Using Automata to Characterise Fixed Point Temporal Logics

Roope Kaivola

Doctor of Philosophy
University of Edinburgh
1996

(Graduation date:July 1997)

Abstract

This work examines propositional fixed point temporal and modal logics called *mu-calculi* and their relationship to automata on infinite strings and trees. We use correspondences between formulae and automata to explore definability in mu-calculi and their fragments, to provide normal forms for formulae, and to prove completeness of axiomatisations. The study of such methods for describing infinitary languages is of fundamental importance to the areas of computer science dealing with non-terminating computations, in particular to the specification and verification of concurrent and reactive systems.

To emphasise the close relationship between formulae of mu-calculi and alternating automata, we introduce a new *first recurrence* acceptance condition for automata, checking intuitively whether the first infinitely often occurring state in a run is accepting. Alternating first recurrence automata can be identified with mu-calculus formulae, and ordinary, non-alternating first recurrence automata with formulae in a particular normal form, the *strongly aconjunctive* form. Automata with more traditional Büchi and Rabin acceptance conditions can be easily unwound to first recurrence automata, i.e. to mu-calculus formulae.

In the other direction, we describe a powerset operation for automata that corresponds to fixpoints, allowing us to translate formulae inductively to ordinary Büchi and Rabin-automata. These translations give easy proofs of the facts that Rabin-automata, the full mu-calculus, its strongly aconjunctive fragment and the monadic second-order calculus of n successors *SnS* are all equiexpressive, that Büchi-automata, the fixpoint alternation class Π_2 and the strongly aconjunctive fragment of Π_2 are similarly related, and that the weak *SnS* and the fixpoint-alternation-free fragment of mu-calculus also coincide. As corollaries we obtain Rabin's complementation lemma and the powerful decidability result of *SnS*.

We then describe a direct tableau decision method for modal and linear-time mu-calculi, based on the notion of *definition trees*. The tableaux can be interpreted as first recurrence automata, so the construction can also be viewed as a transformation to the strongly aconjunctive normal form.

Finally, we present solutions to two open axiomatisation problems, for the linear-time mu-calculus and its extension with path quantifiers. Both completeness proofs are based on transforming formulae to normal forms inspired by automata. In extending the completeness result of the linear-time mu-calculus to the version with path quantifiers, the essential problem is capturing the limit closure property of paths in an axiomatisation. To this purpose, we introduce a new $\exists\nu$*-induction* inference rule.

Acknowledgements

First of all, I would like to express my gratitude to my supervisor Colin Stirling, for setting an example of intellectual clarity and encouraging me to tackle problems I otherwise would not have ventured to take up. I would also like to thank other members of the Edinburgh concurrency community, in particular Julian Bradfield, for helpful discussions at various stages of the work, and my examiners Mark Jerrum and Damian Niwiński for their constructive and insightful comments. The Laboratory for Foundations of Computer Science provided a stimulating environment for my work.

My stay in Edinburgh was made possible by generous funding from the Academy of Finland, the British Council, the CVCP Overseas Research Students Awards Scheme and the Finnish Cultural Foundation.

On a more personal note, I would like to thank Neil Ghani, Ally Hume, Christoph Lüth, Alex Mifsud, Simon McPartlin, Claudio Russo, Jitka Stříbrná, numerous members of the Finnish community and other friends in Edinburgh for their company, especially during those days when the sun was not shining (including most of Hibs' away games). Thanks are also due to my family and friends in Finland for their support. Finally, I would like to thank Marjukka Barron for refusing to give up the belief that one day this thing would be finished, and for making my life more complete in the meanwhile.

Declaration

I declare that this thesis was composed by myself, and the work contained in it is my own, unless otherwise stated. Some of the material has been published in [47, 48, 49, 50] and [51].

Table of Contents

Chapter 1

Introduction

1.1 Background

This work examines propositional fixed point temporal and modal logics called *mu-calculi* and their relationship to automata on infinite strings and trees. We point out various correspondences between classes of formulae of mu-calculi and different types of automata, and show how understanding the relationship between the different formalisms allows us, on the one hand, to use techniques from the realm of logic in addressing traditionally automata-theoretic problems, and on the other hand, to take advantage of automata-theoretic insights in solving problems related to logics. For example, using the modal mu-calculus we obtain a simple proof of Rabin's complementation lemma, and using automata-like normal forms for formulae, we show the completeness of axiomatisations for various mu-calculi.

Besides the theoretical interest, the study of infinitary languages is of fundamental importance to the areas of computer science dealing with non-terminating computations. In particular, in recent years a lot of research interest has been shown towards applying work on infinitary languages to understanding the behaviour of concurrent and reactive systems. In the traditional sequential input-output model of computation a program can be viewed as computing a function or transformation between its starting and terminating states. In contrast, a normal behaviour of a reactive program, e.g. an operating system, is a non-terminating computation which maintains an ongoing interaction with the environment. Therefore, the transformation-based semantic models that are used to model sequential programs, and the specification and verification techniques such as Hoare-logics [43] or preconditions [38] that are geared towards such transformational models are not suitable for describing concurrent and reactive systems. Instead, it is natural to base the semantics either on the infinite sequences of events a system can perform, or on an infinite tree representing the sequences of events and choices between different courses of execution, and consider specification techniques that can be used to describe such infinite structures.

Temporal and modal logics are a class of logic-based formalisms that can be used to describe properties of infinite computations. Their use for specifying and verifying concurrent systems was first advocated by Pnueli in [74], and the approach has subsequently been extensively studied. Temporal logics extend an underlying propositional or first-order logic by a set of temporal operators such as *always* or *sometimes* that allow describing how the truth of assertions varies over time. Similarly, modal logics contain operators that allow expressing the possibility or necessity of situations or events. For surveys and introductions of temporal and modal logics in computer science see [25, 83], and in the setting of more traditional philosophical logic see [18, 21, 44, 79].

Propositional mu-calculi, the formalisms discussed in the current work, arise in the framework of modal and temporal logics. The distinguishing feature of these languages is explicit minimal and maximal fixpoint operators μz and νz. The two main variants considered here are the modal mu-calculus [55], obtained by adding fixpoints to modal logic and interpreted over branching structures, and the linear-time mu-calculus [9, 6, 95], obtained similarly from standard linear-time temporal logic and interpreted over linear structures. Usually modal mu-calculus is based on the Hennessy-Milner logic [40], a poly-modal logic containing operators $<a>\phi$, *it is possible to execute an a-action leading to a state where* ϕ *holds* and $[\,a\,]\,\phi$, *every a-action leads to a state where* ϕ *holds*. However, to make the comparison with automata technically easier, we discuss here mainly a variant based on indexed modalities $ⓘ$, to distinguish the successors of a node in a tree [45].

The modal and linear mu-calculi hold a rather special position among the multitude of existing modal and temporal logics. On the one hand, they are syntactically succinct, elegant and tractable, and on the other hand, expressively very powerful so that most other propositional temporal and modal logics can be translated into them. The factor behind both these features is the incorporation of explicit fixpoint constructs in the language.

Using formalisms with fixpoint operators for describing computational phenomena can be traced back at least to the early work of Park [72], de Bakker and de Roever [5]. A basic observation that has lead to such formalisms is that first-order logics are not powerful enough to express many important properties of programs, such as totality and termination [41]. Fixpoints are also one of the central notions that appear in various forms all over computer science: denotational semantics, logics of computation, descriptive complexity theory, database theory, bisimulations etc.

Another construct close to fixpoints is second-order quantification over propositions. As shall be seen later, in the calculi and models considered in the current work, there is no difference between fixpoints and second-order quantifiers as far as expressivity is concerned. However, in a sense fixpoints have a more constructive flavour than quantifiers. This is reflected, for example, by the fact that determining satisfiability for the fixpoint calculi can be done in exponential time, whereas the same problem for the related quantifier-based calculi is non-elementary, i.e. not bounded by any fixed composition of exponential functions.

In the formalisms considered in the current work we start from a propositional language and extend it with a second-order construct, either fixpoints or second-order quantifiers allowing quantification over properties. Effectively this can be viewed as jumping from a propositional or '0-th order' formalism directly to a second-order one, without going via a first order language, with quantification over individuals. We could naturally also extend first-order formalisms with similar second-order constructs [88]. However, in particular with fixpoints, this would lead to less tractable and less well researched formalism [42].

The quantifier-based formalisms considered in the current work are very closely related to two important calculi with second-order quantification which have been examined in the context of decision problems in mathematical logic: the monadic second-order theory of one successor *S1S*, studied by Büchi [15], and the monadic second-order theory of n successors *SnS*, studied by Rabin [76]. The decidability of these calculi, *SnS* in particular, is a fundamental result to which large classes of other decidability results can be reduced [76, 78]. To quote Gurevich and Harrington [39] on this: *If you worked on decision problems you did most probably use Rabin's result.*

Originally, one of the main reasons to study automata on infinite objects was as a tool to show the decidability of these second-order calculi. To this end, Büchi introduced a type of finite automata on infinite strings as an easy-to-handle normal form for *S1S* formulae in his seminal 1962 paper [15]. These automata, nowadays known as Büchi automata, generalise the notion of ordinary nondeterministic finite automata on finite strings by attaching to them an acceptance condition specifying which infinite executions of the automaton count as accepting. In a similar way, to show the decidability of *SnS*, Rabin [76] defined the notion of finite automata on infinite trees, generalising the notion of automata on finite trees [91, 37]. For a survey of automata on infinite strings and trees, see [93].

The core point of the decidability proof for both *S1S* and *SnS* is showing that the automata in question are closed under boolean operations, especially complementation. Already nontrivial for Büchi automata on strings, complementation is notoriously hard for Rabin automata on trees. Due to the central nature of this result, known as Rabin's complementation lemma, several researchers [16, 17, 39, 66, 67, 33] have endeavoured to explain and simplify the original proof [76].

In the area of temporal and modal logics, and mu-calculi in particular, automata on infinite objects have been long applied as a tool for decidability results. For example, the decision procedure of Streett and Emerson [86, 87] for modal mu-calculus works by translating formulae of the logic to Rabin tree automata, and reducing the satisfiability problem of the formula to the emptiness problem for the corresponding automaton. Similarly, the decision procedure of Vardi [95] for linear-time mu-calculus is based on mapping formulae to Büchi automata. Translations from formulae to automata have also been used for model-checking, i.e. determining the validity of a formula over a particular model, especially for linear-time logics [98, 94].

The advantage of translating formulae to automata, thereby reducing a logical problem to an automata-theoretic problem, is that this makes available a whole body of results that have already been established concerning automata, in particular algorithmic and complexity-theoretic results. Thanks to their simpler, non-hierarchic structure, automata are easier to handle automatically than formulae, so once we have managed to translate a formula to an automaton, the rest usually follows easily. Because of this, automata have been proposed as a uniform framework to which various logical formalisms can be translated, and as a specification formalism in their own right [98, 94]. It should be pointed out here that the research on applying automata as a tool for deciding logics has also produced new results concerning automata, in the form of improved algorithms and new types of acceptance conditions.

The drawback in using automata directly as a specification formalism is that they lack a structural theory that the logic-based calculi have, and by which properties can be composed from sub-properties by operators of the language. The reduction of logical problems to automata-theoretic ones has also the negative consequence that results concerning logics tend to be rather indirect. For example, although the decision methods using automata certainly answer the question of whether a formula is satisfiable or not, the required rather complex translations and automata constructions mean that the answer does not give an insight into why or how the formula is satisfiable. Related to this, although decision methods often yield complete axiomatisations, the automata constructions are not very helpful in this respect, and the question of axiomatising linear-time and modal mu-calculi was an open problem until very recently [49, 101].

The technical gap between automata and mu-calculi has lately been narrowing due to a new type of automata on infinite objects called *alternating automata* [66, 67, 68, 64]. The concept of alternation in automata, introduced in [19], generalises the notion of nondeterminism by allowing states to be existential or universal. This means that in addition to being able to represent disjunction, which naturally corresponds to nondeterminism, such automata have also the means to represent conjunction, and consequently they resemble logical formulae more than ordinary automata do. In particular, the structure of alternating automata is very similar to formulae of mu-calculus, to the extent that, quoting Emerson [26], *mu-calculus formulae are really representations of alternating finite-state automata on infinite trees*. An advantage of alternating automata over ordinary ones is that they are easy to complement by dualization. In the case of automata over trees, alternating automata also appear a more natural formalism than ordinary nondeterministic automata [10]. However, since alternating automata have a more complicated structure than ordinary ones, in some respects they are also more difficult to analyse and manipulate. In particular, although trivial for ordinary automata, projection or existential quantification is hard for alternating ones and effectively requires translating them back to ordinary automata. As closure under projection is essential for Rabin's decidability result for SnS, this means that we cannot escape the complications of Rabin's proof by simply using alternating automata instead of ordinary ones. Deciding emptiness is also harder for alternating than for ordinary automata.

The second-order calculi and automata on infinite objects discussed in this work have also deep connections with the theory of infinite games. One of the main lines of research on the complementation problem of Rabin automata, pursued by Büchi [16, 17] and Gurevich and Harrington [39], has been the formulation of the problem in the framework of infinite games. In short, a run of a tree automaton on an input can be viewed as a strategy in an infinite game, where one of the players plays for acceptance and the other for rejection. Conversely, a finite state winning strategy in such a game can be viewed as an automaton. The question of complementation can then be reduced to a theorem about the determinacy of these games and the existence of finite state winning strategies [39]. This approach and the Gurevich-Harrington theorem is also the easiest way to translate alternating automata to ordinary ones. In addition to the complementation problem, games have been used more recently by Stirling to examine the model-checking problem of modal mu-calculus [84]. However, as the developments in the current work are independent of the game-theoretic approach, in the following we will address the issue only in passing.

1.2 Synopsis

A main theme going through the whole work here is following Büchi's idea and viewing ordinary nondeterministic automata as useful normal forms for formulae, or as the means of transforming formulae to such normal forms.

Chapter 2 is a brief introduction to infinite linear and branching structures and some logical languages which can be used to describe these. The most important formalisms are the linear-time mu-calculus μTL and its branching time counterpart, the (indexed) modal mu-calculus μKn, which is interpreted over trees with a fixed branching degree n. In addition to these fixpoint-based languages we examine another second-order construct, second-order quantification, and the linear and branching languages $\exists TL$ and $\exists Kn$ based on quantifiers instead of fixpoints. The languages $\exists TL$ and $\exists Kn$ can be viewed as alternative formulations of the monadic second-order calculus of one successor *S1S* and that of n successors *SnS* mentioned earlier. Furthermore, we define the weak versions $\overset{\mathrm{w}}{\exists} TL$ and $\overset{\mathrm{w}}{\exists} Kn$ of these calculi, corresponding to the weak calculi *WS1S* and *WSnS*, with quantification over finite sets only.

Chapter 3 concentrates on automata and on relating various types of automata and fragments of mu-calculi to each other. We first define the usual notions of ordinary and alternating automata on infinite strings and trees, and the standard Büchi and Rabin acceptance conditions. In Section 3.2 we introduce a new type of automata, the *first recurrence* automata, providing a common ground on which mu-calculi and automata can be related to each other. First recurrence automata are particularly appropriate for understanding mu-calculi, since alternating first recurrence automata correspond precisely to formulae of mu-calculi and vice versa, via easy syntactic translations. Ordinary, non-alternating first recurrence automata correspond to formulae in a restricted normal form, where the most important restriction is the so-called *strong aconjunctivity*. The close connection between first recurrence automata and mu-calculus formulae allows us to rephrase the statement above about mu-calculus formulae being alternating automata in the other direction: *alternating automata are really mu-calculus formulae.*

In Section 3.4 we describe fixpoint constructions for ordinary Büchi and Rabin automata, based on [22, 76, 77]. These allow us to translate mu-calculus formulae inductively to such automata. Using them and translations between Rabin and Büchi automata on the one hand and first recurrence automata on the other, we show the equiexpressivity of ordinary Rabin automata and the full mu-calculus, and that of ordinary Büchi automata and the fixpoint alternation class Π_2. Since the full mu-calculus is trivially closed under complementation, this implies that

Rabin automata are closed under complementation, as well. As a corollary we get an easy inductive translation from $\exists Kn$ or *SnS* to Rabin automata, and the decidability of all the calculi mentioned above. We believe that this provides the simplest proof of Rabin's result so far; in fact, as Rabin's original proof [76] uses constructions which have a close resemblance to the fixpoint constructions used here, although not called as such, we can view the use of mu-calculus and the current approach as a way of structuring Rabin's proof in a more transparent fashion. As side products of the results, we show the equiexpressivity of the languages $\overset{w}{\exists}Kn$, *WSnS* and the fixpoint alternation class Δ_2 of mu-calculus, and obtain one half of Rabin's fundamental correspondence between Büchi recognisable languages and existentially quantified $\overset{w}{\exists}Kn$ or *WSnS*. Although none of these correspondence or decidability results are new in themselves, we feel that the contribution of the current work is that all these results, shown previously by a variety of tools, arise here uniformly from two rather simple concepts: the first recurrence automata, and the fixpoint constructions for ordinary Büchi and Rabin automata. We also believe that first recurrence automata are an interesting formalism in their own right, since they have the same expressive power as Rabin automata, but emptiness is decidable in linear time for them.

The constructions and results listed above work for both linear and branching structures and formalisms. In Section 3.5 we examine differences between the linear and the branching case. In linear case it is easy to see that Büchi and Rabin recognisability coincide, which allows us to show that the full linear-time mu-calculus and its fragment Δ_2 without any proper fixpoint alternation are equiexpressive. For the branching case none of this is true. Another distinguishing feature is determinisation of automata, which leads us to discuss what determinism means in formulae, and define a deterministic normal form for linear-time mu-calculus.

Based on the understanding of the intimate relationship between mu-calculi and automata provided by Chapter 3, the rest of the work examines deciding and axiomatising mu-calculi. In Chapter 4 we describe an elementary tableau decision method for modal and linear-time mu-calculi[1]. The approach is based on the notion of *definition trees*, related to the definition lists of Stirling and Walker [85], and is influenced by the works of Safra [80], Emerson and Jutla [32, 33], and Walukiewicz [100, 101]. The tableau construction also yields a direct translation from any mu-calculus formula to an equivalent strongly aconjunctive one. This provides yet another proof of the fact that Rabin automata are closed under com-

[1]Formulating such a tableau method was stated as an open problem in Jutla's thesis [46]

plementation. What may be more interesting, though, is that the translation to the strongly aconjuctive normal form allows us show the decidability of the quantifier-based calculi $\exists Kn$ and SnS by translating them inductively to the mu-calculus μKn without having to explicitly invoke automata in the process. Compared to the decidability proof of $\exists Kn$ above, which uses fixpoint constructions for automata, the current translation has the added elegance that it only requires the use of one auxiliary formalism μKn instead of two.

In Chapter 5 we present a solution to the previously open problem of completely axiomatising the linear-time mu-calculus μTL. The axiomatisation of μTL has been attempted before by at least Lichtenstein [59] and Dam [22], and the related axiomatisation for the modal mu-calculus by Kozen [55] and Walukiewicz [100, 101]. The axiomatisation of μTL used here is essentially the one proposed by Kozen in [55]. The completeness proof in Chapter 5 is based transforming formulae provably to a new normal form, the *bi-aconjunctive non-alternating form*, inspired by restricted weak alternating automata. The crucial property of such formulae is that not only is it easy to construct a model of a consistent formula, but the same holds also of its negation. Moreover, the semantic equivalence between the full μTL and the normal form can be lifted to the level of provability rather easily on the basis of what is already known about aconjunctivity.

In 1995 Walukiewicz presented a completeness proof for an axiomatisation of the modal mu-calculus [101]. This result naturally carries over from the modal mu-calculus to the the linear mu-calculus, as well. However, the proof requires a rather complex argument using games between tableaux, which can be avoided here using the easy negatability of formulae in the bi-aconjunctive non-alternating normal form. Bar one observation that was used in passing in Walukiewicz's work, the work presented here (published originally in [49]) was carried out independently.

In Chapter 6 we depart slightly from the framework of the rest of the work and examine an extension of the linear-time mu-calculus μTL to a branching time formalism using *path quantifiers* $\overrightarrow{\exists}\phi$ and $\overrightarrow{\forall}\phi$, *for some path* ϕ and *for all paths* ϕ. This way of extending a linear formalism to a branching one is familiar from various temporal logics; probably the best example is the full computation tree logic CTL^* [28, 29], which extends the standard linear-time temporal logic TL with path quantifiers. We call the formalism consisting of μTL and path quantifiers here the *extended computation tree logic* $\overrightarrow{\exists}\mu TL$. In general, due to the interaction of path quantifiers with other operators of the logic, it has turned out to be difficult to axiomatise logics with path quantifiers, even though axiomatisations for the underlying linear-time formalisms would be known. This holds also for the extended computation tree logic $\overrightarrow{\exists}\mu TL$.

Chapter 6 presents a solution to the axiomatisation problem of $\overset{\rightsquigarrow}{\exists}\mu TL$ with respect to the so-called *R-generable* structures, basically normal transition systems where every maximal sequence of pairwise connected states counts as a path. The essential difficulty in such an axiomatisation is characterising *limit closure*. This has become apparent e.g. with the standard computation tree logic, for which the axiomatisation problem has been open for some while [35, 25, 83]. Here limit closure is characterised by a a new inference rule, the $\overset{\rightsquigarrow}{\exists}\nu$*-induction*. The completeness proof is is based on transforming formulae to a strongly aconjunctive deterministic normal form.

An intriguing aspect in this completeness proof is that the ability to transform a formula to the deterministic form requires an arbitrary level of fixpoint alternation. which means that the approach is not directly applicable for the formulation of extended computation tree logic with ω-regular expressions, although this is semantically equiexpressive with $\overset{\rightsquigarrow}{\exists}\mu TL$. The same holds also for CTL^*, so the axiomatisation problem for it remains open. Nevertheless, we believe that the current work outlines one potential way of approaching the completeness problem for CTL^*; first, the presence of the $\overset{\rightsquigarrow}{\exists}\nu$-induction rule here leads us to believe that some similar proof principle will be needed for CTL^* as well, and secondly, we believe that it should be possible to reformulate the current proof so that the transformation of formulae to the deterministic normal form can take place implicitly, in which case the same principles might be used for CTL^*, as well. However, at the current stage this is still speculation.

Chapter 2

Logical calculi

In this chapter we introduce formally the languages of linear and modal mu-calculi and some other logical formalisms examined in later parts of the work. Although modal mu-calculus is probably more widely known than the linear-time one, we discuss the latter first in Section 2.2, as the linear framework is technically somewhat simpler. In Section 2.3 the concepts introduced for the linear calculus are extended to deal with branching tree structures.

Some standard concepts and notations are listed in Section 2.1. After this, Section 2.2 concentrates on linear structures and calculi interpreted over them. First, the linear models, the language of linear-time mu-calculus μTL and the related basic notions are introduced in Subsection 2.2.1. The important concept of alternation of minimal and maximal fixpoints is examined in Subsection 2.2.3. Analogous to the analytical and arithmetical hierarchies based on alternation of quantifiers, the alternation of fixpoints leads to a hierarchy of classes Σ_i, Π_i and Δ_i of formulae, and is one of the main sources, if not *the* main source, of difficulty in understanding mu-calculi. In Subsection 2.2.4 we give a more syntactic account of truth of a formula in a model, by describing an infinitary tableau construction in the spirit of [85].

Subsection 2.2.5 studies some aspects of the relationship of fixpoints and second-order quantification over propositions. We introduce a calculus $\exists TL$ which corresponds to the linear-time mu-calculus μTL, but is based on second-order quantifiers instead of fixpoints, and show that the fixpoint-based calculus can be translated to the quantifier-based one. The language $\exists TL$ can be viewed as an alternative formulation of the monadic second-order calculus of one successor *S1S* mentioned earlier; $\exists TL$ and *S1S* can be straightforwardly translated syntactically to each other. In Subsection 2.2.6 we discuss the weak versions $\overset{\mathrm{w}}{\exists} TL$ and *WS1S* of these calculi, with quantification over finite sets only, and point out the important relation between these languages, non-alternating fixpoints and continuity.

Section 2.3 extends the ideas of the previous section from linear structures to branching ones. The language of the indexed modal mu-calculus μKn, with modalities $\textcircled{i}\phi$, *for the* i*-th successor* ϕ, is introduced in Subsection 2.3.1, and related basic concepts and techniques are generalised from the linear to the branching case in Subsection 2.3.2. The second-order quantifier based linear calculus $\exists TL$ and its weak variant $\overset{\mathrm{w}}{\exists} TL$ are extended to branching formalisms $\exists Kn$ and $\overset{\mathrm{w}}{\exists} Kn$ in Subsection 2.3.3. Analogous to the situation above, the calculus $\exists Kn$ is essentially an alternative formulation of the important monadic second-order calculus of n successors *SnS*, studied above all by Rabin, and $\overset{\mathrm{w}}{\exists} Kn$ a formulation of the weak calculus *WSnS*.

2.1 Preliminaries

Let us first list some notations for sets and strings.

Definition 2.1.1 We use $\mathbb{N} = \{0, 1, 2, \ldots\}$ to denote the set of natural numbers. For any $n \in \mathbb{N}$ the expression $[n]$ denotes the set $[n] = \{0, \ldots, n-1\}$.

For any set A, we use 2^A to denote the powerset of A, i.e. the set of all subsets of A. For any sets A and B, we use $A \rightarrow B$ to denote the set of functions and $A \rightharpoonup B$ the set of partial functions from A to B. If $f : A \rightharpoonup B$ is a partial function, the *domain* of f, denoted by $\mathrm{dom}(f)$ is the subset of A on which f is defined.

We use the notation $\overline{q}$ to denote a vector and $\overline{q}_i$ to refer to its elements. $\square$

Definition 2.1.2 Let A be an arbitrary set. We use A^* and A^ω to denote the sets of finite and infinite strings of elements of A, respectively, and define $A^\infty = A^* \cup A^\omega$. The symbol ϵ denotes the unique empty string and $|s|$ the length of s. If $s = a_0 a_1 \ldots a_n \ldots$, $s(i)$ is the element a_i, $s[i \ldots j]$ the finite string $a_i a_{i+1} \ldots a_j$, and $s[i \ldots]$ the finite or infinite string $a_i a_{i+1} a_{i+2} \ldots$ If t and s are strings, $t \cdot s$ the concatenation of t and s, and $s \preceq t$ ($s \prec t$) means that s is a prefix (a proper prefix) of t. If $s \preceq t$, $s^{-1}t$ is the s' such that $s \cdot s' = t$. $\square$

Notice that the first element of a string s is $s(0)$, not $s(1)$. Let us then introduce some standard concepts and notation for trees and labelled trees. In the current work we only deal with trees which are ordered and at most countably branching.

Definition 2.1.3 A *tree* is a set $T \subseteq \mathbb{N}^*$ such that for all $t \in \mathbb{N}^*$ and $i \in \mathbb{N}$,

- if $t \cdot i \in T$ then $t \in T$, and
- if $t \cdot (i+1) \in T$ then $t \cdot i \in T$.

Let T be a tree.

- We call every $t \in T$ a *node* of T.
- The *root node* of T is ϵ.
- For every $t \in \mathbb{N}^*$ and $i \in \mathbb{N}$, if $t \cdot i$ is a node of T, we call t the *parent* of $t \cdot i$, and $t \cdot i$ a *child* of t.
- If $t \cdot i$ and $t \cdot j$ are children of node t and $i < j$, we say that the node $t \cdot i$ is *older* than the node $t \cdot j$, and that $t \cdot j$ is *younger* than $t \cdot i$.
- If t is a node of T and t has no children, t is called a *leaf*. Otherwise, t is called an *interior node*.
- We say that a node t is an *ancestor* of node t' and t' a *descendant* of t iff $t \preceq t'$. We call t is a *proper ancestor* of t' and t' a *proper descendant* of t iff $t \prec t'$.
- Let t be a node of T. A finite or infinite sequence $p = t_0 t_1 t_2 \ldots$ of nodes of T is a *path* of T from node t iff $t_0 = t$, each t_{i+1} is a child of t_i, and if the sequence is finite then its last element is a leaf.
- The set of nodes of T on a path $p = t_0 t_1 t_2 \ldots$ of T is denoted by $\mathrm{st}(p) = \{t_0, t_1, t_2, \ldots\}$.
- The set of all paths from a node t of T is denoted by $\mathrm{paths}(T, t)$. The set of all paths of T is denoted by $\mathrm{paths}(T)$. □

Definition 2.1.4 Let T be a tree. We extend the older/younger ordering between children of any node to a total ordering on the nodes of T as follows:

- if t is a proper ancestor of t', then t is older than t', and
- if t_1 is a descendant of t'_1, t_2 is a descendant of t'_2, t'_1 and t'_2 are both children of some node t, and t'_1 is older than t'_2, then t_1 is older than t_2. □

Definition 2.1.5 Let A be an arbitrary set. An *A-labelled tree* T is a partial mapping $T : \mathbb{N}^* \rightharpoonup A$ such that $\mathrm{dom}(T)$ is a tree.

Let T be an A-labelled tree. If t is a node of T, the *subtree of T rooted at node t*, denoted by T^t, is the A-labelled tree defined by $T^t(t') = T(t \cdot t')$, for all $t' \in \mathbb{N}^*$. If $p = t_0 t_1 t_2 \ldots$ is a path of T, we use $T[p]$ to denote the corresponding string $T(t_0)T(t_1)T(t_2)\ldots$ of elements of A. □

We classify trees according to their level of branching and the number of nodes.

Definition 2.1.6 We say that a tree T is *finitely branching* iff every node of T has only finitely many children. For every $n \in \mathbb{N}$, we say that a tree T is *n-branching* or is of *branching degree n* iff every node of T is either a leaf or has exactly n children. If T is finite as a set, we call T a *finite tree* and if it is infinite, we call

T an *infinite tree*. If a tree T has no leaves, we say that T is *total*. The *depth* of a finite tree T is the length of the longest path in T. □

Notice that every total tree is infinite, but not necessarily vice versa, and that a total and infinite tree T is n-branching iff $T = [n]^*$.

2.2 Linear structures

2.2.1 Linear time mu-calculus

The linear-time mu-calculus and linear-time temporal logics in general are logic based languages that can be used to describe properties of linear sequences of situations. Assuming some level of atomicity in a program's actions, an execution of a program or system can be modelled by a linear sequence of execution states. The use of linear-time temporal logics in specifying properties of programs is based on viewing these execution sequences as temporal logic models, i.e. linear structures on which temporal logic formulae can be interpreted.

Let us first define the concept of a linear-time model.

Definition 2.2.1 Fix a countable set $\mathcal{Z}$, the elements of which are called *atomic propositions*, and define the set $\overline{\mathcal{Z}}$ of negated atomic propositions by $\overline{\mathcal{Z}} = \{\neg z \mid z \in \mathcal{Z}\}$. □

Definition 2.2.2 A *linear model* M is an infinite sequence of sets of propositions,

$$M \in (2^{\mathcal{Z}})^{\omega}$$

A *state* s of a linear model M is any $s \in \mathbb{N}$, and the set of states, denoted by $\mathrm{st}(M)$, is $\mathrm{st}(M) = \mathbb{N}$. The set of all linear models is denoted by $\mathcal{M}_{TL}$. □

Notice that we assume for simplicity that all linear models are infinite.

The idea that execution sequences can be viewed as temporal logic models, and that temporal logic formulae can be used to specify required properties of the executions of a system, was first proposed by Pnueli in [74]. Temporal logics take an underlying propositional or first-order language, which is used to describe characteristics of individual states or moments of a model, and extend it with a set of temporal operators. In the current work we will only discuss propositional temporal logics.

In the framework of linear time, the most traditional temporal operators are *sometime/always in the future*, denoted by $\mathbf{F}\phi$ and $\mathbf{G}\phi$, and their past counterparts. The truth definitions for these operators are

- $\mathbf{F}\phi$ is true at moment/state i of model M iff there is some $j \geq i$ such that ϕ is true at moment j of M
- $\mathbf{G}\phi$ is true at moment i of model M iff for all $j \geq i$, ϕ is true at moment j of M

These two operators are *duals*, i.e. $\neg\mathbf{F}\neg\phi \Leftrightarrow \mathbf{G}\phi$.

As far as applications in computer science are concerned, the most common linear-time temporal logic TL is based on the *nexttime* and *until* operators $\bigcirc\phi$ and $\psi\mathbf{U}\phi$. The truth definitions of these are

- $\bigcirc\phi$ is true at moment i of model M iff ϕ is true at moment $i+1$ of M
- $\psi\mathbf{U}\phi$ is true at moment i of model M iff there is some $j \geq i$ such that ϕ is true at moment j of M and for all $i \leq i' < j$, ψ is true at moment i' of M

Often past temporal operators are omitted, since this does not affect the expressive power of the language [36]. It is easy to see that the sometime and always operators can be expressed in terms of the until operator by $\mathbf{F}\phi \Leftrightarrow \top\mathbf{U}\phi$ and $\mathbf{G}\phi \Leftrightarrow \neg(\top\mathbf{U}\neg\phi)$, where $\top$ denotes the everywhere true proposition. The reverse does not hold; there are properties which can be expressed using the until operator, but which cannot be captured by the sometime and always operators alone [52].

Applications of these logics to specification and verification of programs have been extensively studied, and form one of the main research trends in the field. For example, simple safety properties of the type ϕ *never happens* can be expressed by formulae of the type $\mathbf{G}\neg\phi$, and simple liveness properties ϕ *eventually happens* by formulae of the type $\mathbf{F}\phi$.

However, there are some rather straightforward properties that cannot be expressed in terms of the until and nexttime operators, for example *at every even moment* ϕ [102]. Because of this, various more expressive formalisms have been studied, most notably Wolper's *extended temporal logic ETL* [102, 103], and the linear-time mu-calculus. Wolper's ETL is based on an infinite family of temporal operators, one corresponding to each regular grammar.

The linear-time mu-calculus was first introduced by Barringer, Kuiper and Pnueli in [7, 8, 9], to define temporal semantics for recursive procedures. It has also been explicitly examined at least by Banieqbal and Barringer [6], and Vardi [95]. As already mentioned in several occasions, the characteristic feature of mu-calculi is the incorporation of explicit fixpoint operators $\nu z.\phi$ and $\mu z.\phi$ in the language. We only need to take one of these as a primitive operator, since the other can be introduced as a dual.

Definition 2.2.3 The formulae of the *linear-time mu-calculus* μTL are defined by the abstract syntax:

$$\phi ::= z \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \bigcirc\phi \mid \mu z.\phi$$

where z varies over $\mathcal{Z}$. In $\mu z.\phi$, each occurrence of z in ϕ is required to be *positive*, i.e. in the scope of an even number of negations. □

Let us then state the standard truth definitions for the linear-time mu-calculus.

Definition 2.2.4 Let $M \in \mathcal{M}_{TL}$ be a linear model. The set of states of M satisfying a μTL-formula ϕ, denoted by $\|\phi\|_M$, is defined by

$$\begin{aligned}
\|z\|_M &= \{s \in \mathrm{st}(M) \mid z \in M(s)\} \\
\|\neg\phi\|_M &= \mathrm{st}(M) \setminus \|\phi\|_M \\
\|\phi \wedge \phi'\|_M &= \|\phi\|_M \cap \|\phi'\|_M \\
\|\bigcirc\phi\|_M &= \{s \in \mathrm{st}(M) \mid s+1 \in \|\phi\|_M\} \\
\|\mu z.\phi\|_M &= \bigcap\{W \subseteq \mathrm{st}(M) \mid \|\phi\|_{M[W/z]} \subseteq W\}
\end{aligned}$$

where $M[W/z]$ is defined by:

$$M[W/z](s) = \begin{cases} M(s) \cup \{z\} & \text{if } s \in W \\ M(s) \setminus \{z\} & \text{if } s \in \mathrm{st}(M) \setminus W \end{cases}$$

We say that a formula ϕ is *true at state* s *of* M and write $M, s \models \phi$ iff $s \in \|\phi\|_M$. We say that ϕ is *initially true in* M and write $M \models \phi$ iff $M, 0 \models \phi$.

We say that a formula ϕ is *universally valid* and write $\models \phi$ iff $M, s \models \phi$ for all models M and all states s of M. A formula ϕ is *satisfiable* iff there exists a model M and a state s of M such that $M, s \models \phi$.

The *language characterised* by a formula ϕ, denoted by $L(\phi)$, is defined by $L(\phi) = \{M \in \mathcal{M}_{TL} \mid M \models \phi\}$. □

Before discussing the intuitions behind the truth definitions, let us introduce some standard concepts and derived operators, which will be used for all the logical formalisms in the current work.

Definition 2.2.5 We use the notation $\phi \trianglelefteq \phi'$ to mean that ϕ' is a subformula of ϕ, and $\phi \triangleleft \phi'$ that ϕ' is a proper subformula of ϕ. We use $|\phi|$ to denote the length of a formula ϕ, considered as a string.

An occurrence of a variable z in a formula ϕ is *bound* iff it is within a subformula $\mu z.\phi'$ of ϕ and *free* otherwise. □

Definition 2.2.6 Let $\phi, \phi_0, \ldots, \phi_k$ be formulae and $z_k, \ldots, z_k$ distinct variables. Then $\phi[\phi_0/_t z_0, \ldots, \phi_k/_t z_k]$ is the result of simultaneously textually substituting each ϕ_i for all free occurrences of z_i in ϕ.

The notation $\phi[\phi_0/z_0, \ldots, \phi_k/z_k]$ denotes the same substitution operation, but with the following exception: If some free variable z' of ϕ_i would be captured by a fixpoint $\mu z'$ of ϕ in the substitution, the bound variable z' in ϕ is systematically renamed. We extend this notation to a set of formulae by $\Gamma[\phi_0/z_0, \ldots, \phi_k/z_k] = \{\phi[\phi_0/z_0, \ldots, \phi_k/z_k] \mid \phi \in \Gamma\}$. □

Definition 2.2.7 We use the derived operators $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\top$, $\bot$ and νz, the semantics of which are defined by the translations:

$$\begin{aligned}
\phi \vee \phi' &= \neg(\neg\phi \wedge \neg\phi') \\
\phi \Rightarrow \phi' &= \neg(\phi \wedge \neg\phi') \\
\phi \Leftrightarrow \phi' &= (\phi \Rightarrow \phi') \wedge (\phi' \Rightarrow \phi) \\
\top &= a \vee \neg a \qquad \text{where } a \text{ is some fixed atomic proposition} \\
\bot &= \neg\top \\
\nu z.\phi &= \neg\mu z.\neg\phi[\neg z/z]
\end{aligned}$$

The following precedence order is used to reduce the number of parentheses:

$$\bigcirc, \neg, \wedge, \vee, \Leftrightarrow, \Rightarrow, \nu z, \mu z$$

where $\bigcirc$ has the highest precedence and μz the lowest, i.e. $\mu z.a \vee \bigcirc z$ stands for $\mu z.(a \vee (\bigcirc z))$.

The symbol σ refers to both the μ and ν-operators. If $\Gamma = \{\phi_0, \ldots, \phi_n\}$ is a finite set of formulae, we use the abbreviation $\bigwedge\Gamma$ to stand for $\phi_0 \wedge \ldots \wedge \phi_n$. Define inductively the syntactic abbreviation $\bigcirc^n\phi$ for every $n \in \mathbb{N}$ by: $\bigcirc^0\phi = \phi$ and $\bigcirc^{n+1}\phi = \bigcirc(\bigcirc^n\phi)$. We call a formula ϕ *atomic* iff ϕ is $\top$, $\bot$, an atomic proposition $z \in \mathcal{Z}$ or a negated atomic proposition $\neg z \in \overline{\mathcal{Z}}$. □

Notice that the scope of a fixpoint extends as far to the right as possible. As has been already done informally above, we write $\phi \Leftrightarrow \psi$ to mean that $\models \phi \Leftrightarrow \psi$, i.e. that ϕ and ψ are semantically equivalent. With slight abuse of notation, we use the expression $\phi \Leftrightarrow \psi$ to denote the semantical equivalence of ϕ and ψ also when ϕ and ψ are formulae of different languages. The following lemma states the obvious fact that we can replace subformulae of a formula with equivalent ones without affecting its truth value.

Lemma 2.2.8 Let ϕ, ϕ' and ψ be formulae and z a variable. If $\models \phi \Leftrightarrow \phi'$, then $\models \phi[\psi/z] \Leftrightarrow \phi'[\psi/z]$, and if $\models \phi \Rightarrow \phi'$, then $\models \phi[\psi/z] \Rightarrow \phi'[\psi/z]$.

Furthermore, if we assume that both $\psi[\phi/_t z]$ and $\psi[\phi'/_t z]$ are well-defined, then $\models \phi \Leftrightarrow \phi'$ implies $\models \psi[\phi/_t z] \Leftrightarrow \psi[\phi'/_t z]$. If in addition z occurs only positively in ψ, then $\models \phi \Rightarrow \phi'$ implies $\models \psi[\phi/_t z] \Rightarrow \psi[\phi'/_t z]$.

Proof: Easy. □

Notice that the operator $\bigcirc$ is self-dual, i.e. $\neg\bigcirc\neg\phi \Leftrightarrow \bigcirc\phi$, and consequently there is no need to introduce a dual modality for it. However, if we interpreted the language not only over infinite sequences but also over finite ones, this would not hold. Assuming that the interpretation of $\bigcirc\phi$ in this framework would be *there is a next moment and ϕ holds at that moment*, the dual modality for $\bigcirc$ would have the interpretation *if there is a next moment, then ϕ holds at that moment.*

An easy, although not entirely accurate way of understanding the meaning of the fixpoint operators is viewing the minimal fixpoint $\mu z.\phi$ as finite looping, and the maximal fixpoint $\nu z.\phi$ as infinite looping. For example, the sometime, always and until operators of the standard until-based linear-time temporal logic TL correspond to the following expressions

$$\begin{aligned} \mathbf{F}\phi &\Leftrightarrow \mu z.\phi \vee \bigcirc z \\ \mathbf{G}\phi &\Leftrightarrow \nu z.\phi \wedge \bigcirc z \\ \psi\mathbf{U}\phi &\Leftrightarrow \mu z.\phi \vee (\psi \wedge \bigcirc z) \end{aligned}$$

This shows that TL can be easily embedded in the linear-time mu-calculus. The view of minimal and maximal fixpoints as finite and infinite looping, respectively, is especially clear if we look at how ω-regular expressions could be characterised by fixpoints [73]. For example

$$\begin{aligned} a^* \cdot b &\Leftrightarrow \mu z.a \wedge \bigcirc z \ \vee \ b \\ (a \cdot b)^\omega &\Leftrightarrow \nu z.a \wedge \bigcirc b \wedge \bigcirc\bigcirc z \end{aligned}$$

For a definition of ω-regular expressions, see e.g. [93, p. 136].

As a somewhat more complex example, the property *a is true infinitely often* can be expressed by either of the following formulae

$$\nu z.(\mu x.a \vee \bigcirc x) \wedge \bigcirc z$$
$$\nu z.\mu x.(a \wedge \bigcirc z) \vee \bigcirc x$$

The former of these corresponds directly to the formula $\mathbf{GF}a$ of TL. The latter formula, where the inner fixpoint formula makes reference to the outer fixpoint variable, has no such direct counterpart in TL.

The dual property *only finitely often not a* or *almost always a* can be expressed by either of the following formulae

$$\mu z.(\nu x.a \wedge \bigcirc x) \vee \bigcirc z$$
$$\mu z.\nu x.(a \vee \bigcirc z) \wedge \bigcirc x$$

As above, the first of these corresponds directly to the formula $\mathbf{FG}a$ of TL, and the second has no direct counterpart in TL.

Above we mentioned that the property *at every even moment* ϕ cannot be expressed by any formula of the standard linear-time temporal logic TL. In linear-time mu-calculus it can be characterised by the formula $\nu z.\phi \wedge \bigcirc\bigcirc z$.

2.2.2 Fixpoints and approximants

To address the issue of fixpoints with some more rigour, the idea behind the minimal fixpoint operation $\mu z.\phi$ is viewing ϕ as a function mapping the set of states where z is true to the set of states where ϕ is true, i.e. considering the functional

$$f : 2^{\mathrm{st}(M)} \longrightarrow 2^{\mathrm{st}(M)}; W \mapsto \|\phi\|_{M[W/z]}$$

and defining the semantics of $\mu z.\phi$ as the least fixpoint of this functional. Similarly the meaning of $\nu z.\phi$ is the greatest fixpoint of this functional. Since $2^{\mathrm{st}(M)}$, ordered by $\subseteq$, is a complete lattice and since f is monotonic due to the assumption that z only occurs positively in ϕ, the least and greatest fixpoints of f exist by the Knaster-Tarski fixpoint theorem [90] (for an introduction to fixpoints, see e.g. [62, Chapter 4]). The least fixpoint coincides with the intersection of all post-fixpoints, i.e. sets W for which $f(W) \subseteq W$, which justifies the formal definition of $\|\mu z.\phi\|_M$ above.

If we spell out the semantics of $\nu z.\phi$ directly, this is

$$\|\nu z.\phi\|_M = \bigcup\{W \subseteq \mathrm{st}(M) \mid W \subseteq \|\phi\|_{M[W/z]}\}$$

i.e. the semantics is determined as the union of all pre-fixpoints of the functional f, which coincides with the greatest fixpoint of f by the Knaster-Tarski theorem.

For future reference, let us state formally an easy reformulation of the truth conditions for fixpoints.

Lemma 2.2.9 Let M be a linear model, s a state of M and $\mu z.\phi$ a minimal fixpoint formula of μTL. Then

$$M, s \models \mu z.\phi \;\text{ iff }\; \text{for all } W \subseteq \mathrm{st}(M),\; \text{if } \|\phi\|_{M[W/z]} \subseteq W \text{ then } s \in W.$$

Symmetrically,

$$M, s \models \nu z.\phi \text{ iff there exists } W \subseteq \mathrm{st}(M) \text{ such that } s \in W \text{ and } W \subseteq \|\phi\|_{M[W/z]}.$$

Proof: Immediate from definitions. □

From the fact that the semantics of $\mu z.\phi$ and $\nu z.\phi$ are defined as fixpoints of the functional f mapping $W \subseteq \mathrm{st}(M)$ to $\|\phi\|_{M[W/z]}$ it is easy to see validity of the following lemma.

Lemma 2.2.10 For any linear model M and any μTL-formula ϕ,

$$\begin{aligned} \|\mu z.\phi\|_M &= \|\phi[\mu z.\phi/z]\|_M \\ \|\nu z.\phi\|_M &= \|\phi[\nu z.\phi/z]\|_M \end{aligned}$$

Proof: Easy. □

Let us give a particular name to this operation of replacing all free occurrences of the fixpoint variable in the body of a fixpoint formula by the fixpoint formula itself.

Definition 2.2.11 The *unfolding* of a fixpoint formula $\sigma z.\phi$ is the formula $\phi[\sigma z.\phi/z]$. □

Let us also define the concept of the closure set of a formula, which is like the set of subformulae except that fixpoints are unfolded.

Definition 2.2.12 The *closure* of a formula ϕ, denoted $\mathrm{cl}(\phi)$, is the minimal set of formulae that contains ϕ and fulfils:

- if $\psi \wedge \psi' \in \mathrm{cl}(\phi)$ then $\psi, \psi' \in \mathrm{cl}(\phi)$,
- if $\psi \vee \psi' \in \mathrm{cl}(\phi)$ then $\psi, \psi' \in \mathrm{cl}(\phi)$,
- if $\neg\psi \in \mathrm{cl}(\phi)$ then $\psi \in \mathrm{cl}(\phi)$,
- if $\bigcirc\psi \in \mathrm{cl}(\phi)$ then $\psi \in \mathrm{cl}(\phi)$, and
- if $\sigma z.\psi \in \mathrm{cl}(\phi)$ then $\psi[\sigma z.\psi/z] \in \mathrm{cl}(\phi)$. □

By the Knaster-Tarski fixpoint theorem, the least fixpoint of the functional f can be determined by defining $W_0 = \emptyset$, $W_1 = f(W_0) = f(\emptyset)$, $W_2 = f(W_1) = f(f(\emptyset))$ and so on, until we reach the first W_α such that $W_\alpha = f(W_\alpha)$. The number of iterations required depends on the size of the model M and the structure of the formula ϕ. In the general case, iteration up to an arbitrary ordinal is required.

However, as here all models M are countable, we only need iteration up to the first uncountable ordinal.

To formulate this idea precisely, let us define the notion of a fixpoint approximant for minimal and maximal fixpoints. As a technical tool we need to introduce a variant of the mu-calculus with infinitary disjunction and conjunction.

