\preceq} P & \text{if } \ell(s) = \forall \end{cases}$$

  where $p_1 \preceq p_2 :\iff rew(p_1) \leq rew(p_2)$ with $rew(p) = p$ for all even p and $rew(p) = -p$ for all odd p.

- The priority function Ω' with index $m + 2$ is defined as follows:

  $$\Omega' : (l, k, f) \mapsto \begin{cases} f(q) + 2 & \text{if } dom'(f) = \{q\} \text{ and } k = 1 \\ 1 & \text{otherwise} \end{cases}$$

Obviously $\mathcal{A}'$ is total and deterministic. Let now $w \in \Sigma^\omega$ be arbitrary and let $R \in Runs(\mathcal{A}', w)$ be the only run of $\mathcal{A}'$ on w.

We observe that R is not accepting if w is not truly alternating since only alternating transitions possibly yield even priorities. Thus w.l.o.g. let w be truly alternating.

Let $R' : i \mapsto R(i)_3$. First, we show the following property of R' by induction on $i \in \mathbb{N}$:

$$dom'(R'(i)) = \{r(i) \mid r \in Runs(\mathcal{A}, w)\} \tag{B.1}$$

For $i = 0$ this is obvious; for $i \rightsquigarrow i + 1$ we apply a case distinction on whether $R(i)_2 = 1$ or not.

Case $R(i)_2 = 1$: Then the following holds:

$$\begin{aligned} dom'(R'(i+1)) &= g(w_i, 1, R'(i)) \\ &= \delta(dom'(R'(i)), w_i) \\ &\overset{IH}{=} \delta(\{r(i) \mid r \in Runs(\mathcal{A}, w)\}, w_i) \\ &= \{r(i+1) \mid r \in Runs(\mathcal{A}, w)\} \end{aligned}$$

Case $R(i)_2 = 0$: Then the following holds

$$\begin{aligned} dom'(R'(i+1)) &= g(w_i, 0, R'(i)) \\ &= \{q \mid T(R'(i), w_i, q) \neq \emptyset\} \\ &= \{q \mid q \in \delta(dom'(R'(i)), w_i)\} \\ &\overset{IH}{=} \delta(\{r(i) \mid r \in Runs(\mathcal{A}, w)\}, w_i) \\ &= \{r(i+1) \mid r \in Runs(\mathcal{A}, w)\} \end{aligned}$$

Particularly this implies that if $Runs(\mathcal{A}, w)$ is empty, R is also not accepting; thus w.l.o.g. let $Runs(\mathcal{A}, w) \neq \emptyset$.

For the next properties to show, we need to define the alternating set w.r.t. w:

$$Alt := \{i \in \mathbb{N} \mid i = 0 \vee (i > 0 \wedge \ell(w_i) \neq \ell(w_{i-1}))\}$$

Note that Alt is infinite since w is truly alternating; thus the successor access function is welldefined: $su : i \in Alt \mapsto \min\{j \in Alt \mid j > i\}$.

Also note that - by definition of alternating languages - $|dom'(R(i))| = 1$ for all $i \in Alt$; thus the state access function $altq : i \in Alt \mapsto q_i$ with $dom'(R(i)) = \{q_i\}$ is welldefined.

Moreover define the priority descriptor $pr : Runs(\mathcal{A}, w) \times Alt \times \mathbb{N} \to \mathbb{N}$ as follows:

$$pr : (r, i, j) \mapsto \max(\Omega[\{r(i), ..., r(j-1)\}])$$

Now we are able to show the following properties for all $i \in Alt$, all $i < j \leq su(i)$ and all $q \in dom'(R'(j))$:

$$\ell(w_i) = \exists \Rightarrow max_{\preceq} S(i, j, q) = R'(j)(q) \tag{B.2}$$

$$\ell(w_i) = \forall \Rightarrow min_{\preceq} S(i, j, q) = R'(j)(q) \tag{B.3}$$

where $S(i, j, q) := \{pr(r, i, j) \mid r \in Runs(\mathcal{A}, w) \wedge r(j) = q\}$.