Definition 2.2.13 We use Ord to denote the class of ordinals, $\prec$ their standard ordering, $\omega = \omega_0$ the first infinite ordinal, and ω_1 the first uncountable ordinal. □

Definition 2.2.14 The language of *infinitary* μTL extends μTL by an infinitary conjunction operator $\bigwedge_{i\in I}\phi_i$, where I is any class. The semantics of the new operator are defined by

$$\|\bigwedge_{i\in I}\phi_i\|_M = \bigcap_{i\in I}\|\phi_i\|_M$$

Define also the derived operator $\bigvee_{i\in I}\phi_i = \neg\bigwedge_{i\in I}\neg\phi_i$. □

Definition 2.2.15 Let us introduce the derived *fixpoint approximant* operators $\mu^\alpha z.\phi$ and $\nu^\alpha z.\phi$, for every ordinal $\alpha \in$ Ord, the semantics of which are defined by the inductive translations:

$$\begin{aligned}
\mu^0 z.\phi &= \bot\\
\mu^{\alpha+1} z.\phi &= \phi[\mu^\alpha z.\phi/z]\\
\mu^\lambda z.\phi &= \bigvee_{\alpha\prec\lambda}\mu^\alpha z.\phi\\
\nu^0 z.\phi &= \top\\
\nu^{\alpha+1} z.\phi &= \phi[\nu^\alpha z.\phi/z]\\
\nu^\lambda z.\phi &= \bigwedge_{\alpha\prec\lambda}\nu^\alpha z.\phi
\end{aligned}$$

where λ is a limit ordinal. □

Proposition 2.2.16 For any linear model $M \in \mathcal{M}_{TL}$ and any μTL or infinitary μTL formulae $\mu z.\phi$ and $\nu z.\phi$,

$$\|\mu z.\phi\|_M = \|\bigvee_{\alpha\in\mathrm{Ord}}\mu^\alpha z.\phi\|_M = \|\bigvee_{\alpha\prec\omega_1}\mu^\alpha z.\phi\|_M = \|\mu^{\omega_1} z.\phi\|_M$$

$$\|\nu z.\phi\|_M = \|\bigwedge_{\alpha\in\mathrm{Ord}}\nu^\alpha z.\phi\|_M = \|\bigwedge_{\alpha\prec\omega_1}\nu^\alpha z.\phi\|_M = \|\nu^{\omega_1} z.\phi\|_M$$

Proof: Immediate from the Knaster-Tarski fixpoint theorem [90] and the fact that the model M is countable. □

Corollary 2.2.17 Let M be a linear model, s a state of M and $\mu z.\phi$ a minimal fixpoint formula of μTL or infinitary μTL. We have $M, s \models \mu z.\phi$ iff there is an ordinal $\alpha \prec \omega_1$ such that $M, s \models \mu^\alpha z.\phi$, and this holds iff for every ordinal $\beta \prec \alpha$ there is a set W_β such that

- $s \in W_\alpha$
- $W_0 = \emptyset$
- $W_{\beta+1} \subseteq \|\phi\|_{M[W_\beta/z]}$ for every $\beta \prec \alpha$
- $W_\lambda \subseteq \bigcup_{\beta \prec \lambda} W_\beta$ for every limit ordinal $\lambda \prec \alpha$.

Proof: Immediate from 2.2.16. □

Example 2.2.18 We can use approximants to substantiate the earlier claim that $\mu z.a \vee \bigcirc z$ corresponds to $\mathbf{F}a$ and $\nu z.a \wedge \bigcirc z$ to $\mathbf{G}a$. We have:

$$\begin{array}{rcl}
\mu^0 z.a \vee \bigcirc z & = & \bot \\
\mu^1 z.a \vee \bigcirc z & = & (a \vee \bigcirc z)[\bot/z] = a \vee \bigcirc\bot \Leftrightarrow a \\
\mu^2 z.a \vee \bigcirc z & = & (a \vee \bigcirc z)[(a \vee \bigcirc\bot)/z] = a \vee \bigcirc(a \vee \bigcirc\bot) \Leftrightarrow a \vee \bigcirc a \\
 & \vdots & \\
\mu^{n+1} z.a \vee \bigcirc z & = & (a \vee \bigcirc z)[(\mu^n z.a \vee \bigcirc z)/z] \Leftrightarrow \bigvee_{i=0}^{n} \bigcirc^i a \\
\mu^\omega z.a \vee \bigcirc z & \Leftrightarrow & \bigvee_{i \in \mathbb{N}} \bigcirc^i a \\
\mu^{\omega+1} z.a \vee \bigcirc z & = & (a \vee \bigcirc z)[(\mu^\omega z.a \vee \bigcirc z)/z] \Leftrightarrow a \vee \bigcirc \bigvee_{i \in \mathbb{N}} \bigcirc^i a \Leftrightarrow \bigvee_{i \in \mathbb{N}} \bigcirc^i a
\end{array}$$

Since $\mu^\omega z.a \vee \bigcirc z \Leftrightarrow \mu^{\omega+1} z.a \vee \bigcirc z$, we have

$$\mu z.a \vee \bigcirc a \ \Leftrightarrow\ \mu^\omega z.a \vee \bigcirc z \ \Leftrightarrow\ \bigvee_{i \in \mathbb{N}} \bigcirc^i a \ \Leftrightarrow\ \mathbf{F}a$$

Similarly

$$\begin{array}{rcl}
\nu^0 z.a \wedge \bigcirc z & = & \top \\
\nu^1 z.a \wedge \bigcirc z & \Leftrightarrow & a \\
\nu^2 z.a \wedge \bigcirc z & \Leftrightarrow & a \wedge \bigcirc a \\
 & \vdots & \\
\nu^{n+1} z.a \wedge \bigcirc z & \Leftrightarrow & \bigwedge_{i=1}^{n} \bigcirc^i a \\
\nu z.a \wedge \bigcirc z \ \Leftrightarrow\ \nu^\omega z.a \wedge \bigcirc z & \Leftrightarrow & \bigwedge_{i \in \mathbb{N}} \bigcirc^i a \ \Leftrightarrow\ \mathbf{G}a
\end{array}$$

□

Example 2.2.19 To give an example of a situation where ordinals beyond ω are required in the approximants, let $\mu z.\phi$ be the following formula

$$\mu z.\, (a \wedge \bigcirc(\nu x.(a \vee z) \wedge \bigcirc x) \;\vee\; \neg a \wedge \bigcirc(a \vee z))$$

and let M be the linear model defined by

$$M(s) = \begin{cases} \{a\} & \text{if } s = 2^i \text{ for some } i \in \mathbb{N} \\ \emptyset & \text{otherwise} \end{cases}$$

Notice that

$$\|\mu^1 z.\phi\|_M = \{s \in \mathbb{N} \mid \exists i \in \mathbb{N} \text{ such that } s = 2^i - 1 \text{ and } s > 2^{i-1}\}$$

and that for every $n \in \mathbb{N}$,

$$\|\mu^n z.\phi\|_M = \{s \in \mathbb{N} \mid \exists i \in \mathbb{N} \text{ such that } 2^i > s \geq 2^i - n \text{ and } s > 2^{i-1}\}$$

which implies

$$\|\mu^\omega z.\phi\|_M = \{s \in \mathbb{N} \mid \exists i \in \mathbb{N} \text{ such that } 2^i > s > 2^{i-1}\}$$

However,

$$\|\mu z.\phi\|_M = \|\mu^{\omega+1} z.\phi\|_M = \mathbb{N}$$

□

2.2.3 Fixpoint alternation

In recursion theory the alternation of existential and universal quantification results in the arithmetical and analytical hierarchies. A related phenomenon takes place in mu-calculi in the form of alternation of minimal and maximal fixpoints. Besides the theoretical questions related to fixpoint alternation, such as whether the alternation depth hierarchy is proper or not, the issue has also practical significance, as fixpoint alternation appears to be a key ingredient in the model-checking problem.

The simplest way we can approach alternation is by just looking at the syntactic nesting of fixpoint formulae. To make it clearer what constitutes a maximal and what a minimal fixpoint inside a formula, we use a particular normal form where all negations are pushed inwards in a formula so that they only apply to atomic propositions.

Definition 2.2.20 A μTL-formula ϕ is in the *positive normal form* (abbr. *pnf*) iff it only contains atomic propositions, their negations, and the $\top, \bot, \vee, \wedge, \bigcirc, \mu$ and ν-operators.

If ϕ is a formula, pnf(ϕ) is the unique formula in positive normal form obtained from ϕ by pushing negations inwards using DeMorgan's laws and fixpoint dualities:

$$\begin{aligned}
\neg\top &= \bot \\
\neg\bot &= \top \\
\neg\neg\phi &= \phi \\
\neg(\phi \wedge \psi) &= \neg\phi \vee \neg\psi \\
\neg(\phi \vee \psi) &= \neg\phi \wedge \neg\psi \\
\neg\bigcirc\phi &= \bigcirc\neg\phi \\
\neg\mu z.\phi &= \nu z.\neg\phi[\neg z/z] \\
\neg\nu z.\phi &= \mu z.\neg\phi[\neg z/z]
\end{aligned}$$

□

For example,

$$\text{pnf}(\neg\nu z.\mu x.(a \wedge \bigcirc z) \vee \bigcirc x) \;=\; \mu z.\nu x.(\neg a \vee \bigcirc z) \wedge \bigcirc x$$

In the following we often assume that formulae are in the positive normal form, effectively considering them as being generated by the abstract syntax

$$\phi ::= z \mid \neg z \mid \top \mid \bot \mid \phi \wedge \phi' \mid \phi \vee \phi' \mid \bigcirc\phi \mid \mu z.\phi \mid \nu z.\phi$$

When proving results by induction on the structure of formulae, we sometimes implicitly assume a translation to pnf, and sometimes do the induction directly on the basic operators $\neg$, $\wedge$ and μz, whichever happens to be easier.

Examining a formula ϕ is positive normal form, the syntactic notion of alternation is based on considering sequences $\phi \trianglelefteq \sigma_1 x_1.\psi_1 \lhd \sigma_2 x_2.\psi_2 \lhd \ldots \lhd \sigma_n x_n.\psi_n$ where every $\sigma_i \neq \sigma_{i+1}$. Alternation classes based on this notion can be defined as closure classes in the following way.

Definition 2.2.21 Let us define for every $n \in \mathbb{N}$ the *syntactic fixpoint alternation classes* Σ_n^{stx}, Π_n^{stx} and Δ_n^{stx} of μTL-formulae in positive normal form as follows. The class $\Sigma_0^{stx} = \Pi_0^{stx}$ is the set of all formulae in pnf without any fixpoint operators. For every $n > 0$, the class Σ_n^{stx} is the least set of formulae in pnf such that it contains $\Sigma_{n-1}^{stx} \cup \Pi_{n-1}^{stx}$ and fulfils:

1 if $\phi, \psi \in \Sigma_n^{stx}$, then $\phi \wedge \psi, \phi \vee \psi, \bigcirc\phi, \mu z.\phi \in \Sigma_n^{stx}$.

The class Π_n^{stx} is defined analogously, by changing μ to ν in condition 1. For every $n \in \mathbb{N}$, define $\Delta_n^{stx} = \Sigma_n^{stx} \cap \Pi_n^{stx}$.

A formula ϕ which is not in positive normal form belongs to a given alternation class iff pnf(ϕ) does. The same convention applies to the other notions of alternation introduced in the following, as well. □

For example,

$$\begin{array}{rcl} \mu z.a \vee \bigcirc z & \in & \Sigma_1^{stx} \\ \nu x.(\mu z.a \vee \bigcirc z) \wedge \bigcirc x & \in & \Pi_2^{stx} \\ \mu x.\nu z.a \wedge \bigcirc z \vee \bigcirc x & \in & \Sigma_2^{stx} \end{array}$$

and these are the lowest classes in the hierarchy the formulae belong to. It is easy to see that any $\phi \in \Sigma_n^{stx}$ iff there are no $\sigma_1 x_1.\phi_1, \ldots, \sigma_n x_n.\phi_n$ such that $\phi \trianglelefteq \sigma_1 x_1.\psi_1 \triangleleft \ldots \triangleleft \sigma_n x_n.\psi_n$ where $\sigma_i = \nu$ if i is odd, and $\sigma_i = \mu$ if i is even.

In some respects this syntactic view gives a rather rough way of measuring the alternation present in a formula. For example, according to this view, the formula $\nu x.(\mu z.a \vee \bigcirc z) \wedge \bigcirc x$, *infinitely often a* or $\mathbf{GF}a$ in *TL*, would have one level of alternation, since it has a minimal fixpoint within a maximal one. However, although $\mu z.a \vee \bigcirc z$ occurs syntactically in the scope of νx, it is not semantically dependent on x, i.e. $\|\mu z.a \vee \bigcirc z\|_M$ is independent of where x is true in M. This suggests that when defining alternation we should not only look at syntactic embedding, but also at actual dependencies between fixpoints. In other words, when determining the level of alternation present in a formula, we should look at sequences $\phi \trianglelefteq \sigma_1 x_1.\psi_1 \triangleleft \ldots \triangleleft \sigma_n x_n.\psi_n$ where every $\sigma_i \neq \sigma_{i+1}$ and each $\sigma_{i+1} x_{i+1}.\psi_{i+1}$ depends on x_i.

One idea in this direction, due to Emerson and Lei, is to consider subsentences or closed subformulae separately when determining the alternation depth of a formula [34]; since closed formulae have no free variables that could be bound by fixpoints, such subformulae clearly cannot semantically depend on any outer fixpoint variable. To formulate the notion of a closed formula in the present framework, we need to introduce a division of atomic propositions to constants and variables.

Definition 2.2.22 Let us divide the set $\mathcal{Z}$ of propositions into two disjoint parts, the set of *propositional constants* $\mathcal{Z}_c$ and the set of *propositional variables* $\mathcal{Z}_v$, and decree that only propositions of the latter kind can be bound by fixpoint operators. Let us call a formula ϕ *closed* iff it has no free occurrences of any variable $z \in \mathcal{Z}_v$.

Define the *Emerson-Lei fixpoint alternation classes* Σ_n^{el}, Π_n^{el} and Δ_n^{el} as follows. The class $\Sigma_0^{el} = \Pi_0^{el}$ is the set of all formulae in pnf without any fixpoint operators.

For every $n > 0$, the class Σ_n^{el} is the least set of formulae in pnf such that it contains $\Sigma_{n-1}^{el} \cup \Pi_{n-1}^{el}$ and fulfils:

1 if $\phi, \psi \in \Sigma_n^{el}$, then $\phi \wedge \psi, \phi \vee \psi, \bigcirc\phi, \mu z.\phi \in \Sigma_n^{el}$.

2 if $\phi \in \Sigma_n^{el}$ and $z_1, \ldots, z_m$ are distinct variables that occur only positively in ϕ and $\psi_1, \ldots, \psi_m \in \Sigma_n^{el}$ are closed formulae, then $\phi[\psi_1/z_1, \ldots, \psi_m/z_m] \in \Sigma_n^{el}$.

The class Π_n^{el} is defined analogously, by changing μ to ν in condition 1. For every $n \in \mathbb{N}$, define $\Delta_n^{el} = \Sigma_n^{el} \cap \Pi_n^{el}$. □

The way the notion of alternation is technically defined by Emerson and Lei in [34] is different from the treatment here; when determining the alternation depth of a formula, they first remove all closed subsentences from it and then count alternation syntactically. Furthermore, they only count the depth of alternation without paying attention to which way the alternation goes, i.e. not distinguishing between $\mu z.\nu x.\phi$ and $\nu z.\mu x.\phi$. Nevertheless, despite these technical differences it is rather easy to see that the class Δ_{n+1}^{el} here is precisely the same as the class $L\mu_n$ of formulae with alternation depth at most n in the terminology and notation of Emerson and Lei. For an alternative formulation of the same alternation classes as above, due to Bradfield, see [11, Def. 2.30].

Although the Emerson-Lei definition of alternation is widely used, especially in the modal mu-calculus research community, it still does not fully capture the idea that alternation should reflect the nesting of minimal fixpoints within maximal ones, and vice versa, with the inner fixpoint depending on the outer. For example, consider the formula

$$\nu z.\mu x.a \wedge \bigcirc x \ \vee \ (\nu y.c \wedge \bigcirc y \ \vee \ a \wedge \bigcirc z)$$

Notice that both the fixpoints νy and μx depend on z, since z occurs in the scope of both of these, whereas νy does not depend on x although it occurs in the scope of μx. Nevertheless, since neither of the inner fixpoints is closed, we would still regard the formula as having two levels of alternation.

So far we have talked about fixpoints depending on outer fixpoint variables without actually making precise what is meant by 'depend'. Naturally a formula depends on a variable that occurs in it. However, this is not the whole truth. For example, in formula

$$\nu z.\mu x.a \wedge \bigcirc z \ \vee \ (\mu y.b \wedge \bigcirc y \ \vee \ c \wedge \bigcirc x)$$

the fixpoint μy does not contain the variable z. Yet in the context of the whole formula also this fixpoint depends indirectly on z, since μy contains an occurrence of and therefore depends on x, and μx in turn similarly depends on z. To capture this notion of dependence, we introduce the concept of *activeness*. Intuitively this can be viewed an the transitive closure of 'occurring free in'.

Definition 2.2.23 Fix a μTL-formula ϕ. Let ψ be a subformula of ϕ and let z be a variable. We say that z is *active in* ψ (in the context of ϕ) iff either

- there is a free occurrence of z in ψ, or
- there is some fixpoint subformula $\sigma z'.\psi'$ of ϕ such that
  - z is active in $\sigma z'.\psi'$,
  - ψ is a subformula of $\sigma z'.\psi'$, and
  - there is a free occurrence of z' in ψ. □

Rephrasing the intuition in determining the level of alternation in a formula, we are interested in sequences $\phi \trianglelefteq \sigma_1 x_1.\psi_1 \triangleleft \ldots \triangleleft \sigma_n x_n.\psi_n$ where every $\sigma_i \neq \sigma_{i+1}$ and each x_i is active in $\sigma_{i+1}x_{i+1}.\psi_{i+1}$. This intuition is captured precisely in the following elegant characterisation of alternation classes, due to Niwiński [70]. The definition says simply that an alternation class contains all the lower classes in the hierarchy and is closed under usual substitutions and prefixing with the relevant fixpoint. Let us state the definition first and show then that it indeed corresponds to the intended intuition.

Definition 2.2.24 Let us define the *fixpoint alternation classes* Σ_n, Π_n and Δ_n as follows. The class $\Sigma_0 = \Pi_0$ is the set of all formulae in pnf without any fixpoint operators. For every $n > 0$, the class Σ_n is the least set of formulae in pnf such that it contains $\Sigma_{n-1} \cup \Pi_{n-1}$ and fulfils:

1 if $\phi \in \Sigma_n$, then $\mu z.\phi \in \Sigma_n$

2 if $\phi, \psi_1, \ldots, \psi_m \in \Sigma_n$ and $z_1, \ldots, z_m$ are distinct variables that occur only positively in ϕ, then $\phi[\psi_1/z_1, \ldots, \psi_m/z_m] \in \Sigma_n$.

The class Π_n is defined analogously, by changing μ to ν in condition 1. For every $n \in \mathbb{N}$, define $\Delta_n = \Sigma_n \cap \Pi_n$. □

Notice that we have dropped the conditions

$$\text{if } \phi, \psi \in \Sigma_n, \text{ then } \phi \wedge \psi, \phi \vee \psi, \bigcirc\phi \in \Sigma_n$$

that were present in earlier definitions of alternation classes. This causes no harm, since they are subsumed by clause 2 above.

Proposition 2.2.25 For every $n \in \mathbb{N}$ and every μTL-formula ϕ in pnf, $\phi \in \Sigma_n$ iff there are no $\sigma_1 x_1.\psi_1, \ldots, \sigma_n x_n.\psi_n$ such that

1 $\phi \trianglelefteq \sigma_1 x_1.\psi_1 \triangleleft \ldots \triangleleft \sigma_n x_n.\psi_n$,

2 x_i is active in $\sigma_{i+1}x_{i+1}.\psi_{i+1}$ (in the context of ϕ) for every $1 \leq i < n$, and

3 $\sigma_i = \nu$ if i is odd, and $\sigma_i = \mu$ if i is even, for every $1 \leq i \leq n$.

Symmetrically, $\phi \in \Pi_n$ iff there are no $\sigma_1 x_1.\psi_1, \ldots, \sigma_n x_n.\psi_n$ fulfilling 1 and 2 above and

3' $\sigma_i = \mu$ if i is odd, and $\sigma_i = \nu$ if i is even, for every $1 \leq i \leq n$.

Proof: The claim is shown by induction on n. For $n = 0$ the claim holds trivially. Assume then that the claim holds for both Σ_{n-1} and Π_{n-1}. Let us show that it holds for Σ_n. The proof for Π_n is symmetric.

To show the claim for Σ_n from left to right, notice first that by induction assumption, for every $\phi \in \Sigma_{n-1}$ there is no sequence of fixpoints $\sigma_1 x_1.\psi_1, \ldots$ such that clauses 1-3 (w.r.t. $n-1$) would hold, and for every $\phi \in \Pi_{n-1}$ there is no sequence such that clauses 1-3' (w.r.t $n-1$) would hold. This means that for there is no sequence fulfilling 1-3 (w.r.t. n) for any $\phi \in \Sigma_{n-1} \cup \Pi_{n-1}$. By the definition of Σ_n it suffices to show that the two operations Σ_n is closed under, i.e. minimal fixpoints and substitution, preserve the nonexistence of a sequence fulfilling clauses 1-3. It is clear that if there is no fixpoint sequence fulfilling 1-3 for ϕ then there is not any for $\mu z.\phi$ either. As no occurrence of any variable in any ψ_i is captured by a fixpoint of ϕ in $\phi[\psi_1/z_1, \ldots, \psi_n/z_n]$, any fixpoint sequence fulfilling 1-3 in $\phi[\psi_1/z_1, \ldots, \psi_n/z_n]$ would already occur in either ϕ or in some ψ_i, i.e. the substitution preserves the nonexistence of such a sequence, as well.

Proving the induction step for Σ_n from right to left, i.e. that if there is no fixpoint sequence fulfilling 1-3 for some ϕ then $\phi \in \Sigma_n$, is done inductively itself, by induction on the length of the formula ϕ. The base case, when ϕ is atomic, is trivial. Assume then that for all ϕ with $|\phi| \leq m$ we know that if there is no sequence fulfilling 1-3 for ϕ, then $\phi \in \Sigma_n$. Take now some ϕ such that $|\phi| = m+1$. If ϕ is of the forms $\psi \vee \psi'$, $\psi \wedge \psi'$, $\bigcirc\psi$ or $\mu z.\psi$ the induction step is immediate.

Assume then that ϕ is of the form $\nu z.\psi$ and there is no sequence fulfilling 1-3 for ϕ. Notice that this is only possible if $n \geq 2$. Take every fixpoint subformula $\sigma y_i.\chi_i$ of ψ such that z is not active in $\sigma y_i.\chi_i$ (in the context of ψ), but z is active in every $\sigma y'.\chi'$ such that $\psi \trianglelefteq \sigma y'.\chi' \triangleleft \sigma y_i.\chi_i$, and replace $\sigma y_i.\chi_i$ in ψ by a fresh variable v_i. Denote the resulting formula χ. Now $\phi = (\nu z.\chi)[\sigma y_1.\chi_1/v_1, \ldots, \sigma y_k.\chi_k/v_k]$. Since $|\sigma y_i.\chi_i| \leq m$, we know by induction assumption that $\sigma y_i.\chi_i \in \Sigma_n$ for every $1 \leq i \leq k$. As there is no sequence fulfilling 1-3 for ϕ, there is not any for $\nu z.\chi$ either. Since by definition of χ, z is active in every fixpoint subformula of χ, this means that there cannot be any sequence fulfilling 1-3' (w.r.t. $n-1$) for χ. By the main induction assumption this means $\chi \in \Pi_{n-1}$, implying $\nu x.\chi \in \Pi_{n-1} \subseteq \Sigma_n$. Consequently $\phi = (\nu z.\chi)[\sigma y_1.\chi_1/v_1, \ldots, \sigma y_k.\chi_k/v_k] \in \Sigma_n$, as claimed. $\square$

From this we can get natural characterisations of the alternation classes Σ_2 and Π_2.

Corollary 2.2.26 For every μTL-formula ϕ in pnf, $\phi \in \Sigma_2$ iff there are no $\nu x.\psi$ and $\mu z.\chi$ such that

- $\phi \trianglelefteq \nu x.\psi \triangleleft \mu z.\chi$, and
- x occurs free in $\mu z.\chi$

Symmetrically, $\phi \in \Pi_2$ iff there are no $\mu x.\psi$ and $\nu z.\chi$ such that

- $\phi \trianglelefteq \mu x.\psi \triangleleft \nu z.\chi$, and
- x occurs free in $\nu z.\chi$

Proof: Immediate from 2.2.25. □

This characterisation implies that in the alternation class Δ_2 there is no proper alternation between fixpoints at all. Since we will study such formulae in several occasions in the following, let us give them a particular name.

Definition 2.2.27 We call a formula ϕ *non-alternating* iff $\phi \in \Delta_2$. □

It is easy to see that for all three notions of alternation, the Π and Σ classes are closed under $\vee$, $\wedge$ and $\bigcirc$, and the Δ classes are additionally closed under negation. The following proposition helps to relate the different notions of alternation to each other.

Proposition 2.2.28 For every $n \in \mathbb{N}$,

$$\Sigma_n^{stx} \subseteq \Sigma_n^{el} \subseteq \Sigma_n$$

$$\Pi_n^{stx} \subseteq \Pi_n^{el} \subseteq \Pi_n$$

and for $n \geq 2$ the inclusions are proper.

Proof: The inclusions are immediate from definitions. To see that the inclusions are proper, fix some $n \geq 2$, and define:

$$\begin{array}{rcl}
\phi & = & \mu x_0.a_0 \wedge \bigcirc x_0 \ \vee\ c \wedge \bigcirc \phi_1 \\
\phi_i & = & \left\{ \begin{array}{ll} \nu x_i.a_i \wedge \bigcirc x_i \ \vee\ b \wedge \bigcirc \phi_{i+1} \ \vee\ c \wedge \bigcirc x_{i-1} & \text{if } i \text{ is odd} \\ \mu x_i.a_i \wedge \bigcirc x_i \ \vee\ b \wedge \bigcirc \phi_{i+1} \ \vee\ c \wedge \bigcirc x_{i-1} & \text{if } i \text{ is even} \end{array} \right. \quad \text{for } 1 \leq i < n \\
\phi_n & = & x_0 \vee w \\
\psi & = & \mu y_0.\nu y_1.e \wedge \bigcirc y_1 \ \vee\ d \wedge \bigcirc y_0 \\
\psi' & = & \mu y_0.\nu y_1.e \wedge \bigcirc y_1 \ \vee\ d \wedge \bigcirc y_0 \ \vee\ f \wedge \bigcirc x_0
\end{array}$$

It is easy to see that $\phi, \psi \in \Sigma_n^{stx} \subseteq \Sigma_n^{el}$. Since ψ is closed (under the assumption that d and e are constants), we have then $\phi[\psi/_t w] = \phi[\psi/w] \in \Sigma_n^{el}$. However,

$\phi[\psi/_t w] \notin \Sigma_n^{stx}$ since $\phi[\psi/w]$ has at least $n+1$ levels of syntactic alternation. Since $\phi_1, \psi' \in \Sigma_n$, we have $\phi_1[\psi'/_t w] = \phi_1[\psi'/w] \in \Sigma_n$ and

$$\phi[\psi'/_t w] \;=\; \mu x_0.a_0 \wedge \bigcirc x_0 \vee c \wedge \bigcirc(\phi_1[\psi'/_t w]) \;\in\; \Sigma_n$$

However, since ψ' is not closed, the substitution is not allowed in Σ_n^{el}, and $\phi[\psi'/_t w] \notin \Sigma_n^{el}$. The claim concerning the Π alternation classes is shown symmetrically. □

All the notions of alternation above are in a sense syntactic, based on a classification of formulae. We can also take a semantic view, classifying properties rather than formulae, and measure the complexity of a property by the level of alternation that is needed to characterise it by any formula. It is clear that all the hierarchies of alternation classes defined above are proper, i.e. on each level there are formulae which do not belong to any lower level. However, the same question for the semantic notion, i.e. are there properties which can be characterised by some formula in Σ_n but not by any in Σ_{n-1}, is highly nontrivial. We shall return to the issue later in Chapter 3.

2.2.4 Tableaux for linear structures

The semantics of linear-time mu-calculus was defined in Definition 2.2.4 in a way that is rather global and non-constructive. In particular, from the semantics of the minimal fixpoint $\mu z.\phi$ it is not easy to determine whether $M, i \models \phi$ for a given state i and model M, without determining this for every state of the model at the same time.

In this subsection, we give an alternative account of what it means for a formula to be true in a state of a model, based on the idea of semantic tableaux. The intuition behind this can be roughly described as follows.

Assume that we have a model M, a state s of M and a formula ϕ. The aim is to determine whether $M, s \models \phi$ or not. Without loss of generality, we may assume that ϕ is in positive normal form. If ϕ is an atomic proposition or a negation of such, we can immediately find out whether $M, s \models \phi$ by just looking at the set $M(s)$. If ϕ is of the form $\psi \wedge \psi'$, we can divide the task of of determining whether $M, s \models \phi$ holds to the subtasks of determining whether $M, s \models \psi$ and $M, s \models \psi'$ hold. Similarly, if $\phi = \psi \vee \psi'$, it suffices to check whether either $M, s \models \psi$ or $M, s \models \psi'$ holds, and if $\phi = \bigcirc\psi$, we know that $M, s \models \psi$ iff $M, s+1 \models \psi$.

However, for the fixpoint operators μz and νz there is no such direct decomposition. Instead, a natural idea for analysing fixpoints is to consider their unfoldings; to find out whether $M, s \models \mu z.\psi$ holds, determine whether $M, s \models \psi[\mu z.\psi/z]$ holds, and similarly for the maximal fixpoint νz.

Assume that we are interested in finding out whether $M, s \models \nu z.(a \wedge \bigcirc\bigcirc z)$ holds. Unfolding the fixpoint, we can reduce this to determining whether $M, s \models a \wedge \bigcirc\bigcirc \nu z.(a \wedge \bigcirc\bigcirc z)$ holds, i.e. whether both $M, s \models a$ and $M, s \models \bigcirc\bigcirc \nu z.(a \wedge \bigcirc\bigcirc z)$ hold. The former of these can be checked directly by $a \in M(s)$, and the latter reduces to $M, s+2 \models \nu z.(a \wedge \bigcirc\bigcirc z)$. Repeating the process, this reduces to $a \in M(s+2)$ and $M, s+4 \models \nu z.(a \wedge \bigcirc\bigcirc z)$, which reduces again to $a \in M(s+4)$ and $M, s+6 \models \nu z.(a \wedge \bigcirc\bigcirc z)$, and so on ad infinitum. If there is some n such that $a \notin M(s+2n)$, we see that $M, s \not\models \nu z.(a \wedge \bigcirc\bigcirc z)$. Otherwise, if $a \in M(s+2n)$ for all $n \in \mathbb{N}$, the construction effectively shows that if we define $W = \{s' \in \mathbb{N} \mid s' - s \text{ is even}\}$, we have $W \subseteq \|a \wedge \bigcirc\bigcirc z\|_{M[W/z]}$. As stated in Lemma 2.2.9, this guarantees $M, s \models \nu z.a \wedge \bigcirc\bigcirc z$.

Notice that to show $M, s \models \nu z.a \wedge \bigcirc\bigcirc z$ by this construction, it was enough to find some pre-fixpoint W such that $s \in W$, and there was no need to compute the actual maximal fixpoint. For instance, regarding the truth of $M, s \models \nu z.a \wedge \bigcirc\bigcirc z$, it is irrelevant whether the actual maximal fixpoint contains any of those $s' \in \mathbb{N}$ for which $s' - s$ is odd, and we do not need to consider these states.

Unfortunately the same strategy does not work for minimal fixpoints. If we wanted to use the direct characterisation given by Lemma 2.2.9 for $\mu z.\phi$, we would need to show that s belongs to *every* post-fixpoint of the functional corresponding to ϕ. Instead, it is easier to base the approach on fixpoint approximants, as spelt out in Corollary 2.2.17.

Assume that we want to determine whether $M, s \models \mu z.a \vee \bigcirc z$ holds. Unfolding the fixpoint, this reduces to $M, s \models a \vee \bigcirc \mu z.a \vee \bigcirc z$, and further either to $a \in M(s)$ or $M, s+1 \models \mu z.a \vee \bigcirc z$. If we choose the second disjunct, this reduces in the same way either to $a \in M(s+1)$ or $M, s+2 \models \mu z.a \vee \bigcirc z$, and choosing here the second disjunct repeats the process again.

If there exists some $n \in \mathbb{N}$ such that $a \in M(s+n)$, then it is possible to make the choices between the disjuncts in this process so that at the n-th round we choose to verify $a \in M(s+n)$. In this case we know that $M, s+n \models a \vee \bigcirc\bot$, i.e. that $\{s+n\} \subseteq \|\mu^1 z.a \vee \bigcirc z\|_M$. On the basis of this, the construction guarantees that $\{s+n-1\} \subseteq \|\mu^2 z.a \vee \bigcirc z\|_M$, and more generally that $\{s+n-i\} \subseteq \|\mu^{i+1} z.a \vee \bigcirc z\|_M$ for all $0 \leq i \leq n$. Since this means that $\{s\} \subseteq \|\mu^{n+1} z.a \vee \bigcirc z\|_M \subseteq \|\mu z.a \vee \bigcirc z\|_M$, we know that $M, s \models \mu z.a \vee \bigcirc z$ holds.

What we have effectively constructed here is an ordinal α, in this case $n+1$, and a collection of sets $W_0, W_1, \ldots, W_\alpha$ as described in Corollary 2.2.17. What is analogous to the case of maximal fixpoints is that we do not need to compute the entire sets $\|\mu^\beta z.a \vee \bigcirc z\|_M$ nor the entire set $\|\mu z.a \vee \bigcirc z\|_M$, but only enough of these to justify $M, s \models \mu z.a \vee \bigcirc z$.

These intuitions are formalised in the concept of a tableau below. To keep track of the unfolding of different fixpoints, we use the technical tool of definition constants and definition lists, introduced by Stirling and Walker [85].

Definition 2.2.29 Fix a set $\mathcal{U}$ of *definition constants*. The notion of an *extended μTL-formula* is as that of a μTL-formula, but allowing definition constants in place of free atomic propositions. A *definition list* is a finite sequence

$$d = (u_0, \sigma z_0.\phi_0) \ldots (u_k, \sigma z_k.\phi_k)$$

where

- every $u_i \in \mathcal{U}$ and $\sigma z_i.\phi_i$ is an extended fixpoint formula,
- all u_i are distinct, and
- if a constant u occurs in ϕ_i, then $u = u_j$ for some $j < i$.

For every u_i, define $d(u_i) = \sigma z_i.\phi_i$. We call u a *maximal* or *minimal* fixpoint constant of d depending on whether $d(u) = \nu z.\phi$ or $d(u) = \mu z.\phi$.

We say that a constant u_i is *active* in ϕ (relative to the definition list d) iff either

- u_i occurs in ϕ, or
- there is some u_j, $i < j$, such that u_i occurs in $d(u_j)$ and u_j is active in ϕ.

We say that a variable z is active in ϕ iff either

- z occurs free in ϕ, or
- there is some u such that z occurs free in $d(u)$ and u is active in ϕ.

If ϕ is an extended formula and d a definition list, $\phi[d]$ is defined by

- $\phi[\epsilon] = \phi$ and
- $\phi[d \cdot (u, \psi)] = (\phi[\psi/u])[d]$.

If Γ is a set of extended formulae, $\Gamma[d] = \{\phi[d] \mid \phi \in \Gamma\}$. $\square$

Previously in Definition 2.2.23 we gave an account of activeness of a variable in a formula in terms of fixpoint subformulae. As fixpoints correspond to constants in a definition list, there is a natural correspondence between that account and the definition of activeness above; these are two formulations of the same phenomenon. It would be possible to give an account of alternation purely in terms of definition lists, but as this would have required setting up the whole technical machinery before being able to talk about alternation in formulae, we felt this might have obscured the issue.

name	application	name	application
$\vee\boldsymbol{L}$	$\dfrac{s,\ \ \phi\vee\phi',\ \ d}{s,\ \ \phi,\ \ d}$	$\vee\boldsymbol{R}$	$\dfrac{s,\ \ \phi\vee\phi',\ \ d}{s,\ \ \phi',\ \ d}$
$\wedge$	$\dfrac{s,\ \ \phi\wedge\phi',\ \ d}{s,\ \ \phi,\ \ d \qquad\qquad s,\ \ \phi',\ \ d}$		
$\boldsymbol{\sigma}$	$\dfrac{s,\ \ \sigma z.\phi,\ \ d}{s,\ \ u,\ \ d\cdot(u,\sigma z.\phi)}$ **1**	$\boldsymbol{U}$	$\dfrac{s,\ \ u,\ \ d}{s,\ \ \phi[u/z],\ \ d}$ **2**
$\bigcirc$	$\dfrac{s,\ \ \bigcirc\phi,\ \ d}{s+1,\ \ \phi,\ \ d}$		
Note:	**1:** u does not appear in d **2:** $d(u)=\sigma z.\phi$		

Table 2.1: Simple tableau rules for linear structures

The tableau system described next is essentially the local model-checking tableau system of Stirling and Walker [85], which was extended to deal with infinite systems by Bradfield [11, 14]. The main difference is that the tableaux here are infinite, whereas those used for model-checking are finite. This reflects the fact that tableaux as defined now are used as a means of understanding truth and satisfiability, without computational concerns in mind. The issue of decidability and tableaux of a finite variety will be discussed further in Chapter 4.

Definition 2.2.30 Let ϕ be a μTL-formula in pnf. A *simple tableau* T for ϕ is a finite or infinite tree, every node t of which is labelled with a triple (s_t, ϕ_t, d_t) where

- every $s_t \in \mathbb{N}$, ϕ_t is an extended μTL-formula in pnf, and d_t is a definition list containing all definition constants in ϕ_t,
- the root of T is labelled with $(0, \phi, \epsilon)$,
- the children of every node t of T are derived by applying one of the rules in Table 2.1, and
- a node t of T is a leaf only if no rule can be applied to it.

We assume that every constant is defined at most once in a tableau.

We say that a constant u is a maximal (minimal) fixpoint constant in T iff there is some node t of T such that u is a maximal (minimal) fixpoint constant in d_t. □

So far the definition of a linear tableau does not yet take into account the difference between maximal and minimal fixpoints discussed above. For this we add a well-foundedness requirement for minimal fixpoints in a tableau.

Definition 2.2.31 Let T be a simple tableau and let (s_t, ϕ_t, d_t) denote the label of t for every node t of T.

We say that T is *proper* iff there is no minimal fixpoint constant u of T and infinite path p such that $u = \phi_{p(i)}$ for infinitely many $i \in \mathbb{N}$.

Let $M \in M_{TL}$ be a linear model. We say that T *agrees with* M iff for every leaf t of T,

- $\phi_t[d_t] \neq \bot$,
- if $\phi_t[d_t] = z \in \mathcal{Z}$ then $z \in M(s_t)$, and
- if $\phi_t[d_t] = \neg z \in \overline{\mathcal{Z}}$ then $z \notin M(s_t)$. □

Before showing correctness of the tableau account of truth, let us state a simple technical lemma, which will be used later.

Lemma 2.2.32 Let T be a simple tableau. For every infinite path p of T, there is exactly one constant u such that $u = \phi_{p(i)}$ for infinitely many $i \in \mathbb{N}$.

Proof: Easy. □

Proposition 2.2.33 Let ϕ be a μTL-formula in pnf and M a linear model. Then $M \models \phi$ iff there is a simple tableau T for ϕ such that

- T is proper and
- T agrees with M.

Proof: Tableau constructions closely related to the one above and proofs of their correctness have appeared many times in the literature, see e.g. [85, 11, 87, 95, 59]. However, since we will refer to some aspects of the proof later, we have spelt it out here as well.

Suppose first that $M \models \phi$. We can easily produce a tableau T for ϕ agreeing with M by starting from the triple $(0, \phi, \epsilon)$ and applying tableau rules in any order such that the validity of $M, s_t \models \phi_t[d_t]$ is preserved. However, the tableau produced this way is not necessarily proper.

To guarantee this, let us modify the tableau construction slightly by annotating every occurrence of a minimal constant u in every ϕ_t and in every $d_t(u')$ containing u by an ordinal α. Write u^α for this. The definition of $\psi[d]$ is modified accordingly, by defining $\psi[d \cdot (u, \mu z.\phi)] = \psi'[d]$, where ψ' is obtained from ψ by replacing every occurrence of u^α in it by $\mu^\alpha z.\phi$, for any ordinal α. Furthermore, the tableau rules are modified so that the normal σ and U rules only apply to maximal fixpoints and constants, and minimal fixpoints are taken care of by the following new μ and

U' rules:

$$\boldsymbol{\mu}\quad \frac{s,\quad \mu z.\phi,\quad d}{s,\quad u^{\alpha},\qquad d\cdot(u,\sigma z.\phi)}\quad \begin{array}{l}\text{where } u \text{ is new and}\\ \alpha=\min\{\beta\in\text{Ord}\mid M,s\models(\mu^{\beta}z.\phi)[d]\}\end{array}$$

$$\boldsymbol{U'}\quad \frac{s,\quad u^{\alpha},\qquad d}{s,\quad \phi[u^{\beta}/z],\quad d}\quad \begin{array}{l}\text{where } d(u)=\mu z.\phi \text{ and}\\ \beta=\min\{\beta'\in\text{Ord}\mid M,s\models(\mu^{\beta'}z.\phi)[d]\}-1\end{array}$$

By the Knaster-Tarski fixpoint theorem 2.2.16 we know that if $M, s \models (\mu z.\phi)[d]$ then the α in the σ-rule is well-defined, and that if $M, s \models u^{\alpha}[d]$ then the β is the U'-rule is well-defined and smaller than α.

We can now use the original strategy and apply these modified tableau rules in any order to get a simple tableau T agreeing with M. Since a path with infinitely many occurrences of a minimal constant in this T would correspond to an infinite decreasing sequence of ordinals, by the well-foundedness of these T is proper. To get a normal proper tableau from T just erase the ordinal annotations.

Suppose then that there is a proper simple tableau T for ϕ agreeing with M. Since all the tableau rules are backwards truth-preserving, every non-atomic formula in the tableau will be decomposed by an application of a tableau rule, and T agreeing with M guarantees satisfaction of atomic formulae, the only nontrivial issue in showing $M \models \phi$ is the satisfaction of fixpoints.

For every constant u occurring in T, denote by $U(u)$ the set of nodes t of T such that $\phi_t = u$, and define $W(u) = \{s \in \text{st}(M) \mid \exists t \in U(u) : s_t = s\}$. It is easy to see that for all maximal constants u and all $s \in W(u)$,

$$M[W(u_1)/u_1]\ldots[W(u_n)/u_n], s \models \phi[u/z]$$

where $d(u) = \nu z.\phi$ and $u_1, \ldots, u_n = u$ is the subsequence of constants which are active in u, for any d defining u in T. The fact that all maximal fixpoints are satisfied follows then from Lemma 2.2.9.

For every minimal constant u and every node $t \in U(u)$, define inductively an ordinal $\alpha(u, t)$ by

$$\alpha(u,t) \;=\; (\text{lub}\{\alpha(u,t') \mid t' \in U(u) \text{ and } t \prec t'\}) + 1$$

To see that this indeed defines an ordinal for every u and t, notice that if $\alpha(u, t)$ is not defined, then there must be some descendant t' of t such that $t' \in U(u)$ and $\alpha(u, t')$ is not defined either. Repeating this reasoning we could then construct an infinite path p along which the minimal constant u occurs infinitely often, contrary to the assumption that T is proper. Define then

$$W(u,\beta) = \{s \in \text{st}(M) \mid \exists t \in U(u) : s_t = s \text{ and } \alpha(u,t) = \beta\}$$

$$\begin{array}{cccc}
 & 0, & \mu z.a \vee \bigcirc z, & \epsilon \\ \cline{2-4}
\sigma & & & \\
 & 0, & u, & (u, \mu z.a \vee \bigcirc z) \\ \cline{2-4}
U & & & \\
 & 0, & a \vee \bigcirc u, & (u, \mu z.a \vee \bigcirc z) \\ \cline{2-4}
\vee R & & & \\
 & 0, & \bigcirc u, & (u, \mu z.a \vee \bigcirc z) \\ \cline{2-4}
\bigcirc & & & \\
 & 1, & u, & (u, \mu z.a \vee \bigcirc z) \\ \cline{2-4}
U & & & \\
 & 1, & a \vee \bigcirc, & (u, \mu z.a \vee \bigcirc z) \\ \cline{2-4}
\vee L & & & \\
 & 1, & a, & (u, \mu z.a \vee \bigcirc z)
\end{array}$$

Figure 2.1: A simple tableau for $\mu z.a \vee \bigcirc z$

As above, it is easy to see that for all $s \in W(u, \beta + 1)$,

$$M[W(u_1)/u_1] \ldots [W(u_{n-1})/u_{n-1}][W(u_n, \beta)/u_n], s \models \phi[u/z]$$

where $d(u) = \mu z.\phi$ and $u_1, \ldots, u_n = u$ is as above. The fact that all minimal fixpoints are satisfied as well follows then from Corollary 2.2.17. $\square$

Example 2.2.34 A simple tableau for the formula $\mu z.a \vee \bigcirc z$, *sometime* a, is presented in Figure 2.1. Each element of the tableau is listed on a separate line, and the tableau is annotated with names of derivation rules. As the tableau in Figure 2.1. has no infinite paths at all, it is trivially proper. It is also easy to see that the tableau agrees with the model M defined by: $M(0) = \emptyset$ and $M(s) = \{a\}$ for all $s > 0$. $\square$

The simple tableau system which was just described captures elegantly the idea of determining the truth of a formula in a state of a model in the way discussed in the beginning of the current Subsection. However, the approach copes less well with satisfiability, i.e. determining whether a formula has a model or not. The basic reason for this is the branching caused by the $\wedge$-rule. If we tried to determine the satisfiability of a formula by building a simple tableau for it, there is nothing to prevent us from making mutually contradictory choices in different branches of the tableau; for example, one branch might require that a proposition z is true in a certain state and another branch that it is false in the same state.

Because of this, we will introduce another variant of the tableau construction, the *bundled* tableau, where the decomposition of a conjunction $\phi \wedge \phi'$ does not cause the tableau to branch. The advantage of such a tableau is that all the requirements set for some state of a model are spelt out at a single point of the tableau, which means that it is easy to check that they are mutually consistent. The price to pay is added complexity; the tableau nodes are labelled with sets of formulae and not with single formulae as in simple tableaux.

Let us first define the technical concept of guardedness. Intuitively this precludes the possibility that the problem of determining whether $M, s \models \sigma z.\phi$ holds could reduce back to itself after the fixpoint has been unfolded. We also show that every formula can be mechanically transformed to an equivalent guarded formula.

Definition 2.2.35 An occurrence of a variable z or a formula ψ in a formula ϕ is *guarded* iff it is in a subformula of the type $\bigcirc\phi'$. A formula ϕ is guarded iff for every fixpoint subformula $\sigma z.\phi'$ of ϕ, every occurrence of z in ϕ' is guarded. $\square$

Proposition 2.2.36 For every μTL-formula ϕ we can produce a guarded formula ϕ' such that

- $\models \phi \Leftrightarrow \phi'$,
- for every $n \in \mathbb{N}$, if $\phi \in \Sigma_n$ then $\phi' \in \Sigma_n$, and
- for every $n \in \mathbb{N}$, if $\phi \in \Pi_n$ then $\phi' \in \Pi_n$.

Proof: (See also [6, Subsection 2.4]) With propositional manipulation and the rule $\mu z.\psi \vee (z \wedge \psi') \Leftrightarrow \mu z.\psi$, we can delete from any fixpoint formula all unguarded occurrences of the fixpoint variable that are not in the scope of another fixpoint.

We show by induction on the structure of the formula ϕ that for every ϕ there is an equivalent ϕ' such that (1) ϕ' is guarded, (2) there are only guarded occurrences of fixpoint subformulae in ϕ', and (3) if any variable z occurs only positively in ϕ it does so in ϕ', as well. The only nontrivial point is the induction step for $\phi = \mu z.\psi$. Assume that the claim holds for ψ, and take the corresponding ψ'. As all fixpoint subformulae of ψ' occur guarded, there are no unguarded occurrences of z in ψ' inside fixpoint subformulae, and we can delete all unguarded occurrences of z in $\mu z.\psi'$ by the transformation above. Denote the result by $\mu z.\chi$. Choosing then $\phi' = \chi[\mu z.\chi/z]$ fulfils the induction step.

To justify the claim concerning alternation classes, we use the characterisation of Σ_n and Π_n in terms of sequences of nested alternating dependent fixpoints, provided by Proposition 2.2.25. In the transformation from ϕ to ϕ', three kinds of transformations are performed on ϕ: propositional manipulation, erasing occurrences of fixpoint variables, and replacing fixpoint subformulae by their unfoldings. It is clear that the first two do not bring about any new fixpoints or dependencies between fixpoints and therefore they do not increase the alternation class of a

name	application	name	application
$\vee\boldsymbol{L}$	$\dfrac{s,\quad \Gamma\cup\{\phi\vee\phi'\},\quad d}{s,\quad \Gamma\cup\{\phi\},\qquad d}$	$\vee\boldsymbol{R}$	$\dfrac{s,\quad \Gamma\cup\{\phi\vee\phi'\},\quad d}{s,\quad \Gamma\cup\{\phi'\},\qquad d}$
$\wedge$	$\dfrac{s,\quad \Gamma\cup\{\phi\wedge\phi'\},\quad d}{s,\quad \Gamma\cup\{\phi,\phi'\},\quad d}$		
$\boldsymbol{\sigma}$	$\dfrac{s,\quad \Gamma\cup\{\sigma z.\phi\},\quad d}{s,\quad \Gamma\cup\{u\},\qquad d\cdot(u,\sigma z.\phi)}$ **1**	$\boldsymbol{U}$	$\dfrac{s,\quad \Gamma\cup\{u\},\qquad d}{s,\quad \Gamma\cup\{\phi[u/z]\},\quad d}$ **2**
$\bigcirc$	$\dfrac{s,\qquad \{z_1,\ldots,z_m,\bigcirc\phi_1,\ldots,\bigcirc\phi_k\},\quad d}{s+1,\quad \{\phi_1,\ldots,\phi_k\},\qquad d}$ **3**		
Note:	**1:** u does not appear in d **3:** every z_i is atomic. In each rule, Γ is disjoint from the other set		**2:** $d(u)=\sigma z.\phi$

Table 2.2: Bundled tableau rules for linear structures

formula. Moreover, since in $\chi[\mu z.\chi/z]$ no variable occurring free in $\mu z.\chi$ is captured by a fixpoint of the outer χ, the formulae $\phi[\mu z.\chi/w]$ and $\phi[(\chi[\mu z.\chi/z])/w]$ have precisely the same sequences of nested dependent fixpoints. Consequently, replacing fixpoint subformulae by their unfoldings does not affect the alternation class of the formula either. This means that all three transformations preserve the alternation class of the formula they are applied to. $\square$

Definition 2.2.37 Let ϕ be a guarded μTL-formula in pnf. A *bundled tableau* T for ϕ is an infinite sequence $T=(s_0,\Gamma_0,d_0)(s_1,\Gamma_1,d_1)\ldots$ where

- every $s_i\in\mathbb{N}$, Γ_i is a finite set of extended μTL-formulae in pnf, and d_i is a definition list containing all definition constants in Γ_i,
- $(s_0,\Gamma_0,d_0)=(0,\{\phi\},\epsilon)$, and
- every $(s_{i+1},\Gamma_{i+1},d_{i+1})$ is derived from (s_i,Γ_i,d_i) by applying one of the rules in Table 2.2.

We say that a constant u is a maximal (minimal) fixpoint constant in T iff there is some d_j such that u is a maximal (minimal) fixpoint constant in d_j. $\square$

Notice that as a special case, the $\bigcirc$-rule allows deriving $(s+1,\emptyset,d)$ from $(s,\emptyset,d)$ for any s and d. To express the well-foundedness requirement for a bundled tableau, we need to extract sequences of formulae corresponding to the paths of simple tableaux. These are called dependency sequences.

Definition 2.2.38 Let $T=(s_0,\Gamma_0,d_0)(s_1,\Gamma_1,d_1)\ldots$ be a bundled tableau for a μTL-formula ϕ. For every $i\in\mathbb{N}$, the rule applied at point i induces a *dependency relation* $\rightarrow\subseteq\Gamma_i\times\Gamma_{i+1}$ by:

- if the rule is not $\bigcirc$, the formula in Γ_i to which the rule is applied depends on the resulting formulae (e.g. $\phi \vee \phi' \rightarrow \phi$ for $\vee L$) and $\psi \rightarrow \psi$ for every other $\psi \in \Gamma_i$
- if the rule is $\bigcirc$, $\bigcirc\phi \rightarrow \phi$ for every formula of the form $\bigcirc\phi \in \Gamma_i$.

We say that $i \in \mathbb{N}$ is a $\bigcirc$-*point* of T iff the $\bigcirc$-rule is applied at point i of T.

For any $n \in \mathbb{N}$, a finite or infinite sequence $\phi_0, \phi_1, \ldots$ is a *dependency sequence* of T from formula ϕ_0 at point n iff every $\phi_i \in \Gamma_{n+i}$ and $\phi_i \rightarrow \phi_{i+1}$ relative to the rule applied at point $n+i$. The tableau T is *proper* iff there is no minimal fixpoint constant u of T and infinite dependency sequence $\phi_0, \phi_1, \ldots$ from some point n of T such that $\phi_i = u$ for infinitely many $i \in \mathbb{N}$.

The tableau T *agrees* with a linear model M iff for every $\bigcirc$-point i of T,

- $\bot \notin \Gamma_i[d_i]$,
- for every $z \in \mathcal{Z}$, if $z \in \Gamma_i[d_i]$ then $z \in M(s_i)$, and
- for every $z \in \mathcal{Z}$, if $\neg z \in \Gamma_i[d_i]$ then $z \notin M(s_i)$.

The tableau T is *propositionally consistent* iff there is no $i \in \mathbb{N}$ such that either $\bot \in \Gamma_i[d_i]$, or both $z \in \Gamma_i[d_i]$ and $\neg z \in \Gamma_i[d_i]$ for some $z \in \mathcal{Z}$. □

Proposition 2.2.39 Let ϕ be a guarded μTL-formula in pnf and M a linear model. Then $M \models \phi$ iff there is a bundled tableau T for ϕ such that

- T is proper and
- T agrees with M.

Proof: Assume first that $M \models \phi$. As in the proof of Prop. 2.2.33 for simple tableaux, we can easily produce a bundled tableau for ϕ agreeing with M, but special precautions are needed to guarantee that M is proper. For this we use the same technique as there and annotate occurrences of minimal constants u with ordinals α. The tableau rules are correspondingly modified so that σ and U only apply to maximal fixpoints, and minimal fixpoints are taken care of by new μ and U' rules, which are similar to those for simple tableaux in the proof of Prop. 2.2.33. Furthermore, we add the following deletion rule, which has priority over all the other rules:

$$\textbf{del} \quad \frac{s, \quad \Gamma \cup \{\phi[u^\alpha/z], \phi[u^\beta/z]\}, \quad d}{s, \quad \Gamma \cup \{\phi[u^\alpha/z]\}, \quad d} \quad \text{where } \alpha \prec \beta$$

We can now apply these rules in any order, respecting the priority of the deletion rule, to produce a bundled tableau T agreeing with M. Since a dependency sequence with infinitely many occurrences of a minimal constant in this T would correspond to an infinite decreasing sequence of ordinals, T is proper. It is easy to read a normal proper bundled tableau from this T by erasing the ordinal annotations and omitting any applications of the deletion rule. Notice here that after erasing the ordinal annotations, the $\phi[u^\alpha/z]$ and $\phi[u^\beta/z]$ in the premise of the deletion rule become the same formula.

Assume then that there exists a proper bundled tableau T for ϕ agreeing with M. It is easy to read from T a proper simple tableau for ϕ agreeing with M, which guarantees $M \models \phi$ by Prop. 2.2.33. The assumption of guardedness is required here to guarantee that every non-atomic formula in a bundled tableau will be decomposed by an application of a tableau rule sooner or later. $\Box$

Proposition 2.2.40 Let ϕ be a guarded μTL-formula in pnf. Then ϕ is satisfiable iff there is a bundled tableau T for ϕ such that

- T is propositionally consistent and
- T is proper.

Proof: From left to right immediate by Proposition 2.2.39. In the other direction, it is easy to read a model M agreeing with T from a propositionally consistent tableau. Define every $M(s)$ by $M(s) = \Gamma_i[d_i] \cap \mathcal{Z}$, where i is the $\bigcirc$-point of T for which $s_i = s$. $\Box$

Example 2.2.41 A bundled tableau for the formula $\mu z.\nu x.\bigcirc x \wedge (a \vee \bigcirc z)$, *almost always a*, is presented in Figure 2.2. The arrows in the figure specify the dependency relations. It is easy to see that the tableau agrees with the model M defined by: $M(0) = \emptyset$ and $M(s) = \{a\}$ for all $s > 0$. The tableau is also proper; although it has infinite dependency sequences, the only constant u corresponding to a minimal fixpoint occurs only finitely many times in any dependency sequence. $\Box$

2.2.5 Strong second-order quantifiers

In this subsection we will look at another second-order construct related to fixpoints, second-order quantification over propositions, and two calculi $\exists TL$ and *S1S* based on this.

Extending standard linear-time temporal logic TL with quantification over propositions results in a formalism we call $\exists TL$ for uniformity, but which is more widely known as $QPTL$ or *quantified propositional temporal logic* [53]. If we consider $QPTL$ from the point of view of specification and verification of programs, the advantage of quantification is that it naturally corresponds to the operation of hiding internal variables of a program, which means that the language can be used for compositional reasoning about programs.

Notice that here we start from a propositional language and extend it with second-order quantifiers allowing quantification over propositions. This should

$$\begin{tikzcd}[row sep=tiny]
 & 0,\ \{ & & \mu z.\nu x.\bigcirc x \wedge (a \vee \bigcirc z) \arrow[dd] & & \},\ \epsilon \\
\sigma \\
 & 0,\ \{ & & u \arrow[dd] & & \},\ (u, \mu z.\nu x.\bigcirc x \wedge (a \vee \bigcirc z)) \\
U \\
 & 0,\ \{ & & \nu x.\bigcirc x \wedge (a \vee \bigcirc u) \arrow[dd] & & \},\ (u, \ldots) \\
\sigma \\
 & 0,\ \{ & & v \arrow[dd] & & \},\ (u, \ldots)(v, \nu x.\bigcirc x \wedge (a \vee \bigcirc u)) \\
U \\
 & 0,\ \{ & & \bigcirc v \wedge (a \vee \bigcirc u) \arrow[ddl] \arrow[ddr] & & \},\ (u, \ldots)(v, \ldots) \\
\wedge \\
 & 0,\ \{ & \bigcirc v, \arrow[dd] & & a \vee \bigcirc u \arrow[dd] & \},\ (u, \ldots)(v, \ldots) \\
\vee R \\
 & 0,\ \{ & \bigcirc v, \arrow[dd] & & \bigcirc u \arrow[dd] & \},\ (u, \ldots)(v, \ldots) \\
\bigcirc \\
 & 1,\ \{ & v, \arrow[dd] & & u \arrow[dd] & \},\ (u, \ldots)(v, \ldots) \\
U \\
 & 1,\ \{ & v, \arrow[dd] & & \nu x.\bigcirc x \wedge (a \vee \bigcirc u) \arrow[dd] & \},\ (u, \ldots)(v, \ldots) \\
\sigma \\
 & 1,\ \{ & v, \arrow[dd] & & v' \arrow[dd] & \},\ (u, \ldots)(v, \ldots)(v', \nu x.\bigcirc x \wedge (a \vee \bigcirc u)) \\
U \\
 & 1,\ \{ & \bigcirc v \wedge (a \vee \bigcirc u), \arrow[dd] & & v' \arrow[dd] & \},\ (u, \ldots)(v, \ldots)(v', \ldots) \\
U \\
 & 1,\ \{ & \bigcirc v \wedge (a \vee \bigcirc u), \arrow[dd] \arrow[ddr] & & \bigcirc v' \wedge (a \vee \bigcirc u) \arrow[dd] & \},\ (u, \ldots)(v, \ldots)(v', \ldots) \\
\wedge \\
 & 1,\ \{ & \bigcirc v, \arrow[dd] & a \vee \bigcirc u, \arrow[dd] & \bigcirc v' \wedge (a \vee \bigcirc u) \arrow[dd] \arrow[ddl] & \},\ (u, \ldots)(v, \ldots)(v', \ldots) \\
\wedge \\
 & 1,\ \{ & \bigcirc v, \arrow[dd] & a \vee \bigcirc u, \arrow[dd] & \bigcirc v' \arrow[dd] & \},\ (u, \ldots)(v, \ldots)(v', \ldots) \\
\vee L \\
 & 1,\ \{ & \bigcirc v, \arrow[dd] & a, & \bigcirc v' \arrow[dd] & \},\ (u, \ldots)(v, \ldots)(v', \ldots) \\
\bigcirc \\
 & 2,\ \{ & v, \arrow[dd] & & v' \arrow[dd] & \},\ (u, \ldots)(v, \ldots)(v', \ldots) \\
U \\
 & 2,\ \{ & \bigcirc v \wedge (a \vee \bigcirc u), \arrow[dd] & & v' \arrow[dd] & \},\ (u, \ldots)(v, \ldots)(v', \ldots) \\
U \\
 & 2,\ \{ & \bigcirc v \wedge (a \vee \bigcirc u), \arrow[dd] \arrow[ddr] & & \bigcirc v' \wedge (a \vee \bigcirc u) \arrow[dd] & \},\ (u, \ldots)(v, \ldots)(v', \ldots) \\
\wedge \\
 & 2,\ \{ & \bigcirc v, \arrow[dd] & a \vee \bigcirc u, \arrow[dd] & \bigcirc v' \wedge (a \vee \bigcirc u) \arrow[dd] \arrow[ddl] & \},\ (u, \ldots)(v, \ldots)(v', \ldots) \\
\wedge \\
 & 2,\ \{ & \bigcirc v, \arrow[dd] & a \vee \bigcirc u, \arrow[dd] & \bigcirc v' \arrow[dd] & \},\ (u, \ldots)(v, \ldots)(v', \ldots) \\
\vee L \\
 & 2,\ \{ & \bigcirc v, \arrow[dd] & a, & \bigcirc v' \arrow[dd] & \},\ (u, \ldots)(v, \ldots)(v', \ldots) \\
\bigcirc \\
 & 3,\ \{ & v, \arrow[dd] & & v' \arrow[dd] & \},\ (u, \ldots)(v, \ldots)(v', \ldots) \\
U \\
 & & {} & \vdots & {} &
\end{tikzcd}$$

Figure 2.2: A bundled tableau for $\mu z.\nu x.\bigcirc x \wedge (a \vee \bigcirc z)$

not be confused with the step from a propositional to a first-order language with individuals and properties, and quantification over individuals only.

Definition 2.2.42 The formulae of the *quantified linear-time temporal logic* $\exists TL$ are defined by the abstract syntax:

$$\phi ::= z \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \bigcirc\phi \mid \mathbf{G}\phi \mid \exists z.\phi$$

where z varies over $\mathcal{Z}$. Let us define the derived operators $\mathbf{F}$ and $\forall z$ by:

$$\begin{aligned} \mathbf{F}\phi &= \neg\mathbf{G}\neg\phi \\ \forall z.\phi &= \neg\exists z.\neg\phi \end{aligned}$$

□

As opposed to the formula $\mu z.\phi$ of the linear-time mu-calculus μTL, there are no restrictions on the occurrences of z in $\exists z.\phi$; it may also appear in the scope of an odd number of negations. The nexttime operator $\bigcirc$ is as before, $\mathbf{G}\phi$ is the usual *always*-operator of linear-time temporal logic, and $\exists z.\phi$ is quantification over proposition z. As with $\mu z.\phi$ we adopt the convention that the scope of the quantifier in $\exists z.\phi$ extends as far to the right as possible.

Definition 2.2.43 Let $M \in \mathcal{M}_{TL}$ be a linear model. The set of states of M satisfying an $\exists TL$-formula ϕ, denoted by $\|\phi\|_M$, is defined for z, $\neg\phi$, $\phi \wedge \phi'$ and $\bigcirc\phi$ as in Def. 2.2.4 and by

$$\begin{aligned} \|\mathbf{G}\phi\|_M &= \{s \in \mathrm{st}(M) \mid \forall s' \geq s : s' \in \|\phi\|_M\} \\ \|\exists z.\phi\|_M &= \{s \in \mathrm{st}(M) \mid \exists W \subseteq \mathrm{st}(M) : s \in \|\phi\|_{M[W/z]}\} \end{aligned}$$

The notation $M, s \models \phi$ etc. and the related concepts are as in Def. 2.2.4. □

Intuitively, $\exists z.\phi$ is true at state s of model M iff there is a way to choose the interpretation W of z so that ϕ is true at state s of $M[W/z]$. We call this general form of second-order quantification *strong* to distinguish it from *weak* quantification, introduced in next subsection, which only allows quantification over finite sets or propositions which are true in only finitely many states. We use the same notation $\|\phi\|_M$ for both formulae of μTL and those of $\exists TL$. This causes no confusion, since for all formulae that belong to both languages the definitions coincide.

The formula $a\mathbf{U}b$, *a until b*, of standard linear-time temporal logic TL corresponds to the following $\exists TL$-formula:

$$\exists z.z \wedge \mathbf{F}\neg z \wedge \mathbf{G}(z \Rightarrow (b \vee (a \wedge \bigcirc z)))$$

and the property *at every even moment a* can be characterised by the formula

$$\exists z.z \wedge \mathbf{G}(z \Rightarrow (a \wedge \bigcirc\bigcirc z))$$

Given the two calculi μTL and $\exists TL$, it is natural to compare their expressive power. Let us see first that it is rather easy to translate μTL to $\exists TL$.

Proposition 2.2.44 For every μTL-formula ϕ there is an $\exists TL$-formula ψ such that $\phi \Leftrightarrow \psi$, i.e. $\|\phi\|_M = \|\psi\|_M$ for all linear models $M \in \mathcal{M}_{TL}$.

Proof: Let us translate every formula ϕ of μTL inductively to an equivalent formula $f(\phi)$ of $\exists TL$. Define the translation by

$$\begin{aligned} f(z) &= z \\ f(\neg\phi) &= \neg f(\phi) \\ f(\phi \wedge \phi') &= f(\phi) \wedge f(\phi') \\ f(\bigcirc\phi) &= \bigcirc f(\phi) \\ f(\mu z.\phi) &= \forall z.\mathbf{G}(f(\phi) \Rightarrow z) \Rightarrow z \end{aligned}$$

The only nontrivial clause here is the last one, justified by Lemma 2.2.9.

Essentially the same translation has been formulated in several occasions in the literature, at least in [4, 6, 56, 59]. □

If we spell out the effect of the translation above on maximal fixpoint formulae, this is:

$$f(\nu z.\phi) = \exists z.z \wedge \mathbf{G}(z \Rightarrow f(\phi))$$

The converse of Proposition 2.2.44 holds as well; for every $\exists TL$-formula there exists an equivalent μTL-formula. However, the translation is far more complex than the one above and requires insights from automata theory. We shall return to this issue in Chapter 3, once the necessary technical machinery has been introduced.

The linear-time quantified temporal logic $\exists TL$ is closely related to another important and probably more widely known calculus with second-order quantification, the monadic second-order theory of one successor *S1S*. Known also as the *sequential calculus*, *S1S* is a system of monadic second-order logic for the formalisation of properties of sequences, It has been examined above all by Büchi [15], who was motivated by the decision problem of *S1S* to examine the notion of automata on infinite strings as a normal form for *S1S* formulae. Before discussing the intuitions behind *S1S* and its relation to $\exists TL$, let us first formally define the language.

Definition 2.2.45 Fix a countable set $\mathcal{X}$ of *individual variables*, disjoint from the set $\mathcal{Z}$ of atomic propositions, called *predicate variables* in the current context.

The *individual terms* of the *monadic second-order theory of one successor S1S* are defined by the abstract syntax

$$t ::= 0 \mid x \mid s(t)$$

where x varies over $\mathcal{X}$ and 0 and s are fixed symbols.

The *atomic formulae* of *S1S* are defined by the abstract syntax

$$\phi_A ::= z(t) \mid t = t' \mid t < t'$$

where z varies over $\mathcal{Z}$ and t over the set of terms.

The *formulae* of *S1S* are defined by the abstract syntax

$$\phi ::= \phi_A \mid \neg\phi \mid \phi \wedge \phi' \mid \exists x.\phi \mid \exists z.\phi$$

where ϕ_A varies over the set of atomic formulae, x over $\mathcal{X}$ and z over $\mathcal{Z}$.

The abbreviations $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\forall$ etc. are as before. We also use the abbreviation $t \leq t'$ to stand for $(t = t') \vee (t < t')$. We call an *S1S*-formula *closed* iff it has no free occurrences of individual variables. □

The way the calculi *S1S* and $\exists TL$ approach the issue of characterising properties of sequences is somewhat different. In $\exists TL$, and μTL as well, the notion of state is implicit in the level of the logic, but the truth of a formula in a model is defined relative to a state, 'the current moment'. In *S1S*, in contrast, we refer to states directly in the level of the logic by means of individual terms, and compare the temporal ordering of states by the $=$ and $<$ relations and the successor function $s()$. In $\exists TL$, the denotations of atomic propositions $z \in \mathcal{Z}$ are sets of states and the second-order quantifiers range over these sets. The intuitive meaning of second-order quantification is the same in *S1S*, but as the language has individual terms denoting states, the elements of $\mathcal{Z}$ become monadic predicates $z(x)$, meaning z *holds in state* x.

In interpreting *S1S*, the denotations of the individual and predicate variables can vary. The language also contains the constant term 0 and the successor function $s()$ on terms, the interpretation of which is fixed. The natural definition of a structure for interpreting *S1S* is a mapping assigning to each individual variable an element of $\mathbb{N}$ and to each monadic predicate variable a monadic relation, i.e. a subset of $\mathbb{N}$. Such an assignment to predicate variables is clearly just another way of encoding the same information as in the linear models as defined in the current work. Keeping this in mind the semantics of *S1S* can be defined as follows.

Definition 2.2.46 Let $M \in \mathcal{M}_{TL}$ be a linear model and $V : \mathcal{X} \to \mathrm{st}(M)$ a *valuation* assigning to each individual variable $x \in \mathcal{X}$ a state of M. Let us extend the valuation V to all individual terms by defining

$$\begin{aligned} V(0) &= 0 \\ V(s(t)) &= V(t) + 1 \end{aligned}$$

An *S1S*-formula ϕ is true in model M under valuation V, denoted by $M, V \models \phi$, according to the following rules:

$$\begin{array}{lll} M, V \models z(t) & \text{iff} & z \in M(V(t)) \\ M, V \models t = t' & \text{iff} & V(t) = V(t') \\ M, V \models t < t' & \text{iff} & V(t) < V(t') \\ M, V \models \neg\phi & \text{iff} & \text{not } M, V \models \phi \\ M, V \models \phi \wedge \phi' & \text{iff} & M, V \models \phi \text{ and } M, V \models \phi' \\ M, V \models \exists x.\phi & \text{iff} & \text{there is some } s \in \mathrm{st}(M) \text{ such that } M, V[s/x] \models \phi \\ M, V \models \exists z.\phi & \text{iff} & \text{there is some } W \subseteq \mathrm{st}(M) \text{ such that } M[W/z], V \models \phi \end{array}$$

where $V[s/x](x) = s$ and $V[s/x](x') = V(x')$ for all $x' \neq x$. We write $M \models \phi$ iff $M, V \models \phi$ for all valuations V. □

In *S1S* the property *a holds initially and at every even state thereafter* can be expressed by the formula

$$\exists z.z(0) \wedge \forall x.z(x) \Rightarrow a(x) \wedge z(s(s(x)))$$

The $<$-relation as actually superfluous as a primitive construct in *S1S*, since it is second-order definable in terms of the successor function:

$$t < t' \text{ iff } \forall z.(z(s(t)) \wedge \forall x.z(x) \Rightarrow z(s(x))) \Rightarrow z(t')$$

Lemma 2.2.47 Let $M \in \mathcal{M}_{TL}$ be a linear model and ϕ a closed *S1S*-formula. Then for any valuations V and V', $M, V \models \phi$ iff $M, V' \models \phi$, i.e. the validity of ϕ over M is independent of valuation.

Proof: Easy. □

Let us then describe translations between $\exists TL$ and *S1S*. The translations are in both directions so straightforward that we can almost view the two formalisms as two formulations of the same language. Because of this, in the rest of the work we will not be studying *S1S* directly, preferring to examine $\exists TL$, which is closer to our framework, and use the correspondence with $\exists TL$ to derive results concerning *S1S*, as well.

Proposition 2.2.48 Let ϕ be an $\exists TL$-formula. There exists a closed *S1S*-formula ψ such that $M \models \phi$ iff $M \models \psi$, for every linear model M.

Proof: Let us describe an inductive translation f from $\exists TL$-formulae to *S1S*-formulae. Fix an individual variable $w \in \mathcal{X}$. Define

$$\begin{array}{rcl} f(z) & = & z(w) \\ f(\neg\phi) & = & \neg f(\phi) \\ f(\phi \wedge \phi') & = & f(\phi) \wedge f(\phi') \\ f(\bigcirc\phi) & = & f(\phi)[s(w)/w] \\ f(\mathbf{G}\phi) & = & \forall x. w \leq x \Rightarrow f(\phi)[x/w] \\ f(\exists z.\phi) & = & \exists z. f(\phi) \end{array}$$

Notice that every formula has exactly one free individual variable, w. Intuitively w expresses the point or 'current moment' relative to which $\exists TL$-formulae are evaluated.

The important property of the translation is that for every $\exists TL$-formula ϕ, model M and state s of M, $M, s \models \phi$ iff $M, V_s \models f(\phi)$, where $V_s(w) = s$ and $V_s(x)$ is arbitrary for $x \neq w$. Take now any $\exists TL$-formula ϕ. The formula $\psi = f(\phi)[0/w]$ fulfils the claim of the proposition, since $M \models \phi$ iff $M, 0 \models \phi$ iff $M, V_0 \models f(\phi)$ iff $M, V \models f(\phi)[0/w]$ for every valuation V iff $M \models f(\phi)[0/w]$. $\square$

Proposition 2.2.49 Let ϕ be a closed *S1S*-formula. There exists an $\exists TL$-formula ψ such that $M \models \phi$ iff $M \models \psi$, for every linear model M.

Proof: As a preliminary step to a translation from *S1S* to $\exists TL$, notice that by rewriting

$$\begin{array}{rcl} p(t) & \text{as} & \exists x.\, (x = t) \wedge p(x), \\ t = t' & \text{as} & \exists x.\, (x = t) \wedge (x = t'), \\ t < t' & \text{as} & \exists x. \exists x'.\, (x = t) \wedge (x' = t') \wedge (x < x'), \quad \text{and} \\ x = s(t) & \text{as} & \exists x'. (x' = t) \wedge (x = s(x')) \end{array}$$

where x, x' are fresh variables, we can assume that all atomic formulae are of the types $p(x)$, $x = 0$, $x = s(y)$ or $x < y$, where x and y are individual variables.

The only nontrivial step in defining an inductive translation f from *S1S* to $\exists TL$ is dealing with quantification $\exists x$ over individuals or states. This is done by coding individual quantification as second-order quantification over singleton sets. The translation f is:

$$f(p(x)) \quad = \quad \mathbf{G}(x \Rightarrow p)$$

$$\begin{array}{rcl}
f(x=0) & = & x \\
f(x=s(y)) & = & \mathbf{G}(y \Rightarrow \bigcirc x) \\
f(x<y) & = & \mathbf{G}(x \Rightarrow \bigcirc \mathbf{F} y) \\
f(\neg\phi) & = & \neg f(\phi) \\
f(\phi \wedge \psi) & = & f(\phi) \wedge f(\psi) \\
f(\exists x.\phi) & = & \exists x.\text{singleton}(x) \wedge f(\phi) \qquad \text{where} \\
 & & \text{singleton}(x) \;=\; (\forall x'.\mathbf{G}(x' \Rightarrow x) \Rightarrow (\mathbf{G}(x \Rightarrow x') \vee \mathbf{G}\neg x')) \\
f(\exists z.\phi) & = & \exists z.f(\phi)
\end{array}$$

The essential property of the translation f is that for all models M and valuations V, we have $M, V \models \psi$ iff $M_V \models f(\psi)$, where M_V is defined by

$$M_V(s) = M(s) \cup \{x \in \mathcal{X} \mid V(x) = s \text{ and } x \text{ occurs free in } \psi\}$$

Here we have temporarily abused the notation by extending the notion of a model so that some $M_V(s)$ also include some elements of $\mathcal{X}$ and not only elements of $\mathcal{Z}$. It is clear that this causes no harm. □

The translations between $\exists TL$ and *S1S* outlined above are modifications of ones due to Wolper [102, pp. 46-48]. Notice that both translations are polynomial time and the size of the resulting formula is polynomial in the size of the original formula.

The approach with individual terms denoting states and monadic predicates denoting properties of states has been used in addition to *S1S* in a weaker, first-order framework as well. In actual fact, one of the classical results in linear temporal logics is a first-order variant of the correspondence between $\exists TL$ and *S1S*. This result, due to Kamp [52], states that the standard linear temporal logic TL and the first-order theory of one successor and linear order are equiexpressive. In contrast to the relatively easy translation from *S1S* to $\exists TL$ above, Kamp's translation from the first-order theory of one successor to TL is highly nontrivial. The basic reason for the complexity is that in the first-order world we cannot use the technique of coding first-order quantification as second-order quantification over singleton sets, as was done above.

2.2.6 Weak second-order quantifiers

The essential characteristic of both $\exists TL$ and *S1S* is second-order quantification over sets or propositions. There is also another, weaker choice for a second-order construct: quantification over *finite* sets only. One philosophical reason for studying such weak second-order quantification is the aim to extend first-order logic to be able to express finiteness, but to make this extension as small as possible so that it would not interfere with results which are already known about the first-order language. It is also the case that weak or finitary quantification is technically easier to handle than general quantification. Corresponding to $\exists TL$ and *S1S*, respectively, there are two weak second-order languages $\overset{\text{w}}{\exists} TL$ and *WS1S*.

Definition 2.2.50 The formulae of the *weak quantified linear-time temporal logic* $\overset{\text{w}}{\exists} TL$ are defined as those of $\exists TL$ in Definition 2.2.42, except the quantifier $\exists z$ is replaced by the weak quantifier $\overset{\text{w}}{\exists} z$. The semantics of $\overset{\text{w}}{\exists} TL$-formulae are defined as in Definition 2.2.43, except for

$$\|\overset{\text{w}}{\exists} z.\phi\|_M \quad = \quad \{s \in \mathrm{st}(M) \mid \exists W \subseteq \mathrm{st}(M) : W \text{ is finite and } s \in \|\phi\|_{M[W/z]}\}$$

The derived operator $\overset{\text{w}}{\forall}$ is defined by $\overset{\text{w}}{\forall} z.\phi = \neg\overset{\text{w}}{\exists} z.\neg\phi$. □

Definition 2.2.51 The formulae of the *weak monadic second-order theory of one successor WS1S* are defined as those of *S1S* in Definition 2.2.45, except the quantifier $\exists z$ is replaced by the weak quantifier $\overset{\text{w}}{\exists} z$. The semantics of *WS1S*-formulae are defined as in Definition 2.2.46, except for

$$M, V \models \overset{\text{w}}{\exists} z.\phi \quad \text{iff} \quad \text{there is some finite } W \subseteq \mathrm{st}(M) \text{ such that } M[W/z], V \models \phi$$

□

It is obvious that like $\exists TL$ and *S1S*, their weak counterparts are equiexpressive. It comes as no great surprise either, that the weak calculi can be embedded in the strong ones.

Proposition 2.2.52 For every $\overset{\text{w}}{\exists} TL$-formula ϕ there is a closed *WS1S*-formula ψ, and vice versa, such that $M \models \phi$ iff $M \models \psi$, for all linear models M.

Proof: From $\overset{\text{w}}{\exists} TL$ to *WS1S* as in Proposition 2.2.48 and from *WS1S* to $\overset{\text{w}}{\exists} TL$ as in Proposition 2.2.49, by replacing $\exists z$ with $\overset{\text{w}}{\exists} z$. Notice that in the latter proof weak quantification is sufficient for the purpose of coding quantification over individuals as second-order quantification over singletons. □

Proposition 2.2.53 For every $\overset{\text{w}}{\exists} TL$-formula ϕ there is an $\exists TL$-formula ψ such that $\phi \Leftrightarrow \psi$, i.e. $\|\phi\|_M = \|\psi\|_M$ for all linear models M. Similarly, for every *WS1S*-formula ϕ there is an *S1S*-formula ψ such that $M, V \models \phi$ iff $M, V \models \psi$, for all linear models M and valuations V.

Proof: A translation f from $\overset{\mathrm{w}}{\exists}TL$ to $\exists TL$ is trivial for all other operators except for $f(\overset{\mathrm{w}}{\exists}z.\phi) = \exists z.\mathrm{finite}(z) \wedge f(\phi)$, where $\mathrm{finite}(z) = \mathbf{FG}\neg z$, characterising the fact that z should be true in finitely many states only. The proof for *WS1S* is analogous, except that $\mathrm{finite}(z) = \exists x.\forall x'.(x' \geq x) \Rightarrow \neg z(x')$. □

What is more interesting is the relation of weak second-order quantification and fixpoints over non-alternating formulae. It turns out that the expressive power of the weak calculi matches precisely that of the alternation class Δ_2, i.e. the class of formulae without any proper alternation at all. We shall prove one direction of this here and the other in Chapter 3. Let us first point out a particularly simple case of the tableau construction for μTL formulae without any maximal fixpoints.

Lemma 2.2.54 Let $\phi \in \Sigma_1$ be a μTL-formula and T a simple tableau for ϕ. Then T is proper iff T is finite.

Proof: Assume that T is proper. Since $\phi \in \Sigma_1$, the tableau T can have only minimal constants. As by 2.2.32 any infinite path of T would have one of these minimal constants occurring infinitely often along it, T has only finite paths. As T is a finitely branching tree, by König's lemma this means that T is finite. Conversely, if T is finite, then T has only finite paths and is trivially proper. □

Proposition 2.2.55 Let $\mu z.\phi \in \Delta_2$ be a guarded non-alternating μTL-formula. Then for all models M and states s of M, the following statements are equivalent:

- $M, s \models \mu z.\phi$,
- there is some $n \in \mathbb{N}$ such that $M, s \models \mu^n z.\phi$
- there is a finite set $W \subseteq \mathrm{st}(M)$ such that $s \in W$ and $W \subseteq \|\phi\|_{M[W/z]}$.

Proof: By Corollary 2.2.17 $\mu^n z.\phi \Rightarrow \mu z.\phi$, so it suffices to show that the first claim implies the third and the third the second. Without loss of of generality we can assume that $s = 0$.

Assume that $M, 0 \models \mu z.\phi$. Since $\mu z.\phi \in \Delta_2$, by Corollary 2.2.26 it can be expressed in the form $\mu z.\phi = (\mu z.\psi)[\psi_1/z_1, \ldots, \psi_n/z_n]$ where $\mu z.\psi \in \Sigma_1$, every $\psi_i \in \Delta_2$ and z does not occur free in any ψ_i. If we define now $M' = M[W_1/z_1]\ldots[W_n/z_n]$ where each $W_i = \|\psi_i\|_M$, we have $M', 0 \models \mu z.\psi$. By Prop. 2.2.33, there is a proper simple tableau T for $\mu z.\psi$ agreeing with M'. Since $\mu z.\psi \in \Sigma_1$, T is finite by Lemma 2.2.54. The only rule that can be applied initially in T is the σ-rule, replacing $\mu z.\psi$ by some constant u. Define now $W = \{s' \in \mathrm{st}(M) \mid \exists t \in \mathrm{dom}(T) : s_t = s' \text{ and } \phi_t = u\}$, where we use (s_t, ϕ_t, d_t) to denote the label of node t of T. Clearly W is finite and $0 \in W$. From the tableau

T we can also see as in the proof of 2.2.33 that $W \subseteq \|\psi\|_{M'[W/z]}$, which implies $W \subseteq \|\phi\|_{M[W/z]}$, as required.

Assume then that we have a finite set W such that $0 \in W$ and $W \subseteq \|\phi\|_{M[W/z]}$. Let $W_0 \subset W_1 \subset \ldots \subset W_n$ be a sequence of sets such that $W_0 = \emptyset$, $W_n = W$ and each

$$W_{i+1} = \{s' \in W \mid \forall s'' > s' : \text{if } s'' \in W \text{ then } s'' \in W_i\}$$

Since z only occurs guarded in ϕ, the validity of ϕ at any point $s' \in W_{i+1}$ of $M[W/z]$ only depends on the value of z at the points of W after s', i.e. $M[W/z], s' \models \phi$ iff $M[W_i/z], s' \models \phi$, for all $s' \in W_{i+1}$ and $0 \leq i < n$, which implies $W_{i+1} \subseteq \|\phi\|_{M[W_i/z]}$ for all $0 \leq i < n$. Since $W_n = W$ and $0 \in W$, by Corollary 2.2.17 we have then $0 \in \|\mu^n z.\phi\|_M$. □

Notice the close relation of this characterisation of minimal fixpoints in non-alternating formulae and that of maximal fixpoints in Lemma 2.2.9. In both cases we need to find a pre-fixpoint W containing the state i; the only difference is that here with a minimal fixpoint the set W needs to be finite.

Corollary 2.2.56 If $\mu z.\phi \in \Delta_2$, then $\mu z.\phi \Leftrightarrow \mu^\omega z.\phi$, i.e. $\|\mu z.\phi\|_M = \|\mu^\omega z.\phi\|_M$ for all models M.

Proof: Immediate from Prop. 2.2.55. □

This corollary is really a version of Kleene's recursion theorem [54, p. 348], which states that the least fixpoint of a continuous functional f coincides with f^ω, the least upper bound of all $f^i(\bot)$ where $i \in \mathbb{N}$ and f^i is the i-th iterate of f. In actual fact, if $\mu z.\phi \in \Delta_2$, the functional

$$f : 2^{\mathrm{st}(M)} \longrightarrow 2^{\mathrm{st}(M)}; W \mapsto \|\phi\|_{M[W/z]}$$

corresponding to the body of the fixpoint is not only monotonic but also continuous.

We can now easily translate Δ_2 to the weak calculus $\overset{\mathrm{w}}{\exists}TL$.

Theorem 2.2.57 For every non-alternating μTL-formula $\phi \in \Delta_2$, there is a $\overset{\mathrm{w}}{\exists}TL$-formula ψ such that $\phi \Leftrightarrow \psi$, i.e. $\|\phi\|_M = \|\psi\|_M$ for all linear models M.

Proof: As by Prop. 2.2.36 there is an equivalent guarded formula in Δ_2 for any formula in Δ_2, we need to consider only guarded formulae. Define a translation f from the class of guarded formulae in Δ_2 to $\overset{\mathrm{w}}{\exists}TL$ as in the proof of Prop. 2.2.44, except for

$$f(\mu z.\phi) \quad = \quad \overset{\mathrm{w}}{\exists} z.z \wedge \forall(z \Rightarrow f(\phi))$$

The validity of the translation is justified by Prop. 2.2.55. □

Two other translations from non-alternating fixpoints to weak second-order quantifiers are provided by Arnold [1] and Arnold and Niwiński [4]. Both of these as well as the one above are all variations on the same theme of minimal non-alternating fixpoints being essentially finitary.

2.3 Branching structures

In last section we examined various formalisms for describing properties of infinite strings, and introduced some related concepts and techniques. Now we are going to generalise these to a framework of infinite branching structures, i.e. infinite trees, mostly with a fixed degree of branching. As many features of the languages generalise without any difficulty, the treatment is correspondingly more concise.

2.3.1 Modal mu-calculus

In specifying properties of a program by the linear-time formalisms discussed in previous section, what is really being described are the required properties of the execution sequences of a program. Another alternative is to model the execution of a program by an execution tree that records both the possible execution sequences, and the points where non-deterministic choices between various courses of execution are made. Historically, this choice between linear and branching models has been one of the great divides in the semantic models of concurrent programs, and the respective merits of each side have been analysed at length in a number of papers. For an overview, see [75].

The concept of a linear model generalises naturally to the branching case.

Definition 2.3.1 A *branching model* M is a total infinite tree labelled with sets of propositions,

$$M : \mathbb{N}^* \rightharpoonup 2^{\mathcal{Z}}$$

A *state* s of a branching model M is any $s \in \mathrm{dom}(M)$, and the set of states, denoted by $\mathrm{st}(M)$, is $\mathrm{st}(M) = \mathrm{dom}(M)$. The set of all branching models is denoted by $\mathcal{M}_{\leq\omega}$. A branching model M is an *n-branching* model iff it is a total n-branching tree, i.e. iff $\mathrm{st}(M) = [n]^*$. The set of all n-branching models is denoted by $\mathcal{M}_n$. □

As in the linear case, we require for simplicity that models do not have terminal states. Fairly often branching-time formalisms are interpreted directly over graphs modelling the execution of a system. However, as any such graph can be unravelled to a tree, we do not lose anything by considering only tree models here; in a sense we can view an execution graph as a succinct representation of an execution tree.

The modal mu-calculus generalises the linear-time mu-calculus by replacing the $\bigcirc$-operator *at the next moment* by a modal operator or a family of modal operators *at some successor*. Probably the most common variant of modal mu-calculus [55] is based on the Hennessy-Milner logic [40], a poly-modal logic which contains the modal operators $<a>\phi$, *it is possible to execute an a-action leading to a state where ϕ holds* and their duals $[\,a\,]\,\phi$, *every a-action leads to a state where ϕ holds*, and is interpreted over labelled transition systems. In the present framework, where the transitions from a parent to a child in a tree model are implicit and unlabelled, a corresponding natural choice would be to base the language on the modality $\Diamond\phi$, *for some child ϕ* and its dual $\Box\phi$, *for all children ϕ*. If necessary, the transition labels could be coded in the labels of the target states, and the modalities $<a>\phi$ and $[\,a\,]\,\phi$ expressed by $\Diamond(a \wedge \phi)$ and $\Box(a \Rightarrow \phi)$, respectively.

The modal mu-calculus based on $\Diamond$ and $\Box$ and interpreted over arbitrary infinite trees is a natural framework when models is intended to reflect executions of programs. Nevertheless, we will mainly concentrate on a more restricted setting with models of a fixed degree of branching and modalities which distinguish different children of a node, not just picking some or all of them. The main reason for this is that our interest lies particularly in the relations of mu-calculi and automata, and the operation of tree automata is traditionally considered is such a setting. Let us therefore define a modal mu-calculus with indexed modalities $ⓘ\phi$, *for the i-th child ϕ*, interpreted over n-branching trees.

Definition 2.3.2 Fix some $n \in \mathbb{N}$. The formulae of the *indexed modal mu-calculus μKn* are defined by the abstract syntax:

$$\phi ::= z \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid ⓘ\phi \mid \mu z.\phi$$

where z varies over $\mathcal{Z}$ and i over $[n]$. In $\mu z.\phi$, each occurrence of z in ϕ is required to be positive. □

Definition 2.3.3 Let $n \in \mathbb{N}$ and let $M \in \mathcal{M}_n$ be an n-branching model. The set of states of M satisfying a μKn-formula ϕ, denoted by $\|\phi\|_M$, is defined for z, $\neg\phi$, $\phi \wedge \phi'$ and $\mu z.\phi$ as in Def. 2.2.4, and by:

$$\|ⓘ\phi\|_M \;=\; \{s \in \mathrm{st}(M) \mid s \cdot i \in \|\phi\|_M\}$$

The notations $M, i \models \phi$ etc. and the related notions are as in Def. 2.2.4. □

It needs to be stressed that for every n, the indexed mu-calculus μKn is interpreted over infinite n-branching models only. Notice also that like $\bigcirc$, each $\textcircled{i}$ is self-dual, i.e. $\textcircled{i}\phi \Leftrightarrow \neg\textcircled{i}\neg\phi$. We can introduce now $\Diamond$ and $\Box$ as derived operators.

Definition 2.3.4 Relative to any $n \in \mathbb{N}$, let us add to μKn the derived operators $\Diamond$ and $\Box$ by defining $\Diamond\phi = \bigvee_{i\in[n]} \textcircled{i}\phi$ and $\Box\phi = \bigwedge_{i\in[n]} \textcircled{i}\phi$. $\Box$

It is easy to see that 1-branching total infinite trees are isomorphic to infinite strings, which means that we can consider strings a particularly simple instance of trees. The indexed modal mu-calculus $\mu K1$, interpreted over 1-branching trees, also naturally coincides with the linear-time mu-calculus μTL. This means that all the results shown for μKn for an arbitrary n also hold trivially for μTL.

For some examples, fix $n = 2$. The $\mu K2$-formula $\nu z.(a \wedge \Box z)$ expresses the property *a holds everywhere* and $\nu z.(a \wedge \Diamond z)$ the property *on some path a holds everywhere.* The formula $\nu z.(\textcircled{0}(a \wedge z)) \wedge (\textcircled{1}(\neg a \wedge z))$ says *for every node, a holds in the older and fails to hold in the younger child of the node.*

In the linear case we saw in page 17 that the property *a is almost always true* could be expressed by either $\mu z.(\nu x.a \wedge \bigcirc x) \vee \bigcirc z$ or $\mu z.\nu x.(a \vee \bigcirc z) \wedge \bigcirc x$. Consider the corresponding modal formulae, where $\bigcirc$ has been replaced with $\Box$:

$$\mu z.(\nu x.a \wedge \Box x) \vee \Box z$$
$$\mu z.\nu x.(a \vee \Box z) \wedge \Box x$$

These turn out to express different properties. This first says *on every path there is a point such that a holds in every descendant of that node*, and the second that *on every path almost always a*, i.e. *on every path there is a point such that a holds in all subsequent points in that path.* To illustrate the difference, define a 2-branching model M by

$$M(i) = \begin{cases} \emptyset & \text{if } i \text{ is of the form } 0^* \cdot 1 \\ \{a\} & \text{otherwise} \end{cases}$$

Now $M \not\models \mu z.(\nu x.a \wedge \Box x) \vee \Box z$ but $M \models \mu z.\nu x.(a \vee \Box z) \wedge \Box x$.

2.3.2 Generalising linear-time concepts and results

Let us look now at the techniques and concepts introduced for the linear-time mu-calculus μTL and generalise them to deal with branching structures and the indexed modal mu-calculus μKn. In Subsection 2.2.2 we examined the meaning of fixpoint operators and introduced the notion of approximants, and in Subsection 2.2.3 we addressed the phenomenon of fixpoint alternation. The discussion and definitions as well as all the results stated there hold for μKn as well.

name	application
$\textcircled{\scriptsize i}$	$\dfrac{s, \quad \textcircled{\scriptsize i}\phi, \quad d}{s \cdot i, \quad \phi, \quad d}$

Table 2.3: Simple tableau rules for branching structures

Definition 2.3.5 For any $n \in \mathbb{N}$, the language of infinitary μKn extends μKn as in Def. 2.2.14, and the fixpoint approximants $\mu^\alpha z.\phi$ and $\nu^\alpha z.\phi$ are defined for μKn precisely as for μTL in Def. 2.2.15.

The concept of the positive normal form is defined for μKn as for μTL in Def. 2.2.20, except that the $\bigcirc$ operator is replaced by the $\textcircled{\scriptsize i}$ operators. The concept of activeness is defined for μKn exactly as for μTL in Def. 2.2.23.

The syntactic fixpoint alternation classes Σ_n^{stx}, Π_n^{stx} and Δ_n^{stx}, the Emerson-Lei fixpoint alternation classes, Σ_n^{el}, Π_n^{el} and Δ_n^{el}, the fixpoint alternation classes Σ_n, Π_n and Δ_n, and the concepts of non-alternation and guardedness are defined for μKn as for μTL in Definitions 2.2.21, 2.2.22, 2.2.24, 2.2.27 and 2.2.35, except that the $\bigcirc$ operator is replaced by the $\textcircled{\scriptsize i}$ operators. □

Proposition 2.3.6 The reformulations of the truth definitions for fixpoints in Lemmas 2.2.9 and 2.2.10, the characterisations of fixpoints by approximants in Proposition 2.2.16 and Corollary 2.2.17, the characterisations of alternation classes in terms of sequences of nested dependent fixpoints in Proposition 2.2.25 and Corollary 2.2.26, the relations between different notions of alternation in Proposition 2.2.28, and the transformation to guarded form in Proposition 2.2.36 work for μKn exactly as for μTL. □

Let us then extend the tableau constructions for branching structures. The technique of definition constants generalises to μKn without problems. The only real difference in the tableau constructions for the branching case is that the $\bigcirc$-rule needs to be replaced by new ones taking into account the branching of a model.

Definition 2.3.7 The concepts of extended μKn-formulae and definition lists and related notations for μKn are defined exactly as for μTL in Def. 2.2.29.

Let ϕ be a μKn-formula in pnf. The concept of a *simple tableau* T for ϕ is defined exactly as in Def. 2.2.30, except that $s_t \in [n]^*$ in the labelling triples (s_t, ϕ_t, d_t), the root of T is labelled with $(\epsilon, \phi, \epsilon)$, and the $\bigcirc$-rule is replaced by the $\textcircled{\scriptsize i}$-rules (one rule for every $i \in [n]$) in Table 2.3. The concepts of T being proper and T agreeing with M are defined as in Def. 2.2.31. □

name	application
$\bigcirc$	$\dfrac{s,\quad \{z_1,\ldots,z_m, \textcircled{\scriptsize i_1}\phi_1,\ldots,\textcircled{\scriptsize i_k}\phi_k\},\quad d}{s\cdot 0,\ \Gamma_0,\ d \qquad s\cdot 1,\ \Gamma_1,\ d \qquad \ldots \qquad s\cdot(n-1),\ \Gamma_{n-1},\ d}$ **1**
Note:	**1:** every z_j is atomic, and for every $i \in [n]$, $\Gamma_i = \{\phi_j \mid i_j = i\}$.

Table 2.4: Bundled tableau rule for branching structures

Definition 2.3.8 Fix some $n \in \mathbb{N}$, and let ϕ be a guarded μKn-formula in pnf. A *bundled tableau* T for ϕ is an infinite tree T, every node t of which is labelled with a triple (s_t, Γ_t, d_t) where

- every $s_t \in [n]^*$, Γ_t is a finite set of extended μKn-formulae in pnf, and d_t is a definition list containing all definition constants in Γ_t,
- the root of T is labelled with $(\epsilon, \{\phi\}, \epsilon)$, and
- every node of T either has exactly one child which is derived by applying one of the rules $\vee L$, $\vee R$, $\wedge$, σ or U in Table 2.2, or exactly n children which are derived by the rule $\bigcirc$ in Table 2.4.

We assume that every constant u is defined at most once in any tableau. A constant u is a maximal (minimal) fixpoint constant in T iff there is some node t of T such that u is a maximal (minimal) fixpoint constant in d_t. $\Box$

Definition 2.3.9 Let T be a bundled tableau for a μKn-formula ϕ. For every node t of T and every child t' of T, the rule applied at t induces a dependency relation $\rightarrow \subseteq \Gamma_t \times \Gamma_{t'}$, which is defined for all the rules except $\bigcirc$ as for the linear case in Def. 2.2.38. Using the notation of Table 2.4, the dependencies for $\bigcirc$ are defined by: $\textcircled{\scriptsize i}\phi \rightarrow \phi$ for every $\phi \in \Gamma_i$ and every $i \in [n]$.

For any node t of T and path p from node t, a finite or infinite sequence $\phi_0, \phi_1, \ldots$ is a *dependency sequence* of T from formula ϕ_0 at node t along path p iff every $\phi_i \in \Gamma_{p(i)}$ and $\phi_i \rightarrow \phi_{i+1}$ relative to the dependencies between node $p(i)$ of T and its child $p(i+1)$. The tableau T is *proper* iff there is no minimal fixpoint constant u of T and infinite dependency sequence $\phi_0, \phi_1, \ldots$ from some node t of T along some path p of T such that $\phi_i = u$ for infinitely many $i \in \mathbb{N}$.