Let $i \in Alt$ as well as $op_{\preceq} \in \{\max_{\preceq}, \min_{\preceq}\}$ be arbitrary; now induction on j.
For $j = i + 1$ consider the following:

$$\begin{aligned}
R'(j)(q) &= R'(i+1)(q) \\
&= g(w_i, 1, R'(i))(q) \\
&= (\lambda q' \in \delta(dom'(R'(i)), w_i).\Omega(q'))(q) \\
&= (\lambda q' \in dom'(R'(i+1)).\Omega(q'))(q) \\
&= \Omega(q) \\
&= op_{\preceq} S(i, i+1, q) \\
&= op_{\preceq} S(i, j, q)
\end{aligned}$$

For $j \rightsquigarrow j+1 \leq su(i)$ consider the following:

$$\begin{aligned} op_{\preceq} S(i, j+1, q) &= op_{\preceq}\{\max(\Omega(q), op_{\preceq} S(i, j, q')) \mid q \in \delta(q', w_j) \wedge \\ &\qquad\qquad S(i, j, q') \neq \emptyset\} \\ &\overset{IH}{=} op_{\preceq}\{\max(\Omega(q), R'(j)(q')) \mid q \in \delta(q', w_j) \wedge \\ &\qquad\qquad q' \in dom'(R'(j))\} \\ &= op_{\preceq}\{\max(\Omega(q), p) \mid p \in T(R'(j), w_j, q)\} \\ &= h(\ell(w_j), \{\max(\Omega(q), p) \mid p \in T(R'(j), w_j, q)\})(q) \\ &= g(w_j, 0, R'(j))(q) \\ &= R'(j+1)(q) \end{aligned}$$

Both properties B.2 and B.3 particularly imply that the following two properties hold for all $i \in Alt$:

$$\ell(w_i) = \exists \Rightarrow max_{\preceq} S(i, su(i), altq(i)) = \Omega'(R(su(i)) - 2 \tag{B.4}$$

$$\ell(w_i) = \forall \Rightarrow min_{\preceq} S(i, su(i), altq(i)) = \Omega'(R(su(i)) - 2 \tag{B.5}$$

Now consider the following two maps extracting the most significant priorities w.r.t. the non-alternating sections

$$runp : Runs(\mathcal{A}, w) \times Alt \to \mathbb{N},\ (r, i) \mapsto pr(r, i, su(i))$$

$$altp : Alt \to \mathbb{N},\ i \mapsto \Omega'(R(su(i))) - 2$$

and observe that by B.4 and B.5 the following holds for all $i \in Alt$ and all runs $r \in Runs(\mathcal{A}, w)$:

$$\ell(w_i) = \exists \Rightarrow runp(r, i) \preceq altp(i) \tag{B.6}$$

$$\ell(w_i) = \forall \Rightarrow runp(r, i) \succeq altp(i) \tag{B.7}$$

By definition of alternating languages and by properties B.4 and B.5 there is a run $T \in Runs(\mathcal{A}, w)$ with

$$\forall i \in \mathbb{N} :\ runp(T, i) = altp(i) \tag{B.8}$$

By the Covering Corollary (Corollary B.2) moreover holds for all runs $r \in Runs(\mathcal{A}, w)$ that

$$r \text{ is accepting iff } \max(Inf(\lambda(i \in Alt).runp(r, i))) \text{ is even}$$

Similarly holds that

$$R \text{ is accepting iff } \max(Inf(\lambda(i \in Alt).altp(i))) \text{ is even}$$

After all these preparations we are now able to show that

$$\exists r \in Runs(\mathcal{A}, w) : [r]_{\equiv^w_{(\mathcal{A},\ell)}} \subseteq AccRuns(\mathcal{A}, w) \text{ iff } T \text{ is acceping}$$

which completes our proof (since R is accepting iff T is accepting).
"Only-if": Let $r \in Runs(\mathcal{A}, w)$ with $[r]_{\equiv^w_{(\mathcal{A},\ell)}} \subseteq AccRuns(\mathcal{A}, w)$. Construct a run $s \in Runs(\mathcal{A}, w)$ as follows:

$$s : i \mapsto \begin{cases} r(i) & \text{if } \ell(w_i) = \exists \\ T(i) & \text{if } \ell(w_i) = \forall \end{cases}$$