The concepts of a $\bigcirc$-point, agreement with a model and propositional consistency are defined as in Def. 2.2.38. $\Box$

Proposition 2.3.10 Let ϕ be a guarded μKn-formula in pnf and M an n-branching model. Then $M \models \phi$ iff there is a simple or a bundled tableau T for ϕ such that T is proper and T agrees with M. Furthermore, ϕ is satisfiable iff there is a bundled tableau T for ϕ such that T is propositionally consistent and T is proper.

Proof: As in Prop. 2.2.33, 2.2.39 and 2.2.40. $\square$

2.3.3 Second-order quantifiers

Let us then re-examine the calculi with second-order quantifiers introduced in Subsections 2.2.5 and 2.2.6 and generalise these to branching structures. Corresponding to $\exists TL$ and *S1S*, for any $n \in \mathbb{N}$ there are two calculi $\exists Kn$ and *SnS* interpreted over n-branching trees.

Definition 2.3.11 The formulae of the *quantified branching time temporal logic* $\exists Kn$ are defined by the abstract syntax:

$$\phi ::= z \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \textcircled{\scriptsize i}\,\phi \mid \mathbf{G}\phi \mid \exists z.\phi$$

where z varies over $\mathcal{Z}$ and i over $[n]$.

Let $M \in \mathcal{M}_n$ be an n-branching model. The set of states of M satisfying an $\exists Kn$-formula ϕ, denoted by $\|\phi\|_M$, is defined for z, $\neg\phi$, $\phi \wedge \phi'$ and $\exists z.\phi$ as in Def. 2.2.43, for $\textcircled{\scriptsize i}\,\phi$ as in Def. 2.3.3 and by

$$\|\mathbf{G}\phi\|_M \quad = \quad \{s \in \mathrm{st}(M) \mid \forall s' \succeq s : s' \in \|\phi\|_M\}$$

The notation $M, s \models \phi$ etc. and the related concepts are as in Def. 2.2.4. $\square$

Proposition 2.3.12 For every μKn-formula ϕ there is an $\exists Kn$-formula ψ such that $\phi \Leftrightarrow \psi$, i.e. $\|\phi\|_M = \|\psi\|_M$ for all n-branching models $M \in \mathcal{M}_n$.

Proof: A translation f from μKn to $\exists Kn$ is defined as in 2.2.44, except for $f(\textcircled{\scriptsize i}\,\phi) = \textcircled{\scriptsize i}\,f(\phi)$. The correctness of the translation is justified by Lemma 2.2.9 and Prop. 2.3.6. $\square$

Definition 2.3.13 The formulae of the *monadic second-order theory of n successors SnS* are defined as for *S1S* in Def. 2.2.45, except that individual terms are generated by the abstract syntax

$$t ::= \epsilon \mid x \mid s_i(t)$$

where x varies over $\mathcal{X}$ and i over $[n]$, and atomic formulae by the abstract syntax

$$\phi_A ::= z(t) \mid t = t' \mid t \prec t'$$

We use the abbreviation $t \preceq t'$ to stand for $(t = t') \vee (t \prec t')$.

The semantics of *SnS*-formulae with respect to an n-branching model $M \in \mathcal{M}_n$ and a valuation $V : \mathcal{X} \to \mathrm{st}(M)$ are defined as for *S1S* in Def. 2.2.46 except that the valuation V is extended to all terms by

$$\begin{aligned} V(\epsilon) &= \epsilon \\ V(s_i(t)) &= V(t) \cdot i \end{aligned}$$

and the truth definition for $t \prec t'$ is given by

$$M, V \models t \prec t' \quad \text{iff} \quad V(t) \prec V(t')$$

□

Like $\exists TL$ and *S1S*, the branching formalisms $\exists Kn$ and *SnS* are so easily reducible to each other that we can effectively consider them two formulations of the same language.

Proposition 2.3.14 For every $\exists Kn$-formula ϕ, there exists a closed *SnS*-formula ψ such that $M \models \phi$ iff $M \models \psi$, for every n-branching model M. Conversely, for every closed *SnS*-formula ϕ there exists an $\exists Kn$-formula ψ such that $M \models \phi$ iff $M \models \psi$, for every n-branching model M.

Proof: A translation f from $\exists Kn$ to *SnS* is defined as in 2.2.48, except for

$$\begin{aligned} f(\textcircled{i}\phi) &= f(\phi)[s_i(w)/w] \\ f(\mathbf{G}\phi) &= \forall x. w \preceq x \Rightarrow f(\phi)[x/w] \end{aligned}$$

In the other direction, as in 2.2.49 we can rewrite atomic formulae so that they are of the forms $p(x)$, $x = 0$, $x = s_i(y)$ or $x \prec y$, where x and y are individual variables. The translation f from *SnS* to $\exists Kn$ is then defined as in 2.2.49, except for

$$\begin{aligned} f(x = s_i(y)) &= \mathbf{G}(y \Rightarrow \textcircled{i}x) \\ f(x \prec y) &= \mathbf{G}(x \Rightarrow \bigvee_{i \in [n]} \textcircled{i}\mathbf{F}y) \end{aligned}$$

□

The weak variants $\overset{\mathrm{w}}{\exists} Kn$ and *WSnS* of $\exists Kn$ and *SnS* are formed precisely as in the linear case and have the same relations to each other. The only points that require some care are the finiteness requirements when embedding the weak calculi in the strong ones.

Definition 2.3.15 The formulae of the *weak quantified branching time temporal logic* $\overset{w}{\exists}Kn$ are defined as those of $\exists Kn$ in Def. 2.3.11, except the quantifier $\exists z$ is replaced by the weak quantifier $\overset{w}{\exists}z$. The semantics of $\overset{w}{\exists}Kn$-formulae are defined for all other operators as in Definition 2.3.11 and for $\overset{w}{\exists}z.\phi$ as in Def. 2.2.50.

The formulae of the *weak monadic second-order theory of n successors WSnS* are defined as those of *SnS* in Definition 2.3.13, except the quantifier $\exists z$ is replaced by the weak quantifier $\overset{w}{\exists}z$. The semantics of *WSnS*-formulae are defined for all other operators as in Definition 2.3.13 and for $\overset{w}{\exists}z.\phi$ as in Def. 2.2.51. □

Proposition 2.3.16 For every $\overset{w}{\exists}Kn$-formula ϕ there is a closed *WSnS*-formula ψ, and vice versa, such that $M \models \phi$ iff $M \models \psi$, for all n-branching models M.

For every $\overset{w}{\exists}Kn$-formula ϕ there is an $\exists Kn$-formula ψ such that $\phi \Leftrightarrow \psi$, i.e. $\|\phi\|_M = \|\psi\|_M$, for all n-branching models M. Similarly, for every *WSnS*-formula ϕ there is an *SnS*-formula ψ such that $M, V \models \phi$ iff $M, V \models \psi$, for all n-branching models M and valuations V.

Proof: The first claim is shown as in Prop. 2.2.52, and the second as in Prop. 2.2.53, except that for $\overset{w}{\exists}Kn$,

$$\text{finite}(z) \;=\; \neg\exists z'.z' \wedge \mathbf{G}(z' \Rightarrow (\mathbf{F}z) \wedge \bigvee_{i\in[n]} \circled{i}\mathbf{F}z')$$

and for *WSnS*

$$\text{finite}(z) \;=\; \neg\exists z'.z'(0) \wedge \forall x.(z'(x) \Rightarrow (\exists x'.(x \prec x') \wedge z(x')) \wedge (\exists x'.(x \prec x') \wedge z'(x')))$$

□

The relation between the non-alternating fragment Δ_2 of the modal mu-calculus μKn and the calculi with weak second-order quantification is analogous to the linear case.

Proposition 2.3.17 The characterisation of Lemma 2.2.54 for proper simple tableaux for Σ_1-formulae, the statements of Proposition 2.2.55 and Corollary 2.2.56 that only approximants up to $\mu^\omega z$ are needed for Δ_2-formulae, and the translation from Δ_2 to a weak second-order language in Theorem 2.2.57 work for μKn as for μTL.

Proof: Easy. □

	linear structures		n-branching structures	
fixpoints	μTL	(Def. 2.2.3)	μKn	(Def. 2.3.2)
strong quantifiers	$\exists TL$ *S1S*	(Def. 2.2.42) (Def. 2.2.45)	$\exists Kn$ *SnS*	(Def. 2.3.11) (Def. 2.3.13)
weak quantifiers	$\overset{\mathrm{w}}{\exists} TL$ *WS1S*	(Def. 2.2.50) (Def. 2.2.51)	$\overset{\mathrm{w}}{\exists} Kn$ *WSnS*	(Def. 2.3.15) (Def. 2.3.15)

Table 2.5: A menagerie of logical formalisms

2.4 Summary

In this chapter we have introduced a variety of logical calculi for describing properties of infinite strings and trees. The most important of these, as far as the current work is concerned, are the linear-time mu-calculus μTL and its branching time counterpart, the (indexed) modal mu-calculus μKn. Various basic techniques and concepts for these fixpoint-based formalisms, including fixpoint approximants, tableaux and the phenomenon of fixpoint alternation, were discussed in Subsections 2.2.2–2.2.4. In Subsections 2.2.5 and 2.2.6 we examined another second-order construct, quantification over propositions, and the related calculi $\exists TL$, the quantified linear time temporal logic, and *S1S*, the monadic second-order theory of one successor, and their weak versions $\overset{\mathrm{w}}{\exists} TL$ and *WS1S*. Branching variants for these were introduced, as well. Table 2.5 presents a quick overview of all the logical calculi discussed in the current work. We also described translations between the different formalisms. These have been summarised in Figures 2.3 and 2.4.

$$\begin{tikzcd}
\mu TL \arrow[r, "(2.2.44)"] & \exists TL \arrow[r, shift left, "(2.2.48)"] & \mathit{S1S} \arrow[l, shift left, "(2.2.49)"] \\
\Delta_2 \text{ (of } \mu TL) \arrow[u, "(\text{obv.})"'] \arrow[r, "(2.2.57)"'] & \overset{\text{w}}{\exists} TL \arrow[u, "(2.2.53)"'] \arrow[r, leftrightarrow, "(2.2.52)"'] & \mathit{WS1S} \arrow[u, "(2.2.53)"']
\end{tikzcd}$$

Figure 2.3: Relations of linear formalisms

$$\begin{tikzcd}
\mu Kn \arrow[r, "(2.3.12)"] & \exists Kn \arrow[r, leftrightarrow, "(2.3.14)"] & \mathit{SnS} \\
\Delta_2 \text{ (of } \mu Kn) \arrow[u, "(\text{obv.})"'] \arrow[r, "(2.3.17)"'] & \overset{\text{w}}{\exists} Kn \arrow[u, "(2.3.16)"'] \arrow[r, leftrightarrow, "(2.3.16)"'] & \mathit{WSnS} \arrow[u, "(2.3.16)"']
\end{tikzcd}$$

Figure 2.4: Relations of branching formalisms

Chapter 3

Automata

Where previous chapter examined the topic of characterising infinite strings and trees in terms of different logical calculi, we now take a step back and approach the issue from a different angle, by using automata on infinite objects. Although at face value very different, the two approaches turn out to be deeply interconnected.

First, in Section 3.1, we define the notions of automata on infinite strings and trees, explain how these can be generalised to alternating automata, and describe the usual Büchi and Rabin acceptance conditions. As we relate automata on strings to linear-time mu-calculus and automata on trees to modal mu-calculus by the same constructions, in the following paragraphs we just talk about relating automata to mu-calculus, meaning both of these.

In Section 3.2 we introduce a new type of automata, the *first recurrence* automata. The acceptance condition for these is based on requiring that an automaton has a tree-like structure, and checking whether the oldest infinitely often occurring state in a path of a run belongs to a designated set of accepting states. These automata can be seen as a simplification of the parity automata of [65, 33]. We show in Subsection 3.2.2 that first recurrence automata are particularly appropriate for understanding mu-calculi, since alternating first recurrence automata correspond to formulae of mu-calculi and vice versa, via easy syntactic translations. Ordinary, non-alternating first recurrence automata correspond to formulae in a restricted normal form. The most important restriction in this form is the so-called *strong aconjunctivity*[1], which severely limits situations in which conjunction operators may occur. We view first recurrence automata and mu-calculus formulae as two different syntactic representations of essentially the same object, allowing us to rephrase the statement about mu-calculus formulae being alternating automata in the other direction: *alternating automata are really mu-calculus formulae.* In our opinion the choice between the two representations is above all a matter of taste.

[1]This is different from Kozen's notion of aconjunctivity [55], although closely related.

The first recurrence automata provide a common ground on which mu-calculi and automata with more usual Büchi and Rabin acceptance conditions can be related to each other. In Subsection 3.2.3 we show that alternating first recurrence automata can be easily translated to alternating Rabin automata and vice versa, and that the same holds for ordinary, non-alternating automata. This means that alternating Rabin automata and the mu-calculus are equiexpressive, and that ordinary Rabin automata and the restricted fragment of mu-calculus are similarly related. By translations between Büchi and first recurrence automata, we also show that alternating Büchi automata and the fixpoint alternation class Π_2 of mu-calculus are equiexpressive, and that the same holds for ordinary Büchi automata and the restricted fragment of Π_2.

Subsection 3.2.4 examines the problem of deciding the emptiness of an ordinary first recurrence automaton, i.e. deciding whether there exists some input string or tree accepted by a given automaton. The problem turns out to be very easy; emptiness can be decided in a time which is linear in the size of the automaton.

In Section 3.3 we address the role of ordinary automata in decision procedures for second-order calculi like $\exists Kn$ and *SnS*. These decision procedures work by inductively translating formulae to automata, which can be viewed as a convenient normal form for formulae. For this it is essential that the class of automata is closed under operators of the language. A key advantage of ordinary automata in this task is that they are trivially closed under existential second-order quantification, but a drawback is that complementation is hard for them. On the other hand, for alternating automata, especially alternating first recurrence automata or mu-calculus formulae, the situation is directly opposite: complementation is easy but quantification hard. What would allow us to enjoy the best of both worlds is a translation from alternating automata to ordinary ones, as then both quantification and complementation would be possible. For a first example of such a translation, we show in Section 3.3 how a simple subclass of mu-calculus formulae, the fixpoint alternation class Σ_1, can be translated to restricted mu-calculus formulae in Σ_1, i.e. how a subclass of alternating first recurrence automata can be transformed to ordinary first recurrence automata. A modification of the construction allows us to translate the weak calculus $\overset{\mathrm{w}}{\exists}Kn$ (i.e. *WSnS*) inductively to the class Δ_2 of mu-calculus formulae without any proper alternation of fixpoints, showing the equiexpressivity of these languages. For mu-calculus formulae in this class Δ_2, the acceptance condition in the first recurrence automaton corresponding to the formula coincides with the *weak acceptance* of Muller, Saoudi and Schupp [68], which means that formulae in Δ_2 correspond to weak alternating automata, and vice versa.

Translations from mu-calculus to the restricted fragment of it, or from alternating automata to ordinary, non-alternating automata are further examined in Section 3.4. First, in Subsection 3.4.1 we describe a translation from the fixpoint alternation class Π_2 of mu-calculus to ordinary Büchi automata. This is done inductively on the structure of formulae, by providing for each operator of the calculus a corresponding construction on ordinary automata. The novelty here is a powerset construction, related to [22, 77], which corresponds to maximal and minimal fixpoints. Since we already know how to map Büchi automata via first recurrence automata to mu-calculus formulae in Π_2, we have then a precise correspondence between Π_2 and ordinary Büchi automata. Another way of formulating this is that Π_2 and the restricted fragment of Π_2 are equiexpressive. As side products of the result, we get the decidability of $\overset{w}{\exists}Kn$ or *WSnS*, and one half of Rabin's fundamental characterisations of Büchi recognisable languages as those which are characterised by existentially quantified $\overset{w}{\exists}Kn$ or *WSnS* formulae, and of $\overset{w}{\exists}Kn$ or *WSnS* as the languages for which both the language and its complement are Büchi recognisable [77].

In Subsection 3.4.2 we generalise the fixpoint operations on ordinary automata from Büchi to Rabin acceptance, and provide an inductive translation from the full mu-calculus to ordinary Rabin automata. This shows that the full mu-calculus, i.e. alternating first recurrence automata, and ordinary Rabin automata are equiexpressive. Another way of looking at this correspondence is that the full mu-calculus and its restricted fragment are equiexpressive. An important corollary of the correspondence is that Rabin automata are closed under complementation, i.e. the result referred to above as Rabin's complementation lemma. The result also allows translating any $\exists Kn$-formula (i.e. any *SnS*-formula) inductively to an ordinary Rabin or first recurrence automaton, showing the equiexpressivity of $\exists Kn$, *SnS* and μKn, and the decidability of $\exists Kn$ and *SnS*. We believe that this provides the simplest proof of Rabin's result so far.

It should be pointed out that none of the above correspondence results are new in themselves. The equivalence between *SnS* and Rabin automata was originally shown by Rabin [76], using the extraordinarily difficult direct complementation construction for Rabin automata. The natural translation from modal mu-calculus to *SnS* has been formulated at least in [4, 6, 56, 59], and translations between Rabin automata and what is essentially the strongly aconjunctive fragment of modal mu-calculus were given by Niwiński in [70, 71]. Emerson and Jutla [33] described a translation from the full mu-calculus to Rabin-automata by using the correspondence between mu-calculus and alternating tree automata, and reducing these to ordinary automata by a construction related to Safra's [80]. Furthermore, the correspondence between Büchi-automata and Π_2 was shown in [3], and the relationship between *WSnS* and the fixpoint-alternation-free fragment Δ_2 of mu-calculus in [1, 4, 68].

What we feel the contribution of the current work in this respect is that all these results that were shown by a variety of tools, arise here uniformly from two rather simple concepts: the notion of first recurrence automata, and the fixpoint constructions for ordinary Büchi and Rabin-automata. The inductive mapping of formulae to automata also allows us to concentrate on one operator of the logic at a time.

The constructions and results discussed above work for both the modal mu-calculus over trees and linear-time mu-calculus over strings. In Section 3.5 we examine differences between the linear and the branching case. In linear case it is easy to see that the more general acceptance condition of a Rabin automaton brings no greater expressive power that that of a Büchi automaton. Combined with the earlier results on the relation of Büchi automata and the fixpoint alternation class Π_2 of mu-calculus, this implies that in the linear case the full mu-calculus and its fragment Π_2 are equiexpressive, and further, that already the fragment Δ_2 without any proper fixpoint alternation is equiexpressive with the full language. For the branching case none of this is true. Another issue which distinguishes between linear and branching cases is determinisation of automata: automata on strings can be determinised, ones on trees cannot. Related to this, we discuss what determinism means in formulae, and define a deterministic normal form for linear-time mu-calculus.

3.1 Automata on infinite objects

The concepts of automata on infinite strings and trees generalise usual automata on finite objects by notions of acceptance which are appropriate for structures which do not have a final state or final boundary. Pioneering work on infinitary automata was done by Büchi [15], McNaughton [63] and Rabin [76, 77]. For an overview of the area, see [93]. We start here from the simplest concept, automata on infinite strings, and then look how these are first generalised to automata on trees and then further to alternating automata.

3.1.1 Ordinary automata

A usual non-deterministic automaton on finite strings with a finite input alphabet Σ consists of a finite set of states Q, an initial state q_{init}, a transition relation

$\Delta \subseteq Q \times \Sigma \times Q$, and a set of accepting final states. A transition (q, a, q') specifies that if the automaton is in state q and the next input letter is a, the automaton may move to state q' after reading the input letter. Disregarding the issue of an acceptance condition for the moment being, the behaviour of such an automaton generalises naturally to infinite input strings.

In the current context we would like automata to operate on the same structures that serve as models of the logical calculi discussed in previous chapter. Because of this, we modify the meaning of a transition in an automaton slightly. Instead of specifying exactly what the label of a state must be in order for a transition to be possible, a transition label of an automaton specifies a condition which must be true for the transition to be possible, but which does not completely characterise the truth set labelling the current input state. A transition of an automaton is labelled with a set Z of atomic propositions and their negations $Z \subseteq \mathcal{Z} \cup \overline{\mathcal{Z}}$. The intuition is that the propositions in $Z \cap \mathcal{Z}$ must be true and the negated propositions in $Z \cap \overline{\mathcal{Z}}$ false in order for the transition to be available in an input state.

As we shall introduce later various other type of automata, we use the qualifier 'ordinary' with usual non-deterministic automata to stress the difference. Notice that we only talk about automata operating on infinite objects in the current work: 'ordinary' does not imply finiteness.

Definition 3.1.1 An *ordinary automaton A on (infinite) strings* is a 4-tuple $A = (Q, q_{init}, \Delta, \Omega)$ where

- Q is a finite set of *states*,
- q_{init} is the *initial state*,
- $\Delta \subseteq Q \times 2^{\mathcal{Z} \cup \overline{\mathcal{Z}}} \times (Q \setminus \{q_{init}\})$ is a *transition relation* such that for every $(q, Z, q') \in \Delta$ the set Z is finite, and
- Ω is an acceptance condition, to be defined later.

We use the notation $q \xrightarrow{Z} q'$ to mean that $(q, Z, q') \in \Delta$, and $q \longrightarrow q'$ to mean that there is some Z such that $q \xrightarrow{Z} q'$. We use $\longrightarrow^*$ to denote the reflexive and transitive closure of $\longrightarrow$. □

The restriction that no transition can lead back to the initial state is inessential and purely for technical convenience.

Definition 3.1.2 Let $A = (Q, q_{init}, \Delta, \Omega)$ be an ordinary automaton on strings. A *run* π of A on a linear model $M \in \mathcal{M}_{TL}$ is an infinite sequence of transitions of A, $\pi = (q_0, Z_0, q_0')(q_1, Z_1, q_1') \ldots$ such that

- $q_0 = q_{init}$,

- for every $i \in \mathbb{N}$, $q'_i = q_{i+1}$, and
- for every $i \in \mathbb{N}$ and every $z \in \mathcal{Z}$, if $z \in Z_i$ then $z \in M(i)$, and if $\neg z \in Z_i$ then $z \notin M(i)$.

Given a run π of A, define the sequences π^{fr}, π^{lab} and π^{to} by: for every $i \in \mathbb{N}$, if $\pi(i) = (q_i, Z_i, q'_i)$ then $\pi^{\mathrm{fr}}(i) = q_i$, $\pi^{\mathrm{lab}}(i) = Z_i$ and $\pi^{\mathrm{to}}(i) = q'_i$. □

Let us then define two standard forms of acceptance condition, Büchi and Rabin acceptance. The acceptance condition of a Büchi automaton (called a *special* automaton in [77]) consists of a set of accepting states, at least one of which must occur infinitely often in a run [15]. A Rabin acceptance condition consists of a sequence of acceptance pairs, where each pair consists of a set of accepting and a set of rejecting states [76]. In order for a run to be accepting, there must be some acceptance pair for which at least one accepting state and no rejecting state occurs infinitely often in the run.

Definition 3.1.3 An *ordinary Büchi automaton on strings* is an ordinary automaton $A = (Q, q_{init}, \Delta, \Omega)$ such that the acceptance condition Ω is of the form $\Omega = F$ where $F \subseteq Q$.

A run π of an ordinary Büchi automaton A on strings is *accepting* iff $\pi^{\mathrm{fr}}(i) \in F$ for infinitely many $i \in \mathbb{N}$, i.e. iff accepting states occur infinitely often along π.

An *ordinary Rabin automaton on strings* is an ordinary automaton A where the acceptance condition Ω is of the form $\Omega = ((G_0, R_0) \ldots (G_{m-1}, R_{m-1}))$, where $m \in \mathbb{N}$, and for every $i \in [m]$, $G_i \subseteq Q$ and $R_i \subseteq Q$. We call m the *index* of the Rabin automaton A.

A run π of an ordinary Rabin automaton A on strings is *accepting* iff there is some $k \in [m]$ such that

- $\pi^{\mathrm{fr}}(i) \in G_k$ for infinitely many $i \in \mathbb{N}$, and
- $\pi^{\mathrm{fr}}(i) \in R_k$ for only finitely many $i \in \mathbb{N}$,

i.e. iff there is some acceptance pair such that accepting states in that pair occur infinitely often and rejecting states only finitely often along π.

The *language accepted* by an ordinary Büchi/Rabin automaton A, denoted by $L(A)$, is defined by $L(A) = \{M \in \mathcal{M}_{TL} \mid A \text{ has an accepting run on } M\}$. □

Figure 3.1 depicts an ordinary Büchi automaton on strings recognising the property *at every even moment a holds*. In the graph the states of the automaton are represented by small circles and transitions by arrows annotated with the transition label, except that transitions (q, Z, q') with $Z = \emptyset$ are represented by arrows without labels. The initial state is marked by a short arrow which does not originate from any state.

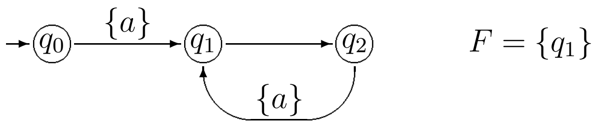


Figure 3.1: An ordinary Büchi automaton for *at every even moment a*

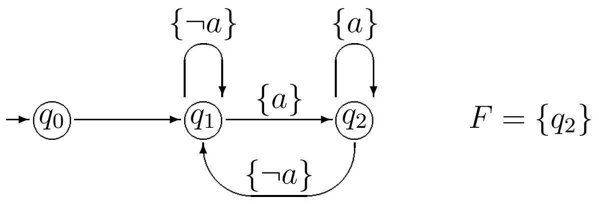


Figure 3.2: An ordinary Büchi automaton for *infinitely often a*

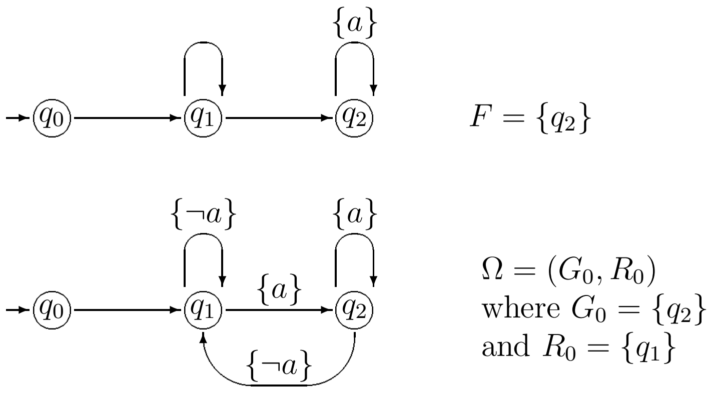


Figure 3.3: Ordinary Büchi and Rabin automata for *almost always a*

Figure 3.2 presents an ordinary Büchi automaton recognising the property *a holds infinitely often*, and Figure 3.3 Büchi and Rabin automata both recognising the property *a holds almost always* or *only finitely often not a*.

Automata on trees generalise the notion of usual string automata so that they are able to operate on branching stuctures. As before, an automaton on n-branching trees can take a transition $(q, Z, (q'_0, \dots, q'_{n-1}))$ iff it is in state q and is currently looking at an input node satisfying the condition Z. In executing the transition, the automaton intuitively splits itself into n copies, one for each child of the input node, and moves to state q'_i in the copy corresponding to the i-th child. For an introduction to automata on finite trees, see [37].

Definition 3.1.4 An *ordinary automaton A on n-branching trees* is a 4-tuple $A = (Q, q_{init}, \Delta, \Omega)$ where Q, q_{init} and Ω are as in Definition 3.1.1, and the transition relation Δ is of the form $\Delta \subseteq Q \times 2^{\mathcal{Z} \cup \overline{\mathcal{Z}}} \times (Q \setminus \{q_{init}\})^n$, where for every $(q, Z, \bar{q}) \in \Delta$ the set Z is finite.

We use the notation $q \xrightarrow{Z} \bar{q}$ to mean that $(q, Z, \bar{q}) \in \Delta$, and the notation $q \xrightarrow{Z} q'$ to mean that there is some $\bar{q}$ and $i \in [n]$ such that $(q, Z, \bar{q}) \in \Delta$ and $q' = \bar{q}_i$. We also use the notations $q \longrightarrow q'$ and $\longrightarrow^*$, defined as in Def. 3.1.1. □

Definition 3.1.5 Let $A = (Q, q_{init}, \Delta, \Omega)$ be an ordinary automaton on n-branching trees. A *run* π of A on an n-branching model $M \in \mathcal{M}_n$ is an infinite total n-branching tree labelled with transitions of A such that

- $\pi(\epsilon) = (q_{init}, Z, \overline{q})$ for some Z and $\overline{q}$,
- for every $t \in [n]^*$ and every $i \in [n]$, if we write $\pi(t) = (q, Z, \overline{q})$ and $\pi(t \cdot i) = (q', Z', \overline{q}')$, then $q' = \overline{q}_i$, and
- for every $t \in [n]^*$, if we write $\pi(t) = (q, Z, \overline{q})$, then for every $z \in \mathcal{Z}$, if $z \in Z$ then $z \in M(t)$, and if $\neg z \in Z$ then $z \notin M(t)$.

Given a run π of A, define the trees π^{fr}, π^{lab} and π^{to} as in Definition 3.1.2. □

Definition 3.1.6 The concepts of *ordinary Büchi* and *Rabin automata on n-branching trees* are defined analogously to Definition 3.1.3.

A run π of an ordinary Büchi automaton $A = (Q, q_{init}, \Delta, F)$ on n-branching trees is *accepting* iff for every path p of π, $\pi^{\text{fr}}(p(i)) \in F$ for infinitely many $i \in \mathbb{N}$, i.e. iff accepting states occur infinitely often along every path of π.

A run π of an ordinary Rabin automaton $A = (Q, q_{init}, \Delta, \Omega)$ on n-branching trees, where $\Omega = ((G_0, R_0) \dots (G_{m-1}, R_{m-1}))$, is *accepting* iff for every path p of π there is some $k \in [m]$ such that

- $\pi^{\text{fr}}(p(i)) \in G_k$ for infinitely many $i \in \mathbb{N}$, and
- $\pi^{\text{fr}}(p(i)) \in R_k$ for only finitely many $i \in \mathbb{N}$,

i.e. iff for every path of π there is some acceptance pair such that accepting states in that pair occur infinitely often and rejecting states only finitely often along the path. The notation $L(A)$ is defined for automata on trees as in Def. 3.1.3. $\square$

It is easy to see than Büchi automata are a subclass of Rabin automata.

Lemma 3.1.7 For every ordinary Büchi automaton A on strings/n-branching trees, there is an ordinary Rabin automaton A' on strings/n-branching trees such that $L(A) = L(A')$.

Proof: Let $A = (Q, q_{init}, \Delta, F)$ be a Büchi automaton. The corresponding Rabin automaton is $A' = (Q, q_{init}, \Delta, \Omega)$, where $\Omega = (F, \emptyset)$. $\square$

3.1.2 Alternating automata

The transition relation of an automaton on strings may have non-deterministic branching, i.e. there may be states q from which there are transitions (q, Z', q') and (q, Z'', q'') such that $q' \neq q''$ and Z' and Z'' are not mutually exclusive. By its nature such branching is disjunctive; if we consider that the state in which an automaton is at a certain moment of its execution specifies a requirement for the rest of the input, in a branching state q the automaton needs to verify either that the current state satisfies Z' and the rest of the input the requirement corresponding to q', or that the current state satisfies Z'' and the rest the requirement corresponding to q''.

When usual automata are generalised to so-called *alternating* automata, such disjunctive branching is complemented by conjunctive branching [19]. Intuitively, in conjunctive or and-branching the meaning of two transitions (q, Z', q') and (q, Z'', q'') to different states q' and q'' from q is that the automaton needs to verify that the current state satisfies both Z' and Z'' and that the rest of the input satisfies both the requirement corresponding to q' and the one corresponding to q''. Automata on trees already have conjunctive branching in a restricted form; a transition $(q, Z, \overline{q})$ contains the requirement that the oldest child of the current input node satisfies the requirement corresponding to $\overline{q}_0$, the next child the requirement corresponding to $\overline{q}_1$ and so on.

In automata on strings we could accommodate both disjunctive and conjunctive type of branching by changing the form of the transition relation from $\Delta \subseteq Q \times 2^{\mathcal{Z} \cup \overline{\mathcal{Z}}} \times (Q \setminus \{q_{init}\})$ to $\Delta \subseteq Q \times 2^{\mathcal{Z} \cup \overline{\mathcal{Z}}} \times 2^{(Q \setminus \{q_{init}\})}$. Intuitively the choice between two transitions (q, Z', Q') and (q, Z'', Q'') would represent or-branching, and the multiple target states $q' \in Q'$ of a transition and-branching. However, it is technically more convenient, particularly for automata on trees, to represent alternating automata by finite graphs, where every node with multiple successors is defined either as and-branching or as or-branching. Two other types of nodes are used to check for properties of the current input state, and to advance to next input state. We define alternating automata immediately for trees; alternating automata on strings are an easy subcase of these.

Definition 3.1.8 An *alternating automaton A on n-branching trees* is a 5-tuple $A = (Q, q_{init}, \Delta, l, \Omega)$, where

- Q is a finite set of *states*,
- q_{init} is the *initial state*,
- $\Delta \subseteq Q \times (Q \setminus \{q_{init}\})$ is a *transition relation*,
- $l : Q \to (\{\wedge, \vee, \top, \bot\} \cup \{ⓘ \mid i \in [n]\} \cup \mathcal{Z} \cup \overline{\mathcal{Z}})$ is a function labelling each state with a symbol $\wedge$, $\vee$, $\top$, $\bot$ or ⓘ or a (negated) atomic proposition, and
- Ω is an acceptance condition, to be defined later.

We require that for every $q \in Q$ the following restrictions hold:

- If $l(q) \in \{\top, \bot\} \cup \mathcal{Z} \cup \overline{\mathcal{Z}}$, then there is no $q' \in Q$ such that $(q, q') \in \Delta$. In these cases we call the state q *atomic*.
- If $l(q) =$ ⓘ for some $i \in [n]$, then there is exactly one $q' \in Q$ such that $(q, q') \in \Delta$. In this case we call q an ⓘ-state.
- If $l(q) \in \{\wedge, \vee\}$, then there is at least one $q' \in Q$ such that $(q, q') \in \Delta$. In these cases we call q a $\wedge$ or a $\vee$-state, respectively.

An alternating automaton on strings is defined as an alternating automaton on 1-branching trees, equating ⓪ with ◯. We use the notation $q \longrightarrow q'$ to mean $(q, q') \in \Delta$, and $\longrightarrow^*$ to denote the reflexive and transitive closure of $\longrightarrow$. □

Assume that $(q, Z_0, q_0), \ldots, (q, Z_k, q_k)$ are all the transitions from a state q of an ordinary automaton on strings or 1-branching trees, and that each $Z_i = \{z^i_0, \ldots, z^i_{m^i}\}$. In an alternating automaton this would correspond to the fragment of transition graph presented in Figure 3.4.

Definition 3.1.9 Let $A = (Q, q_{init}, \Delta, l, \Omega)$ be an alternating automaton on n-branching trees. A *run* π of A on an n-branching model $M \in \mathcal{M}_n$ is a finite or infinite tree π, every node t of which is labelled with a pair (s_t, q_t) such that

- every $s_t \in [n]^*$ and $q_t \in Q$,
- the root of π is labelled with (ϵ, q_{init}),
- the children of every node t of π are derived by applying one of the rules in Table 3.1, and
- a node t of π is a leaf only if no rule can be applied to it.

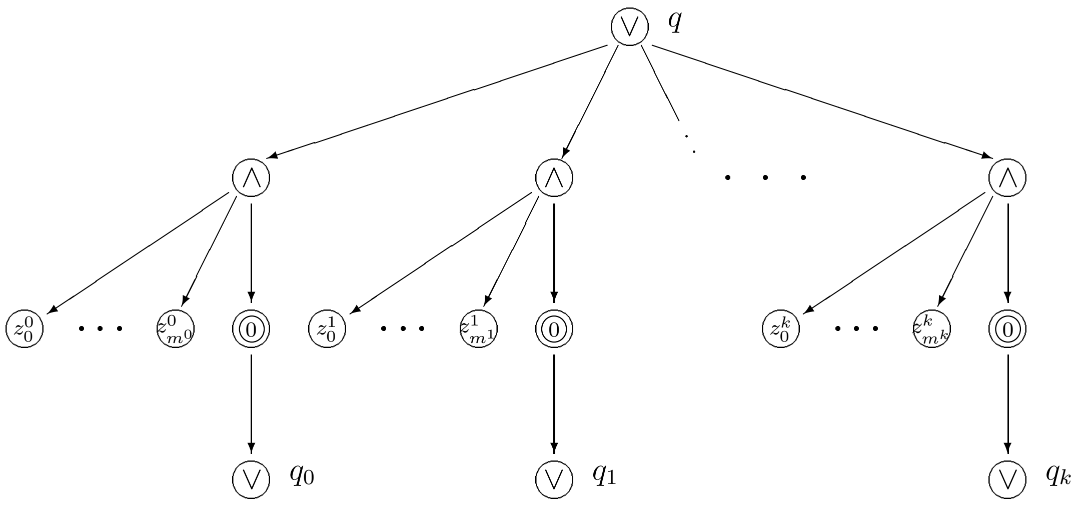


Figure 3.4: Fragment of an alternating automaton

name	application	
$\vee$	$\dfrac{s,\quad q}{s,\quad q'}$	where $l(q) = \vee$ and $(q, q') \in \Delta$
$\wedge$	$\dfrac{s,\ q}{s,\ q_0 \qquad \cdots \qquad s,\ q_k}$	where $l(q) = \wedge$ and $\{q_0, \ldots, q_k\} = \{q' \in Q \mid (q, q') \in \Delta\}$
$\textcircled{i}$	$\dfrac{s, \qquad q}{s \cdot i, \quad q'}$	where $l(q) = \textcircled{i}$ and $(q, q') \in \Delta$

Table 3.1: Alternating automaton run rules

Furthermore, every leaf t of π is required to fulfil the following:

- $l(q_t) \neq \bot$,
- if $l(q_t) = z \in \mathcal{Z}$, then $z \in M(s_t)$, and
- if $l(q_t) = \neg z \in \overline{\mathcal{Z}}$, then $z \notin M(s_t)$.

Given a run π of A, define trees π^{fr} and π^{st} by: for every $t \in \mathrm{dom}(\pi)$, if $\pi(t) = (s, q)$, then $\pi^{\mathrm{fr}}(t) = q$ and $\pi^{\mathrm{st}}(t) = s$. □

Notice the close resemblance between a run of an alternating automaton and a simple tableau for a mu-calculus formula (Def. 2.2.30). The Büchi and Rabin acceptance conditions generalise naturally from ordinary to alternating automata.

Definition 3.1.10 An *alternating Büchi automaton on n-branching trees* is an alternating automaton $A = (Q, q_{init}, \Delta, \Omega)$ such that the acceptance condition Ω is of the form $\Omega = F$ where $F \subseteq Q$. A run π of an alternating Büchi automaton A on n-branching trees is accepting iff for every infinite path p of π, $\pi^{\mathrm{fr}}(p(i)) \in F$ for infinitely many $i \in \mathbb{N}$.

An *alternating Rabin automaton on n-branching trees* is an alternating automaton $A = (Q, q_{init}, \Delta, \Omega)$ where the acceptance condition Ω is of the form $\Omega = ((G_0, R_0) \dots (G_{m-1}, R_{m-1}))$, where $m \in \mathbb{N}$, and for every $i \in [m]$, $G_i \subseteq Q$ and $R_i \subseteq Q$. We call m the index of A. A run π of an alternating Rabin automaton A on n-branching trees is accepting iff for every infinite path p of π there is some $k \in [m]$ such that

- $\pi^{\mathrm{fr}}(p(i)) \in G_k$ for infinitely many $i \in \mathbb{N}$, and
- $\pi^{\mathrm{fr}}(p(i)) \in R_k$ for only finitely many $i \in \mathbb{N}$.

The notation $L(A)$ is defined for alternating automata as in Def. 3.1.3. □

The use of alternating automata for studying infinite tree languages has been advocated especially by Muller and Schupp [66, 67], who state: *[...] we do feel that alternating automata provide the 'natural' theory of automata on trees* [67]. In addition to some inessential technical differences with the definition here, the alternating automata of Muller and Schupp use a more general and powerful type of acceptance condition, allowing Ω to be any Borel subset of Q^{ω} (for discussion on the Borel hierarchy in this context, see [93, pp. 152-156]). The advantage of this is that alternating automata with the more general acceptance condition can be trivially complemented. This is not true for the Büchi or Rabin acceptance conditions, as the complements of these conditions are not in the same form as the conditions themselves. However, we will not discuss the Muller and Schupp type acceptance condition in the current work, since the *first recurrence* automata, to be introduced in the next section, are easy to complement but their acceptance condition is still much simpler than the general Borel case. Alternating automata on infinite strings have also been examined by Miyano and Hayashi [64] and Lindsay [61].

Ordinary automata, without general and-branching, correspond to a particular subclass of alternating automata, called here *restricted* alternating automata. The most important limitation imposed on this class is *strong aconjunctivity*, which restricts severely situations in which $\wedge$-states may occur.

Definition 3.1.11 Let $A = (Q, q_{init}, \Delta, l, \Omega)$ be an alternating automaton on n-branching trees. We say that A is *strongly aconjunctive* iff for every $\wedge$-state $q \in Q$ the following restrictions hold:

- for every $q' \in Q$ such that $(q, q') \in \Delta$, q' is either an atomic state or a $\textcircled{i}$-state for some $i \in [n]$, and
- for every $q' \in Q$ and $q'' \in Q$ such that $(q, q') \in \Delta$, $(q, q'') \in \Delta$ and $q' \neq q''$, if $l(q') = \textcircled{i}$ and $l(q'') = \textcircled{j}$, then $i \neq j$. □

To characterise the class of alternating automata corresponding to ordinary ones, we also need some more technical side conditions.

Definition 3.1.12 Let $A = (Q, q_{init}, \Delta, l, \Omega)$ be an alternating automaton on n-branching trees. We say that A is *guarded* iff for every sequence of states $q_0, \ldots, q_k$ such that $q_0 = q_k$, $k > 0$, and $(q_i, q_{i+1}) \in \Delta$ for every $0 \leq i < k$, there is some $0 \leq j < k$ such that q_j is a $\textcircled{i}$-state, i.e. iff every loop in the transition graph has at least one $\textcircled{i}$-state in it. □

Definition 3.1.13 Let $A = (Q, q_{init}, \Delta, l, \Omega)$ be an alternating automaton. We call a state $q \in Q$ *principal* iff either $q = q_{init}$ or there is some $\textcircled{i}$-state $q' \in Q$ such that $(q', q) \in \Delta$. We call a state $q \in Q$ *auxiliary* iff there is some $\wedge$ or $\vee$-state $q' \in Q$ such that $(q', q) \in \Delta$. We say that an auxiliary state q' is a *subordinate* of a principal state q iff there is a sequence $q_0, \ldots, q_k$ of states, where $k \geq 0$, such that $q_0 = q$, $q_k = q'$, $(q_i, q_{i+1}) \in \Delta$ for every $0 \leq i < k$, and q_i is an auxiliary state for every $1 \leq i \leq k$. □

In other words, a state is principal iff it is the initial state or there is a transition from some $\textcircled{i}$-state to it, and it is auxiliary iff there is a transition from some $\wedge$ or $\vee$-state to it. Notice that a state can be both principal and auxiliary at the same time.

Definition 3.1.14 Let A be an alternating automaton. We say that A is *restricted* iff

- A is strongly aconjunctive,
- A is guarded,
- only principal states appear in the acceptance condition of A, and
- every auxiliary state is a subordinate of exactly one principal state. □

The next two lemmas state that ordinary automata are really restricted alternating automata and vice versa. Since the proofs are not very interesting, they have been postponed to Appendix A. In the following we regularly use the correspondence, allowing us to use whichever formulation happens to be more convenient for the task at hand.

Lemma 3.1.15 For every ordinary Büchi automaton A there exists a restricted alternating Büchi automaton A', and vice versa, such that $L(A) = L(A')$.

Proof: See Appendix A. □

Lemma 3.1.16 For every ordinary Rabin automaton A there exists a restricted alternating Rabin automaton A', and vice versa, such that $L(A) = L(A')$.

Proof: See Appendix A. □

3.2 First recurrence automata

In this section we introduce a new type of automata, the *first recurrence* automata. Their acceptance condition can be seen as a simplification of the parity acceptance condition used by Mostowski [65] and Emerson and Jutla [33]. Due to their structure, first recurrence automata are so close to formulae of mu-calculi in positive normal form that we can identify the two. What is more, there are easy translations between first recurrence automata on the one hand, and the more traditional Büchi and Rabin automata on the other hand. This means that first recurrence automata provide a useful common ground on which mu-calculi and automata can be related to each other. Moreover, deciding emptiness is very easy for frst recurrence automata, as this can be done in a linear time.

3.2.1 Preliminaries

The acceptance condition of first recurrence automata is based on imposing a tree structure on the transition graph of an automaton so that all transitions from a state lead to its children or (not necessarily proper) ancestors. For every path of a run of such an automaton there is a unique state which is the most senior, in the ancestral relation, of all states which occur infinitely often along the path. The acceptance is then based on looking at whether this state belongs to a specified set of accepting states or not. Let us start by defining both ordinary and alternating versions of first recurrence automata.

Definition 3.2.1 Let $A = (Q, q_{init}, \Delta, \Omega)$ be an ordinary automaton on n-branching trees. We say that A is *tree-like* iff

- the set of states $Q \subseteq \mathbb{N}^*$ forms a finite tree (see Def. 2.1.3),
- the initial state q_{init} is the root ϵ, and
- for every transition $(q, Z, \overline{q}) \in \Delta$ and every $i \in [n]$, the state $\overline{q}_i$ is either a child or an ancestor of q.

Let $A = (Q, q_{init}, \Delta, l, \Omega)$ be an alternating automaton on n-branching trees. We say that A is *tree-like* iff

- $Q \subseteq \mathbb{N}^*$ forms a finite tree,
- the initial state q_{init} is the root ϵ, and
- for every transition $(q, q') \in \Delta$ the state q' is a child or an ancestor of q.

We call a state q of a tree-like ordinary or alternating automaton a *loop state* iff there is a transition $q' \longrightarrow q$ from some descendant q' of q back to q. The *depth* of a tree-like ordinary or alternating automaton is the depth of the tree Q. □

Definition 3.2.2 An *ordinary* or *alternating first recurrence automaton* (abbreviated *FR-automaton*) on n-branching trees, is an ordinary or alternating automaton A on n-branching trees such that

- A is tree-like, and
- the acceptance condition Ω of A is of the form $\Omega = (G, R)$, where $G \cap R = \emptyset$ and $G \cup R$ is the set of loop states of A.

A run π of an ordinary or alternating FR-automaton A is *accepting* iff for every infinite path p of π, $q \in G$ where q is the element of Q such that

- $\pi^{\mathrm{fr}}(p(i)) = q$ for infinitely many $i \in \mathbb{N}$, and
- for every proper ancestor q' of q, $\pi^{\mathrm{fr}}(p(i)) = q'$ for only finitely many $i \in \mathbb{N}$.

An ordinary or alternating FR-automaton on strings is defined as an ordinary or alternating FR-automaton on 1-branching trees. □

Notice that an infinite path p of a run π of an FR-automaton fails the acceptance condition iff $q \in R$ where q is defined as above. As with Büchi and Rabin automata, ordinary first recurrence automata correspond to restricted alternating first recurrence automata.

Lemma 3.2.3 For every ordinary first recurrence automaton A there exists a restricted alternating first recurrence automaton A', and vice versa, such that $L(A) = L(A')$.

Proof: See Appendix A. $\square$

3.2.2 First recurrence automata and mu-calculi

The tree-like structure and the acceptance condition of the first recurrence automata mean that alternating FR-automata resemble closely mu-calculus formulae in positive normal form. In fact, the correspondence is so close that we can view FR-automata as a representation of mu-calculus formulae, and vice versa. Let us make this correspondence explicit.

Definition 3.2.4 Let $A = (Q, q_{init}, \Delta, l, (G, R))$ be an alternating FR-automaton on n-branching trees. Fix for every loop state $q \in G \cup R$ a fresh distinct variable $z(q)$. Define the *μKn-formula corresponding to A*, denoted by $\phi(A)$, by $\phi(A) = \phi(q_{init})$, where for every $q \in Q$ the formula $\phi(q)$ is defined inductively as follows:

$$\phi(q) \quad = \quad \begin{cases} \nu z(q).\psi(q) & \text{if } q \in G \\ \mu z(q).\psi(q) & \text{if } q \in R \\ \psi(q) & \text{if } q \text{ is not a loop state} \end{cases}$$

where

$$\psi(q) \quad = \quad \begin{cases} \bigvee_{(q,q')\in\Delta} \psi(q,q') & \text{if } l(q) = \vee \\ \bigwedge_{(q,q')\in\Delta} \psi(q,q') & \text{if } l(q) = \wedge \\ \textcircled{\scriptsize i}\,\psi(q,q') & \text{if } l(q) = \textcircled{\scriptsize i} \\ z & \text{if } q \text{ is atomic and } l(q) = z \end{cases}$$

where

$$\psi(q,q') \quad = \quad \begin{cases} \phi(q') & \text{if } q' \text{ is a child of } q \\ z(q') & \text{if } q' \text{ is an ancestor of } q \end{cases}$$

For an ordinary FR-automaton A, the formula $\phi(A)$ is defined by viewing A as a restricted alternating FR-automaton, as in Lemma 3.2.3. $\square$

Definition 3.2.5 Let ϕ be a μKn-formula in pnf. We define the *alternating FR-automaton corresponding to ϕ*, denoted by $A(\phi)$, by the following construction. Define inductively a finite tree T labelled with subformulae of ϕ by:

- the root of T is labelled with $T(\epsilon) = \phi$,
- if $T(t)$ is of the forms $\psi \wedge \psi'$ or $\psi \vee \psi'$, then t has two children, labelled with ψ and ψ', respectively,

- if $T(t)$ is of the form $\textcircled{i}\psi$ then t has one child labelled with ψ,
- if $T(t)$ is of the form $\sigma z.\psi$ then t has one child labelled with ψ,
- if $T(t)$ is labelled with an atomic formula, then t is a leaf.

Define the automaton $A(\phi) = (Q, q_{init}, \Delta, l, (G, R))$ by:

- for every $t \in \mathrm{dom}(T)$, t belongs to Q, except if $T(t) = z \in \mathcal{Z}$ and there is some ancestor t' of t such that $T(t') = \sigma z.\psi$ for some ψ,
- $q_{init} = \epsilon$,
- $(t, t') \in \Delta$ iff $t, t' \in Q$ and either t' is a child of t, or t' is an ancestor of t and there is some child t'' of t such that t'' is a leaf of T, $T(t'') = z$ and $T(t') = \sigma z.\psi$ for some ψ.
- for every $t \in Q$,
  - $l(t) = \vee$ iff $T(t)$ is of the forms $\psi \vee \psi'$ or $\sigma z.\psi$,
  - $l(t) = \wedge$ iff $T(t)$ is of the form $\psi \wedge \psi'$,
  - $l(t) = \textcircled{i}$ iff $T(t)$ is of the form $\textcircled{i}\psi$, and
  - $l(t) = z$ for an atomic z iff $T(t) = z$,
- $G = \{t \in Q \mid T(t) \text{ is of the form } \nu z.\psi\}$, and
- $R = \{t \in Q \mid T(t) \text{ is of the form } \mu z.\psi\}$. □

Proposition 3.2.6 For every alternating FR-automaton A on n-branching trees, $L(A) = L(\phi(A))$. Conversely, for every μKn-formula ϕ in pnf, $L(\phi) = L(A(\phi))$.

Proof: Take any model $M \in \mathcal{M}_n$. By Propositions 2.2.33 and 2.3.10, $M \models \phi(A)$ iff there is a proper simple tableau T for $\phi(A)$ agreeing with M. From such a tableau T is easy to read an accepting run π of A on M, and vice versa. The converse claim is shown in the same way. □

An important feature which alternating first recurrence automata share with mu-calculus formulae is that complementation is easy for them. Remembering that FR-automata can be viewed as formulae in positive normal form, complementation corresponds to negating the formula and transforming the negated formula back to positive normal form. Phrased in automata-theoretic terminology the complementation of an alternating first recurrence automaton amounts to changing all $\vee$-states to $\wedge$-states and vice versa, negating the labels of all atomic states, and complementing the acceptance condition by changing (G, R) to (R, G).

Thanks to the correspondence between FR-automata and mu-calculus formulae, most concepts related to formulae are also meaningful in connection with automata, in particular the notion of activeness of a variable in a formula, and the fixpoint alternation classes discussed in the context of formulae in Subsection 2.2.3.

Definition 3.2.7 Let $A = (Q, q_{init}, \Delta, l, (G, R))$ be an alternating FR-automaton, and let $q, q' \in Q$ be states of A such that q is an ancestor of q'. We say that q is *active* in q' iff either

- there is a transition to q from some descendant of q', or
- there is some q'' such that q is an ancestor of and active in q'', q'' is an ancestor of q', and there is a transition to q'' from some descendant of q'. □

Definition 3.2.8 Let us define classes Π_m, Σ_m and Δ_m of alternating FR-automata as follows, for every $m \in \mathbb{N}$.

An alternating FR-automaton $A = (Q, q_{init}, \Delta, l, (G, R))$ belongs to the class Σ_m iff there is no sequence $q_1, \ldots, q_m$ of loop states of A such that

1 q_i is an ancestor of and active in q_{i+1}, for every $1 \leq i < m$, and

2 $q_i \in G$ iff i is odd, and $q_i \in R$ iff i is even, for every $1 \leq i \leq m$.

Symmetrically, $A \in \Pi_m$ iff iff there is no sequence $q_1, \ldots, q_m$ of loop states of A such that condition 1 above holds and

2' $q_i \in R$ iff i is odd, and $q_i \in G$ iff i is even, for every $1 \leq i \leq m$.

Furthermore, $A \in \Delta_m$ iff $A \in \Sigma_m$ and $A \in \Pi_m$. □

This characterisation of fixpoint alternation classes matches exactly the earlier one for formulae. The simple characterisation of the alternation classes Σ_2 and Π_2 in Corollary 2.2.26 also carries over to automata.

Lemma 3.2.9 For any $m \in \mathbb{N}$ and μKn-formula ϕ, $\phi \in \Sigma_m$ iff $A(\phi) \in \Sigma_m$, and the same holds for Π_m and Δ_m. Conversely, for any $m \in \mathbb{N}$ and alternating FR-automaton A, $A \in \Sigma_m$ iff $\phi(A) \in \Sigma_m$, and the same holds for Π_m and Δ_m.

Proof: Obvious from Definitions 3.2.4 and 3.2.5 of $\phi(A)$ and $A(\phi)$, Definition 3.2.8 of Σ_m for automata, and the characterisation of Σ_m for formulae in Proposition 2.2.25. □

Lemma 3.2.10 For any alternating FR-automaton $A = (Q, q_{init}, \Delta, l, (G, R))$, we have $A \in \Sigma_2$ iff there are no loop states q and q' of A such that

- q is an ancestor of q',
- there is a transition to q from some descendant of q',
- $q \in G$ and $q' \in R$.

Symmetrically, $A \in \Pi_2$ iff there are no loop states q and q' of A such that

- q is an ancestor of q',
- there is a transition to q from some descendant of q',
- $q \in R$ and $q' \in G$.

Proof: Immediate from definitions. □

The correspondence between first recurrence automata and mu-calculus formulae also allows transport of ideas in the other direction. Let us characterise the class of formulae which correspond to restricted alternating FR-automata, i.e. ordinary FR-automata.

Definition 3.2.11 Let ϕ be a μKn-formula in pnf. We say that ϕ is *strongly aconjunctive* iff for all subformulae of ϕ of the form $\phi_0 \wedge \ldots \wedge \phi_{m-1}$,

- for every $i \in [m]$, either
  - ϕ_i is an atomic formula not bound by a fixpoint in ϕ, or
  - ϕ_i is of the form $\textcircled{k}\psi$, and
- for every $i, j \in [m]$ such that $i \neq j$, if $\phi_i = \textcircled{k}\psi$ and $\phi_j = \textcircled{l}\psi'$, then $k \neq l$.

□

For linear-time μTL-formulae, the conditions degenerate to: if $\phi_i = \bigcirc\psi$, then for every $j \neq i$, ϕ_j is atomic and not bound by any fixpoint.

Definition 3.2.12 Let ϕ be a μKn-formula in pnf. We say that ϕ is *strongly guarded* iff every fixpoint subformula $\sigma z.\psi$ of ϕ is immediately enclosed in a $\textcircled{i}$-operation, and every occurrence of z in ψ is also immediately enclosed in a $\textcircled{i}$-operation. □

For example, a formula of the type $\textcircled{2}(\sigma z.\psi \vee \sigma z'.\psi')$ does not fulfil the requirements of strong guardedness, since the fixpoints are not immediately enclosed in a $\textcircled{i}$-operation, unlike e.g. in the formula $(\textcircled{2}\sigma z.\psi) \vee (\textcircled{2}\sigma z'.\psi')$. It is easy to see that strong guardedness implies guardedness for formulae.

Definition 3.2.13 Let ϕ be a μKn-formula in pnf. We say that ϕ is *restricted* iff

- ϕ is strongly aconjunctive, and
- ϕ is strongly guarded. □

Lemma 3.2.14 If ϕ is a restricted μKn-formula in pnf, then the alternating FR-automaton $A(\phi)$ is restricted, as well. Conversely, if A is a restricted alternating FR-automaton, then the μKn-formula $\phi(A)$ is also restricted.

Proof: Straightforward from Definitions 3.1.14 and 3.2.13. □

3.2.3 First recurrence vs. Büchi and Rabin acceptance

As mentioned earlier, an important reason for introducing first recurrence automata in the current work has been to enable examination of the relations of fixpoint calculi and automata. In previous subsection we studied the direct connection between first recurrence automata and mu-calculus formulae. Now we shall look at the relation of FR-automata and the more traditional Büchi and Rabin acceptance conditions. Although not quite as immediate as the correspondence between FR-automata and mu-calculus, the relations between first reccurrence and the other acceptance conditions are not too involved either. We shall see that Rabin and first recurrence automata can be translated to each other, and that Büchi automata correspond precisely to first recurrence automata in class Π_2. Let us start from Büchi automata, since the constructions for these are slightly easier.

Definition 3.2.15 Let $A = (Q, q_{init}, \Delta, l, (G, R))$ be an alternating first recurrence automaton such that $A \in \Pi_2$. We define the *alternating Büchi automaton A' corresponding to A* as $A' = (Q, q_{init}, \Delta, l, G)$. □

Lemma 3.2.16 Let A and A' be as in Def. 3.2.15. Then

- $L(A) = L(A')$, and
- if A is restricted then so is A'.

Proof: To show the first claim, let π be a run of A. If π is accepting according to the first recurrence criterion, then it is obviously accepting also according to the Büchi criterion. Assume then that π is not accepting according to the FR-criterion, i.e. that there is a path p of π and $q \in Q$ such that $q \notin G$ (implying $q \in R$), $q = \pi^{\text{fr}}(p(i))$ for infinitely many $i \in \mathbb{N}$, and $q' = \pi^{\text{fr}}(p(i))$ for only finitely many $i \in \mathbb{N}$ for any proper ancestor q' of q. This means that there is some $k \in \mathbb{N}$ such that for every $i \geq k$, q is an ancestor of and active in $\pi^{\text{fr}}(p(i))$. Now, for any loop state q' such that $q' = \pi^{\text{fr}}(p(i))$ for some $i \geq k$, it must be the case that $q' \in R$ (implying $q' \notin G$), as otherwise by Lemma 3.2.10 the pair q, q' would violate the assumption that $A \in \Pi_2$. But then $\pi^{\text{fr}}(p(i)) \notin G$ for all $i \geq k$, and π is not accepting according to the Büchi criterion, either. Finally, it is obvious that if A is restricted then so is A'. □

Definition 3.2.17 Let $A = (Q, q_{init}, \Delta, l, F)$ be an alternating Büchi automaton. We define the *alternating FR-automaton A' corresponding to A* by the following construction.

Define inductively a finite tree T labelled with states $q \in Q$ by:

- the root of T is labelled with $T(\epsilon) = q_{init}$,

- if $T(t) = q$ and t is not a leaf (see below), then t has a child labelled with q' for every $q' \in Q$ such that $(q, q') \in \Delta$, and

t is a leaf of T iff either

1 there is a proper ancestor t' of t such that $T(t') = T(t)$ and $T(t') \in F$, or

2 there is a proper ancestor t' of t such that $T(t') = T(t)$ and for all t'' such that $t' \preceq t'' \preceq t$, $T(t'') \notin F$.

In both cases we call t' the loop node corresponding to t. We say that a node t' is a loop node of type 1 (type 2) iff there is some node t of T such that t' is the loop node corresponding to t and clause 1 (clause 2) above holds.

Define now $A' = (Q', q'_{init}, \Delta', l', (G, R))$ by:

- $Q' = \{t \in \mathrm{dom}(T) \mid t \text{ is not a leaf}\}$,
- $q'_{init} = \epsilon$,
- $(t, t') \in \Delta$ iff $t \in Q'$ and either
  - t' is a child of t and t' is not a leaf, or
  - there is some child t'' of t such that t'' is a leaf and t' the loop node corresponding to t'',
- for every $t \in Q$, $l'(t) = l(T(t))$,
- $G = \{t' \in Q' \mid t' \text{ is a loop node of type 1}\}$, and
- $R = \{t' \in Q' \mid t' \text{ is a loop node of type 2}\}$. □

Lemma 3.2.18 Let A and A' be as in Def. 3.2.17. Then

- $L(A) = L(A')$,
- if A is restricted then so is A',
- $A' \in \Pi_2$, and
- the depth of A' is at most $|Q|^2$.

Proof: Let us show the last claim first, since it also implies that A' is finite and in that sense well-defined. Let T be the tree used in defining A', and let $t_0, \ldots, t_k$ be any prefix of a path of T for which no t_i is a leaf. Notice first that there must be fewer than $|Q|$ points $0 \leq i \leq k$ for which $T(t_i) \in F$, since otherwise some t_i would satisfy leaf condition 1. Secondly, the distance between any two such points must be less than $|Q|$, since otherwise leaf condition 2 would be satisfied. Consequently $k \leq |Q|^2$.

To see that $L(A) = L(A')$, let us see that there is a one-to-one correspondence between the runs π of A and π' of A' such that $\mathrm{dom}(\pi) = \mathrm{dom}(\pi')$, and for all $t \in \mathrm{dom}(\pi)$, if $\pi(t) = (s, q)$ and $\pi'(t) = (s', q')$, then $s' = s$ and $T(q') = q$, where T is the tree involved in the construction of A'. For any run π of A, we can define the corresponding run π' of A' inductively by: $\pi'(\epsilon) = (\epsilon, q'_{init})$, and if $\pi'(t) = (s', q')$ and $\pi(t) = (s, q)$, where $s' = s$ and $T(q') = q$, then for every $i \in \mathbb{N}$, if $\pi(t \cdot i) = (s_i, q_i)$ then $\pi'(t \cdot i) = (s'_i, q'_i)$, where $s'_i = s_i$ and q'_i is the element of Q' such that $(q', q'_i) \in \delta$ and $T(q'_i) = q_i$. By definition of Q', such a q'_i exists and is unique. For any run π' of A', we can define the corresponding run π of A by: for all $t \in \text{dom}(\pi')$, if $\pi'(t) = (s', q')$, then $\pi(t) = (s, q)$, where $s = s'$ and $q = T(q')$.

Let us show then that a run π of A is Büchi accepting iff the corresponding run π' of A' is FR-accepting, and vice versa. Take any infinite path p of π and π'. If π' satisfies the FR-acceptance condition (G, R) along p, there is some loop state $q' \in G$ such that $q' = \pi'^{\text{fr}}(p(i))$ for infinitely many $i \in \mathbb{N}$. Defining $q = T(q')$, this implies that $q \in F$ and $q = \pi^{\text{fr}}(p(i))$ for infinitely many $i \in \mathbb{N}$, i.e. π satisfies the Büchi acceptance condition F along p. If π' does not satisfy the FR-acceptance condition (G, R) along p, there is some loop state $q' \in R$ and a bound $k \in \mathbb{N}$ such that $q' = \pi'^{\text{fr}}(p(i))$ for infinitely many $i \in \mathbb{N}$, and q' is an ancestor of and active in $\pi'^{\text{fr}}(p(i))$ for all $i \geq k$. But by the structure of A', the latter means that $\pi^{\text{fr}}(p(i)) = T(\pi'^{\text{fr}}(p(i))) \notin F$ for all $i \geq k$, i.e. π fails the Büchi acceptance condition F along p, as well.

The claim $A' \in \Pi_2$ follows immediately from the definition of A' and Lemma 3.2.10. Assume then that A is restricted. It is clear that A' is guarded and strongly aconjunctive. Loop states in G are principal by the assumption that every element of F in A is principal, and loop states in R are principal, since any path with two occurrences of an auxiliary state q must have already passed through two occurrences of the principal state that q is subordinate to. Finally, the tree-like structure of A' and the fact that loop states of A' are principal guarantee that every auxiliary state is a subordinate of exactly one principal state. □

We can now pull together some results tying together the alternation class Π_2 and Büchi acceptance.

Theorem 3.2.19 For any language $L \subseteq \mathcal{M}_n$, the following statements are mutually equivalent:

- $L = L(\phi)$ for some μKn-formula $\phi \in \Pi_2$,
- $L = L(A)$ for some alternating first recurrence automaton $A \in \Pi_2$, and
- $L = L(A)$ for some alternating Büchi automaton A.

Furthermore, the following statements are also mutually equivalent:

- $L = L(\phi)$ for some restricted μKn-formula $\phi \in \Pi_2$,
- $L = L(A)$ for some restricted alternating first recurrence automaton $A \in \Pi_2$, and
- $L = L(A)$ for some ordinary or restricted alternating Büchi automaton A.

Proof: Immediate from Proposition 3.2.6 and Lemmas 3.2.9, 3.2.16, 3.2.18, 3.2.14 and 3.1.15. □

Let us then present translations between Rabin and first recurrence automata.

Definition 3.2.20 Let $A = (Q, q_{init}, \Delta, l, (G, R))$ be an alternating first recurrence automaton. We define the *alternating Rabin automaton A' corresponding to A* as $A' = (Q, q_{init}, \Delta, l, \Omega)$, where $\Omega = ((G_0, R_0), \ldots, (G_{m-1}, R_{m-1}))$ is defined by

$$\begin{aligned} G_0 &= G \\ R_i &= \{q \in R \mid \exists q' \in G_i \text{ such that } q \text{ is active in } q'\} \\ G_{i+1} &= \{q \in G \mid \exists q' \in R_i \text{ such that } q \text{ is active in } q'\} \end{aligned}$$

and $m = \min\{i \in \mathbb{N} \mid G_i = \emptyset\}$. □

Lemma 3.2.21 Let A and A' be as in Def. 3.2.20. Then

- $L(A) = L(A')$,
- if A is restricted then so is A', and
- if $A \in \Sigma_{m+2}$ then the index of A' is at most m, for every $m \in \mathbb{N}$.

Proof: To see $L(A) = L(A')$ let π be a run of A. Assume first that π is accepting according to the Rabin condition Ω and let p be an infinite path of π. Take some $k \in [m]$ and some $q \in G_k$ such that q occurs infinitely often along p in π. Since every state $q' \in R$ which is active in q belongs to R_k by definition, no such state can occur infinitely often along p in π. Therefore, the oldest state which occurs infinitely often along p in π must belong to G and p satisfies the FR-condition (G, R). Assume then that π is accepting according to the FR-condition (G, R) and let p be an infinite path of π. Let $q \in G$ be the oldest state which occurs infinitely often along p in π. Notice that by its definition q is active in any state q' occurring infinitely often along p in π. Take then the largest $k \in [m]$ such that $q \in G_k$. Since $q \notin G_{k+1}$, it cannot be active in any $q' \in R_k$, implying that no $q' \in R_k$ can occur infinitely often along p in π. Consequently, p satisfies the Rabin acceptance pair (G_k, R_k). Finally, the second claim of the lemma is obvious and the third follows directly from the definitions of Σ_{m+2} and Ω. □

Definition 3.2.22 Let $A = (Q, q_{init}, \Delta, l, \Omega)$ be an alternating Rabin automaton, where $\Omega = ((G_0, R_0), \ldots, (G_{m-1}, R_{m-1}))$. We define the *alternating first recurrence automaton A' corresponding to A* by the following construction.

Define inductively a finite tree T, each state of which is labelled with a pair (q, c) of a state $q \in Q$ and a permutation of indices of acceptance pairs, i.e. a permutation of the string $01 \ldots (m-1)$, as follows:

- the root of T is labelled with $T(\epsilon) = (q_{init}, 01 \ldots (m-1))$,

- if $T(t) = (q, c)$ and t is not a leaf (see below), then t has a child labelled with (q', c') for every $q' \in Q$ such that $(q, q') \in \Delta$, where the sequence c' is derived from the sequence c by taking the first element $c(i)$ such that $q \in R_{c(i)}$, if any exists, and moving it to the end of the sequence.

Let t be a node of T and $T(t) = (q, c)$. Define

$$k \;=\; \max\{j \in [m] \mid \text{ for all } 0 \leq i \leq j : q \notin G_{c(i)} \cup R_{c(i)}\}$$

Then t is a leaf of T iff there is a proper ancestor t' of t labelled with $T(t') = (q', c')$ such that

- $q = q'$,
- $c[0 \ldots k] = c'[0 \ldots k]$ and either $k = m - 1$ or $c(k+1) = c'(k+1)$,
- $q'' \notin G_{c''(i)} \cup R_{c''(i)}$, for all $0 \leq i \leq k$ and all $t' \preceq t'' \preceq t$, where $T(t'') = (q'', c'')$,

and either

1 $k < m - 1$, $q \in G_{c(k+1)}$ and $q'' \notin R_{c''(k+1)}$ for all $t' \preceq t'' \preceq t$, where $T(t'') = (q'', c'')$, or

2 $k = m - 1$ or $q \in R_{c(k+1)}$.

The concepts of a loop node of type 1 and type 2 are defined as in Def. 3.2.17. The FR-automaton A' is also defined on the basis of the tree T precisely as in Def. 3.2.17. □

Lemma 3.2.23 Let A and A' be as in Def. 3.2.22. Then

- $L(A) = L(A')$,
- if A is restricted then so is A',
- $A' \in \Sigma_{m+2}$, and
- the depth of A' is at most $|Q|^{2m+1}$,

where m is the index of A.

Proof: Let us show the last claim first, since it also implies that A' is finite and in that sense well-defined. Let T be the tree used in defining A'. We shall show by descending induction on $0 \leq h \leq m$ that

> for any node t of T and any ancestor t' of t, if for every $t' \preceq t'' \preceq t$, t'' is not a leaf, and for every $t' \preceq t'' \preceq t$ and every $0 \leq i < h$, $q'' \notin G_{c''(i)} \cup R_{c''(i)}$ where $T(t'') = (q'', c'')$, then the distance between t' and t is at most $|Q|^{2(m-h)+1}$.

The claim concerning the depth of A' follows from the case $h = 0$.

The base case of the induction is $h = m$. Take any t and t' as in the induction claim. Notice that now $c = c''$ and $q'' \notin G_{c''(i)} \cup R_{c''(i)}$ for every $i \in [m]$ and every

$t' \preceq t'' \preceq t$, where $T(t'') = (q'', c'')$. But then the distance between t' and t must be less than $|Q|$, since otherwise leaf condition 2 would be satisfied by some t'' between t' and t, with $k = m - 1$.

Assume then that the claim holds for $h + 1$, and take any t and t' as in the induction claim. Notice first that there can be at most $|Q|$ points t'' between t' and t for which $q'' \in R_{c''(h+1)}$, as otherwise leaf condition 2 would be satisfied by some such t'' and $k = h - 1$. Secondly, between any two such points t'' there can be at most $|Q|$ points t''' for which $q''' \in G_{c'''(h)}$, as otherwise leaf condition 1 would be satisfied by some such t''' and $k = h-1$. Thirdly, by the induction assumption the distance between any two such t''' must be less than $|Q|^{2(m-(h+1))+1}$. Consequently, the distance between t' and t must be less than $|Q|^{2(m-h)+1}$, and the induction claim holds for h.

To see that $L(A) = L(A')$, notice that as in the proof of Lemma 3.2.18, there is a one-to-one correspondence between the runs π of A and π' of A' such that $\mathrm{dom}(\pi) = \mathrm{dom}(\pi')$, and for all $t \in \mathrm{dom}(\pi)$, if $\pi(t) = (s, q)$ and $\pi'(t) = (s', q')$, then $s' = s$ and $T(q') = (q, c)$ for some c. Let us show that a run π of A is Rabin accepting iff the corresponding run π' of A' is FR-accepting, and vice versa. Define $q_t = \pi^{\mathrm{fr}}(t)$ and $q'_t = \pi'^{\mathrm{fr}}(t)$, and write c_t for the sequence such that $T(q'_t) = (q_t, c_t)$, for every $t \in \mathrm{dom}(\pi)$.

Notice first that for any loop state q' of A' and any state q'_x of A' such that q' is an ancestor of and active in q'_x, if we write $T(q') = (q, c)$ and $T(q'_x) = (q_x, c_x)$ and define k as in Def. 3.2.22, then $c_x[0 \ldots k] = c[0 \ldots k]$ and $q_x \notin G_{c_x(i)} \cup R_{c_x(i)}$ for all $0 \le i \le k$. If additionally $q' \in G$, then $q_x \notin R_{c_x(k+1)}$ and $c(k + 1) = c_x(k + 1)$.

Take then any infinite path p of π and π'. There is some loop state q' of A' and a bound $l \in \mathbb{N}$ such that $q' = q'_{p(j)}$ for infinitely many $j \in \mathbb{N}$, and q' is an ancestor of and active in $q'_{p(j)}$ for all $j \ge l$. Let us write $T(q') = (q, c)$ and define k as in Def. 3.2.22. If π' satisfies the FR-acceptance condition (G, R) along p, i.e. if $q' \in G$, the observations of the previous paragraph mean that $c_{p(l)}(k+1) = c_{p(j)}(k+1)$ for all $j \ge l$, and if we define $h = c_{p(l)}(k + 1)$, then $q_{p(j)} \in G_h$ for infinitely many $j \in \mathbb{N}$, and $q_{p(j)} \notin R_h$ for all $j \ge l$, which means that π satisfies the Rabin acceptance pair (G_h, R_h) along p.

If π' does not satisfy the FR-acceptance condition (G, R), i.e. if $q' \in R$, the observations above mean that

- for all $0 \le i \le k$ and all $j \ge l$, $q_{p(j)} \notin G_{c_{p(j)}(i)}$ and $c_{p(j)}(i) = c_{p(l)}(i)$, and
- either $k = m - 1$ or $q_{p(j)} \in R_{c_{p(j)}(k+1)}$ for infinitely many $j \in \mathbb{N}$. By the rules of updating c in transitions, this means that for every $h \in [m]$ such that $h = c_{p(l)}(i)$ for some $k < i \le m - 1$, we have $q_{p(j)} \in R_h$ for infinitely many $j \in \mathbb{N}$.

Together these mean that for every $h \in [m]$ either $q_{p(j)} \notin G_h$ for all $j \geq l$, or $q_{p(j)} \in R_h$ for infinitely many $j \in \mathbb{N}$. Consequently, π does not satisfy the Rabin acceptance condition Ω along p either.

Let us show then $A' \in \Sigma_{m+2}$. Take any loop states q'_1 and q'_2 of A' such that q'_1 is active in q'_2, write $T(q'_1) = (q_1, c_1)$, $T(q'_2) = (q_2, c_2)$, let k_1 be defined on the basis of (q_1, c_1) as in Def. 3.2.22 and let k_2 be defined similarly. Then, if $q'_1 \in G$ and $q'_2 \in R$, we have $k_1 < k_2$, and if $q'_1 \in R$ and $q'_2 \in G$, we have $k_1 \leq k_2$. Since all such k's belong to $[m]$, the claim $A' \in \Sigma_{m+2}$ follows then directly from the definition of the class Σ_{m+2}. Finally, the fact that if A is restricted then so is A' can be seen as in Lemma 3.2.18. □

Related translations between Rabin automata and parity automata can be found in [65]. Translations between ordinary Rabin automata and what is essentially the restricted fragment of the modal mu-calculus μKn, i.e. ordinary FR-automata, have also been described by Niwiński in [70, 71]. The relation between the index of Rabin automata and the alternation class of the corresponding formula was first stated there.

We can now summarise the equiexpressiveness results obtained so far concerning the modal mu-calculus μKn, first recurrence automata and Rabin automata.

Theorem 3.2.24 For any language $L \subseteq \mathcal{M}_n$ and any $m \in \mathbb{N}$, the following statements are mutually equivalent:

- $L = L(\phi)$ for some μKn-formula $\phi \in \Sigma_{m+2}$,
- $L = L(A)$ for some alternating first recurrence automaton $A \in \Sigma_{m+2}$, and
- $L = L(A)$ for some alternating Rabin automaton A with index at most m.

Furthermore, the following statements are also mutually equivalent:

- $L = L(\phi)$ for some restricted μKn-formula $\phi \in \Sigma_{m+2}$,
- $L = L(A)$ for some ordinary or restricted alternating first recurrence automaton $A \in \Sigma_{m+2}$, and
- $L = L(A)$ for some ordinary or restricted alternating Rabin automaton A with index at most m.

Proof: Immediate from Proposition 3.2.6 and Lemmas 3.2.9, 3.2.21, 3.2.23, 3.2.14, 3.2.3 and 3.1.16. □

3.2.4 Decidability of ordinary FR-automata

In this subsection we show that it is easy to decide the emptiness of an ordinary or restricted alternating first recurrence automaton A, i.e. to answer the question whether $L(A) = \emptyset$ or not.

Proposition 3.2.25 Let A be an ordinary or a restricted alternating first recurrence automaton. We can determine whether $L(A) = \emptyset$ or not, and this can be done in a time which is linear in the size of A.

Proof: Let $A = (Q, q_{init}, \Delta, l, (G, R))$ be a restricted alternating FR-automaton on n-branching trees. Let us call a transition $(q, q') \in \Delta$ *bad* iff q' is an ancestor of q and $q' \in R$, i.e. iff (q, q') is a transition back to a loop state in R. Define inductively the concept of a *bad* state $q \in Q$ by:

- if $l(q) = \vee$, then q is bad iff for every $q' \in q$ such that $(q, q') \in \Delta$, either the transition (q, q') is bad, or q' is a child of q and q' is bad,
- if $l(q) = \wedge$, then q is bad iff there is some $q' \in q$ such that $(q, q') \in \Delta$ and either 1) the transition (q, q') is bad, or 2) q' is a child of q and q' is bad, or 3) there are some $q', q'' \in Q$ and $z \in \mathcal{Z}$ such that $(q, q') \in \Delta$, $(q, q'') \in \Delta$, $l(q') = z$ and $l(q'') = \neg z$
- if $l(q) = \textcircled{\scriptsize i}$, then q is bad iff for the $q' \in q$ such that $(q, q') \in \Delta$, either the transition (q, q') is bad, or q' is a child of q and q' is bad, and
- lf q is atomic, then q is bad iff $l(q) = \perp$.

It is clear that we can compute the set of bad states in a time linear in $|Q| + |\Delta|$ by an ordinary depth-first search. We claim now that $L(A) = \emptyset$ iff q_{init} is bad.

Assume first that q_{init} is bad, and take any run π of A on any model M. Let us construct inductively an infinite path p of π such that for every $i \in \mathbb{N}$, every ancestor of $\pi^{\text{fr}}(p(i))$ (including the state itself) is bad, and if $\pi^{\text{fr}}(p(i+1))$ is not a child of $\pi^{\text{fr}}(p(i))$, then the transition $(\pi^{\text{fr}}(p(i)), \pi^{\text{fr}}(p(i+1)))$ is bad, implying that $\pi^{\text{fr}}(p(i+1)) \in R$. Such a path p cannot fulfil the FR-acceptance condition (G, R), since in order to be accepted p would need to have infinitely many transitions back to an ancestor in G, i.e. infinitely many $i \in \mathbb{N}$ such that $\pi^{\text{fr}}(p(i+1))$ is an ancestor of $\pi^{\text{fr}}(p(i))$ and $\pi^{\text{fr}}(p(i+1)) \in G$.

We can construct such a path p by starting from $p(0) = \epsilon$ and defining $p(i+1)$ for every $i \in \mathbb{N}$ as some child of $p(i)$ in π such that either $\pi^{\text{fr}}(p(i+1))$ is an ancestor of $\pi^{\text{fr}}(p(i))$ and $(\pi^{\text{fr}}(p(i)), \pi^{\text{fr}}(p(i+1)))$ is a bad transition, or $\pi^{\text{fr}}(p(i+1))$ is a child of $\pi^{\text{fr}}(p(i))$ and $\pi^{\text{fr}}(p(i+1))$ is bad. This p is well-defined, since q_{init} is bad, and since by the definition of a run there can be neither any $t \in \text{dom}(\pi)$ such that $l(\pi^{\text{fr}}(t)) = \perp$, nor any $t \in \text{dom}(\pi)$, $z \in \mathcal{Z}$ and $q', q'' \in Q$ such that $l(\pi^{\text{fr}}(t)) = \wedge$, $l(q') = z$, $l(q'') = \neg z$ and q' and q'' are both children of $\pi^{\text{fr}}(t)$. Since we can construct such a non-accepting path for any run π of A, we have $L(A) = \emptyset$.

Assume then that q_{init} is not bad. Let us define inductively an accepting run π of A such that for all $t \in \text{dom}(\pi)$, no ancestor of $\pi^{\text{fr}}(t)$ (including the state itself) is bad, and for all children t' of t, if $\pi^{\text{fr}}(t')$ is an ancestor of $\pi^{\text{fr}}(t)$, then $\pi^{\text{fr}}(t') \in G$.

The run π is defined by

- the root of π is labelled with $\pi(\epsilon) = (\epsilon, q_{init})$,
- if $\pi(t) = (s, q)$ and $l(q) = \wedge$, then t has a child labelled with (s, q') for every $q' \in Q$ such that $(q, q') \in \Delta$,
- if $\pi(t) = (s, q)$ and $l(q) = \vee$, then t has one child labelled with (s, q'), where q' is some element of Q such that $(q, q') \in \Delta$ and neither the state q' nor the transition (q, q') is bad,
- if $\pi(t) = (s, q)$ and $l(q) = \textcircled{i}$, then t has one child labelled with $(s \cdot i, q')$, where q' is the element of Q such that $(q, q') \in \Delta$, and
- if $\pi(t) = (s, q)$ and q is atomic, then t is a leaf.

Take any infinite path p of π. If p would fail the FR-acceptance condition (G, R), there would have to be infinitely many $i \in \mathbb{N}$ such that $\pi^{\mathrm{fr}}(p(i+1))$ is an ancestor of $\pi^{\mathrm{fr}}(p(i))$ and $\pi^{\mathrm{fr}}(p(i+1)) \in R$. But this is impossible by the properties of π, which means that π is an accepting run.

We still need to show that π is actually a run of A on some model M. Define a model M as follows. For all states $s \in \mathrm{st}(M)$ and all $z \in \mathcal{Z}$, $z \in M(s)$ iff there is some $t \in \mathrm{dom}(\pi)$ such that $\pi^{\mathrm{st}}(t) = s$ and $l(\pi^{\mathrm{fr}}(t)) = z$. To see that π is consistent with the model M, notice that due to the strong aconjunctivity of A, whenever there are $t', t'' \in \mathrm{dom}(\pi)$ such that $\pi^{\mathrm{st}}(t') = \pi^{\mathrm{st}}(t'')$ and both $\pi^{\mathrm{fr}}(t')$ and $\pi^{\mathrm{fr}}(t'')$ are atomic, then t' and t'' are both children of some t such that $l(\pi^{\mathrm{fr}}(t)) = \wedge$, which means in turn by the definition of badness for $\wedge$-states that $l(\pi^{\mathrm{fr}}(t'))$ and $l(\pi^{\mathrm{fr}}(t''))$ cannot be mutually contradictory or $\bot$. Consequently there is a model $M \in L(A)$ and $L(A) \neq \emptyset$. □

As restricted FR-automata are really restricted μKn-formulae, and vice versa, we can formulate the decision method also in terms of formulae. In this setting it amounts to the following. When determining the satisfiability of a restricted μKn-formula ϕ in pnf, replace first every subformula of ϕ of the form $\mu z.\psi$ by $\mu^1 z.\psi$, i.e. by $\psi[\bot/z]$, and every subformula of the form $\nu z.\psi$ by $\nu^1 z.\psi$, i.e. by $\psi[\top/z]$. Reduce then subformulae of the resulting formula according to the rules:

$$\begin{array}{rcl}
\psi \wedge \bot & \text{reduces to} & \bot \\
\bot \vee \bot & \text{reduces to} & \bot \\
\textcircled{i}\bot & \text{reduces to} & \bot \\
z \wedge \neg z & \text{reduces to} & \bot
\end{array}$$

If the whole formula reduces to $\bot$ then it is not satisfiable and otherwise it is satisfiable.

The decision procedure for restricted alternating first recurrence automata also allows us to solve the emptiness problems for ordinary FR-automata and ordinary or restricted alternating Büchi and Rabin automata. It suffices to observe that all the translations which we have presented between these types of automata are clearly computable. Notice though, that unwinding automata to a tree-like form causes an exponential penalty, which means that implementing the translations directly does not lead to an optimal decision procedure. The decision problem for ordinary Büchi tree automata is known to be logspace complete for PTIME [77, 97] and the same problem for ordinary Rabin tree automata is NP-complete [31].

However, if we are interested in efficient algorithms, with certain modifications to the current framework it appears to be possible to retain the essence of it but avoid the exponential blowup caused by unwinding graphs to trees. Let us just outline this briefly. The basic idea is to relax the structural requirements for a first recurrence automaton so that it is no longer 'a tree with back transitions' but 'a rooted dag (directed acyclic graph) with back transitions' fulfilling the following hierarchy condition: If we take any state q of the automaton and any two non-looping paths $q_0, \dots, g_k$ and $q'_0, \dots, q'_{k'}$ from the initial state q_{init} to q and define

$$\begin{aligned} g &= \max\{0 \le i \le k \mid q_i \in G \text{ and } q_i \text{ is active in } q\} \\ r &= \max\{0 \le i \le k \mid q_i \in R \text{ and } q_i \text{ is active in } q\} \\ gr &= \max\{0 \le i \le k \mid q_i \in G \cup R \text{ and } q_i \text{ is active in } q\} \end{aligned}$$

and g', r' and gr' analogously, then

- if $q \in G$ then r is well-defined iff r' is, and if both are then $q_r = q'_{r'}$,
- if $q \in R$ then g is well-defined iff g' is, and if both are then $q_g = q'_{g'}$, and
- if $q \notin G \cup R$ then gr is well-defined iff gr' is, and if both are then $q_{gr} = q'_{gr'}$.

Although in this more general framework it is no longer the case that for every infinite path of a run there would be a unique state such that it occurs infinitely often along the path but none of its proper ancestors does so, the hierarchy condition guarantees that either all such states belong to G or they all belong to R. This means that the first recurrence acceptance condition (G, R) still makes sense. Furthermore, the basis of the decision procedure above, a depth-first traversal where each state is visited at most once, can also be augmented to work in this more general setting. It then appears to be possible to translate Rabin automata to these more general first recurrence automata in the same way and within the same $|Q|^{2m+1}$ depth bound as here, where m is the Rabin index of the automaton, and to match the $(|Q| \cdot m)^{\mathcal{O}(m)}$ time complexity of the decision algorithm in [31, 33].

However, since efficiency is not our main concern in the current work and this line of research is still partly under development, we shall not examine the more general framework further here. We would still like to point out, though, that analogous to the relation between FR-automata in the usual sense and modal mu-calculus, there is a natural correspondence between these more general automata and the logical language extending mu-calculus with simultaneous vectorial fixpoints $\nu(x_0, \ldots, x_k).(\phi_0, \ldots, \phi_k)$ and $\mu(x_0, \ldots, x_k).(\phi_0, \ldots, \phi_k)$.

3.3 Quantification and ordinary automata

As we have mentioned before, a central motivation for studying ordinary non-alternating automata on infinite objects in the first place was the decidability of the monadic second-order calculi *S1S* and *SnS*. The function of the automata in this task is to work as a normal form to which every formula can be inductively transformed and which can relatively easily be checked for satisfiability. Crucial to the translation of formulae to automata is that the automata are closed under the logical connectives of the language. As shall be seen in the following sections, closure under disjunction, conjunction and modal operators is not problematic. Negation or complementation, on the other hand, is highly complicated.

However, what is probably the most valuable property of ordinary automata as far as inductive translations from second-order calculi are concerned, is that existential second-order quantification over sets or properties is trivial for ordinary automata. In fact, if $L(A) = L(\phi)$ for an ordinary automaton A, all that needs to be done to obtain an automaton A' such that $L(A') = L(\exists z.\phi)$ is to erase all references to z from A. At the heart of this property is a certain 'localisation' of all requirements that an automaton sets to a particular state of a model; they are all present at a particular point of a run of the automaton. In both automata and formulae the feature guaranteeing such localisation of requirements is strong aconjunctivity, which prevents different conjuncts from setting requirements to the same future state. Let us discuss the relation between strong aconjunctivity, ordinary automata and second-order quantification in some more detail.

Lemma 3.3.1 Let ϕ be a strongly aconjunctive μKn-formula in pnf and z a variable. Then there exists a μKn-formula ϕ' such that

- for every model $M \in \mathcal{M}_n$, $M \models \phi'$ iff $M[W/z] \models \phi$ for some $W \subseteq \mathrm{st}(M)$,
- if ϕ is restricted then so is ϕ', and
- for any alternation class Π_m, Σ_m or Δ_m, ϕ' belongs to the class iff ϕ does so.

Proof: Without loss of generality we can assume that ϕ has no subformulae of the type $z \wedge \neg z$, as these can be replaced by $\perp$. We obtain the required ϕ' from ϕ by replacing every occurrence of both z and $\neg z$ in it by $\top$.

Take any model M and assume that there is some $W \subseteq \mathrm{st}(M)$ such that $M[W/z] \models \phi$. By Prop. 2.2.33 there is a proper simple tableau T for ϕ agreeing with $M[W/z]$. By replacing every occurrence of both z and $\neg z$ in T with $\top$, we can read from T a proper tableau T' for ϕ' agreeing with M, which implies that $M \models \phi'$.

Assume then that $M \models \phi'$. By Prop. 2.2.33 there is a proper simple tableau T' for ϕ' agreeing with M. By changing some $\top$'s in T' to z or $\neg z$, we can read from T' a proper simple tableau T for ϕ. Defining W as the set containing those $s \in \mathrm{st}(M)$ for which there is some $t \in \mathrm{dom}(T)$ such that $T(t) = (s, z, d)$ for some d, the tableau T agrees with $M[W/z]$, implying that $M[W/z] \models \phi$. Notice that due to the assumption that ϕ is strongly aconjunctive, we know that if $t' \neq t''$ and $T(t') = (s', x', d')$ and $T(t'') = (s'', x'', d'')$ such that $s' = s''$ and $x', x'' \in \{z, \neg z\}$, then both t' and t'' are children of some node t, derived by the $\wedge$-rule, and by the assumption in the beginning it cannot be the case that x and x' would be mutually contradictory. This guarantees that T indeed agrees with $M[W/z]$. $\square$

Proposition 3.3.2 Let A be a restricted alternating first recurrence automaton on n-branching trees, and z a variable. Then there exists a restricted alternating first recurrence automaton A' such that for any model $M \in \mathcal{M}_n$, $M \in L(A')$ iff there is some $W \subseteq \mathrm{st}(M)$ such that $M[W/z] \in L(A)$. The same claim holds also with respect to ordinary FR-automata, and ordinary and restricted alternating Büchi and Rabin automata.

Proof: Let A be a restricted alternating FR-automaton. By 3.2.6 we can view any such automaton as a restricted μKn-formula, and vice versa. Being restricted, A is strongly aconjunctive. We obtain the required A' by the construction of Lemma 3.3.1, effectively by erasing all references to z from A. As this does not affect the acceptance condition of the automaton in any way, the same construction works also for restricted alternating Büchi and Rabin automata. The claims concerning ordinary automata are obvious due to the correspondence with restricted alternating automata. $\square$

So, closure under existential second-order quantification is easy for ordinary automata, and conjunction, disjunction and modal operators are not too hard either. What makes an inductive translation from calculi like $\exists Kn$ or SnS to ordinary automata difficult is negation. The situation is directly opposite for

name	application	name	application
$\vee$	$\dfrac{\Gamma \cup \{\psi \vee \psi'\}}{\Gamma \cup \{\psi\} \qquad \Gamma \cup \{\psi'\}}$	$\wedge$	$\dfrac{\Gamma \cup \{\psi \wedge \psi'\}}{\Gamma \cup \{\psi, \psi'\}}$
$\boldsymbol{\sigma}$	$\dfrac{\Gamma \cup \{\sigma x.\psi\}}{\Gamma \cup \{\psi[\sigma x.\psi/x]\}}$		
$\bigcirc$	$\dfrac{\{z_1, \ldots, z_k, \textcircled{i_1}\psi_1, \ldots, \textcircled{i_m}\psi_m\}}{\Gamma_0 \quad \Gamma_1 \quad \ldots \quad \Gamma_{n-1}}$ **1**		
Note:	**1:** each z_j is atomic, and $\Gamma_i = \{\psi_j \mid i_j = i\}$. In each rule, Γ is disjoint from the other set		

Table 3.2: Derivation rules for Lemma 3.3.3

alternating automata in general and for alternating first recurrence automata in particular: complementation is easy, but due to lack of aconjunctivity it is not at all clear that closure under quantification holds.

If we were able to translate alternating automata to equivalent ordinary ones, we could enjoy the best of both scenarios and obtain an inductive translation from a calculus with second-order quantifiers to ordinary automata. In this case we could complement an automaton by viewing it as an alternating automaton, complementing this alternating automaton, and translating the result back to an ordinary automaton. Alternatively, we could formulate the translation in terms of alternating automata, and deal with quantification by translating an alternating automaton to an ordinary one, erasing the occurrences of the quantified variable in the ordinary automaton, and interpreting the result as an alternating automaton.

Because of this, it is worth looking at the issue of relating ordinary and alternating automata to each other, beyond the rather trivial observation that ordinary automata are a subclass of alternating automata. Let us see first for a restricted subcase that some alternating automata can be transformed to ordinary automata. For technical convenience, the following lemma has been formulated in terms of μKn-formulae, but as has already been pointed out in countless occasions, these are just alternating first recurrence automata.

Lemma 3.3.3 For every μKn-formula $\phi \in \Sigma_1$, there exists a restricted μKn-formula $\psi \in \Sigma_1$ such that $\models \phi \Leftrightarrow \psi$.

Proof: By Prop. 2.2.36 we can assume without loss of generality that ϕ is guarded. Let us define inductively a finite tree T labelled with sets of formulae:

- The root of T is labelled with $T(\epsilon) = \{\phi\}$.

- If a node t is not a leaf (see below), the children of t are derived by one of the rules in Table 3.2. Depending on the rule applied at a node t, t is called a $\vee, \wedge, \sigma$ or $\bigcirc$-node, respectively.

A node t is a leaf iff either

1 it is labelled with the empty set, or

2 it is derived from its parent by the $\bigcirc$-rule, and there is some proper ancestor t' of t such that $T(t') = T(t)$ and t' is also derived from its parent by the $\bigcirc$-rule. In this case we call t' the loop node corresponding to t.

Fix for every loop node t of T a distinct fresh variable $z(t)$. Define the formula ψ by $\psi = \chi(\epsilon)$, where $\chi(t)$ is defined inductively for every node t of T by:

$$\chi(t) \quad = \quad \begin{cases} \mu z(t).\theta(t), & \text{if } t \text{ is a loop node} \\ \theta(t), & \text{otherwise} \end{cases}$$

where

$$\theta(t) \quad = \quad \begin{cases} \bigvee\{\theta(t') \mid t' \text{ is a child of t}\} & \text{if } t \text{ is a } \vee\text{-node} \\ \theta(t') & \text{if } t \text{ is a } \wedge\text{-node and } t' \text{ the child of } t \\ \theta(t') & \text{if } t \text{ is a } \sigma\text{-node and } t' \text{ the child of } t \\ (\bigwedge\{z \in T(t) \mid z \text{ is atomic }\}) \wedge \bigwedge_{i\in[n]} \textcircled{\scriptsize i}\,\chi(t\cdot i) & \text{if } t \text{ is a } \bigcirc\text{-node} \\ \top & \text{if } t \text{ is a leaf of type 1} \\ z(t') & \text{if } t \text{ is a leaf of type 2 and } t' \\ & \text{the loop node corresponding to } t \end{cases}$$

It should be clear that ψ is restricted and that $\psi \in \Sigma_1$. To show that $\models \phi \Leftrightarrow \psi$, take any model $M \in \mathcal{M}_n$. By Prop. 2.2.39 and 2.3.10, $M \models \phi$ iff there is a proper bundled tableau T for ϕ agreeing with M, and $M \models \psi$ iff there is a proper bundled tableau T' for ψ agreeing with M. Since $\phi \in \Sigma_1$, from Lemma 2.2.54 and Prop. 2.3.17 we can see that T is proper iff for every infinite path p of T, $\Gamma_{p(i)} = \emptyset$ for some $i \in \mathbb{N}$, and that the same holds for T'. As it is easy to read a tableau T' for ψ agreeing with M, from a tableau T for ϕ agreeing with M, and vice versa, this means that we can read a proper tableau T' for ψ from any proper tableau T for ϕ, and vice versa. Consequently, $M \models \phi$ iff $M \models \psi$. □

A formula $\phi \in \Sigma_1$ with only minimal fixpoints can be viewed as an alternating finite automaton on finite strings. The construction of the lemma above corresponds then to mapping such automata to normal non-deterministic finite automata on finite strings.

We can take advantage of the transformation in the lemma above to show our first full correspondence result between a formalism with second-order quantifiers and one with fixpoints. This result is for the weak language $\overset{\text{w}}{\exists}Kn$, essentially another formulation of *WSnS*, and the fixpoint alternation-free fragment Δ_2 of μKn.

Lemma 3.3.4 Let ϕ be a non-alternating μKn-formula, i.e. $\phi \in \Delta_2$, and z a variable. Then there exists a non-alternating μKn-formula ϕ' such that for any model $M \in \mathcal{M}_n$, $M \models \phi'$ iff there exists a finite set $W \subseteq \mathrm{st}(M)$ such that $M[W/z] \models \phi$.

Proof: By Prop. 2.2.36, we can assume without loss of generality that ϕ is guarded. Let us construct a finite tree T labelled with sets of formulae precisely as in the proof of Lemma 3.3.3 (notice here that the σ-rule also applies to maximal fixpoints). Define also for every node t of T a formula $\chi(t)$ as in the proof of Lemma 3.3.3, except that if t is a loop node then $\chi(t) = \mu z(t).\theta(t) \vee x(t)$ (instead of $\chi(t) = \mu z(t).\theta(t)$), where $x(t)$ is a fresh variable. Define then $\chi = \chi(\epsilon)$ and $\psi = \chi[\alpha(t_1)/x(t_1), \ldots, \alpha(t_m)/x(t_m)]$, where $t_1, \ldots, t_m$ are the loop nodes of T, and every $\alpha(t_i) = (\bigwedge T(t_i))[\bot/z]$, i.e. $\alpha(t_i)$ is the conjunction of all formulae labelling t_i, with the exception that z has been replaced by $\bot$. It should be clear that without the $[\bot/z]$-part, the formula ψ would be equivalent to ϕ.