Then, since $r \equiv^w_{(\mathcal{A},\ell)} s$, s is accepting w.r.t. $\mathcal{A}$. But then particularly T is accepting since $runp(s, i) \preceq runp(T, i)$ for all $i \in \mathbb{N}$.
"If": Let T be accepting w.r.t. $\mathcal{A}$ and let $r \in Runs(\mathcal{A}, w)$ with $T \equiv^w_{(\mathcal{A},\ell)} r$ be arbitrary. Then particularly r is accepting since $runp(T, i) \preceq runp(r, i)$ for all $i \in \mathbb{N}$. □

The combination of the two parity automata from above finally results in a deterministic parity automaton accepting alternating languages. Although the whole construction seems to be a very complicated way to achieve this goal, an easier approach using more general well-known construction principles results in exponentially worse automata sizes.

For instance, building firstly an alternating parity automaton accepting alternating languages, secondly simulating this automaton by an NBA and finally transforming the NBA into a DPA leads to an overall size of roughly $2^{O(n^m \cdot 2^{2 \cdot n^m})}$ with index $2^{O(n^p)}$.

The combination of both DPAs from above runs as follows: Both automata are simultaneously simulated. Let m be a fixed but arbitrary even priority that is greater than all priorities occurring in the finally stationary DPA. Whenever

there is an alternation in the processed word a special priority $p+m$ is returned where p is the greatest priority occurring in the simulated run of the truly alternating DPA between the current position and the last alternating position.

If there is no alternation at the current position, the priority assignment of the finally stationary DPA is applied. The resulting DPA now accepts the alternating language: If a word is finally stationary the priority assignment function returns priorities not obtained by the assignment function of the finally stationary automaton only finitely often. Otherwise, if a word is truly alternating, the respective automaton infinitely often returns the greatest priority of the corresponding segment and thus handles the truly alternating word correctly by the Covering Corollary B.2.

Theorem B.11 (Alternating language to DPA)**.** *Let $\mathcal{L}(\mathcal{A}, \ell)$ be an alternating language s.t. $\mathcal{A}$ is an NPA with n states and index m. There is a DPA $\mathcal{A}'$ with at most $64 \cdot (n \cdot \lceil \frac{m+1}{2} \rceil)^{n \cdot (2 \cdot m+4)+5}$ states and at most index $(n+1) \cdot (m+2)+4$ s.t. $\mathcal{L}(\mathcal{A}, \ell) = \mathcal{L}(\mathcal{A}')$.*

Proof. Let $\mathcal{L}(\mathcal{A}, \ell)$ be an alternating language s.t. $\mathcal{A}$ is an NPA with n states and index m.
Let $\mathcal{A}_{\exists\forall}$ be a total DPA with $m_{\exists\forall}$ states and index $m_{\exists\forall}$ s.t.

$$\mathcal{L}(\mathcal{A}, \ell)_{\exists} \dot{\cup} \mathcal{L}(\mathcal{A}, \ell)_{\forall} = \mathcal{L}(\mathcal{A}_{\exists\forall})$$

which exists due to Lemma B.9.
Let $\mathcal{A}_{\infty}$ be a total DPA with n_{∞} states and index m_{∞} s.t.

$$\mathcal{L}(\mathcal{A}, \ell)_{\infty} = \mathcal{L}(\mathcal{A}_{\infty})$$

which exists due to Lemma B.10.

The DPA $\mathcal{A}' = (Q', \Sigma, q_0', \delta', \Omega')$ is constructed as follows:

- $Q' := \{\exists, \forall\} \times \{0,1\} \times \{0, ..., m_\infty\} \times Q_{\exists\forall} \times Q_\infty$ i.e. Q' has $n' := 4 \cdot m_\infty \cdot n_{\exists\forall} \cdot n_\infty$ states
- $q_0' := (\exists, 1, \Omega_\infty(q_0^\infty), q_0^{\exists\forall}, q_0^\infty)$
- $\delta' : (l, k, p, q^{\exists\forall}, q^\infty), s \mapsto (\ell(s), \mathbb{1}_{\ell(s)\neq l}, p', \delta^{\exists\forall}(q^{\exists\forall}), \delta^\infty(q^\infty))$ where

$$p' := \begin{cases} \max(p, \Omega_\infty(q^\infty)) & \text{if k} = 0 \\ \Omega_\infty(q^\infty) & \text{otherwise} \end{cases}$$