Take any model M. We claim now that there is a finite W such that $M[W/z] \models \phi$ iff there is some W' such that $M[W'/z] \models \psi$. Take some finite W such that $M[W/z] \models \phi$, which implies by Prop. 2.2.39 and 2.3.10 that there is a proper bundled tableau T for ϕ agreeing with $M[W/z]$. Due to the finiteness of W we know that

- for every $s \in \mathrm{st}(M)$, if there is no descendant s' of s such that $s' \in W$, then for any μKn-formula α, $M[W/z], s \models \alpha$ iff $M[\emptyset/z], s \models \alpha$ iff $M, s \models \alpha[\bot/z]$, and
- on any infinite path p of M there is some point $i \in \mathbb{N}$ such that $s' \notin W$ for all descendants s' of $p(i)$

These allow us to read a tableau T' for ψ agreeing with $M[W/z]$ from T, implying that $M[W/z] \models \psi$.

Assume then that there is some W' (which is not necessarily finite) such that $M[W'/z] \models \psi$, which means that there is a proper bundled tableau T' for ψ agreeing with $M[W'/z]$. Define now a finite set W by:

$$W = \{s \in W' \mid \exists t \in \mathrm{dom}(T) : T(t) = (s, \Gamma, d) \text{ and } z \in \Gamma \text{ for some } \Gamma, d\}$$

This set is indeed finite, since z does not appear in any $\alpha(t_i)$ in ψ and the μ-fixpoints of $\chi(\epsilon)$ can be unfolded only finitely many times. We can read a tableau T for ϕ agreeing with $M[W/z]$ from T', implying that $M[W/z] \models \phi$.

Notice then that the formula χ is restricted, hence strongly aconjunctive, and that $\chi \in \Sigma_1$. Consequently, by Lemma 3.3.1 there exists a μKn-formula $\chi' \in \Sigma_1$ such that for every model $M \in \mathcal{M}_n$, $M \models \chi'$ iff there is some $W \subseteq \mathrm{st}(M)$ such that $M[W/z] \models \chi$. Define $\phi' = \chi'[\alpha(t_1)/x(t_1), \ldots, \alpha(t_m)/x(t_m)]$. Since z does not occur in any of $\alpha(t_i)$, this implies that for every $M \in \mathcal{M}_n$, $M \models \phi'$ iff there is some $W \subseteq \mathrm{st}(M)$ such that $M[W/z] \models \psi$. But by the intermediate claim above we know that the latter holds iff there is some finite $W \subseteq \mathrm{st}(M)$ such that $M[W/z] \models \phi$. This concludes the proof. $\Box$

The construction of Lemma 3.3.4 above is a reformulation of [68, Lemma 1].

Theorem 3.3.5 For every $\overset{\mathrm{w}}{\exists}Kn$-formula ψ there exists a μKn-formula ϕ such that $\phi \Leftrightarrow \psi$, and $\phi \in \Delta_2$.

Proof: The claim is shown inductively on the structure of ψ. For atomic formulae the claim is obvious, since they belong to both languages. Assume then that for ψ and ψ', there are ϕ and ϕ', respectively, satisfying the claim. For $\textcircled{i}\psi$, $\psi \wedge \psi'$ and $\neg\psi$, the formulae $\textcircled{i}\phi$, $\phi \wedge \phi'$ and $\neg\phi$ satisfy the claim, respectively, and for $\mathbf{G}\psi$, the formula $\nu z.\phi \wedge \bigcirc z$, where z is fresh, does so as well. Since $\phi \in \Delta_2$, by Lemma 3.3.4 there is some $\phi' \in \Delta_2$ such that $M \models \phi'$ iff there is a finite $W \subseteq \mathrm{st}(M)$ such that $M[W/z] \models \phi$, i.e. iff $M \models \overset{\mathrm{w}}{\exists}z.\psi$, justifying the induction step for $\overset{\mathrm{w}}{\exists}z.\psi$, as well. $\Box$

Corollary 3.3.6 For any language $L \subseteq \mathcal{M}_n$, the following statements are mutually equivalent:

- $L = L(\phi)$ for some $\overset{\mathrm{w}}{\exists}Kn$-formula ϕ,
- $L = L(\phi)$ for some *WSnS*-formula ϕ, and
- $L = L(\phi)$ for some μKn-formula ϕ such that $\phi \in \Delta_2$.

Proof: Immediate from Prop. 2.2.52, Theorem 2.2.57, Prop. 2.3.16, Prop. 2.3.17 and Theorem 3.3.5. $\Box$

This result has been shown previously in [4]. The alternating first recurrence automata in the class Δ_2, i.e. the non-alternating μKn-formulae, have a close connection to the so-called *weak alternating* automata of Muller, Saoudi and Schupp [68, 69]. In fact the only real difference is that FR-automata are required to be tree-like, whereas weak alternating automata are not. In this framework Muller, Saoudi and Schupp prove a correspondence theorem equivalent to the one above [68]. Weak alternating automata have attracted interest, because many programming logics are easily reducible to such automata, and they therefore provide a universal framework in which to study the decidability of these logics. For more discussion of this aspect, see [69].

3.4 Fixpoints for ordinary automata

As explained in last section, if we are able to devise a translation from alternating automata to ordinary ones, this will also yield an inductive translation from calculi like $\exists Kn$ and SnS to ordinary automata and show their decidability. The current section will describe such translations from alternating formalisms to non-alternating ones.

3.4.1 Fixpoints and Büchi acceptance

In this subsection we describe a construction translating μKn-formulae in the fixpoint alternation class Π_2 to ordinary Büchi automata so that for a given formula ϕ and the corresponding automaton A_ϕ, $L(\phi) = L(A_\phi)$.

Let us assume that a formula ϕ is in positive normal form. The method of constructing A_ϕ is inductive. For each basic μKn-formula we define a corresponding automaton directly, and for conjunction, disjunction and the modal and fixpoint operators we describe a method for obtaining the required automaton from the automata corresponding to the formulae the operator is applied to. Here the fixpoint operators are naturally the essential problem. The constructions related to them are based on the powerset construction of Dam [22] and the methods of Niwiński [70].

This approach may look contradictory, as seemingly we could then map the full modal mu-calculus to ordinary Büchi automata, and although we have not mentioned it yet, this is known to be impossible. However, although the constructions for all other operators work on any ordinary Büchi automaton, the minimal fixpoint construction relies on a structural property that only the automata corresponding to Π_2-formulae have.

Definition 3.4.1 Let $A = (Q, q_{init}, \Delta, F)$ be an ordinary Büchi automaton on n-branching trees, and $z \in \mathcal{Z}$ an atomic proposition. We say that A *refers to* z iff there are some $q, q' \in Q$ such that $q \xrightarrow{Z} q'$ and either $z \in Z$ or $\neg z \in Z$. We say that A *refers to* z *initially* iff there is some $q \in Q$ such that $q_{init} \xrightarrow{Z} q$ and either $z \in Z$ or $\neg z \in Z$. We say that A *refers to* z *only before accepting states* iff there are no $q, q', q'' \in Q$ and $Z \subseteq \mathcal{Z} \cup \overline{\mathcal{Z}}$ such that $q_{init} \longrightarrow^* q \longrightarrow^* q' \xrightarrow{Z} q''$ in A, the state q is accepting, i.e. $q \in F$, and either $z \in Z$ or $\neg z \in Z$. □

It turns out that the automata for Π_2-formulae refer to μ-variables only before accepting states.

Given a formula $\phi \in \Pi_2$, we show by induction that for all subformulae ψ of ϕ, we can construct the required automaton A_ψ. In each step of the induction proof we show that for the current ψ we can construct the A_ψ so that both $L(\psi) = L(A_\psi)$ and A_ψ meets the relevant structural property, provided that we know how to do this for the immediate subformulae of ψ. The automata for atomic formulae are trivial, the constructions for disjunction and conjunction standard [93], and that for the modal operator is easy, as well.

Definition 3.4.2 Define $A_\top$, $A_\bot$ and A_z, where $z \in \mathcal{Z} \cup \overline{\mathcal{Z}}$, as the following ordinary Büchi automata on n-branching trees:

$$\begin{array}{rcl} A_\top & = & (\{q_0, q_1\}, q_0, \{(q_0, \emptyset, \overline{q}), (q_1, \emptyset, \overline{q})\}, \{q_1\}), \text{ where } \overline{q}_i = q_1 \text{ for all } i \in [n] \\ A_\bot & = & (\{q_0\}, q_0, \emptyset, \emptyset) \\ A_z & = & (\{q_0, q_1\}, q_0, \{(q_0, \{z\}, \overline{q}), (q_1, \emptyset, \overline{q})\}, \{q_1\}), \text{ where } \overline{q}_i = q_1 \text{ for all } i \in [n] \end{array}$$

□

Definition 3.4.3 Let $A_0 = (Q_0, q^0_{init}, \Delta_0, F_0)$ and $A_1 = (Q_1, q^1_{init}, \Delta_1, F_1)$ be ordinary Büchi automata on n-branching trees. Define $A_0 \vee A_1$ as the ordinary Büchi automaton $A_0 \vee A_1 = (Q, q_{init}, \Delta, F)$, where

- $Q = Q_0 \cup Q_1 \cup \{q_x\}$, where q_x is a fresh state and we assume that Q_0 and Q_1 are disjoint,
- $q_{init} = q_x$,
- $\Delta = \Delta_1 \cup \Delta_2 \cup \{(q_x, Z, \bar{q}) \mid (q^0_{init}, Z, \bar{q}) \in \Delta_0\} \cup \{(q_x, Z, \bar{q}) \mid (q^1_{init}, Z, \bar{q}) \in \Delta_1\}$,
- $F = F_1 \cup F_2$,

Define $A_0 \wedge A_1$ as the ordinary Büchi automaton $A_0 \wedge A_1 = (Q, q_{init}, \Delta, F)$, where

- $Q = Q_0 \times Q_1 \times \{0, 1, 2\}$,
- $q_{init} = (q^0_{init}, q^1_{init}, 0)$,
- if $(q_0, Z_0, \bar{q}^0) \in \Delta_0$, $(q_1, Z_1, \bar{q}^1) \in \Delta_1$ and $(q_0, q_1, c) \in Q$, then

$$((q_0, q_1, c), Z_0 \cup Z_1, ((\bar{q}^0_0, \bar{q}^1_0, c_0), \ldots, (\bar{q}^0_{n-1}, \bar{q}^1_{n-1}, c_{n-1}))) \in \Delta$$

  where for every $i \in [n]$:

$$\begin{array}{ll} \text{if } c = 0 \text{ and } \bar{q}^0_i \notin F_0 & \text{then } c_i = 0 \\ \text{if } c = 0 \text{ and } \bar{q}^0_i \in F_0 & \text{then } c_i = 1 \\ \text{if } c = 1 \text{ and } \bar{q}^1_i \notin F_1 & \text{then } c_i = 1 \\ \text{if } c = 1 \text{ and } \bar{q}^1_i \in F_1 & \text{then } c_i = 2 \\ \text{if } c = 2 & \text{then } c_i = 0 \end{array}$$

- $F = Q_0 \times Q_1 \times \{2\}$.

Define $\textcircled{i} A_0$ as the ordinary Büchi automaton $\textcircled{i} A_0 = (Q, q_{init}, \Delta, F)$, for any $i \in [n]$, where

- $Q = Q_0 \cup \{q_x, q_{acc}\}$ where q_x and q_{acc} are fresh,

- $q_{init} = q_x$,
- $\Delta = \Delta_0 \cup \{(q_x, \emptyset, \bar{q}) \mid \bar{q}_i = q^0_{init}$ and for all $j \neq i : \bar{q}_j = q_{acc}\} \cup$ $\{(q_{acc}, \emptyset, (q_{acc}, \ldots, q_{acc}))\}$,
- $F = F_0 \cup \{q_{acc}\}$. □

The fixpoint operations are based on a powerset construction related to those in [22] and [77]. Supposing that automaton A corresponds to ϕ, the idea can be described as follows. In order to decide whether $M \in \mathcal{M}_n$ is a model of $\nu z.\phi$, we start by running A down M. Each time A requires z to be true, we start a new copy of A running down that particular subtree, and repeat the process with the new copies. When two copies of A are in the same state running down the same subtree, they are joined as one. If all the runs of copies of A are accepting, M is a model of $\nu z.\phi$.

Definition 3.4.4 Let $A = (Q, q_{init}, \Delta, \Omega)$ be an ordinary Büchi or Rabin automaton on n-branching trees, and let z be a variable. The *intermediate automaton* based on A, denoted by $\text{fix}_z A$, is a triple $\text{fix}_z A = (Q', q'_{init}, \Delta')$, where

- $Q' = 2^Q$ is the set of states,
- $q'_{init} = \{q_{init}\}$ the initial state, and
- $\Delta' \subseteq Q' \times 2^{\mathcal{Z} \cup \overline{\mathcal{Z}}} \times Q'^n \times (Q \rightharpoonup \Delta)$ the transition relation,

and where $(P, Z, \overline{P}, \delta) \in \Delta'$ iff

1 The domain of δ is $P \subseteq Q$, i.e. δ is a function $\delta : P \to \Delta$. Define functions δ^{fr}, δ^{lab} and δ^{to} by: for all $q \in P$, $\delta^{\text{fr}}(q) = q'$, $\delta^{\text{lab}}(q) = Z'$ and $\delta^{\text{to}}(q) = \bar{q}'$, where $\delta(q) = (q', Z', \bar{q}')$.
2 $\delta^{\text{fr}}(q) = q$ for all $q \in P$.
3 $q_{init} \in P$ iff $P = \{q_{init}\}$ or there is a $q \in P \setminus \{q_{init}\}$ such that $z \in \delta^{\text{lab}}(q)$.
4 $Z = \bigcup_{q \in P} \delta^{\text{lab}}(q) \setminus \{z\}$.
5 $\overline{P}_i \setminus \{q_{init}\} = \{\delta^{\text{to}}(q)_i \mid q \in P\}$ for all $i \in [n]$,

A *run* Π of $\text{fix}_z A$, and the functions Π^{fr}, Π^{lab} and Π^{to} are defined as in 3.1.5. □

The states of $\text{fix}_z A$ are sets of states of A, intuitively sets of copies of A, and a transition of $\text{fix}_z A$ corresponds to a set of simultaneous transitions of A. In a transition of the intermediate automaton $\text{fix}_z A$, the final component δ attaches to each state q in the originating state P of $\text{fix}_z A$ a transition of A. The intuition here is that δ specifies for each $q \in P$ which particular transition from state q of A is taken as part of the set of simultaneous transitions. This corresponds to conditions 1 and 2 above, and the information is used later to characterise acceptance conditions for the intermediate automaton. Condition 3 above states that a new copy of A is started initially and when a running copy refers to z, condition

4 states that a transition of $\text{fix}_z A$ is possible iff all the underlying transitions of A are, and condition 5 says that the state of $\text{fix}_z A$ after a transition is specified by the targets of the underlying transitions of A, with possibly a new copy of A started.

To decide whether a given run Π of $\text{fix}_z A$ corresponds to a set of accepting runs of A, we extract structures corresponding to these individual runs of A from Π.

Definition 3.4.5 Let A and $\text{fix}_z A$ be as in 3.4.4, Π a run of $\text{fix}_z A$, t a node of Π, $\Pi(t) = (P, Z, \overline{P}, \delta)$, and $q \in P$. The *trail of* Π *starting from state* q *at node* t is the tree $\pi(t,q) : [n]^* \to \Delta$, such that

- $\pi(t,q)(\epsilon) = \delta(q)$, and
- for every $t' \in [n]^*$ and $i \in [n]$, we have $\pi(t,q)(t' \cdot i) = \delta'(\pi(t,q)^{\text{to}}(t')_i)$, where $\Pi(t \cdot t' \cdot i) = (P', Z', \overline{P}', \delta')$.

The functions $\pi(t,q)^{\text{fr}}$, $\pi(t,q)^{\text{lab}}$ and $\pi(t,q)^{\text{to}}$ are defined as in 3.1.2. □

According to the intuitions discussed above, in order to decide whether a model $M \in \mathcal{M}_n$ is a model of $\nu z.\phi$, we have to check that all the runs of the individual copies of A within Π are Büchi accepting. As these runs correspond to the trails of Π, this leads to the following definition.

Definition 3.4.6 Let $A = (Q, q_{init}, \Delta, F)$ be an ordinary Büchi automaton, $\text{fix}_z A$ the intermediate automaton based on A, and Π a run of $\text{fix}_z A$. We say that Π is *ν-Büchi-accepting* iff for every node t of Π, every state $q \in \Pi^{\text{fr}}(t)$ and every infinite path p of the trail $\pi(t,q)$, $\pi(t,q)^{\text{fr}}(p(i)) \in F$ for infinitely many $i \in \mathbb{N}$. □

Lemma 3.4.7 Let ϕ be a μKn-formula and A an ordinary Büchi automaton such that $L(\phi) = L(A)$. Then for every model $M \in \mathcal{M}_n$, $M \models \nu z.\phi$ iff $\text{fix}_z A$ has a ν-Büchi-accepting run on M.

Proof: By Lemma 2.2.9, $M \models \nu z.\phi$ iff there is a set $W \subseteq \text{st}(M)$ such that $\epsilon \in W$ and $M[W/z] \models \phi$ for every $t \in W$, i.e. iff there is $W \subseteq \text{st}(M)$ such that $\epsilon \in W$ and $M[W/z]^t \in L(A)$ for every $t \in W$, If we have a ν-Büchi-accepting run Π of $\text{fix}_z A$ on M, then $W = \{t \in \text{st}(M) \mid q_{init} \in \Pi^{\text{fr}}(t)\}$ fulfils the required properties.

Suppose then that we have such a set W. For every $t \in W$, A has an accepting run π_t on $M[W/z]^t$, since $M[W/z]^t \in L(A)$. We build a ν-accepting run Π of $\text{fix}_z A$ on M from these π_t. For each $t \in \text{st}(M)$, we define inductively a set of runs π_t included in Π at node t. The only run included for $t = \epsilon$ is π_ϵ. For $t \neq \epsilon$, the runs included in Π at t are the same as for t's parent, with two exceptions. First, if two included runs $\pi_{t'}$ and $\pi_{t''}$ coincide at t, i.e. $\pi^{\mathrm{fr}}_{t'}(t'^{-1}t) = \pi^{\mathrm{fr}}_{t''}(t''^{-1}t)$, we discard the run with greater index in the prefix-ordering of strings, i.e. $\pi_{t'}$ if $t'' \prec t'$ and vice versa. Secondly, if an included run $\pi_{t'}$ refers to z at t, i.e. $z \in \pi^{\mathrm{lab}}_{t'}(t'^{-1}t)$, we add the newly started run π_t to the included runs. Since we discard a run only when is coincides with another run with a smaller index in the well-founded prefix-ordering, every path of every trail of Π eventually coincides everywhere with a path of some Büchi accepting π_t, implying that Π is ν-Büchi-accepting. □

To construct an ordinary Büchi automaton for $\nu z.\phi$ on the basis of $\mathrm{fix}_z A$, we express the property of being ν-Büchi-accepting in a manner closer to the form of a Büchi acceptance condition.

Definition 3.4.8 Let A, $\mathrm{fix}_z A$ and Π be as in 3.4.6. We say that Π is *ν'-Büchi-accepting* iff for every path p of Π there is an infinite strictly increasing sequence $i_1 < i_2 < \ldots$ such that for every $j \in \mathbb{N}$ and for every $q \in \Pi^{\mathrm{fr}}(p(i_j))$, there is a t such that $p(i_j) \prec t \prec p(i_{j+1})$ and $\pi^{\mathrm{fr}}(p(i_j), q)(p(i_j)^{-1}t) \in F$. □

The maximal fixpoint automaton works like $\mathrm{fix}_z A$ equipped with a mechanism for checking that for each path p the strictly increasing sequence required by ν'-Büchi-acceptability exists. The states of the maximal fixpoint automaton are pairs (P, P'), where P is a state of the intermediate automaton, i.e. a set of states of the original automaton, and P' a subset of P. Intuitively, P expresses the different copies of the original automaton currently running and P' the copies we are currently waiting to reach an accepting state. The transitions correspond to those of the intermediate automaton, and the second component of a state is updated by removing from it all the elements in the acceptance set of the original automaton. When the second element P' becomes empty, i.e. we have seen an accepting state for all the copies of the original automaton for which we were looking for such a state, we start again waiting for accepting states for all the currently running copies. If a path of a run leads from a state $(P_1, \emptyset)$ to a state $(P_2, \emptyset)$, we know that all the copies of the original automaton corresponding to P_1 have reached an accepting state at the latest at the state $(P_2, \emptyset)$ along that path. Consequently, if for every path of a run there are infinitely many visits to states of the type $(P, \emptyset)$, we know that there is such an infinite strictly increasing sequence as required by ν'-Büchi-acceptability.

Definition 3.4.9 Let $A = (Q, q_{init}, \Delta, F)$ be an ordinary Büchi automaton on n-branching trees, and let us write $\mathrm{fix}_z A = (Q_{\mathrm{fix}}, q^{\mathrm{fix}}_{init}, \Delta_{\mathrm{fix}})$. The ordinary Büchi automaton $\nu z.A$ is defined as $\nu z.A = (Q_\nu, q^\nu_{init}, \Delta_\nu, F_\nu)$, where

- $Q_\nu = \{(P, P') \mid P' \subseteq P \subseteq Q\}$,
- $q^\nu_{init} = (\{q_{init}\}, \{q_{init}\})$,
- if $(P, P') \in Q_\nu$ and $(P, Z, (P_0, \ldots, P_{n-1}), \delta) \in \Delta_{\mathrm{fix}}$, then

$$((P, P'), Z, ((P_0, P'_0), \ldots, (P_{n-1}, P'_{n-1}))) \in \Delta_\nu$$

where for every $i \in [n]$:

$$P'_i = \begin{cases} \{\delta^{\mathrm{to}}(q)_i \mid q \in P'\} \setminus F & \text{iff } P' \neq \emptyset \\ P_i \setminus F & \text{iff } P' = \emptyset \end{cases}$$

- $F_\nu = \{(P, \emptyset) \mid P \subseteq Q\}$ □

Lemma 3.4.10 Let ϕ be a μKn-formula and A an ordinary Büchi automaton such that $L(\phi) = L(A)$. Then $L(\nu z.\phi) = L(\nu z.A)$.

Proof: We have $M \models \nu z.\phi$ iff (by 3.4.7) $\mathrm{fix}_z A$ has a ν-Büchi-accepting run Π on M iff (easy) $\mathrm{fix}_z A$ has a ν'-Büchi-accepting run Π on M iff $M \in L(\nu Z.A)$. □

The only difference between the constructions corresponding to the minimal and the maximal fixpoint operators is that in the minimal case we have to prevent infinite regeneration of the fixpoint formula. Intuitively, such regeneration occurs whenever an individual copy of A running as a part of the intermediate automaton $\mathrm{fix}_z A$ takes a transition where the label refers to the fixpoint variable z. The following definition expresses this intuition of one copy of A requiring the starting of another copy, this another one requiring the starting of a third copy etc.

Definition 3.4.11 Let $A = (Q, q_{init}, \Delta, \Omega)$ be an ordinary automaton, $\mathrm{fix}_z A$ the intermediate automaton based on A, Π a run of $\mathrm{fix}_z A$, t a node of Π and q a state $q \in \Pi^{\mathrm{fr}}(t)$. A sequence $d = (t_0, q_0)(t_1, q_1)\ldots$ of pairs of nodes of Π and states $q_i \in \Pi^{\mathrm{fr}}(t_i)$ is a *dependency sequence* of Π from (t, q) iff

- $t \preceq t_0$, i.e. t is an ancestor of t_0,
- $q_0 = \pi(t, q)^{\mathrm{fr}}(t^{-1}t_0)$,
- $z \in \pi(t, q)^{\mathrm{lab}}(t^{-1}t_0)$,

and for all $0 \leq i < |d|$,

- $t_i \preceq t_{i+1}$, i.e. t_i is an ancestor of t_{i+1},
- $q_{i+1} = \pi(t_i, q_{init})^{\mathrm{fr}}(t_i^{-1}t_{i+1})$, and
- $z \in \pi(t_i, q_{init})^{\mathrm{lab}}(t_i^{-1}t_{i+1})$.

We say that d is *proper* iff $|d| > 1$, and if d is finite, that it *leads to* the last element (t_m, q_m). □

Definition 3.4.12 A run Π of the intermediate automaton $\mathrm{fix}_z A$ is *μ-Büchi-accepting* iff

- Π is ν-Büchi-accepting, and
- Π has no infinite dependency sequences. □

Lemma 3.4.13 Let ϕ be a μKn-formula and A an ordinary Büchi automaton such that $L(\phi) = L(A)$. Then for every model $M \in \mathcal{M}_n$, $M \models \mu z.\phi$ iff $\mathrm{fix}_z A$ has a μ-Büchi-accepting run on M.

Proof: By the Knaster-Tarski fixpoint theorem and its formulation in Corollary 2.2.17, we know that $M \models \mu z.\phi$ iff there is an ordinal α and a collection of sets $W_\beta \subseteq \mathrm{st}(M)$ such that $\epsilon \in W_\alpha$, $W_0 = \emptyset$, for all $t \in W_{\beta+1}$, $M[W_\beta/z]^t \models \phi$, and for all limit ordinals λ, $W_\lambda = \bigcup_{\beta \prec \lambda} W_\beta$. If $\mathrm{fix}_z A$ has a μ-Büchi-accepting run Π on M, we have the required W_β by defining

$$\begin{aligned} W_{\beta+1} = \{t \in \mathrm{st}(M) \mid\ & q_{init} \in \Pi^{\mathrm{fr}}(t) \text{ and} \\ & \forall t' \in [n]^* :\ \text{if } z \in \pi(t, q_{init})^{\mathrm{lab}}(t') \text{ then } t \cdot t' \in W_\beta\} \end{aligned}$$

Suppose then that we have the required W_β. For every β and $t \in W_{\beta+1}$, A has an accepting run $\pi_{\beta+1,t}$ on $M[W_\beta/z]^t$. We build a μ-Büchi-accepting run Π from these $\pi_{\beta,t}$ as in 3.4.7. but with following modifications: for $t = \epsilon$, the only included run is $\pi_{\alpha,\epsilon}$; if $\pi_{\beta',t'}$ and $\pi_{\beta'',t''}$ coincide at t, we discard the run with greater index in the lexicographic order of pairs of ordinals and strings; and if $\pi_{\beta',t'}$ refers to z at t, we add $\pi_{\beta,t}$ to the included runs, where $\beta = \min\{\beta'' \mid \pi_{\beta'',t} \text{ is defined}\}$ (notice that $\beta' < \beta$ here). The run Π is ν-accepting as in 3.4.7, since the lexicographic order applied is well-founded. Furthermore, Π has no infinite dependancy sequences, since by the remark above any dependancy sequence corresponds to a strictly decreasing sequence of ordinals. □

In order to construct an ordinary Büchi automaton for $\mu z.\phi$ on the basis of $\mathrm{fix}_z A$, we need a more 'Büchi-like' way of deciding whether a run Π is a μ-accepting one or not. This is where we will take advantage of the fact that for all $\mu z.\phi$ subformulae of any Π_2-formula, the automaton A corresponding to ϕ refers to the μ-variable z only before accepting states. Intuitively, the following concept of a μ'-Büchi-accepting run disallows infinite dependency sequences by requiring that on each path p of the run one eventually reaches a point $i \in \mathbb{N}$ after which the fixpoint z is not unfolded anywhere in the subtree rooted at $P(i)$.

Definition 3.4.14 Let A, $\mathrm{fix}_z A$ and Π be as in 3.4.6. The run Π is *μ'-Büchi-accepting* iff

- Π is ν'-Büchi-accepting and
- for every infinite path p of Π, there is an $i \in \mathbb{N}$ such that for all $t \in \mathrm{dom}(\Pi)$ such that $p(i) \preceq t$, $q_{init} \notin \Pi^{\mathrm{fr}}(t)$. □

Lemma 3.4.15 Let A be an ordinary Büchi automaton which refers to variable z only before accepting states and does not refer to z initially, and let Π be a run of $\mathrm{fix}_z A$. Then Π is μ-Büchi-accepting iff Π is μ'-Büchi-accepting.

Proof: Clearly any μ'-Büchi-accepting run Π is also μ-Büchi-accepting. Take then a μ-Büchi-accepting run Π. As it is ν-Büchi-accepting, it is also ν'-Büchi-accepting. Take then any infinite path p of Π. As Π is ν'-Büchi-accepting, for any $i \in \mathbb{N}$ and $q \in \Pi^{\mathrm{fr}}(p(i))$, there is a future point $j \geq i$ such that $\pi(p(i), q)^{\mathrm{fr}}(p(i)^{-1}p(j)) \in F$. As A refers to z only before accepting states, $\pi(p(i), q)$ then refers to z only finitely many times along p. As A is does not refer to z initially, we also know that for any dependency sequence $(t_0, q_0)(t_1, q_1)\ldots$, the sequence $t_0 \prec t_1 \prec \ldots$ is strictly increasing. From these observations we see inductively that for every $k \in \mathbb{N}$ there are only finitely many dependency sequences of length k along p. Since there are no infinite dependency sequences along p, by König's lemma there is a bound k such that $|d| < k$ for all dependency sequences d along p, and consequently, a bound $m \in \mathbb{N}$ such that $t_i \prec p(m)$ for all elements (t_i, q_i) of all dependency sequences d along p, which implies $q_{init} \notin \Pi^{\mathrm{fr}}(p(j))$ for all $j \geq m$. Since Π is ν'-Büchi-accepting, there is some m' such that all trails $\pi(p(m), q)$ from node $p(m)$ have visited an accepting state before $p(m')$. But since A only refers to z before accepting states, this means that no trail $\pi(p(m'), q)$ from $p(m')$ refers to z anywhere in the subtree $\Pi^{p(m')}$, implying that $q_{init} \notin \Pi^{\mathrm{fr}}(t)$ for all $t \succeq p(m')$. □

Intuitively the Büchi automaton $\mu z.A$ consists of two parts. Initially it behaves exactly like the intermediate automaton $\mathrm{fix}_z A$, without any accepting states. At any transition it can make a non-deterministic choice between staying in the initial part or moving on to the second part, which behaves exactly like the $\nu z.A$ automaton, but without the ability to start any new copies of A.

Definition 3.4.16 Let $A = (Q, q_{init}, \Delta, F)$ be an ordinary Büchi automaton on n-branching trees, and let us write

- $\mathrm{fix}_z A = (Q_{\mathrm{fix}}, q_{init}^{\mathrm{fix}}, \Delta_{\mathrm{fix}})$ and
- $\nu z.A = (Q_\nu, q_{init}^{\nu}, \Delta_\nu, F_\nu)$.

The Büchi-automaton $\mu z.A$ is defined by $\mu z.A = (Q_\mu, q_{init}^{\mu}, \Delta_\mu, F_\mu)$ where

- $Q_\mu = Q_{\text{fix}} \cup Q_\nu$,
- $q^\mu_{init} = q^{\text{fix}}_{init}$,
- if $(P, Z, \overline{P}, \delta) \in \Delta_{\text{fix}}$ then $(P, Z, \overline{P}) \in \Delta_\mu$ and $(P, Z, ((\overline{P}_0, \overline{P}_0), \ldots, (\overline{P}_{n-1}, \overline{P}_{n-1}))) \in \Delta_\mu$,
- if $(P, Z, \overline{P}) \in \Delta_\nu$, $q_{init} \notin P$ and $q_{init} \notin \overline{P}_i$ for every $i \in [n]$, then $(P, Z, \overline{P}) \in \Delta_\mu$, and
- $F_\mu = F_\nu$. □

Lemma 3.4.17 Let ϕ be a μKn-formula and A an ordinary Büchi automaton such that $L(\phi) = L(A)$, and A refers to z only before accepting states, Then $L(\mu z.\phi) = L(\mu z.A)$.

Proof: Straightforward from 3.4.13 and 3.4.15. □

We are now ready to state the main result of the section.

Theorem 3.4.18 Let ϕ be a μKn-formula such that $\phi \in \Pi_2$. Then there exists an ordinary Büchi automaton A such that $L(\phi) = L(A)$.

Proof: Take a formula $\phi \in \Pi_2$. Without loss of generality we can assume that ϕ is guarded and in positive normal form. We show by induction that for all subformulae ψ of ϕ, we can construct an automaton A_ψ such that

1) $L(\psi) = L(A_\psi)$,
2) A_ψ makes reference to a variable $z \in \mathcal{Z}$ only if z occurs free in ψ,
3) if ψ is guarded with respect to a variable $z \in \mathcal{Z}$ then A_ψ does not refer to z initially, and
4) A_ψ refers to any μ-variable of ϕ only before accepting states.

The automata for the base cases of the induction, when ψ is atomic, are provided by Def. 3.4.2, and the constructions corresponding to $\wedge$, $\vee$ and $\textcircled{i}$ by Def. 3.4.3. It is obvious that these fulfil the induction claim.

The construction for $\nu z.\psi$ is provided by Def. 3.4.9, and for $\mu z.\psi$ by Def. 3.4.16. The correctness of these constructions is shown in Lemmas 3.4.10 and 3.4.17. It is clear that induction claims 2 and 3 are fulfilled by $\nu z.A$ and $\mu z.A$. It is also easy to see that if A refers to any x only before accepting states, then so does $\mu z.A$. This is not generally true for $\nu z.A$. However, if x is a μ-variable of ϕ and $\phi \trianglelefteq \mu x.\psi' \triangleleft \nu z.\psi$, then the assumption that $\phi \in \Pi_2$ implies by Corollary 2.2.26 that x cannot occur free in ψ. By induction assumption this means that A does not refer to x at all, and the same holds for $\nu z.A$. □

Corollary 3.4.19 For any language $L \subseteq \mathcal{M}_n$, the following statements are mutually equivalent:

- $L = L(\phi)$ for some (restricted) μKn-formula $\phi \in \Pi_2$,
- $L = L(A)$ for some ordinary or alternating FR-automaton $A \in \Pi_2$, and
- $L = L(A)$ for some ordinary or alternating Büchi automaton A.

Moreover, satisfiability or emptiness is decidable for all these formalisms.

Proof: Immediate from Theorems 3.2.19 and 3.4.18, Prop. 3.2.25, and and the fact that all the translations are clearly computable. □

Let us then examine briefly the relations of Büchi recognisability, the fixpoint alternation free fragment Δ_2 of mu-calculus and the weak calculi $\overset{\text{w}}{\exists}Kn$ and *WSnS*. The results so far allow us to prove one half of Rabin's classical characterisation of *WSnS* as the class of properties for which both the property itself and its complement are Büchi recognisable [77]. We will only state the other half of the characterisation without a proof, since the observations in the current work do not shed any new light on this direction of the correspondence.

Theorem 3.4.20 For every language $L \subseteq \mathcal{M}_n$, the following statements are mutually equivalent:

- $L = L(\phi)$ for some $\overset{\text{w}}{\exists}Kn$ or *WSnS* formula ϕ,
- $L = L(\phi)$ for some μKn-formula ϕ such that $\phi \in \Delta_2$,
- $L = L(\phi)$ and $\mathcal{M}_n \setminus L = L(\bar{\phi})$ for some μKn-formulae ϕ and $\bar{\phi}$ such that $\phi \in \Pi_2$ and $\bar{\phi} \in \Pi_2$, and
- $L = L(A)$ and $\mathcal{M}_n \setminus L = L(\overline{A})$ for some ordinary or alternating Büchi automata A and $\overline{A}$.

Proof: The first two statements are equivalent by Corollary 3.3.6. The second statement implies the third by the fact that if $\phi \in \Delta_2$, then $\phi \in \Pi_2$ and $\neg\phi \in \Pi_2$. The third and the fourth statement are equivalent by Theorems 3.2.19 and 3.4.18. Finally, the fourth statement implies the first by [77, Thm. 29]. □

The results shown so far also allow us to prove Rabin's characterisation of Büchi recognisable languages as those corresponding to existentially quantified *WSnS* formulae. Originally this was shown in [77].

Theorem 3.4.21 For any language $L \subseteq \mathcal{M}_n$, the following statements are mutually equivalent:

- $L = L(\phi)$ for some (restricted) μKn-formula ϕ such that $\phi \in \Pi_2$,
- $L = L(A)$ for some ordinary or alternating Büchi automaton A,

- $L = L(\phi)$ for some $\exists Kn$ or *SnS* formula ϕ of the form $\phi = \exists z_0 \ldots \exists z_n.\psi$ where ψ is a $\overset{\mathrm{w}}{\exists} Kn$ or *WSnS* formula, i.e. a formula with only weak quantifiers.

Proof: The first two statements are mutually equivalent by Corollary 3.4.19.

Take any μKn formula $\phi \in \Pi_2$. Then there are formulae $\psi_0, \ldots, \psi_k \in \Delta_2$ and variables $z_1, \ldots, z_k$ and $x_1, \ldots, x_k$ such that $\phi = \phi_0$ where ϕ_i are defined by: $\phi_i = \psi_i[\nu z_{i+1}.\phi_{i+1}/x_{i+1}]$ for every $0 \leq i < k$, and $\phi_k = \psi_k$. By a generalisation of Lemma 2.2.9 (also Prop. 2.3.6) we can show that for any model M and state s of M, we have $M, s \models \phi$ iff there are sets $W_1, \ldots, W_k \subseteq \mathrm{st}(M)$ such that if we define $M' = M[W_1/x_1] \ldots [W_k/x_k]$, then $M', \epsilon \models \psi_0$ and $M', s \models \psi_i$ for every $1 \leq i \leq k$ and $s \in W_i$. By Theorem 2.2.57 (also Prop. 2.3.17) there are $\overset{\mathrm{w}}{\exists} Kn$-formulae $\bar{\psi}_0, \ldots, \bar{\psi}_k$ such that each $\psi_i \Leftrightarrow \bar{\psi}_i$. Define then a formula $\bar{\phi}$ by

$$\bar{\phi} \;=\; \exists x_1 \ldots \exists x_n.\bar{\psi}_0 \wedge \bigwedge_{i=1}^{n} \mathbf{G}(x_i \Rightarrow \bar{\psi}_i)$$

It is easy to see that $\phi \Leftrightarrow \bar{\phi}$ and that $\bar{\phi}$ is of the form required in the third statement. Consequently, the first statement implies the third.

Take then a formula ϕ as in the third statement. By Corollary 3.3.6 there is a μKn-formula $\psi' \in \Delta_2 \subseteq \Pi_2$ such that $\psi \Leftrightarrow \psi'$, and by Corollary 3.4.19 a restricted μKn-formula $\psi'' \in \Pi_2$ such that $\psi \Leftrightarrow \psi''$. Applying Lemma 3.3.1 n times, we see that there is a restricted μKn-formula $\phi' \in \Pi_2$ such that $L(\phi') = L(\phi)$. Consequently the third statement implies the first. □

Since all the translations between different calculi encountered so far in the current work are clearly computable, the results suffice to show the decidability of *WSnS* and $\overset{\mathrm{w}}{\exists} Kn$. For any *WSnS* or $\overset{\mathrm{w}}{\exists} Kn$ formula ϕ we can produce an equivalent μKn-formula $\phi' \in \Delta_2 \subseteq \Pi_2$ by Cor. 3.3.6, and further an equivalent ordinary Büchi automaton A by Theorem 3.4.18. Deciding the emptiness of the automaton A is then easy. Originally the decidability result for *WSnS* was first established by Doner [23] and for *WS1S* by Läuchli [57].

This correspondence of Theorem 3.4.18 between Büchi automata and the alternation class Π_2 was first reported by Arnold and Niwiński in [3] with a proof outline using a different technique from the one here. For a weaker language, essentially without conjunction, the result was shown earlier by Niwiński [70] and Takahashi [89]. The result can also be obtained indirectly [1], using the equiexpressivity of *WSnS* and the fragment Δ_2 of μKn, stated here in Theorem 2.2.57, the correspondence between maximal fixpoints and existential quantification, and Rabin's characterisation of Büchi recognisability as stated here in Theorem 3.4.21.

3.4.2 Fixpoints and Rabin acceptance

In this subsection we continue the programme of translating mu-calculus formulae inductively to ordinary automata, by generalising the constructions introduced above, especially those for fixpoints, to deal with Rabin instead of Büchi acceptance. As above, the idea is to define an automaton for each atomic μKn-formula directly, and describe a corresponding operation on automata for each boolean, modal and fixpoint operator. Remembering that every Büchi automaton can be viewed as a Rabin automaton, we can use the same automata for atomic formulae as before. The constructions for disjunction, conjunction and the modal operators are also easy to lift from Büchi to Rabin acceptance.

Definition 3.4.22 Let $A_0 = (Q_0, q^0_{init}, \Delta_0, \Omega_0)$ and $A_1 = (Q_1, q^1_{init}, \Delta_1, \Omega_1)$, where $\Omega_0 = ((G^0_0, R^0_0) \dots (G^0_{m_0-1}, R^0_{m_0-1}))$ and $\Omega_1 = ((G^1_0, R^1_0) \dots (G^1_{m_1-1}, R^1_{m_1-1}))$, be ordinary Rabin automata on n-branching trees, Define $A_0 \vee A_1$ as the ordinary Rabin automaton $A_0 \vee A_1 = (Q, q_{init}, \Delta, \Omega)$, where

- $Q = Q_0 \cup Q_1 \cup \{q_x\}$, where q_x is a fresh state and we assume that Q_0 and Q_1 are disjoint,
- $q_{init} = q_x$,
- $\Delta = \Delta_1 \cup \Delta_2 \cup \{(q_x, Z, \bar{q}) \mid (q^0_{init}, Z, \bar{q}) \in \Delta_0\} \cup \{(q_x, Z, \bar{q}) \mid (q^1_{init}, Z, \bar{q}) \in \Delta_1\}$,
- $\Omega = ((G_0, R_0) \dots (G_{m-1}, R_{m-1}))$, where (assuming that $m_0 \leq m_1$) $m = m_1$, $G_i = G^0_i \cup G^1_i$ and $R_i = R^0_i \cup R^1_i$ for all $0 \leq i < m_0$, and $G_i = G^1_i$ and $R_i = R^1_i$ for all $m_0 \leq i < m_1$.

Define $A_0 \wedge A_1$ as the ordinary Rabin automaton $A_0 \wedge A_1 = (Q, q_{init}, \Delta, \Omega)$, where

- $Q = Q_0 \times Q_1 \times ([m_0] \times [m_1] \to \{0, 1, 2\})$,
- $q_{init} = (q^0_{init}, q^1_{init}, c)$, where $c(k_0, k_1) = 0$ for all $k_0 \in [m_0]$ and $k_1 \in [m_1]$,
- if $(q_0, Z_0, \bar{q}^0) \in \Delta_0$, $(q_1, Z_1, \bar{q}^1) \in \Delta_1$ and $(q_0, q_1, c) \in Q$, then

$$((q_0, q_1, c), Z_0 \cup Z_1, ((\bar{q}^0_0, \bar{q}^1_0, c_0), \dots, (\bar{q}^0_{n-1}, \bar{q}^1_{n-1}, c_{n-1}))) \in \Delta$$

  where for every $i \in [n]$, $k_0 \in [m_0]$ and $k_1 \in [m_1]$:

$$\begin{array}{ll} \text{if } c(k_0,k_1) = 0 \text{ and } \bar{q}^0_i \notin G^0_{k_0} & \text{then } c_i(k_0,k_1) = 0 \\ \text{if } c(k_0,k_1) = 0 \text{ and } \bar{q}^0_i \in G^0_{k_0} & \text{then } c_i(k_0,k_1) = 1 \\ \text{if } c(k_0,k_1) = 1 \text{ and } \bar{q}^1_i \notin G^1_{k_1} & \text{then } c_i(k_0,k_1) = 1 \\ \text{if } c(k_0,k_1) = 1 \text{ and } \bar{q}^1_i \in G^1_{k_1} & \text{then } c_i(k_0,k_1) = 2 \\ \text{if } c(k_0,k_1) = 2 & \text{then } c_i(k_0,k_1) = 0 \end{array}$$

- $\Omega = ((G_0, R_0), \dots, (G_{m-1}, R_{m-1}))$, where $m = m_0 \cdot m_1$ and for every $k_0 \in [m_0]$ and $k_1 \in [m_1]$, $G_{k_0 \cdot m_1 + k_1} = \{(q_0, q_1, c) \in Q \mid c(k_0, k_1) = 2\}$ and $R_{k_0 \cdot m_1 + k_1} = \{(q_0, q_1, c) \in Q \mid q_0 \in R_{k_0} \text{ or } q_1 \in R_{k_1}\}$.

Define $\textcircled{i} A_0$ as the ordinary Rabin automaton $\textcircled{i} A_0 = (Q, q_{init}, \Delta, \Omega)$, for any $i \in [n]$, where

- $Q = Q_0 \cup \{q_x, q_{acc}\}$ where q_x and q_{acc} are fresh,
- $q_{init} = q_x$,
- $\Delta = \Delta_0 \cup \{(q_x, \emptyset, \bar{q}) \mid \bar{q}_i = q^0_{init}$ and for all $j \neq i : \bar{q}_j = q_{acc}\} \cup$ $\{(q_{acc}, \emptyset, (q_{acc}, \dots, q_{acc}))\}$,
- $\Omega = ((G_0, R_0), \dots, (G_{m-1}, R_{m-1}))$, where $m = m_0$, $G_0 = G^0_0 \cup \{q_{acc}\}$, $R_0 = R^0_0$, and $G_i = G^0_i$, $R_i = R^0_i$ for all $1 \leq i < m$. □

The fixpoint operations for Rabin automata are based on the same construction of an intermediate automaton as for Büchi automata.

Definition 3.4.23 Let $A = (Q, q_{init}, \Delta, \Omega)$ be an ordinary Rabin automaton, where $\Omega = ((G_0, R_0), \dots, (G_{m-1}, R_{m-1}))$, $\mathrm{fix}_z A$ the intermediate automaton based on A, and Π a run of $\mathrm{fix}_z A$. We say that Π is *ν-Rabin-accepting* iff for every node t of Π, every state $q \in \Pi^{\mathrm{fr}}(t)$ and every infinite path p of the trail $\pi(t, q)$, there is some $k \in [m]$ such that

- $\pi(t, q)^{\mathrm{fr}}(p(i)) \in G_k$ for infinitely many $i \in \mathbb{N}$.
- $\pi(t, q)^{\mathrm{fr}}(p(i)) \in R_k$ for only finitely many $i \in \mathbb{N}$. □

Lemma 3.4.24 Let ϕ be a μKn-formula and A an ordinary Rabin automaton such that $L(\phi) = L(A)$. Then for every model $M \in \mathcal{M}_n$, $M \models \nu z.\phi$ iff $\mathrm{fix}_z A$ has a ν-Rabin-accepting run on M.

Proof: Exactly as the proof of Lemma 3.4.7. □

Let us then express ν-Rabin-acceptance as a condition making reference to the paths of the run Π of the intermediate automaton, instead of the paths of the individual trails $\pi(t, q)$ in Π.

Definition 3.4.25 Let A, Ω and Π be as in Def. 3.4.23. We say that Π is *ν'-Rabin-accepting* iff for every infinite path p of Π there is

- a point $k \in \mathbb{N}$ of path p,
- a set of states $P_d \subseteq \Pi^{\mathrm{fr}}(p(k))$,
- for every $q \in P_d$ an index $m_q \in [m]$, and
- an infinite strictly increasing sequence of points $k = k_0 < k_1 < k_2 < \dots$ of path p

such that

1 for all $q, q' \in P_d$ and all $i \geq k$,

$$\pi(p(k), q)^{\mathrm{fr}}(p(k)^{-1} p(i)) = \pi(p(k), q')^{\mathrm{fr}}(p(k)^{-1} p(i)) \quad \text{iff} \quad q = q'$$

2 for every $j \in \mathbb{N}$ and every $q' \in \Pi^{\mathrm{fr}}(p(k_j))$, there is a $q \in P_d$ such that

$$\pi(p(k), q)^{\mathrm{fr}}(p(k)^{-1}p(k_{j+1})) = \pi(p(k_j), q')^{\mathrm{fr}}(p(k_j)^{-1}p(k_{j+1}))$$

3 for every $q \in P_d$ and every $i \geq k$,

$$\pi(p(k), q)^{\mathrm{fr}}(p(k)^{-1}p(i)) \notin R_{k_q}$$

4 for every $j \in \mathbb{N}$ and every $q \in P_d$, there is a $k_j \leq i \leq k_{j+1}$ such that

$$\pi(p(k), q)^{\mathrm{fr}}(p(k)^{-1}p(i)) \in G_{k_q}$$

□

Intuitively, ν'-Rabin-acceptance states that for any path p of Π, there is a finite set of 'designated' trail paths, each of these trail paths has an associated acceptance pair in Ω, and there is an infinite sequence of 'checkpoints' along p such that (1) all the designated trail paths are entirely separate, (2) any trail path from any checkpoint along p coincides with a designated trail path since the next checkpoint at the latest, (3) no designated trail path ever passes through an associated red state, and (4) every designated trail path passes through an associated green state between any two checkpoints.

Lemma 3.4.26 A run Π of $\mathrm{fix}_z A$ is ν-Rabin-accepting iff it is ν'-Rabin-accepting

Proof: It is easy to see that if Π is ν'-Rabin-accepting, it is also ν-Rabin-accepting. Suppose then that Π is ν-Rabin-accepting, and take any infinite path p of Π. Notice that if there are $k \in \mathbb{N}$, $P_d \subseteq \Pi^{\mathrm{fr}}(p(k))$ and indices m_q for every $q \in P_d$ such that conditions 1 and 3 of ν'-acceptance are fulfilled and the following conditions 2' and 4' hold, then we also have a sequence of points $i_0 < i_1 < \ldots$ along p so that the conditions 2 and 4 of ν'-Rabin-acceptance are fulfilled:

2' for every $i \geq k$ and every $q' \in \Pi^{\mathrm{fr}}(p(i))$, there are $q \in P_d$ and $i' \geq i$ such that $\pi(p(k), q)^{\mathrm{fr}}(p(k)^{-1}p(i')) = \pi(p(i), q')^{\mathrm{fr}}(p(i)^{-1}p(i'))$, and

4' for every $q \in P_d$ and infinitely many $i \geq k$, $\pi(p(k), q)^{\mathrm{fr}}(p(k)^{-1}p(i)) \in G_{m_q}$.