- The priority assignment function Ω' with index $m' := m^* + m_\infty$ where $m^* := 2 \cdot \lceil \frac{m_{\exists\forall}}{2} \rceil$ is defined as follows:

$$\Omega' : (l, k, p, q^{\exists\forall}, q^\infty) \mapsto \begin{cases} p + m^* & \text{if k} = 1 \\ \Omega_{\exists\forall}(q^{\exists\forall}) & \text{otherwise} \end{cases}$$

It remains to show that $\mathcal{L}(\mathcal{A}, \ell) = \mathcal{L}(\mathcal{A}')$ holds; hence let $w \in \Sigma^\omega$ be arbitrary and let $r' \in Runs(\mathcal{A}', w)$ be the (only) run of $\mathcal{A}'$ on w.

If w is finally stationary by definition of δ' holds that $\exists i \in \mathbb{N} \forall j \geq i : r'(j)_2 = 0$ i.e. $\exists i \in \mathbb{N} \forall j \geq i : \Omega'(r'(j)) = \Omega_{\exists\forall}(r'(j)_4)$ and therefore $w \in \mathcal{L}(\mathcal{A}')$ iff $w \in \mathcal{L}(\mathcal{A}, \ell)_\exists \dot\cup \mathcal{L}(\mathcal{A}, \ell)_\forall$.

If otherwise w is truly alternating then by definition of δ' holds that $\forall i \in \mathbb{N} \exists j \geq i : r'(j)_2 = 1$ i.e. $\forall i \in \mathbb{N} \exists j \geq i : \Omega'(r'(j)) = r'(j)_3 + m^*$. Hence we need to show that $w \in \mathcal{L}(\mathcal{A}_\infty)$ iff r' is accepting which follows by Corollary B.2. □

We note that one can reduce the prefactor 64 down to 4 by a refined construction sharing the two "information bits" $\{\exists, \forall\} \times \{0, 1\}$.

We briefly present a simplified version of the alternating tree automaton presented in [Wil01]. Most importantly, the transition conditions of this automaton can contain states not occurring under any modality which is particularly useful for following model-checking tableaux branches with only finitely many applications of the modal rule.

Definition B.12 (Alternating tree automaton)**.** An *alternating tree automaton* is a tuple $\mathcal{A} = (Q, q_0, \delta, \Omega)$ where

1. Q is a finite set of states with $q_0 \in Q$ being the *initial state*
2. $\delta : Q \to Tr_Q$ is the transition function
3. $\Omega : Q \to \mathbb{N}$ is the priority function

where the *set of transition conditions* Tr_Q is defined as follows:

$$Tr_Q := \mathcal{L}(tr_Q)$$

with

$$tr_Q ::= \texttt{tt} \mid \texttt{ff} \mid tr_Q \wedge tr_Q \mid tr_Q \vee tr_Q \mid Q \mid \Box Q \mid \Diamond Q \mid \mathcal{P} \mid \neg \mathcal{P}$$

where Q and $\mathcal{P}$ stand for "any element" out of the respective set.

In this context, alternating automata are interpreted over labelled transitions systems. A *run* R of $\mathcal{A}$ on an LTS $\mathcal{T} = (\mathcal{S}, \theta, \to, \lambda)$ is a map $R : T \to Q \times \mathcal{S}$ with T being a tree s.t. the following holds:

- Initial condition: $R(\varepsilon) = (q_0, \theta)$
- Local consistency: For every $v \in T$ holds that $\mathcal{T}, R, v \models \delta(R(v)_1)$

whereas

$$\mathcal{T}, R, v \models \texttt{tt} \qquad \text{and} \qquad \mathcal{T}, R, v \not\models \texttt{ff}$$

$$\mathcal{T}, R, v \models t_1 \wedge t_2 \iff \mathcal{T}, R, v \models t_1 \text{ and } \mathcal{T}, R, v \models t_2$$

$$\mathcal{T}, R, v \models t_1 \vee t_2 \iff \mathcal{T}, R, v \models t_1 \text{ or } \mathcal{T}, R, v \models t_2$$

$$\mathcal{T}, R, v \models q \iff (q, R(v)_2) \in R[Succ_T(v)]$$

$$\mathcal{T}, R, v \models \Diamond q \iff \exists s \in \mathcal{S} : [(R(v)_2 \to s) \wedge (q, s) \in R[Succ_T(v)]]$$

$$\mathcal{T}, R, v \models \Box q \iff \forall s \in \mathcal{S} : [(R(v)_2 \to s) \Rightarrow (q, s) \in R[Succ_T(v)]]$$

$$\mathcal{T}, R, v \models p \iff p \in \lambda(R(v)_2)$$

$$\mathcal{T}, R, v \models \neg p \iff p \notin \lambda(R(v)_2)$$

A run R is *accepting* iff each infinite path in R fulfils the standard parity condition w.r.t. Ω. R is *minimally accepting* iff for every accepting $R' \subseteq R$ it holds that $R' = R$. Clearly, if R is accepting then there is also a minimally accepting run.

Alternating tree automata can be simulated by formulas of the Modal μ-Calculus. This helps us to show that $sCTL^*_\mu \leq \mathcal{L}_\mu$ by building an alternating tree automaton accepting the same models as a given formula of $sCTL^*_\mu$.

Theorem B.13 (From alternating automata to the Modal μ-Calculus, [Niw97])**.** *Let $\mathcal{A}$ be an alternating tree automaton. There is a closed formula φ of the Modal μ-Calculus s.t. $Mod(\varphi) = \mathcal{L}(\mathcal{A})$.*

C. PARITY GAMES

A parity game is played by two players moving a token in a directed graph. Every node in the game graph is given an integral priority that is associated with either one of the players. A play in a parity game is the sequence of nodes the token visited. Such a play is won by one of the two players depending on the priorities occurring infinitely often.

Definition C.1 (Parity games)**.**
A *parity game* is a quadruple $G = (Pos_\exists, Pos_\forall, \delta, \Omega)$ where

- $Pos_\exists \cap Pos_\forall = \emptyset$ are the disjoint sets of player positions
- $Pos_G := Pos_\exists \cup Pos_\forall$ is the position set
- $\delta \subseteq Pos_G \times Pos_G$ is the transition relation
- $\Omega : Pos_G \mapsto \mathbb{N}$ is the priority function

A *play* π *starting in* $p_0 \in Pos_G$ is a finite or infinite sequence of positions $p_0, p_1, ...$ s.t. $\forall i \in \mathbb{N} : (p_i, p_{i+1}) \in \delta$.

The *player* $\exists$ *wins* a play π iff either π is of finite length and the last position is in $Pos_\forall$ or π is of infinite length and the greatest priority occurring infinitely often in π is even. Likewise $\forall$ wins a play π iff either π is finite and ends in a position of $\exists$ or π is infinite and the greatest priority occurring infinitely often is odd.

A *positional strategy for player* $P \in \{\exists, \forall\}$ is a partial map $s : Pos_P \to Pos_G$. A play π starting in $dom(s)$ is *s-compliant* iff $s(\pi_i) = \pi_{i+1}$ for all i with $\pi_i \in dom(s)$.

A positional strategy is a *positional winning strategy* iff for all s-compliant plays π starting in $dom(s)$ holds that π is won by P.

A position $p \in Pos_G$ is a *winning position for player* P iff there is a positional winning strategy s s.t. each s-compliant play π starting in p is won by player P.

The *positional determinism* is a crucial property of parity games stating that the whole set of nodes can be partitioned into the winning sets for both player.

Theorem C.2 (Positional determinism of parity games, [Mar75, GH82, EJ91])**.** *Let $G = (Pos_\exists, Pos_\forall, \delta, \Omega)$ be a parity game. Then $W_\exists \cup W_\forall = Pos$ holds and there are positional winning strategies for $\exists$ and for $\forall$.*

The problem of *solving* a (finite) parity game is to compute the winning sets of both players along with the winning strategies on the respective winning sets. It is known to be in $NP \cap co - NP$, even in $UP \cap co - UP$ [Jur98].

There are several algorithms for solving parity games leading to slightly different running times. A very efficient algorithm for solving parity games with a small number of different priorities in practice is the "Strategy improvement algorithm" due to Jurdziński and Vöge [JPZ06].