To see that we can pick k and P_d so that conditions 1 and 2' are satisfied, it suffices to notice that the sets $\Pi^{\mathrm{fr}}(p(i))$ are all bounded. Since Π is ν-accepting, it is easy to satisfy conditions 3 and 4', as well. □

To build a Rabin-automaton on the basis of the intermediate automaton, we attach to the intermediate automaton a mechanism checking for every infinite path p whether there exists a point k of p and a set of states P_d fulfilling the requirements of ν'-Rabin-acceptance. This mechanism consists of a table $\bar{e}$, each element

$\bar{e}_i$ of which is used to check for ν'-Rabin-acceptance for some particular set of designated trail paths and indices m_q. An element $\bar{e}_i$ is a 5-tuple $(P, P_d, \mathbf{m}, P_a, P_e)$, where P is the current state of the intermediate automaton, P_d specifies the states in P that are on the designated trails, $\mathbf{m}$ associates with each state in P_d an acceptance pair in Ω, P_a specifies the states in P_d for which the corresponding trail has passed through a green state in the related acceptance pair after last checkpoint, and P_e specifies the states in P on trail paths that are not in the designated trail path set, that have started before last checkpoint, and have not yet coincided with a designated trail path.

In the definition of the transition relation of the Rabin-automaton $\nu z.A$ below, the components of the table $\bar{e}$ are updated according to their intended meaning in a transition. To keep the table $\bar{e}$ bounded, duplicate entries are deleted, keeping only the leftmost occurrence. Furthermore, after each transition, we add to empty places in $\bar{e}$ elements checking ν'-Rabin-acceptance for every possible combination of designated trail paths and indices that is not already being checked by some entry in $\bar{e}$.

Definition 3.4.27 Let $A = (Q, q_{init}, \Delta, \Omega)$ be an ordinary Rabin automaton on n-branching trees, where $\Omega = ((G_0, R_0), \ldots, (G_{m-1}, R_{m-1}))$, and let us write $\text{fix}_z A = (Q_{\text{fix}}, q^{\text{fix}}_{init}, \Delta_{\text{fix}})$. Define first

$$\begin{aligned} E \;=\; & \{(P, P_d, \mathbf{m}, P_a, P_e) \in 2^Q \times 2^Q \times (Q \rightharpoonup [m]) \times 2^Q \times 2^Q \mid \\ & \qquad P_a \subseteq P_d = \text{dom}(\mathbf{m}) \subseteq P,\ P_e \subseteq P,\ P_e \cap P_d = \emptyset\} \end{aligned}$$

For every $P' \subseteq Q$ define

$$E(P') = \{(P, P_d, \mathbf{m}, P_a, P_e) \in E \mid P = P',\ P_a = \emptyset,\ P_e = P \setminus P_d\}$$

Let $\perp$ be an empty element not in E, and define $m_\nu = 2 \cdot |E|$, $\mathcal{E} = (E \cup \{\perp\})^{m_\nu}$.

Define the ordinary Rabin automaton $\nu z.A = (Q_\nu, q^\nu_{init}, \Delta_\nu, \Omega_\nu)$ by:

- Q_ν is the smallest subset of $Q' \times \mathcal{E}$ that contains q^ν_{init} and is closed under the transition function Δ_ν,
- $q^\nu_{init} = (\{q_{init}\}, \bar{e})$, where $\bar{e}$ contains one copy of every element of $E(\{q_{init}\})$, listed in some fixed order, and all other elements of $\bar{e}$ have the empty value $\perp$.
- If $(P, \bar{e}) \in Q_\nu$ and $(P, Z, (P^0, \ldots, P^{n-1}), \delta) \in \Delta_{\text{fix}}$ then

  $$((P, \bar{e}), Z, ((P^0, \bar{e}^0), \ldots, (P^{n-1}, \bar{e}^{n-1}))) \in \Delta_\nu$$

  where for every $j \in [n]$, $\bar{e}^j$ is defined as follows.

  Fix $j \in [n]$, and define a vector $\bar{e}'$. For any $i \in [m_\nu]$, if $\bar{e}_i = \perp$ then $\bar{e}'_i = \perp$. Otherwise, assume that $\bar{e}_i = (P, P_d, \mathbf{m}, P_a, P_e)$. Define

- $P' = P^j$, and
- $P'_d = \{\delta^{\mathrm{to}}(q)_j \mid q \in P_d\}$.
- For every $q' \in P'_d$, if there is a unique $q \in P_d$ such that $\delta^{\mathrm{to}}(q)_j = q'$, then $\mathbf{m}'(q') = \mathbf{m}(q)$ for this unique q, and otherwise $\mathbf{m}'(q')$ is undefined.
- If $P_a = P_d$ and $P_e = \emptyset$, then $P'_a = \emptyset$ and $P'_e = P' \setminus P'_d$, and otherwise

$$\begin{array}{rcl} P'_a & = & \{\delta^{\mathrm{to}}(q)_j \mid q \in P_a\} \cup \{q' \in P'_d \mid q' \in G_{\mathbf{m}'(q')}\} \\ P'_e & = & \{\delta^{\mathrm{to}}(q)_j \mid q \in P_e\} \setminus P'_d \end{array}$$

If either

a) there are $q_1, q_2 \in P_d$ such that $q_1 \neq q_2$ but $\delta^{\mathrm{to}}(q_1)_j = \delta^{\mathrm{to}}(q_2)_j$, or

b) there is some $q' \in P'_d$ such that $q' \in R_{\mathbf{m}'(q')}$,

then $\bar{e}'_i = \bot$, and otherwise $\bar{e}'_i = (P', P'_d, \mathbf{m}', P'_a, P'_e)$.

The vector $\bar{e}^j$ is derived from $\bar{e}'$ by first replacing every entry $\bar{e}_i$ for which there is some $i' < i$ such that $\bar{e}_{i'} = \bar{e}_i$ by $\bot$, and then adding one copy of every value in $E(P^j)$ that does not already occur in $\bar{e}'$ to some place $\bar{e}'_i$ that was not used in $\bar{e}$ (i.e. for which $\bar{e}_i = \bot$), according to some fixed strategy.

- $\Omega_\nu = ((G^\nu_1, R^\nu_1), \ldots, (G^\nu_{m_\nu}, R^\nu_{m_\nu}))$, where for every $i \in [m_\nu]$,

$$\begin{array}{rcl} G^\nu_i & = & \{(P, \bar{e}) \in Q_\nu \mid \bar{e}_i = (P, P_d, \mathbf{m}, P_a, P_e), P_a = P_d \text{ and } P_e = \emptyset\} \\ R^\nu_i & = & \{(P, \bar{e}) \in Q_\nu \mid \bar{e}_i = \bot\} \end{array}$$

□

Lemma 3.4.28 Let ϕ be a μKn-formula and A an ordinary Rabin automaton such that $L(\phi) = L(A)$. Then $L(\nu z.\phi) = L(\nu z.A)$.

Proof: We have $M \models \nu z.\phi$ iff (by Lemma 3.4.24) $\mathrm{fix}_z A$ has a ν-Rabin-accepting run Π on M iff (by Lemma 3.4.26) $\mathrm{fix}_z A$ has a ν'-Rabin-accepting run Π on M iff (easy) $M \in L(\nu Z.A)$. □

Analogous to case of Büchi automata, the only difference between the maximal and minimal fixpoint constructions for Rabin automata is that in the minimal case we have to disallow infinite regeneration of the fixpoint.

Definition 3.4.29 A run Π of the intermediate automaton $\mathrm{fix}_z A$ is *μ-Rabin-accepting* iff

- Π is ν-Rabin-accepting, and
- Π has no infinite dependency sequences. □

Lemma 3.4.30 Let ϕ be a μKn-formula and A an ordinary Rabin automaton such that $L(\phi) = L(A)$. Then for every model $M \in \mathcal{M}_n$, $M \models \mu z.\phi$ iff $\text{fix}_z A$ has a μ-Rabin-accepting run on M.

Proof: Exactly as the proof of lemma 3.4.13. □

Definition 3.4.31 Let A, Ω and Π be as in Def. 3.4.23. We say that Π is *μ'-Rabin-accepting* iff for every infinite path p of Π there is

- a point $k \in \mathbb{N}$ of path p,
- a set of states $P_d \subseteq \Pi^{\text{fr}}(p(k))$,
- for every $q \in P_d$ an index $m_q \in [m]$, and
- an infinite strictly increasing sequence of points $k = k_0 < k_1 < k_2 < \ldots$ of path p

such that they fulfil conditions 1-4 of Definition 3.4.25 and:

5 for every $i \in \mathbb{N}$, every $q' \in \Pi^{\text{fr}}(p(k_i))$, and every proper dependency sequence $(t_0, q_0) \ldots (t_h, q_h)$ from $(p(k_i), q')$ for which t_h is on path p and $t_{h-1} \prec p(k_{i+1}) \preceq t_h$, there are $j \in [h]$ and $q \in P_d$ such that $\pi(p(k), q)^{\text{fr}}(p(k)^{-1}t_j) = q_j$, and

6 for every $q \in P_d$ and every $i \geq k$, there is no proper dependency sequence from $(p(k), q)$ to $(p(i), \pi(p(k), q)^{\text{fr}}(p(k)^{-1}p(i)))$. □

Intuitively, condition (5) states that every proper dependency sequence from a checkpoint extending beyond the following checkpoint passes through a state on some designated trail path between the checkpoints, and condition (6) that there is no proper dependency sequence from a designated trail path back to the same trail path.

Lemma 3.4.32 Let A be an ordinary Rabin automaton which does not refer to z initially, and Π a run of $\text{fix}_z A$. Then Π is μ-Rabin-accepting iff it is μ'-Rabin-accepting

Proof: Assume first that Π is μ'-Rabin-accepting, hence ν'-Rabin-accepting, and ν-Rabin-accepting by Lemma 3.4.26. Suppose then that there is some infinite path p of Π and an infinite dependency sequence $(t_0, q_0)(t_1, q_1) \ldots$ along p. Let k and P_d be as in Def. 3.4.31. Since A does not refer to z initially, $t_0 \prec t_1 \prec \ldots$. By condition 5 of μ'-Rabin-acceptance and the finiteness of P_d, this means that there is some $q \in P_d$ such that $q_i = \pi(p(k), q)^{\text{fr}}(p(k)^{-1}t_i)$ for infinitely many $i \in \mathbb{N}$. But then there is a dependency sequence violating condition 6 for q.

Assume then that Π is μ-Rabin-accepting, hence ν-Rabin-accepting, and ν'-Rabin-accepting by Lemma 3.4.26. Take any infinite path p of Π. As Π is ν'-Rabin-accepting, there are k, P_d etc. fulfilling conditions 1-4 of μ'-acceptance. Notice that for every $q \in P_d$ there is some $i \geq k$ such that there is no proper dependency sequence from $(p(i), \pi(p(k), q)^{\mathrm{fr}}(p(k){-}1p(i)))$ to $(p(i'), \pi(p(k), q)^{\mathrm{fr}}(p(k)^{-1}p(i')))$ for any $i' \geq i$, as otherwise we could construct an infinite dependency sequence along p in Π. Therefore, there are k, P_d, $k_0 < k_1 < \ldots$ fulfilling conditions 1-4 and 6. To see that Π is μ'-Rabin-accepting, let us show that we can pick a subsequence of $k_0 < k_1 < \ldots$ so that condition 5 is satisfied, as well.

To see this, take any k_i and $q' \in \Pi^{\mathrm{fr}}(p(k_i))$. By condition 2 of ν'-Rabin-acceptance, if $(t_0, q_0)(t_1, q_1) \ldots$ is a dependency sequence from $(p(k_i), q')$, and $q_0 \neq \pi(p(k), q)^{\mathrm{fr}}(p(k)^{-1}t_0)$ for all $q \in P_d$, then $t_0 \prec p(k_{i+1})$. This means that for any $h \in \mathbb{N}$, there are only finitely many dependency sequences $(t_0, q_0) \ldots (t_h, q_h)$ of length h from $(p(k_i), q')$ such that $\pi(p(k), q)^{\mathrm{fr}}(p(k)^{-1}t_j) \neq q_j$ for all j and $q \in P_d$. Since there are no infinite dependency sequences in Π, this implies by König's lemma that there are only finitely many dependency sequences $(t_0, q_0) \ldots$ from $(p(k_i), q')$ such that $\pi(p(k), q)^{\mathrm{fr}}(p(k)^{-1}t_j) \neq q_j$ for all j and $q \in P_d$. □

The construction of the Rabin-automaton $\mu z.A$ uses the same strategy as in the case of maximal fixpoints. However, the elements of the table $\bar{e}$ have a new component $\mathbf{d}$, specifying for each designated trail path the states to which there is a proper dependency sequence from some earlier point in the trail. The component P_e is also used to track dependency sequences from trails outside the designated trail path set.

Definition 3.4.33 Let A and Ω be as in Definition 3.4.23, and let us write $\mathrm{fix}_z A = (Q_{\mathrm{fix}}, q^{\mathrm{fix}}_{init}, \Delta_{\mathrm{fix}})$. Define

$$\begin{aligned} E \;=\; \{(P, P_d, \mathbf{m}, \mathbf{d}, P_a, P_e) \in 2^Q \times 2^Q \times (Q \rightharpoonup [m]) \times (Q \rightharpoonup 2^Q) \times 2^Q \times 2^Q \mid \\ P_a \subseteq P_d = \mathrm{dom}(\mathbf{m}) = \mathrm{dom}(\mathbf{d}) \subseteq P, \\ P_e \subseteq P,\; P_e \cap P_d = \emptyset,\; \forall q \in P_d : \mathbf{d}(q) \subseteq P\} \end{aligned}$$

For every $P' \subseteq Q$ define

$$\begin{aligned} E(P') \;=\; \{(P, P_d, \mathbf{m}, \mathbf{d}, P_a, P_e) \in E \mid \\ P = P',\; P_a = \emptyset,\; P_e = P \setminus P_d,\; \forall q \in P_d : \mathbf{d}(q) = \emptyset\} \end{aligned}$$

Define $m_\mu = 2 \cdot |E|$ and $\mathcal{E} = (E \cup \{\bot\})^{m_\mu}$. In the Rabin-automaton

$$\mu z.A = (Q_\mu, q^\mu_{init}, \Delta_\mu, \Omega_\mu)$$

the components Q_μ, q^μ_{init} and Ω_μ are analogous to the Q_ν, q^ν_{init} and Ω_ν of Definition 3.4.27, and Δ_μ is defined as follows.

If $(P, \bar{e}) \in Q_\mu$ and $(P, Z, (P^0, \ldots, P^{n-1}), \delta) \in \Delta'$ then

$$((P, \bar{e}), Z, ((P^0, \bar{e}^0), \ldots, (P^{n-1}, \bar{e}^{n-1}))) \in \Delta_\mu$$

where for every $j \in [n]$, $\bar{e}^j$ is derived from the following $\bar{e}'$ as in Def. 3.4.27.

For any $i \in [m_\mu]$, if $\bar{e}_i = \bot$ then $\bar{e}'_i = \bot$. Assume then $\bar{e}_i = (P, P_d, \mathbf{m}, \mathbf{d}, P_a, P_e)$. Let P', P'_d, $\mathbf{m}'$ and P'_a be as in Definition 3.4.27, and define:

- For every $q' \in P'_d$, if there is a unique $q \in P_d$ such that $\delta^{\text{to}}(q)_j = q'$, then

  $$\mathbf{d}'(q') = \{\delta^{\text{to}}(q'')_j \mid q'' \in \mathbf{d}(q)\} \cup \{\delta^{\text{to}}(q_0)_j \mid \exists q'' \in \mathbf{d}(q) \cup \{q\} : z \in \delta^{\text{lab}}(q'')\},$$

  and if there is no such unique q, then $\mathbf{d}'(q')$ is undefined.
- If $P_a = P_d$ and $P_e = \emptyset$, then $P'_e = P' \setminus P'_d$, and otherwise

  $$P'_e \;=\; (\{\delta^{\text{to}}(q)_j \mid q \in P_e\} \cup \{\delta^{\text{to}}(q_0)_j \mid \exists q'' \in P_e : z \in \delta^{\text{lab}}(q'')\}) \setminus P'_d$$

If condition **a** or **b** of Definition 3.4.27 holds, or if

c) there is some $q' \in P'_d$ such that $q' \in \mathbf{d}'(q')$,

then $\bar{e}'_i = \bot$, and otherwise $\bar{e}'_i = (P', P'_d, \mathbf{m}', \mathbf{d}', P'_a, P'_e)$. □

Lemma 3.4.34 Let ϕ be a μKn-formula and A an ordinary Rabin automaton such that $L(\phi) = L(A)$, and A does not refer to z initially. Then $L(\mu z.\phi) = L(\mu z.A)$.

Proof: Straightforward from 3.4.30 and 3.4.32. □

We are now ready to pull the pieces together and show that the full mu-calculus can be translated to ordinary Rabin automata.

Theorem 3.4.35 For every μKn-formula ϕ there exists an ordinary Rabin automaton A such that $L(\phi) = L(A)$.

Proof: Take a formula ϕ. Without loss of generality we can assume that ϕ is guarded and in positive normal form. We show by induction that for all subformulae ψ of ϕ, we can construct an automaton A_ψ such that

1) $L(\psi) = L(A_\psi)$, and
2) if ψ is guarded with respect to a variable $z \in \mathcal{Z}$ then A_ψ does not refer to z initially,

The automata for the base cases of the induction, when ψ is atomic, are provided by Def. 3.4.2, remembering that every Büchi automaton is also trivially a Rabin automaton. The constructions corresponding to $\wedge$, $\vee$ and $\textcircled{\scriptsize i}$ are described Def. 3.4.22. It is obvious that these fulfil the induction claim.

The construction for $\nu z.\psi$ is provided by Def. 3.4.27, and for $\mu z.\psi$ by Def. 3.4.33. The correctness of these constructions is shown in Lemmas 3.4.28 and 3.4.34. It is obvious that clause 2 in the induction claim is also fulfilled by $\nu z.A$ and $\mu z.A$. $\square$

Let us point out some consequences of this result.

Corollary 3.4.36 For every μKn-formula ϕ, we can compute a restricted μKn-formula ϕ' such that $\models \phi \Leftrightarrow \phi'$.

Proof: Take any μKn-formula ϕ. By Theorem 3.4.35 there is an equivalent ordinary Rabin automaton and by Lemma 3.2.23 an equivalent restricted first recurrence automaton, i.e. a restricted μKn-formula. It is clear that both of these translations are computable. $\square$

Corollary 3.4.37 [**Rabin's complementation lemma**] For every ordinary Rabin automaton A on n-branching trees, there exists an ordinary Rabin automaton $\overline{A}$ such that $L(\overline{A}) = \mathcal{M}_n \setminus L(A)$.

Proof: Take any ordinary Rabin automaton A. By 3.2.23 there exists an equivalent alternating FR-automaton, i.e a μKn-formula ϕ such that $L(A) = L(\phi)$. By Theorem 3.4.35 there exists an ordinary Rabin automaton $\overline{A}$ such that $L(\overline{A}) = L(\neg\phi) = (\mathcal{M}_n \setminus L(\phi)) = (\mathcal{M}_n \setminus L(A))$. $\square$

We are now ready to show the equiexpressivity of $\exists Kn$ and μKn, and the decidability of $\exists Kn$ and *SnS*.

Theorem 3.4.38 For every $\exists Kn$-formula ψ, there exists a μKn formula ϕ such that $\phi \Leftrightarrow \psi$, i.e. $\|\phi\|_M = \|\psi\|_M$ for every $M \in \mathcal{M}_n$.

Proof: Let us translate every formula ψ of $\exists Kn$ inductively to an equivalent formula $f(\phi)$ of μKn. Define the translation for all the operators except $\exists z.\psi$ by

$$\begin{aligned}
f(z) &= z \\
f(\neg\psi) &= \neg f(\psi) \\
f(\psi \wedge \psi') &= f(\psi) \wedge f(\psi') \\
f(\textcircled{\scriptsize i}\,\psi) &= \textcircled{\scriptsize i}\, f(\psi) \\
f(\mathbf{G}\psi) &= \nu z.f(\psi) \wedge \bigwedge_{i \in [n]} \textcircled{\scriptsize i}\, z
\end{aligned}$$

Assume that ψ is an $\exists Kn$-formula and that $f(\psi)$ is the corresponding μKn-formula. By Corollary 3.4.36 there is a restricted, hence strongly aconjunctive μKn-formula ϕ such that $\phi \Leftrightarrow f(\psi)$, and by Lemma 3.3.1 there is a μKn-formula ϕ' such that $\phi' \Leftrightarrow \exists z.\psi$. Take this ψ' as $f(\exists z.\psi)$. $\square$

Corollary 3.4.39 For any language $L \subseteq \mathcal{M}_n$, the following statements are mutually equivalent:

- $L = L(\phi)$ for some (restricted) μKn-formula ϕ,
- $L = L(\phi)$ for some $\exists Kn$ or *SnS* formula ϕ,
- $L = L(A)$ for some ordinary or alternating first recurrence automaton A,
- $L = L(A)$ for some ordinary or alternating Rabin automaton A.

Moreover, satisfiability or emptiness is decidable for all these formalisms.

Proof: Immediate from Prop. 2.3.12, 2.3.14 and 3.2.25, Theorems 3.2.24 and 3.4.35, and the fact that all the translations are clearly computable. □

As mentioned in several occasions above, Rabin's complementation lemma and the decidability of *SnS* were first proved in [76] by a direct construction. When comparing the proof of the complementation lemma here and in Rabin's paper, it becomes apparent that the structure of the proof and the constructions used in it are quite similar. In particular, Rabin uses automata constructions which resemble the fixpoint constructions used here, although they are not called as such. In this way we can also view the use of mu-calculus and the present approach as a way of structuring Rabin's original proof in a more transparent fashion [2].

3.5 Linear vs. branching structures

All the constructions and relations discussed so far in this chapter work uniformly for both the linear-time and branching-time worlds, for any arbitrary fixed branching degree. However, due to the more restricted structure of linear models or 1-branching trees compared to general trees, the linear-time structures and calculi enjoy some special properties which do not hold in the general case. Phrased in automata-theoretic terms, this is reflected in two issues:

- in the linear case the class of Büchi automata and that of Rabin automata are equiexpressive, which is not true in the branching case, and
- in the linear case the class of Rabin automata and that of deterministic Rabin automata are equiexpressive, which again is not true in the branching case.

Let us examine these differences and their implications to the logical calculi in some detail. First, translating Rabin string automata to Büchi ones is easy.

Proposition 3.5.1 For every language $L \subseteq \mathcal{M}_{TL}$ of infinite strings, the following statements are mutually equivalent:

- $L = L(A)$ for some ordinary Büchi automaton A on strings,
- $L = L(A)$ for some ordinary Rabin automaton A on strings.

Proof: As any Büchi automaton can be viewed as a Rabin automaton by Lemma 3.1.7, the first statement implies the second. Take then any ordinary Rabin string automaton $A = (Q, q_{init}, \Delta, \Omega)$, where $\Omega = ((G_0, R_0) \dots (G_{m-1}, R_{m-1}))$. Define the corresponding Büchi automaton A' by $A = (Q', q'_{init}, \Delta', F')$, where

$$\begin{aligned}
Q' &= Q \times [m+1] \\
q'_{init} &= (q_{init}, m) \\
\Delta' &= \{((q,m), Z, (q',i)) \mid (q,Z,q') \in \Delta \text{ and } i \in [m+1]\} \cup \\
&\quad \{((q,i), Z, (q',i)) \mid (q,Z,q') \in \Delta \text{ and } i \in [m] \text{ and } q' \notin R_i\} \\
F' &= \bigcup_{i \in [m]} G_i \times \{i\}
\end{aligned}$$

It is easy to see that $L(A) = L(A')$. Consequently, the second statement implies the first. □

In the light of the results we have already obtained, the equivalence of Rabin and Büchi acceptance conditions on string automata has several implications. First of all, since by Rabin's complementation lemma the class of Rabin recognisable languages is closed under complementation, the class of Büchi recognisable string languages is also closed under complementation.

Corollary 3.5.2 For every ordinary Büchi automaton A on strings, there exists an ordinary Büchi automaton $\overline{A}$ such that $L(\overline{A}) = \mathcal{M}_{TL} \setminus L(A)$.

Proof: Immediate from Corollary 3.4.37 and Prop. 3.5.1. □

If we were just interested in complementing Büchi automata, it would be rather uneconomical to approach the problem as in the Corollary above, i.e. by showing first the more general complementation result for Rabin tree automata and then proceeding to the more restricted case of Büchi string automata. In fact, although complementation for Büchi string automata is not at all as easy as for ordinary automata on finite strings, it is still considerably less involved than for Rabin tree automata, and techniques for complementing Büchi automata were known well before Rabin's complementation result [15, 63].

The equivalence of Büchi and Rabin acceptance conditions for string automata has further interesting consequences. Remember that we noticed before that the expressive power of the weak quantifier-based formalisms $\overset{\text{w}}{\exists}Kn$ or $WSnS$ and the non-alternating fragment Δ_2 of the mu-calculus μKn comprises precisely those languages for which both the language itself and its complement are Büchi-recognisable. Since for linear structures Büchi recognisability is closed under complementation and coincides with Rabin recognisability, this means that the fragment Δ_2 of μTL has the same expressive power as the whole language, i.e. that the fixpoint alternation hierarchy for the linear-time mu-calculus collapses very low indeed.

Proposition 3.5.3 For any linear time mu-calculus μTL-formula ϕ, there exists a non-alternating μTL-formula $\phi' \in \Delta_2$ such that $\models \phi \Leftrightarrow \phi'$.

Proof: Take any μTL-formula ϕ. By Theorem 3.4.35 there are ordinary Rabin automata, and by Theorem 3.5.1 ordinary Büchi automata A and $\overline{A}$ such that $L(\phi) = L(A)$ and $L(\neg\phi) = L(\overline{A})$. By Corollary 3.4.20 there exists then a non-alternating μTL-formula ϕ' such that $L(\phi') = L(A) = L(\phi)$. $\square$

Joined together, these results mean that for the linear-time world, all the formalisms introduced in this work are equiexpressive. In particular, both weak and strong second-order quantification have the same expressive power.

Corollary 3.5.4 For any string language $L \subseteq \mathcal{M}_{TL}$, the following statements are mutually equivalent:

- $L = L(\phi)$ for some (restricted) μTL-formula ϕ,
- $L = L(\phi)$ for some μTL-formula ϕ such that $\phi \in \Delta_2$,
- $L = L(\phi)$ for some $\exists TL$ or *S1S* formula ϕ,
- $L = L(\phi)$ for some $\overset{w}{\exists} TL$ or *WS1S* formula ϕ,
- $L = L(A)$ for some ordinary or alternating first recurrence automaton A,
- $L = L(A)$ for some ordinary or alternating Rabin automaton A.
- $L = L(A)$ for some ordinary or alternating Büchi automaton A.

Proof: Immediate from Prop. 3.5.3, Corollary 3.3.6 and Prop. 3.5.1. $\square$

Let us see then that for the truly branching scenario the statements above do not hold: Büchi automata are not closed under complementation, and Rabin automata are strictly more powerful than Büchi automata.

Proposition 3.5.5 For every branching degree $n \geq 2$, there exists a language $L \subseteq \mathcal{M}_n$ such that $L = L(A)$ for some ordinary Büchi automaton A, but $\mathcal{M}_n \setminus L \neq L(A)$ for every ordinary Büchi automaton A.

Proof: The result was originally shown in [77, Lemma 7], and the proof is also presented in [93, Thm 8.2]. An example of such a language L is the language of all n-branching trees for which a is true in only finitely many states along any path. $\square$

Corollary 3.5.6 For every branching degree $n \geq 2$, there exists a language $L \subseteq \mathcal{M}_n$ for which there is an ordinary Rabin automaton A such that $L = L(A)$, but there is no ordinary Büchi automaton A such that $L = L(A)$.

Proof: Immediate from Prop. 3.5.5, Lemma 3.1.7 and Corollary 3.4.37. $\square$

This implies that the alternation depth hierarchy does not collapse at Δ_2 or Π_2 in the branching case.

Proposition 3.5.7 For every branching degree $n \geq 2$,

- there is a language L such that $L = L(\phi)$ for some μKn-formula $\phi \in \Sigma_2$, but $L \neq L(\psi)$ for every $\psi \in \Pi_2$, and
- there is a language L such that $L = L(\phi)$ for some μKn-formula $\phi \in \Pi_2$, but $L \neq L(\psi)$ for every $\psi \in \Delta_2$.

Proof: Take a language L as in Prop. 3.5.5, and define $\overline{L} = \mathcal{M}_n \setminus L$. By Theorem 3.4.21, there exists a μKn-formula $\phi \in \Pi_2$ such that $L = L(\phi)$, but there exists no μKn-formula $\psi \in \Pi_2$ such that $\overline{L} = L(\psi)$. Since $\Delta_2 \subseteq \Pi_2$ and Δ_2 is closed under negations, the language L fulfils the second claim. Notice then that $\overline{L} = L(\neg\phi)$ and that $\phi \in \Pi_2$ implies $\neg\phi \in \Sigma_2$. Consequently, the language $\overline{L}$ fulfils the first claim. $\square$

The question of whether the alternation depth hierarchy is proper beyond Π_3 used to be a longstanding open problem, which was only recently solved by Bradfield [12] and Lenzi [58] independently. They show that the hierarchy does not collapse at any stage.

Proposition 3.5.7 also shows that for a truly branching framework, the set of formalisms mentioned in Theorem 3.4.20, that in Theorem 3.4.21 and that in Corollary 3.4.39 all have different expressive power. In particular, weak second-order quantification is strictly less expressive than strong quantification.

Let us then look at the other feature distinguishing the linear and branching worlds: determinism and determinisation. So far all the automata we have examined have been non-deterministic, i.e. they may contain states from which more than one transition are enabled on some inputs. In deterministic automata this is not allowed and in any state of the automaton and any input state there can be at most one enabled transition. Since in the current model the transition labels of automata do not have to match the labels of input structures exactly, we have to modify the usual definition of determinism accordingly.

Definition 3.5.8 Let $A = (Q, q_{init}, \Delta, \Omega)$ be an ordinary automaton on n-branching trees. We say that A is *deterministic* iff for every $q \in Q$ and every pair of transitions $(q, Z, \bar{q}) \in \Delta$, $(q, Z', \bar{q}') \in \Delta$ from state q, there is some $z \in \mathcal{Z}$ such that either

- $z \in Z$ and $\neg z \in Z'$, or
- $\neg z \in Z$ and $z \in Z'$. □

Ordinary automata on finite strings can be determinised by the well-known powerset construction. For automata on infinite strings the situation is not quite so easy, but determinisation is possible for them as well. Since we know already that for linear structures Büchi and Rabin recognisability coincide, it is sufficient to consider determinisation of Büchi automata. However, it turns out that in the class of deterministic automata Büchi acceptance is weaker than Rabin acceptance, and the result of determinising a Büchi automaton is no longer necessarily a Büchi automaton itselft.

Proposition 3.5.9 [**McNaughton's Theorem**] For any ordinary Büchi automaton A, there exists an ordinary deterministic Rabin automaton A' such that $L(A) = L(A')$.

Proof: This result was first proved by McNaughton in [63], and an optimal determinisation construction was provided by Safra in [80]. It is easy to see that automata and models considered in the current work, with transition labels only partially specifying the input state, can be translated into the more usual kind of automata and models with transition labels specifying the input state exactly. Translation back is trivial and preserves determinism. This means that we can determinise automata (as considered in the current work) using either McNaughton's or Safra's construction. □

It is easy to see that deterministic Rabin and first recurrence automata have the same expressive power.

Lemma 3.5.10 For any ordinary deterministic Rabin automaton A, there exists an ordinary deterministic first recurrence automaton A' such that $L(A) = L(A')$, and vice versa.

Proof: It suffices to observe that the transformations of Lemmas 3.2.21 and 3.2.23 between Rabin and first recurrence automata preserve determinism. □

The notion of determinism can be extended from automata to formulae. The intuition behind this is to view disjunctions inside a formula as potential sources of non-determinism, and to restrict the structure of a formula so that for any disjunctive subformula $\phi \vee \phi'$ and any possible state of a model, at most one of ϕ or ϕ' can be propositionally consistent with the state.

Definition 3.5.11 Let ϕ and ϕ' be μKn-formulae, and ψ a propositional μKn-formula, i.e. one containing only atomic formulae and the propositional connectives $\wedge$, $\vee$ and $\neg$. We say that ψ is a *deterministic choice* for $\phi \vee \phi'$ iff

- $\models \phi \Rightarrow \psi$ and
- $\models \phi' \Rightarrow \neg\psi$.

Let ϕ be a μKn-formula in pnf. We say that ϕ is *deterministic* iff for every subformula of ϕ of the form $\phi_1 \vee \phi_2$, there is a propositional formula ψ such that

- ψ is a deterministic choice for $\phi_1 \vee \phi_2$, and
- ψ does not contain any occurrences of variables bound by fixpoints in ϕ. $\Box$

Lemma 3.5.12 Let A be an ordinary first recurrence automaton on strings, and $\phi(A)$ the formula corresponding to A as in Def. 3.2.4. The formula $\phi(A)$ is restricted, and if A is deterministic then so is $\phi(A)$.

Proof: Easy. $\Box$

These observations allow us to add some new points to the list of equiexpressive formalisms for linear structures.

Corollary 3.5.13 For any string language $L \subseteq \mathcal{M}_{TL}$, the following statements are equivalent with each other and with any of the statements in Cor. 3.5.4:

- $L = L(\phi)$ for some μTL-formula ϕ,
- $L = L(\phi)$ for some restricted deterministic μTL-formula ϕ,
- $L = L(A)$ for some ordinary deterministic FR-automaton A, and
- $L = L(A)$ for some ordinary deterministic Rabin automaton A.

Proof: Immediate from Corollary 3.5.4, Proposition 3.5.9 and Lemmas 3.5.10 and 3.5.12. $\Box$

For the branching time world determinisation becomes impossible already for automata on finite trees. For example, it is easy to construct an ordinary non-deterministic automaton A recognising the language of all trees where the proposition a is true in at least one state; intuitively A just guesses a path to a state where a is true and verifies that a is true there. However, there is no deterministic automaton recognising this language. The same counterexample works also for all types of automata on infinite trees, so there is no hope of determinisation for them either.

$$\begin{tikzcd}
\begin{array}{c}\text{restricted } \mu Kn,\\ \text{ord./alt. FR,}\\ \text{ord./alt. Rabin}\end{array} \arrow[r, leftrightarrow] \arrow[d, dashed, shift right] & \mu Kn \arrow[r, shift left] & \exists Kn \arrow[l, shift left, "(3.4.38)"] \arrow[r, leftrightarrow] & SnS \\
\begin{array}{c}\Pi_2 \text{ of restr. } \mu Kn,\\ \Pi_2 \text{ of ord./alt. FR,}\\ \text{ord./alt. Büchi}\end{array} \arrow[r, leftrightarrow] \arrow[u, shift right] \arrow[d, dashed, shift right, "(3.5.4)"'] & \begin{array}{c}\Pi_2 \text{ of}\\ \mu Kn\end{array} \arrow[r, leftrightarrow, "(3.4.21)"'] \arrow[u] & \begin{array}{c}\exists\text{-quantified}\\ \overset{\text{w}}{\exists} Kn\end{array} \arrow[r, leftrightarrow] \arrow[u] & \begin{array}{c}\exists\text{-quantified}\\ WSnS\end{array} \arrow[u] \\
\begin{array}{c}\Delta_2 \text{ of alt. FR}\\ L \text{ and } \overline{L} \text{ are Büchi}\\ L \text{ and } \overline{L} \text{ are in } \Pi_2\end{array} \arrow[r, leftrightarrow, "(3.4.20)"'] \arrow[u, shift right] & \begin{array}{c}\Delta_2 \text{ of}\\ \mu Kn\end{array} \arrow[r, shift left] \arrow[u] & \overset{\text{w}}{\exists} Kn \arrow[l, shift left, "(3.3.5)"] \arrow[r, leftrightarrow] \arrow[u] & WSnS \arrow[u]
\end{tikzcd}$$

In the figure arcs represent inclusion of expressive power. Unlabelled arcs correspond to trivial inclusions or to relations present in Fig. 2.4 (page 59) or Fig. 3.6. The relations corresponding to dashed arcs hold only for linear structures and calculi. Notice that in linear case all the formalisms are equiexpressive.

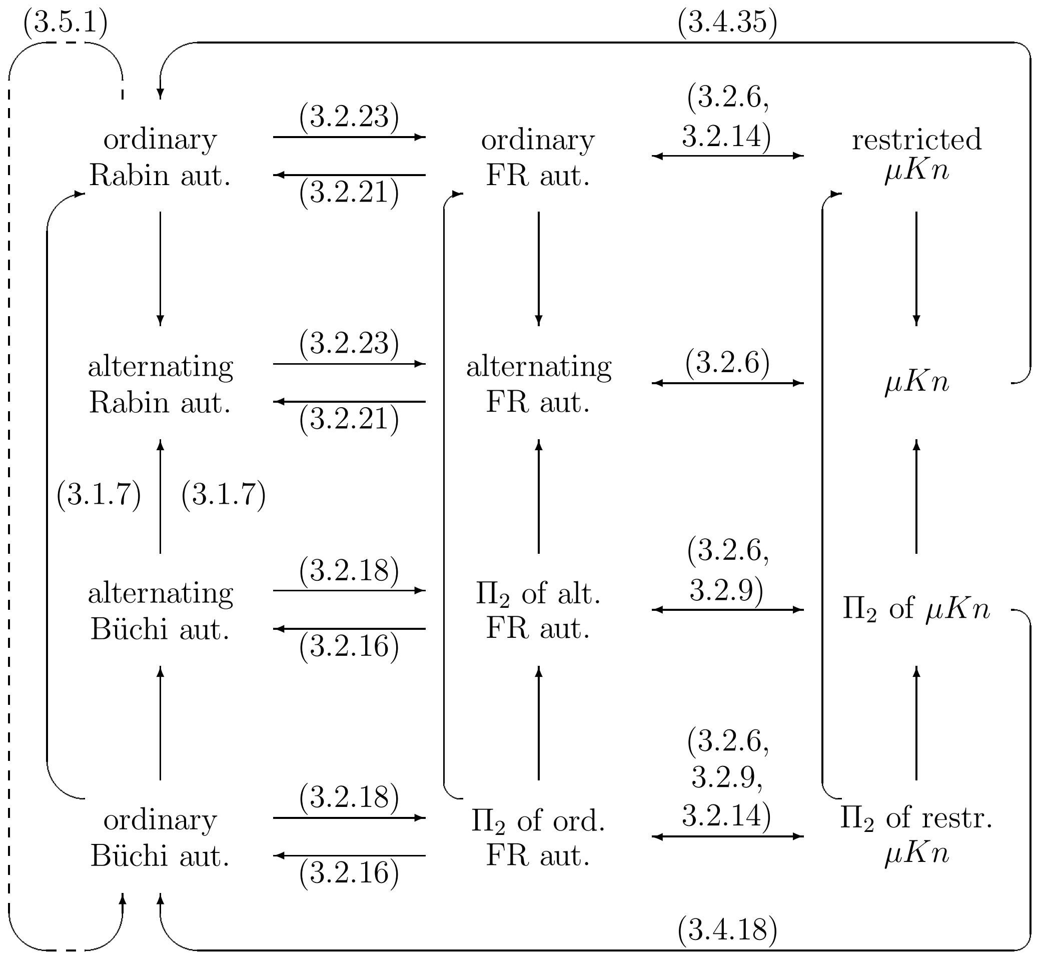
Figure 3.5: Relations of automata and logical calculi

3.6 Summary

In this chapter we have discussed several different types of automata and examined their relations to the logical formalisms discussed in the previous chapter. The main types of automata and acceptance conditions are listed in the following table, along with references to the corresponding definitions.

	ordinary		alternating
	on strings	on trees	
general	Def. 3.1.1	Def. 3.1.4	Def. 3.1.8
Büchi & Rabin	Def. 3.1.3	Def. 3.1.6	Def. 3.1.10
first recurrence	Def. 3.2.2		Def. 3.2.2
restricted	-		Def. 3.1.14
deterministic	Def. 3.5.8		-

Figure 3.5 presents a comparison of various formalisms according to their expressive power, and Figure 3.6 summarises the most important translations between automata and mu-calculus exhibited above. Thanks to the translations between the formalisms, all the formalisms listed in these figures are decidable.



Unlabelled arcs correspond to trivial inclusions or translations between ordinary and restricted alternating automata, (Lemmas 3.1.15, 3.1.16 and 3.2.3). The translation corresponding to the dashed arc is available only in linear structures.

Figure 3.6: Translations between automata and mu-calculus

Chapter 4

Deciding mu-calculi

In this chapter we examine the decision problem for linear-time and modal mu-calculi, i.e. the problem of determining whether a given formula ϕ is satisfiable or not. Using the translations and correspondences of Chapter 3, the satisfiability of a mu-calculus formula can be decided by translating it to a Rabin or first recurrence automaton. However, the drawback of the inductive nature of the translation is that it involves a non-elementary blowup, and is therefore of little practical interest.

In Section 2.2.4 we gave an account of satisfiability in terms of infinite bundled tableaux. The basic idea of all direct decision methods for mu-calculi is a similar analysis of formulae by decomposition and unfolding of fixpoints. Effectively these decision procedures work by determining whether there exists a proper bundled tableau for the given formula ϕ. The easy part of the problem is determining whether there exists any bundled tableau for ϕ at all, proper or not. Where the difficulty lies is in determining in a finitary way whether among the possible bundled tableaux for ϕ there is some proper one. In other words, the difficult question is whether it is possible to avoid unfolding minimal fixpoints infinitely many times in potential models of the formula.

Assume that we are interested in the satisfiability of ϕ. The standard decision procedure first builds on the basis of ϕ a finite tableau-like structure, known as a *Hintikka structure*, which takes care of the satisfaction of ϕ as far as propositional connectives, unfolding of fixpoints, and consistency between successive states is concerned. In the terminology of the current work, we can view this Hintikka structure as a finite encoding of all bundled tableaux for ϕ. It naturally gives rise to an automaton A_{loc} on infinite trees recognising structures that are models of ϕ as far as local aspects are concerned. To guarantee well-foundedness of minimal fixpoint unfoldigs, another automaton A_{wf} on infinite strings is constructed. The task of this is to detect paths of the automaton A_{loc} where some minimal fixpoint is unfolded infinitely many times. By complementing A_{wf} we get an automaton $\overline{A}_{\mathrm{wf}}$ accepting those paths of A_{loc} on which every minimal fixpoint is unfolded only finitely many times. An intersection of A_{loc} and $\overline{A}_{\mathrm{wf}}$ yields then an automaton A_ϕ on infinite trees recognising the proper tableaux for ϕ. Finally, checking A_ϕ for emptiness answers the question of satisfiability of ϕ. This decision procedure was first presented for modal mu-calculus by Streett and Emerson [86, 87]. It has been applied to the linear-time mu-calculus, including past operators, by Vardi [95]. A variation of the method for linear-time mu-calculus works by constructing the Hintikka structure or automaton A_{loc} as above, and by using special graph marking algorithms to either extract from this structure a proper linear bundled tableau, or to detect that no such tableau exists [6, 59].

A drawback in these approaches is the non-transparency caused by the rather complex automata constructions, especially complementation of A_{wf}, or graph marking algorithms. This means that although they certainly give an answer to the question of whether ϕ is satisfiable, they do not necessarily give an insight into why or how this happens. Moreover, in applying these methods it is difficult for a human verifier to take advantage of her intuitions about the formula to ease the task, by guiding the decision procedure to a suitable direction.

Direct tableaux methods enjoying precisely these properties of transparency and possibility of human guidance have been developed with great success for the model checking problem, i.e. for deciding whether a given formula is true in a state of a given model, by Bradfield, Stirling and Walker [11, 13, 14, 85]. However, no direct tableau construction for the satisfiability problem of mu-calculi has been presented so far, and exhibiting such a construction for the modal mu-calculus was in fact mentioned as an open problem in [46].

Below we describe an elementary tableau decision method for both the linear-time and modal mu-calculi. The basis of the method is a generalisation of definition lists called *definition trees.* The usage of definition trees in the tableaux here corresponds implicitly to the string automaton A_{wf} and its determinisation and complementation by Safra's construction [80] in the decision method above. It builds on the work of Walukiewicz [100], who described an infinitary tableau system where the tableau nodes were labelled with trees of sets of formulae. Here the information coded in these trees has been represented in the definition trees. A related infinitary tableau construction has also been presented by Emerson and Jutla [33, 46].

The essential difference between the tableau system below and those of [33] and [100] lies in the step from an infinitary system to a finitary one. Here we formulate a simple leaf condition which allows us to stop the construction of any branch of a tableau after a finite number of steps, and which tells directly whether a tableau is succesful or not. In contrast, Walukiewicz [100] interprets a finite representation of a tableau as a Rabin automaton on infinite trees, reducing the satisfiability of the formula to the non-emptiness of this automaton. The treatment of [33] works along similar lines. A very early version of the current tableau method was described in [48]; this worked only for the linear-time mu-calculus, and the notion of definition trees was not yet present.

In addition to providing a solution to the decision problem of mu-calculi, the tableau method can also be viewed as a direct transformation of mu-calculus formulae to the strongly aconjunctive form. As we noticed in last chapter, this allows us to translate the calculi $\exists Kn$ and SnS inductively to the modal mu-calculus, giving yet another proof of the decidability of these formalisms with second-order quantifiers. Compared to the decidability proof in last chapter, the proof using the tableau construction has the added elegance that there is no need to explicitly invoke the notion of automata at all.

4.1 Definition trees

Let us start by defining the concept of a definition tree, which extends the notion of a definition list, and by describing some basic operations for definition trees.

Definition 4.1.1 A *definition tree* is a finite sequence

$$d = (u_0, \phi_0) \dots (u_k, \phi_k)$$

where

- every u_i is a definition constant, $u_i \in \mathcal{U}$, and every ϕ_i is either a definition constant $u \in \mathcal{U}$ or an extended fixpoint formula $\sigma z_i.\psi_i$,
- all u_i are distinct, and
- if a constant u occurs in ϕ_i, then $u = u_j$ for some $j < i$.

For every u_i, define $d(u_i) = \phi_i$. Define also the operation d^* inductively by

- if $d(u) = \sigma z.\phi$, then $d^*(u) = d(u)$, and
- if $d(u) = u'$, then $d^*(u) = d^*(u')$.

We call u a *maximal*, *minimal* or *auxiliary* constant of d depending on whether $d(u) = \nu z.\phi$, $d(u) = \mu z.\phi$ or $d(u) = u'$. The concept of *activeness* and the notation $\phi[d]$ and $\Gamma[d]$ are defined for definition trees as for definition lists in Def. 2.2.29. We say that a constant u is defined *before* v and that v is defined *after* u in d iff $u = u_i$ and $v = u_j$ for some $0 \leq i < j \leq k$. We say that u is defined *immediately before* v and that v is defined *immediately after* u in d iff $u = u_i$ and $v = u_{i+1}$ for some $0 \leq i \leq k$. □

The difference between definition lists and definition trees is that the latter may contain pairs (u, u'), where an auxiliary constant u is defined to have the same meaning as u'. The auxiliary constants are used in the tableau construction below to keep track of the unfoldings of minimal fixpoints. Intuitively, when a minimal constant u corresponding to a minimal fixpoint $\mu z.\phi$ is unfolded in the tableau, we replace u in $\phi[u/z]$ by a fresh constant u', which is defined to stand for u by adding the pair (u', u) to the definition list.

The reason for calling these structures definition *trees* is to stress the fact that the tableaux below depend on the tree structure implicitly defined in definition trees. Intuitively, a constant v is an ancestor of u iff v is active in u, relative to the definition tree d. Naturally, a similar tree structure arises also in ordinary definition lists, but this is not taken advantage of.

Definition 4.1.2 A definition tree d generates a finite tree labelled with constants defined in d as follows:

- the root of the tree is labelled with a dummy constant u_ϵ, and has a child labelled with u for every constant u defined in d such that u does not occur in $d(v)$ for any constant v defined in d, and
- if a node of the tree, other than the root, is labelled with a constant u, then the node has a child labelled with v for every constant v such that u occurs in $d(v)$ and for which there is no u' such that u would occur in $d(u')$ and u' be active in $d(v)$.

The ordering of children of any node of this tree is defined by:

- any child labelled with an auxiliary constant is older than any child labelled with a minimal or maximal constant,
- a child labelled with an auxiliary constant u is older than a child labelled with an auxiliary constant v iff u is defined before v in d, and
- a child labelled with a minimal or maximal constant u is older than a child labelled with a minimal or maximal constant v iff u is defined before v in d.