Theorem C.3 (Solving parity games, [JPZ06])**.** *Given a parity game, there is an algorithm that computes winning sets for both players.*
It works in space $O(dn)$, and its running time is

$$O\left(d \cdot m \cdot \left(\frac{n}{\lfloor d/2 \rfloor}\right)^{\lfloor d/2 \rfloor}\right)$$

where n is the number of vertices, m is the number of edges, and d is the maximum priority in the parity game.

INDEX

BIBLIOGRAPHY

[Dax06] Christian Dax. Games for the linear time μ-calculus. Master's thesis, Dep. of Computer Science, University of Munich, 2006. Available from http://www.tcs.ifi.lmu.de/lehre/da_fopra/Christian_Dax.pdf.

[DHL06] C. Dax, M. Hofmann, and M. Lange. A proof system for the linear time μ-calculus. In *Proc. of the 26th Conf. on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'06*, volume 4337 of *LNCS*, pages 274–285. Springer, 2006.

[EJ91] E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd Symp. on Foundations of Computer Science*, pages 368–377, San Juan, Puerto Rico, 1991. IEEE.

[FL08] Oliver Friedmann and Martin Lange. The pgsolver collection of parity game solvers, 2008. Dep. of Computer Science, University of Munich. Available from http://www.tcs.ifi.lmu.de/~mlange/pgsolver.

[Fri06] Oliver Friedmann. A proof system for the modal μ-calculus, 2006. Dep. of Computer Science, University of Munich. Available from http://www.tcs.ifi.lmu.de/lehre/da_fopra/.

[GH82] Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proc. 14th Annual ACM Symp. on Theory of Computing, STOC'82*, pages 60–65. ACM, ACM Press, 1982.

[JPZ06] Marcin Jurdzinski, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. *Proceedings of*

ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, January 2006. Available from http://www.dcs.warwick.ac.uk/~mju/Papers/JPZ06-SODA.pdf.

[Jur98] Marcin Jurdzinski. Deciding the winner in parity games is in $up \cap co - up$. *Information Processing Letters*, 68(3):119–124, 1998.

[KP05] Yonit Kesten and Amir Pnueli. A compositional approach to ctl* verification. *Theor. Comput. Sci.*, 331(2-3):397–428, 2005.

[Mar75] D. A. Martin. Borel determinacy. *Ann. Math.*, 102:363–371, 1975.

[Mat02] Radu Mateescu. Local model-checking of modal mu-calculus on acyclic labeled transition systems. *Proc. 8th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS02*, 2002. Available from ftp://ftp.inrialpes.fr/pub/vasy/publications/cadp/Mateescu-02.pdf.

[Niw97] Damian Niwinski. Fixed point characterization of infinite behavior of finite-state systems. *Theoretical Computer Science*, 189(1-2):1–69, 1997.

[Pit06] Nir Piterman. From nondeterministic buchi and streett automata to deterministic parity automata. *Proc. 21st Symposium on Logic in Computer Science*, 2006. Available from http://mtc.epfl.ch/~piterman/publications/2006/Pit06.pdf.

[Rey02] M. Reynolds. An axiomatization of full computation tree logic. *Journal of Symbolic Logic*, 66(3):1011–1057, 2002.

[SC82] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. In *STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 159–168, New York, NY, USA, 1982. ACM.

[Tar55] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics, 5:285309*,

1955. Available from http://projecteuclid.org/Dienst/UI/1.0/Summarize/euclid.pjm/1103044538.

[Tho89] Wolfgang Thomas. Computation tree logic and regular omega-languages. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, School/Workshop*, pages 690–713, London, UK, 1989. Springer-Verlag.

[Var88] M. Y. Vardi. A temporal fixpoint calculus. In *POPL '88: Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 250–259, New York, NY, USA, 1988. ACM.

[VS85] Moshe Y. Vardi and Larry J. Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 240–251, New York, NY, USA, 1985. ACM. Available from http://www.cs.rice.edu/~vardi/papers/ctl_star_lower_bound.pdf.

[Wal00] Igor Walukiewicz. Completeness of kozens axiomatisation of the propositional μ-calculus. *Inf. Comput., 157(1-2):142182*, 2000. Available from https://www.labri.fr/publications/l3a/2000/Wal00/igw_kozen_compl.pdf.

[Wil01] Thomas Wilke. Alternating tree automata, parity games, and modal mu-calculus. 2001.