We say that a constant u is a *parent/child/(proper) descendant/(proper) ancestor* of a constant v in d iff the unique node of this tree labelled with u stands in that relation to the unique node labelled with v. We say that a constant u is a *leaf* of d iff the nodel labelled with u is a leaf. We also say that u is *older than* v in d iff the node labelled with u is older than the node labelled with v, according to the global 'older than' ordering induced by the ordering of the children of nodes (Def. 2.1.4). We say that u is an *unfolding* of v iff u is a proper descendant of v and $d^*(v) = d^*(u)$.

$$\begin{tikzcd}
 & & & & u_\epsilon \arrow[dll] \arrow[drr] & & \\
 & & u \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & & & w \\
 & u' \arrow[dl] \arrow[dr] & u''' & v & v' & & \\
u'' & & v'' & & & &
\end{tikzcd}$$

$$d = (u, \mu z.\phi(z) \wedge \nu x.\psi(z,x))\,(v, \nu x.\psi(u,x))\,(u',u)$$
$$(v', \nu x.\psi(u,x))\,(w, \mu y.\chi(y))\,(v'', \nu x.\psi(u',x))\,(u'',u')\,(u''',u)$$

Figure 4.1: The structure of a definition tree

Let us define a linear ordering $\prec_d$ on the constants defined in d by: $u \prec_d v$ iff either u is a proper descendant of v or u is older than but not an ancestor of v in d. □

For an example of a definition tree and the tree structure induced by it, see Fig. 4.1. We have used there the convention of writing $\psi(x,z)$ to mean that z and x occur free in ψ. In the definition list of Fig. 4.1, the constant u'' is an unfolding of both u' and u, and u' is an unfolding of u. The constant v'' is older than for example u''' or v.

Definition 4.1.3 Let d be a definition tree and ϕ an extended formula. We say that ϕ is *well-formed* with respect to d iff for all constants u and v occurring in ϕ, either u is an ancestor of v or vice versa. If ϕ is well-formed with respect to d, we use $u(\phi)$ to denote the unique constant u such that

- either $u = u_\epsilon$ or u occurs in ϕ, and
- no proper descendant of u occurs in ϕ.

If ϕ and ψ are well-formed with respect to d, we say that ϕ is *older than* ψ relative to d iff $u(\phi)$ is older than $u(\psi)$. In the same way, we say that ϕ is a *(proper) ancestor/(proper) descendant* of ψ iff $u(\phi)$ stands in that relation to $u(\psi)$. We write $\phi \prec_d \psi$ iff $u(\phi) \prec_d u(\psi)$. □

Let us then define two operations for manipulating definition trees: deletion of constants and substitution.

Definition 4.1.4 Let $d = (u_0, \phi_0) \ldots (u_k, \phi_k)$ be a definition tree, and U a set of constants defined in d, $U \subseteq \{u_0, \ldots, u_k\}$. Then the expression $d \setminus U$ denotes the definition tree obtained from d by removing every pair (u_i, ϕ_i) for which $u_i \in U$.

Let $v_0, \ldots, v_m$ be constants and $\psi_0, \ldots, \psi_m$ extended formulae. Then the expression $d[\psi_0/v_0, \ldots, \psi_m/v_m]$ denotes the definition tree $d' = (u_0, \phi'_0) \ldots (u_k, \phi'_k)$ where each $\phi'_i = \phi_i[\psi_0/v_0, \ldots, \psi_m/v_m]$. If ψ is a formula and $U = \{v_0, \ldots, v_m\}$ a set of constants, $d[\psi/U]$ is an abbreviation for $d[\psi/v_0, \ldots, \psi/v_m]$. We use the same abbreviation for substitution to a formula $\phi[\psi/U]$ or a set of formulae $\Gamma[\psi/U]$. $\square$

We will also need the notion of similarity between formulae. Intuitively this means that two formulae are the same except for the choice of names of definition constants in them.

Definition 4.1.5 Let d and d' be two definition lists. Define the concept of extended formulae ϕ and ϕ' being *similar* (relative to d and d') inductively as follows:

- if $\phi = \phi'$ and ϕ and ϕ' do not contain any definition constants, then ϕ and ϕ' are similar, and
- if there is a formula ψ which does not contain any definition constants, variables $z_0, \ldots, z_k$ and constants $u_0, \ldots, u_k$ and $v_0, \ldots, v_k$ such that
  - $\phi = \psi[u_0/z_0, \ldots, u_k/z_k]$,
  - $\phi' = \psi[v_0/z_0, \ldots, v_k/z_k]$ and
  - each $d^*(u_i)$ and $d'^*(v_i)$ are similar,

  then ϕ and ϕ' are similar.

If $d = d'$ above, we say simply that ϕ and ϕ' are similar relative to d. $\square$

Notice that if ϕ and ϕ' are similar relative to d and d', then $\phi[d] = \phi'[d']$. However, the converse does not necessarily hold. For example, if the definition tree d is as in Fig. 4.1, then $a \wedge \bigcirc u''$ and $a \wedge \bigcirc u$ are similar relative to d, the same holds for $\bigcirc b \vee v'$ and $\bigcirc b \vee v''$, but $a \wedge \bigcirc w$ and $a \wedge \bigcirc v$ are not similar, and neither are v and $\nu x.\psi(u, x)$, although $v[d] = (\nu x.\psi(u, x))[d]$.

4.2 Tableaux with definition trees

As a preliminary step towards the tableau-based decision procedure, let us formulate an infinitary tableau system analogous to bundled tableaux but based on decision trees.

Definition 4.2.1 Let ϕ be a guarded μTL-formula in pnf. A *definition tree tableau* T for ϕ is an infinite sequence $T = (s_0, \Gamma_0, d_0)(s_1, \Gamma_1, d_1) \ldots$ where

name	application
del-ϕ	$\dfrac{s,\quad \Gamma \cup \{\phi, \phi'\},\quad d}{s,\quad \Gamma \cup \{\phi\},\quad d}$ where ϕ and ϕ' are similar relative to d and $\phi \prec_d \phi'$
del-U	$\dfrac{s,\quad \Gamma,\quad d}{s,\quad \Gamma,\quad d \setminus \{u\}}$ where u does not occur in any $\phi \in \Gamma$, and u is a leaf of d
contr-U	$\dfrac{s,\quad \Gamma,\quad d}{s,\quad \Gamma[u/U],\quad (d \setminus U)[u/U]}$ where u does not occur in any $\phi \in \Gamma$, every child of u in d is an unfolding of u, and U is the set of unfoldings of u in d
$\vee$**L**	$\dfrac{s,\quad \Gamma \cup \{\phi \vee \phi'\},\quad d}{s,\quad \Gamma \cup \{\phi\},\quad d}$
$\vee$**R**	$\dfrac{s,\quad \Gamma \cup \{\phi \vee \phi'\},\quad d}{s,\quad \Gamma \cup \{\phi'\},\quad d}$
$\wedge$	$\dfrac{s,\quad \Gamma \cup \{\phi \wedge \phi'\},\quad d}{s,\quad \Gamma \cup \{\phi, \phi'\},\quad d}$
$\boldsymbol{\sigma}$	$\dfrac{s,\quad \Gamma \cup \{\sigma z.\phi\},\quad d}{s,\quad \Gamma \cup \{u\},\quad d \cdot (u, \sigma z.\phi)}$ where u does not appear in d
$\boldsymbol{U\nu}$	$\dfrac{s,\quad \Gamma \cup \{u\},\quad d}{s,\quad \Gamma \cup \{\phi[u/z]\},\quad d}$ where $d^*(u) = \nu z.\phi$
$\boldsymbol{U\mu}$	$\dfrac{s,\quad \Gamma \cup \{u\},\quad d}{s,\quad \Gamma \cup \{\phi[u'/z]\},\quad d \cdot (u', u)}$ where $d^*(u) = \mu z.\phi$ and u' does not appear in d
$\bigcirc$	$\dfrac{s,\quad \{z_1, \ldots, z_m, \bigcirc\phi_1, \ldots, \bigcirc\phi_k\},\quad d}{s+1,\quad \{\phi_1, \ldots, \phi_k\},\quad d}$ where every z_i is atomic
Note:	In each rule, Γ is disjoint from the other set

Table 4.1: Definition tree tableau rules for μTL

- every $s_i \in \mathbb{N}$, Γ_i is a finite set of extended μTL-formulae in pnf, and d_i is a definition tree,
- $(s_0, \Gamma_0, d_0) = (0, \{\phi\}, \epsilon)$, and
- every $(s_{i+1}, \Gamma_{i+1}, d_{i+1})$ is derived from (s_i, Γ_i, d_i) by applying one of the rules in Table 4.1. The rules have the following priority order:
  - del-ϕ (highest priority),
  - del-U,
  - contr-U and
  - all other rules (lowest priority)

  A rule with a lower priority can be applied only if there is no applicable rule with a higher priority. Furthermore, the del-U and contr-U rules have to be applied to the oldest u in d that they can be applied to.

  In an application of the del-U rule, we say that the constant u is *deleted* by the rule, and in an application of the contr-U rule we say that u has been *contracted* and every u_i *deleted* by the rule.

We say that T is *proper* iff there is no constant u such that
- u is contracted in infinitely many points of T, and
- u is deleted in only finitely many points of T.

The concept of the *propositional consistency* of a tableau, and that of a tableau *agreeing* with a model are defined as in Def. 2.2.38. □

Those definition tree tableau rules which are the same as for bundled tableaux need no explanation. The U rule for constants is split in two, depending on whether the constant u refers to a maximal or a minimal fixpoint. In the first case the rule is the same as with bundled tableaux. In the case of a minimal fixpoint, however, the formula u is not just replaced by $\phi[u/z]$, but a new auxiliary constant u' is introduced as an unfolding of u, and u is replaced by $\phi[u'/z]$ in Γ instead. Since the meaning of the new auxiliary constant u' is defined as u, extended formulae $\psi[u/z]$ and $\psi[u'/z]$ have the same meaning in the sense that $(\psi[u/z])[d] = (\psi[u'/z])[d]$. However, regarding well-foundedness of minimal fixpoint unfoldings, the constant u' is 'more dangerous' than u in that the minimal fixpoint formula corresponding to both u and u' has been unfolded once more in a derivation leading to $\psi[u'/z]$ than in a derivation leading to $\psi[u/z]$.

If then later u' is unfolded, a new auxiliary constant u'' is again introduced as an unfolding of u', u' replaced by $\phi[u''/z]$ and so on. In this way, the definition trees in a tableau keep track of the unfolding of minimal fixpoints. Assuming for the moment that there were no del-ϕ, del-U and contr-U rules, we could find out whether some minimal fixpoint is unfolded infinitely many times in a tableau by looking at whether some branch of the definition trees grows infinitely long, or to be more accurate, whether there are infinite sequences $u_0, u_1, \ldots$ of constants and $i(0), i(1), \ldots$ of points in the tableau such that each u_{j+1} is an unfolding of u_j in the definition tree $d_{i(j)}$.

The purpose of the del-ϕ rule is to bound the number of formulae in any set Γ_i in a tableau by deleting multiple copies of similar formulae, i.e. formulae which differ only by the choice of constant names. For the correctness of the tableau system it is crucial that formulae are not deleted in an arbitrary manner. Instead, the choice of which of two similar formulae to erase is fixed by the ordering $\prec_d$ induced by the current definition tree d; the formula which is smaller in the ordering $\prec_d$ is preserved and the other deleted. Notice that if u' is an unfolding of u in d and ϕ is a formula containing a free occurrence of z, then $\phi[u'/z] \prec_d \phi[u/z]$, and an application of the del-ϕ rule would delete the 'less dangerous' $\phi[u/z]$ and preserve the 'more dangerous' $\phi[u'/z]$.

The rule del-U is a housekeeping rule, which allows us to delete constants u which are not active in any formula $\phi \in \Gamma$. Such constants cannot have any effect on the subsequent steps in a tableau anyway, since they cannot ever reappear and be unfolded.

Probably the most interesting rule in the tableau is the contraction rule contr-U. Consider a situation in which the rule is applicable to constant u. First of all u does not occur in any $\phi \in \Gamma$ but it must be active in some, as otherwise the del-U rule would be applicable. Secondly, every child u' of u in d is an unfolding of u, i.e. an auxiliary constant for which $d(u') = u$. Forgetting the existence of the contr-U rule for the moment, we can see from the tableau rules that this means that no sequence of derivations can lead to a point where u could reappear and be unfolded. In effect, the interior node u of the definition tree d has become useless. What the contr-U operation does is to compress the node corresponding to u and those descendants of it which are unfoldings of u together to form a single node.

For the purposes of the proofs below, let us define the notion of dependencies also for definition tree tableaux. Notice that this was not necessary in order to spell out the condition of a tableau being proper.

Definition 4.2.2 Let $T = (s_0, \Gamma_0, d_0)(s_1, \Gamma_1, d_1)\ldots$ be a dependency tree tableau. For every point $j \in \mathbb{N}$ of T, define the dependency relation $\rightarrow \subseteq \Gamma_j \times \Gamma_{j+1}$ for the other rules as in Def. 2.2.38 for a bundled tableau, and for the del-ϕ, del-U, and contr-U rules as follows:

- If the rule applied at point j of T is del-ϕ, then $\psi \rightarrow \psi$ for every $\psi \in \Gamma_j \cup \{\phi\}$. The formula $\phi' \in \Gamma_j$ deleted by the rule has no dependants in Γ_{j+1}.

- If the rule applied at point j of T is del-U, then $\psi \rightarrow \psi$ for every $\psi \in \Gamma_j$.
- If the rule applied at point j of T is contr-U, then $\psi \rightarrow \psi[u/U]$ for every $\psi \in \Gamma_j$.

For any $n \in \mathbb{N}$, a sequence $\phi_0, \phi_1, \ldots$ is a *direct dependency sequence* of T from ϕ_0 at point n iff for every ϕ_i in the sequence $\phi_i \rightarrow \phi_{i+1}$ relative to the rule applied at point $n+i$.

For any $n \in \mathbb{N}$, a sequence $\phi_0, \phi_1, \ldots$ is an *indirect dependency sequence* of T from ϕ_0 at point n iff for every ϕ_i in the sequence either $\phi_i \rightarrow \phi_{i+1}$ relative to the rule applied at point $n+i$, or the del-ϕ rule is applied at point $n+i$ to delete formula ϕ_i, the formulae ϕ_i and ϕ_{i+1} are similar relative to d_{n+i}, and $\phi_{i+1} \prec_{d_{n+i}} \phi_i$.

Let $\phi_0, \phi_1, \ldots$ be an infinite direct or indirect dependency sequence from point n of T. We say that the sequence is *bad* iff there is some minimal or auxiliary constant u of d_n such that u is active in every ϕ_i (relative to d_{n+i}), and $\phi_i = u$ for infinitely many $i \in \mathbb{N}$. □

Notice that in the dependency relations for the contr-U rule, if $\phi \rightarrow \psi$ and $\phi' \rightarrow \psi$, then $\phi = \phi'$, as otherwise the del-ϕ rule would be applicable. Let us list some important properties of definition tree tableaux.

Lemma 4.2.3 Let $T = (s_0, \Gamma_0, d_0)(s_1, \Gamma_1, d_1)\ldots$ be a definition tree tableau. Then

1 There is a bound $n \in \mathbb{N}$ such that $|\Gamma_i| \leq n$ for any $i \in \mathbb{N}$.

2 There is a bound $n \in \mathbb{N}$ such that $|d_i| \leq n$ and the depth of d_i (when considered as a tree) is at most n, for any $i \in \mathbb{N}$.

3 Let u and u' be constants which are defined in d_i and d_{i+1}, and assume that $u \prec_{d_i} u'$ and no common ancestor of u and u' in d_i is contracted at point i of T. Then $u \prec_{d_{i+1}} u'$.

4 Let u be a constant which is defined in d_i and $\phi \in \Gamma_i$ a descendant of u in d_i, and assume that there is a direct or indirect dependency sequence (of one step) from ϕ in Γ_i to ψ in Γ_{i+1}. Then either ψ is a descendant of u in d_{i+1}, or ψ is older than u in d_{i+1}, or u does not exist in d_{i+1} due to the contraction of some proper ancestor v of u at point i of T.

5 Let i and j be points of T such that $i < j$. Let u be a minimal or auxiliary constant which is defined in d_i, is not deleted in T between points i and j, and is contracted in T at both point i and point j. Then there is some $\phi \in \Gamma_j$ such that u is active in ϕ relative to d_j. Furthermore, for every $\phi \in \Gamma_j$ for which u is active in ϕ relative to d_j, there is a formula $\psi \in \Gamma_i$ and a direct dependency sequence from ψ at point i to ϕ at point j such that u

is active in every element of the sequence, and at some point between i and j u occurs in the seqeunce and the $U\mu$ rule is applied to it.

6 Let i and j be points of T such that $i < j$. Let u be a minimal or auxiliary constant such that $u \in \Gamma_i$ and the $U\mu$ rule is applied to u at point i of T. If $u \in \Gamma_j$ and there is a direct or indirect dependency sequence from u at point i to u at point j such that u is active in every element of the sequence, then u is contracted somewhere between points i and j of T.

Proof: Straightforward. Notice that for claim 6 it is essential that among the children of any constant in a definition tree, all auxiliary constants are always older than all minimal or maximal constants, and that in the deletion rule the formula which is smaller in the $\prec_d$ order is the one which is preserved. □

Let us see that we can read a proper definition tree tableau from a proper bundled tableau, and vice versa. Since the proofs of the lemmas below are straightforward but tedious, they have been postponed to Appendix A.

Lemma 4.2.4 Let T be a definition tree tableau. The following statements are mutually equivalent:

- T is not proper,
- there is a bad direct dependency sequence in T, and
- there is a bad indirect dependency sequence in T.

Proof: The first condition implies the second by clauses 1 and 5 of Lemma 4.2.3 and by König's lemma. The second trivially implies the third. The third implies the first by clause 6 of Lemma 4.2.3. For full details, see Appendix A. □

Lemma 4.2.5 Let $T = (s_0, \Gamma_0, d_0) \ldots$ be a proper definition tree tableau, n a point of T and $\phi_0, \phi_1, \ldots$ an infinite direct or indirect dependency sequence from point n of T. Then there is some $m \geq 0$ and constant u such that

- u is active in ϕ_i, relative to d_{n+i} for every $i \geq m$, and
- $\phi_i = u$ for infinitely many $i \geq m$.

Proof: Given an infinite dependency sequence, we can inductively find the required u on the basis of clauses 2, 3 and 4 of Lemma 4.2.3. For full details, see Appendix A. □

Lemma 4.2.6 Let T be a proper and propositionally consistent bundled tableau for a guarded μTL-formula ϕ in pnf. There exists a proper and propositionally consistent definition tree tableau T' for ϕ such that for any model $M \in \mathcal{M}_{TL}$, if T agrees with M then T' agrees with M.

Proof: Reading a definition tree tableau T' from a bundled tableau T is straightforward. What is more, we can project any bad direct dependency sequence in T' back to T to a dependency sequence with infinitely many occurrences of some minimal fixpoint constant. Therefore by Lemma 4.2.4, T' is proper, if T is proper. For full details, see Appendix A. □

Lemma 4.2.7 Let T be a proper and propositionally consistent definition tree tableau for a guarded μTL-formula ϕ in pnf. There exists a proper and propositionally consistent bundled tableau T' for ϕ such that for any model $M \in \mathcal{M}_{TL}$, if T agrees with M then T' agrees with M.

Proof: Reading a bundled tableau T' from definition tree tableau T is straightforward. We can also project any dependency sequence of T' back to an indirect dependency sequence in T. Furthermore, by Lemma 4.2.5 we can do this in such a way that if the dependency sequence in T' has infinitely many occurrences of some minimal fixpoint constant, then the dependency sequence in T is bad. Therefore by Lemma 4.2.4, T' is proper, if T is proper. For full details, see Appendix A. □

So far we have showed that definition tree tableaux characterise satisfiability in the same sense as bundled tableaux.

Theorem 4.2.8 Let ϕ be a guarded μTL-formula in pnf. Then ϕ is satisfiable iff there is a proper and propositionally consistent definition tree tableau T for ϕ.

Proof: By Prop. 2.2.40 we know that ϕ is satisfiable iff there is a proper and propositionally consistent bundled tableau T for ϕ. The claim follows then immediately from Lemmas 4.2.6 and 4.2.7. □

4.3 Satisfiability tableaux

We are now ready to formulate a finitary tableau system for the satisfiability of μTL formulae. This is based on characterising a suitable termination condition, which specifies when we can stop extending a tableau. This condition is motivated by the following observation.

Lemma 4.3.1 Let $T = (s_0, \Gamma_0, d_0)(s_1, \Gamma_1, d_1)\ldots$ be a definition tree tableau. The tableau T is proper iff there is some point $n \in \mathbb{N}$ and a constant u defined in d_n such that

- neither u nor any constant defined before it in d_n is deleted or contracted anywhere in T after point n, and

- there are infinitely many points $i \geq n$ such that either u is the last constant defined in d_i, or the constant defined immediately after u in d_i is deleted at point i of T.

Proof: Straightforward. $\square$

Definition 4.3.2 Let ϕ be a guarded μTL-formula in pnf. A *satisfiability tableau* T for ϕ is a finite sequence $T = (\Gamma_0, d_0) \dots (\Gamma_n, d_n)$, where

- Γ_i is a finite set of extended μTL-formulae in pnf, and d_i is a definition tree,
- $(\Gamma_0, d_0) = (\{\phi\}, \epsilon)$,
- every (Γ_{i+1}, d_{i+1}) is derived from (Γ_i, d_i) by applying one of the rules in Table 4.2, using the same priority order as in Def. 4.2.1,
- point n of T is terminal, defined below, and
- for every $0 \leq k < n$, point k of T is not terminal.

We say that T is *proper* iff n is an accepting terminal, defined below. $\square$

Before being able to define the concept of a terminal point, we need the notion of a companion.

Definition 4.3.3 Let $T = (\Gamma_0, d_0) \dots (\Gamma_k, d_k)$ be a satisfiability tableau for a μTL-formula ϕ, and let m and n be points of T such that $0 \leq m < n \leq k$. We say that point m is a *companion* of point n in T iff $(\Gamma_m, d_m) = (\Gamma_n, d_n)$, and there is some constant u defined in d_m such that

0 for every constant v such that $v = u$ or v is defined before u in d_m, v is neither deleted nor contracted between points m and n of T

and either

1 u is the last constant defined in d_m, or

2 the constant v defined immediately after u in d_m is deleted at point m of T, or

3 the constant v defined immediately after u in d_m is contracted at point m, and v is not deleted at any point between m and n in T.

In cases 1 and 2 we say that m is an *accepting* companion of n, and in case 3 that m is a *rejecting* companion. $\square$

Definition 4.3.4 Let $T = (\Gamma_0, d_0) \dots (\Gamma_k, d_k)$ be a satisfiability tableau for a μTL-formula ϕ, and let $0 \leq n \leq k$ be a point of T. We say that the point n is *terminal* iff either

- $\bot \in \Gamma_n$, or
- there is some ϕ such that $\phi \in \Gamma_n$ and $\neg\phi \in \Gamma_n$, or

name	application
del-$\boldsymbol{\phi}$	$\dfrac{\Gamma \cup \{\phi, \phi'\}, \quad d}{\Gamma \cup \{\phi\}, \qquad d}$ where ϕ and ϕ' are similar relative to d and $\phi \prec_d \phi'$
del-$\boldsymbol{U}$	$\dfrac{\Gamma, \quad d}{\Gamma, \quad d \setminus \{u\}}$ where u does not occur in any $\phi \in \Gamma$, and u is a leaf of d
contr-$\boldsymbol{U}$	$\dfrac{\Gamma, \qquad d}{\Gamma[u/U], \quad (d \setminus U)[u/U]}$ where u does not occur in any $\phi \in \Gamma$, every child of u in d is an unfolding of u, and U is the set of unfoldings of u in d
$\vee\boldsymbol{L}$	$\dfrac{\Gamma \cup \{\phi \vee \phi'\}, \quad d}{\Gamma \cup \{\phi\}, \qquad d}$
$\vee\boldsymbol{R}$	$\dfrac{\Gamma \cup \{\phi \vee \phi'\}, \quad d}{\Gamma \cup \{\phi'\}, \qquad d}$
$\wedge$	$\dfrac{\Gamma \cup \{\phi \wedge \phi'\}, \quad d}{\Gamma \cup \{\phi, \phi'\}, \qquad d}$
$\boldsymbol{\sigma}$	$\dfrac{\Gamma \cup \{\sigma z.\phi\}, \quad d}{\Gamma \cup \{u\}, \qquad d \cdot (u, \sigma z.\phi)}$ where u does not appear in d
$\boldsymbol{U\nu}$	$\dfrac{\Gamma \cup \{u\}, \qquad d}{\Gamma \cup \{\phi[u/z]\}, \quad d}$ where $d^*(u) = \nu z.\phi$
$\boldsymbol{U\mu}$	$\dfrac{\Gamma \cup \{u\}, \qquad d}{\Gamma \cup \{\phi[u'/z]\}, \quad d \cdot (u', u)}$ where $d^*(u) = \mu z.\phi$ and u' does not appear in d
$\bigcirc$	$\dfrac{\{z_1, \ldots, z_m, \bigcirc\phi_1, \ldots, \bigcirc\phi_k\}, \quad d}{\{\phi_1, \ldots, \phi_k\}, \qquad d}$ where every z_i is atomic
Note:	In each rule, Γ is disjoint from the other set

Table 4.2: Satisfiability tableau rules for μTL

- there is some earlier point $m < n$ of T such that m is a rejecting companion of n in T, or
- there is some earlier point $m < n$ of T such that m is an accepting companion of n in T.

In the first three cases we say that n is a *rejecting terminal* and in the last case that n is an *accepting terminal* in T. □

Before showing the correctness and completeness of the tableau system, let us state one more technical lemma.

Lemma 4.3.5 Let $T = (\Gamma_0, d_0) \dots (\Gamma_k, d_k)$ be a structure which is like a satisfiability tableau, except without the requirement that only the last point may be terminal. Let m and n be points of T such that m is an accepting companion of n, and let m' and n' be points of T such that m' is a rejecting companion of n'. Then $m \neq m'$, $n \neq n'$, and neither $m' < m < n' < n$ nor $m < m' < n < n'$ can hold.

Proof: By definition there is a constant u such that neither u nor any constant defined before it in d_m is deleted or contracted between points m and n, and u is either the last constant defined in d_m or the constant v defined immediately after u in d_m is deleted at point m of T. Similarly, there is a constant u' such that neither u' nor any constant defined before it in $d_{m'}$ is deleted or contracted between points m' and n', and the constant v' defined immediately after u' in $d_{m'}$ is contracted at point m', and is not deleted at any point between m' and n'.

Since v' is contracted at point m', and no constant is contracted at point m, we have $m \neq m'$. Furthermore, $(\Gamma_m, d_m) \neq (\Gamma_{m'}, d_{m'})$, since otherwise v' would have to be contracted at point m. Since $(\Gamma_n, d_n) = (\Gamma_m, d_m)$ and $(\Gamma_{n'}, d_{n'}) = (\Gamma_{m'}, d_{m'})$, this implies $n \neq n'$.

Assume then that either $m' < m < n' < n$ or $m < m' < n < n'$ would hold. Consider the former case first. Since neither v' nor any constant defined before it in $d_{m'}$ is deleted at point m, v' must either be u or defined before u in d_m. Notice then that since v' is contracted at point m', it is also contracted at point n'. but then v' cannot be u or defined before u in d_m, which is a contradiction.

Assume then that $m < m' < n < n'$. Since neither u nor any constant defined before it in d_m is contracted or deleted at point m', v' must be defined after u in $d_{m'}$. However, since u is either the last constant defined in d_n or the constant defined immediately after it is deleted at n, v' must be u or defined before u in d_n, again a contradiction. □

Theorem 4.3.6 Let ϕ be a guarded μTL-formula in pnf. Then ϕ is satisfiable iff there is a proper satisfiability tableau T for ϕ.

Proof: By Theorem 4.2.8 we know that ϕ is satisfiable iff there is a proper and propositionally consistent definition tree tableau for ϕ.

Assume first there is a proper satisfiability tableau T for ϕ. By unwinding T to an infinite sequence, it is easy to construct a propositionally consistent definition tree tableau T' for ϕ. Lemma 4.3.1 implies that this T' is proper. Consequently ϕ is satisfiable.

Assume then that ϕ is satisfiable, and that there is a proper and propositionally consistent definition tree tableau $T = (s_0, \Gamma_0, d_0)(s_1, \Gamma_1, d_1)\ldots$. By clause 2 of Lemma 4.2.3, there is a bound k such that $|d_i| \leq k$ for all $i \in \mathbb{N}$. This means that by reusing constants we can assume without loss of generality that there are only k different constant names in T, which means that there are only finitely many different definition lists in T. Since T is proper, by Lemma 4.3.1 there are then points m and n of T and a constant u defined in d_m such that $m < n$, $\Gamma_m = \Gamma_n$, $d_m = d_n$ and neither u nor any constant defined before it in d_m is deleted or contracted between points m and n of T, and either u is the last constant defined in d_m or the constant v defined immediately aften u in d_m is deleted at point m of T.

Consider now the sequence $T' = (\Gamma_0, d_0)\ldots(\Gamma_n, d_n)$. By the choice of m and n, we know that m is an accepting companion of n in T', i.e. that n is an accepting terminal. If no $n' < n$ is a terminal, T' is then an accepting satisfiability tableau. Otherwise, let us see that we can remove segments from T' until no point except the last is terminal.

Assume then that some $n' < n$ is also a terminal, i.e. that there is some m' such that m' is a companion of n' in T'. If m' is an accepting companion, define $T'' = (\Gamma_0, d_0)\ldots(\Gamma_{n'}.d_{n'})$. By assumption the last element of this sequence is an accepting terminal. If m' is a rejecting companion of n', by Lemma 4.3.5 we know that it is not possible that $m < m' < n < n'$ or $m' < m < n' < n$. Defining then $T'' = (\Gamma_0, d_0)\ldots(\Gamma_{m'-1}, d_{m'-1})(\Gamma_{n'}, d_{n'})\ldots(\Gamma_n, d_n)$, it is easy to see that the last element of T'' is still an accepting terminal. In both cases $|T''| < |T'|$. Therefore, after chopping off a finite number of segments from T', we obtain a sequence where the last element is an accepting terminal and no other element is a terminal, i.e. a proper satisfiability tableau for ϕ. □

Let us see then that we can derive an exponential bound for the size of tableaux that need to be considered for the satisfiability of a given formula ϕ. This gives us immediately an elementary decision procedure.

Proposition 4.3.7 There exists a polynomial p such that for any μTL-formula ϕ, at most $p(|\phi|)$ distinct constant names are needed in constructing a satisfiability tableau for ϕ. Furthermore, there exists a polynomial p' such that for every μTL-formula ϕ and every satisfiability tableau T for ϕ, if T uses only $p(|\phi|)$ distinct constant names, then the length of T is at most $2^{p'(|\phi|)}$.

Proof: For the first claim, take any satisfiability tableau $T = (\Gamma_0, d_0) \ldots (\Gamma_n, d_n)$ for ϕ. Notice that if we identify similar formulae, there are at most $|\phi|$ different formulae that can occur in T. Take any point i of T where a new constant is added to d_i by the rule applied at point i. Since then neither del-ϕ nor del-U can be applicable at that point, $|\Gamma_i| \leq |\phi|$ and d_i has at most ϕ leaves. Furthermore, since contr-U cannot be applicable at point i either, we can see that the depth of d_i must be less than $2 \cdot |\phi|$, implying that $|d_i| < 2 \cdot |\phi|^2$. Consequently, if we reuse constants, we need no more than $2 \cdot |\phi|^2$ different constant names when building a satisfiability tableau for ϕ.

Let us then derive a bound on the number of different pairs (Γ_i, d_i) that may occur in T, if T uses at most $|\phi|^2$ constant names. Define $k = |\phi|^2$. Since up to similarity there are at most $|\phi|$ different formulae that may occur in T, each of these formulae has at most $|\phi|$ occurrences of constants, and there are k choices for each constant, there are at most $|\phi| \cdot k^{|\phi|}$ syntactically different formulae that may occur in T. Since $|\Gamma_i| \leq |\phi|$ for every i, there are at most $(|\phi| \cdot k^{|\phi|})^{|\phi|} \leq 2^{2 \cdot k \cdot \log k}$ different choices for the set Γ_i. For each element (u_j, ϕ_j) of d_i, there are at most k different choices for u_j and $|\phi| \cdot k^{|\phi|}$ different choices for ϕ_j. Since $|d_i| \leq k$, there are then at most $(k \cdot |\phi| \cdot k^{|\phi|})^k \leq 2^{2 \cdot |\phi| \cdot k \cdot \log k}$ different choices for d_i. Consequently, there are at most $2^{2 \cdot k \cdot \log k + 2 \cdot |\phi| \cdot k \cdot \log k} \leq 2^{4 \cdot k^2}$ different choices for the pair (Γ_i, d_i). Define then $x = 2^{4 \cdot k^2}$.

Let us denote for every i the definition tree d_i by $d_i = (u_{i0}, \phi_{i0}) \ldots (u_{in_i}, \phi_{in_i})$. We shall show next by descending induction on h that for every $0 \leq h \leq k$ the following holds:

> For any points $0 \leq m \leq m' \leq n$, if no point i between m and m' is terminal and no u_{ij} is either deleted or contracted at any point $m \leq i \leq m'$ for any $0 \leq j < h$, then $m' - m \leq x^{2 \cdot (k-h)+1}$.

From case $h = 0$ it follows that $n \leq x^{2 \cdot k+1} = 2^{4 \cdot k^2 \cdot (2 \cdot k+1)} \leq 2^{12 \cdot k^3} = 2^{12 \cdot |\phi|^6}$, establishing the required bound on the size of tableaux for ϕ.

The base case for the induction is $h = k$. As the analysis above shows that there are at most x different pairs (Γ_i, d_i), the induction claim holds in this case, since otherwise some point $m \leq i \leq m'$ would satisfy clause 1 of Def. 4.3.3 and be terminal, contrary to assumption. Assume then that the induction claim holds

name	application
$\bigcirc$	$\dfrac{\{z_1, \ldots, z_m, \textcircled{\scriptsize i_1}\phi_1, \ldots, \textcircled{\scriptsize i_k}\phi_k\},\quad d}{\Gamma_0,\ d \qquad \Gamma_1,\ d \qquad \ldots \qquad \Gamma_{n-1},\ d}$ **1**
Note:	**1:** every z_j is atomic, and for every $i \in [n]$, $\Gamma_i = \{\phi_j \mid i_j = i\}$.

Table 4.3: Satisfiability tableau rule for branching structures

for some $0 < h \leq k$. Take any points $0 \leq m \leq m' \leq n$ such that no $m \leq i \leq m'$ is terminal and no u_{ij} is deleted or contracted at any point $m \leq i \leq m'$ for any $0 \leq j < h-1$. There can be at most x points $m \leq i \leq m'$ such that $u_{i(h-1)}$ is deleted at point i, since otherwise clause 2 of Def. 4.3.3 would be satisfied. Between any two such points there can be at most x points i such that $u_{i(h-1)}$ is contracted at point i, since otherwise clause 3 of Def. 4.3.3 would be satisfied. By induction assumption the distance between any two of these points is at most $x^{2\cdot(k-h)+1}$, which implies that $m' - m \leq x \cdot x \cdot x^{2\cdot(k-h)+1} = x^{2\cdot(k-(h-1))+1}$ and the induction claim holds for $h-1$. $\square$

4.4 Extension to branching structures

Although above we formulated the satisfiability tableau system only for the linear time mu-calculus, the proofs do not essentially depend on the linearity of the structures. Therefore it is easy to generalise the tableaux also for the modal mu-calculus.

Definition 4.4.1 Fix some $n \in \mathbb{N}$, and let ϕ be a guarded μKn-formula in pnf. A *satisfiability tableau* T for ϕ is a finite tree T, every node t of which is labelled with a pair (Γ_t, d_t) where

- every Γ_t is a finite set of extended μKn-formulae in pnf, and d_t is a definition tree,
- the root of T is labelled with $(\{\phi\}, \epsilon)$,
- for every interior node t of T, either t has exactly one child which is derived by applying one of the rules del-ϕ, del-U, contr-U, $\vee L$, $\vee R$, $\wedge$, σ, $U\nu$ or $U\mu$ in Table 4.2, or exactly n children which are derived by the rule $\bigcirc$ in Table 4.3, and where the priority order of the rules is as in Def. 4.2.1,
- every leaf t of T is terminal, defined below, and
- no interior node of T is a terminal.

We say that T is *proper* iff every leaf t of T is an accepting terminal, defined below. □

Definition 4.4.2 Let T be a satisfiability tableau for a μKn-formula ϕ, and let t and t' be nodes of T such that t' is a proper ancestor of t. We say that t' is a *companion* of t in T iff $(\Gamma_{t'}, d_{t'}) = (\Gamma_t, d_t)$, and there is some constant u defined in $d_{t'}$ such that

0 for every constant v such that $v = u$ or v is defined before u in $d_{t'}$, v is neither deleted nor contracted between t' and t in T

and either

1 u is the last constant defined in $d_{t'}$, or

2 the constant v defined immediately after u in $d_{t'}$ is deleted at node t' of T, or

3 the constant v defined immediately after u in $d_{t'}$ is contracted at node t', and v is not deleted between t' and t in T.

In cases 1 and 2 we say that t' is an *accepting* companion of t, and in case 3 that m is a *rejecting* companion. □

Definition 4.4.3 Let T be a satisfiability tableau for a μKn-formula ϕ, and let t be a node of T. We say that t is *terminal* iff either

- $\bot \in \Gamma_t$, or
- there is some ϕ such that $\phi \in \Gamma_t$ and $\neg\phi \in \Gamma_t$, or
- there is some proper ancestor t' of t such that t' is a rejecting companion of t in T, or
- there is some proper ancestor t' of t such that t' is an accepting companion of t in T.

In the first three cases we say that t is a *rejecting terminal* and in the last case that t is an *accepting terminal* in T. □

Theorem 4.4.4 Let ϕ be a guarded μKn-formula in pnf. Then ϕ is satisfiable iff there is a proper satisfiability tableau T for ϕ.

Proof: The correctness and completeness of the tableau system for the branching case is shown using the same techniques as for the linear case. First, it is easy to extend the concept of a definition tree tableau to the branching case and show analogously to Theorem 4.2.8 that a formula is satisfiable iff there is a proper and propositionally consistent definition tree tableau for it.

Assuming that there is a proper satisfiability tableau T for ϕ, is is easy to unwind T to an infinite definition tree tableau which is proper and propositionally consistent, implying the satisfiability of ϕ.

In the other direction, take an infinite proper definition tree tableau T for ϕ. By cutting off the rest of the tree T in any path as soon as an accepting terminal has been reached, we can read from T a finite structure T' which is like a proper satisfiability tableau, except that some interior nodes of T' may still be terminals. As in the proof of Theorem 4.3.6, we can then chop off pieces from T' so that all its leaves stay accepting, until no interior node of the tree remains terminal. □

Proposition 4.4.5 There exists a polynomial p such that for any μKn-formula ϕ, at most $p(|\phi|)$ distinct constant names are needed in constructing a satisfiability tableau for ϕ. Furthermore, there exists a polynomial p' such that for every μKn-formula ϕ and every satisfiability tableau T for ϕ, if T uses only $p(|\phi|)$ distinct constant names, then the length of any branch of T is at most $2^{p'(|\phi|)}$.

Proof: Analogous to Prop. 4.3.7 □

4.5 Discussion

In the sections above, we have described a tableau system for deciding the satisfiability of μTL and μKn-formulae, yielding an elementary decision procedure. What is more, we can use the same tableau system to transform mu-calculus formulae directly to the strongly aconjunctive form.

Theorem 4.5.1 For every μTL-formula ϕ, there is an equivalent strongly aconjunctive μTL-formula ψ. The same holds for μKn, as well.

Proof: Take any μTL-formula ϕ. By Prop. 2.2.36 we can assume without loss of generality that ϕ is guarded and in pnf. Construct then a finite tree T labelled with pairs (Γ, d) as in satisfiability tableaux. The tree T is defined inductively as follows.

- The root of T is labelled with $(\{\phi\}, \epsilon)$,
- For any interior node t of T with label (Γ, d), either
  - t has one child t' labelled with (Γ', d'), where (Γ', d') is derived from (Γ, d) by any satisfiability tableau rule (Table 4.2) except $\vee L$ or $\vee R$, or
  - t has two children t' and t'', labelled with (Γ', d') and (Γ'', d'') respectively, where (Γ', d') is derived from (Γ, d) by applying $\vee L$ to some formula $\phi \in \Gamma$, and (Γ'', d'') is derived from (Γ, d) by applying $\vee R$ to the same formula ϕ, and
- node t is a leaf of T iff t is terminal in the sense of Def. 4.4.3. (Notice that the definition makes sense, although T is strictly speaking not a satisfiability tableau.)

By reusing constants whenever possible, we can guarantee that this T is indeed finite.

Let us then define the formula ψ on the basis of the tree T. Fix first a separate variable $z(t)$ for every node t of T. Define then $\psi = \psi(\epsilon)$, where for every node t of T, $\psi(t)$ is defined inductively by:

$$\psi(t) \quad = \quad \begin{cases} \nu z(t).\phi(t) & \text{if } t \text{ is an accepting companion for some } t' \\ \mu z(t).\phi(t) & \text{if } t \text{ is a rejecting companion for some } t' \\ \phi(t) & \text{otherwise} \end{cases}$$

where

- $\phi(t) = \psi(t') \vee \psi(t'')$, if t is not a leaf and the children t' and t'' of t were derived by the $\vee L$ and $\vee R$ rules,
- $\phi(t) = (\bigwedge\{z \in \Gamma_t \mid z \text{ is atomic }\}) \wedge \bigcirc\psi(t')$, if t is not a leaf and the child t' of t was derived by the $\bigcirc$ rule,
- $\phi(t) = \psi(t')$, if t is not a leaf and the child t' of t was derived by some other rule,
- $\phi(t) = z(t')$, if t is a leaf and there is a proper ancestor t' of t such that t' is a companion of t, and
- $\phi(t) = \bot$, if t is a leaf and it has no companions. Notice that in this case either $\bot \in \Gamma_t$ or $\{\phi, \neg\phi\} \subseteq \Gamma_t$ for some ϕ.

Remembering Lemma 4.3.1 and the definition of a companion, it is easy to see that for any model M, there is a proper bundled tableau for ψ agreeing with M iff there is a proper definition tree tableau for ϕ agreeing with M. In other words, $\models \phi \Leftrightarrow \psi$. The generalisation of the construction to μKn is obvious. □

We already showed this result in the previous chapter, using the inductive translation of mu-calculus formulae to ordinary Rabin automata. However, the current translation which uses the satisfiability tableau system has the added elegance that it does not require the introduction of auxiliary formalisms like Rabin automata. In fact, it allows us to translate *SnS* or $\exists Kn$ to the strongly aconjunctive fragment of μKn, thereby showing the decidability of these formalisms, without ever explicitly invoking the notion of automata at all. To see this, notice that the inductive translation of Theorem 3.4.38 from $\exists Kn$ to μKn requires only the ability to transform any μKn-formula to the strongly aconjunctive form. This approach to the decidability of *SnS* effectively continues the programme of Emerson and Jutla [33, 46]. However, while their work still used explicit automata constructions, the tableau system here allows a direct proof.

Chapter 5

Axiomatising linear time mu-calculus

In this chapter we present a solution to the previously open problem of providing a complete axiomatisation for the linear-time mu-calculus μTL, and see how we can take advantage of various constructions and semantic results above in proving the completeness of the axiomatisation.

Although μTL is syntactically concise and straightforward, the problem of axiomatising it has turned out to be rather intricate. The main culprit for this is the minimal fixpoint operator μ, or more exactly, the prevention of infinite regeneration of minimal fixpoints when trying to build a model for a consistent formula. Previously the axiomatisation of μTL has been addressed by at least Lichtenstein [59] and Dam [22]. The closely related question of axiomatising the modal mu-calculus, has been examined by Kozen [55] and Walukiewicz [100, 101].

Generalising, there have been two approaches to showing the satisfiability of a consistent formula, the essential problem of the completeness proof of an axiomatisation. First, one may try to devise a method of constructing a model directly from a given consistent formula. In this line, Kozen [55] introduced the concept of aconjunctivity, to make it easier to build a model of a consistent formula, and showed the completeness of an axiomatisation of the modal mu-calculus restricted to the aconjunctive fragment of the language. The same approach was pursued by Lichtenstein in [59] to show the completeness of an axiomatisation of μTL restricted to a class of aconjunctive formulae.

Another approach, and the one adopted here, is using a normal form and showing that any formula can be provably transformed to this normal form. If we know how to build a model for a consistent formula in this form, the satisfiability of any consistent formula has been shown. In the context of *S1S*, this approach was already used early by Siefkes [81], and for the linear-time mu-calculus μTL, Dam [22] used Büchi automata -like normal forms to show the completeness of an axiomatisation of μTL containing an 'impure' axiom stating that a formula and its normal form are equivalent.

The completeness proof of the axiomatisation below is based on transforming formulae provably to a normal form. The crucial property of the normal form used here, the *bi-aconjunctive non-alternating form*, is that not only is it easy to construct a model of a consistent formula in this form, but the same holds also of its negation. In our opinion, the remarkable thing about the normal form and the completeness proof here is that the semantic equivalence between the full μTL and the normal form can be lifted to the level of provability rather elegantly on the basis of what is already known about aconjunctivity.

5.1 Axiomatisation

This far we have operated purely on the semantic level in the current work. Let us define now an axiomatic system for the linear time mu-calculus μTL. This is essentially the same as the one used by Kozen in [55] and Lichtenstein in [59].

Definition 5.1.1 We say that a μTL-formula ϕ is *provable* and write $\vdash \phi$, iff it is derivable in the following deductive system.
Axiom schemas:

- **ax1** All propositional tautologies
- **ax2** $\bigcirc(\phi \Rightarrow \psi) \Rightarrow (\bigcirc\phi \Rightarrow \bigcirc\psi)$
- **ax3** $\bigcirc\phi \Leftrightarrow \neg\bigcirc\neg\phi$
- **ax4** $\phi[\mu z.\phi/z] \Rightarrow \mu z.\phi$

Rules of inference:

modus ponens:	from ϕ and $\phi \Rightarrow \psi$ infer ψ
necessitation:	from ϕ infer $\bigcirc\phi$
fixpoint induction:	from $\phi[\psi/z] \Rightarrow \psi$ infer $\mu z.\phi \Rightarrow \psi$

We say that a formula ϕ is *consistent* iff not $\vdash \neg\phi$. □

Showing the soundness of this axiom system is easy.

Theorem 5.1.2 [**Soundness for** μTL] For any μTL-formula ϕ, if $\vdash \phi$ then $\models \phi$.

Proof: All instances of the axiom schemas are clearly universally valid, and the modus ponens and necessitation rules validity-preserving. To see that also the fixpoint induction rule preserves universal validity, assume that $\not\models \mu z.\phi \Rightarrow \psi$. As by the Knaster-Tarski fixpoint theorem 2.2.16, $\mu z.\phi = \bigvee_\alpha \mu^\alpha z.\phi$, there is an α such that $\not\models \mu^\alpha z.\phi \Rightarrow \psi$ but $\models \mu^{\alpha'} z.\phi \Rightarrow \psi$ for all $\alpha' \prec \alpha$. This α cannot be 0 or a limit ordinal. Consequently, there are M and s such that $M, s \models \mu^\alpha z.\phi \wedge \neg\psi$, and by definition of μ^α, $M, s \models \phi[\mu^{\alpha-1} z.\phi/z] \wedge \neg\psi$. But as $\models \mu^{\alpha-1} z.\phi \Rightarrow \psi$ and z occurs only positively in ϕ, $\models \phi[\mu^{\alpha-1} z.\phi/z] \Rightarrow \phi[\psi/z]$, implying $M, s \models \phi[\psi/z] \wedge \neg\psi$, i.e. $\not\models \phi[\psi/z] \Rightarrow \psi$. □

5.2 Normal form

The completeness proof for the axiomatisation of μTL is based on transforming all formulae to a certain normal form. We define next this bi-aconjunctive non-alternating form, and show that the fragment of μTL consisting of formulae in this normal form has the same expressive power as the whole μTL. Let us first introduce a useful variant of aconjunctivity.

Definition 5.2.1 Let ϕ be a μTL or μKn-formula in pnf, and z a variable which is bound by a unique fixpoint in ϕ and occurs only bound in ϕ. We say that ϕ is *strongly aconjunctive relative to* z iff for all subformulae of ϕ of the form $\phi_0 \wedge \ldots \wedge \phi_{m-1}$,

A for every $i \in [m]$, either

1 z is not active in ϕ_i, in the context of ϕ, or

2 z is active in ϕ_i and $\phi_i = \circled{k}\psi$ for some $k \in [n]$ and ψ, and

B for every $i, j \in [m]$ such that $i \neq j$ and both ϕ_i and ϕ_j fulfil condition 2 above, if $\phi_i = \circled{k}\psi$ and $\phi_j = \circled{l}\psi'$, then $k \neq l$.

Let ϕ be a μTL or μKn formula in pnf. We say that ϕ is *μ-aconjunctive* iff ϕ is strongly aconjunctive relative to every z bound by a minimal fixpoint in ϕ. An arbitrary formula ϕ is μ-aconjunctive iff pnf(ϕ) is. □

For μTL-formulae the definition of μ-aconjunctivity implies that if ϕ is μ-aconjunctive, then there is no subformula $\mu z.\phi'$ of ϕ and subformula $\phi_1 \wedge \phi_2$ of ϕ' such that z is active in both ϕ_1 and ϕ_2. In other words, for μTL the definition of μ-aconjunctivity here implies aconjunctivity in the sense of Kozen [55].

The new concept of bi-aconjunctivity requires not only that a formula itself is μ-aconjunctive, but also that its dual is, as well.

Definition 5.2.2 A formula ϕ is *bi-aconjunctive* iff ϕ and $\neg\phi$ are μ-aconjunctive. □

We can now define the normal form that forms the basis of the completeness proof in next section.

Definition 5.2.3 We say that a μTL-formula ϕ is in the *bi-aconjunctive non-alternating normal form* (abbreviated *banan-form*) iff

- ϕ is guarded,
- ϕ is bi-aconjunctive, and
- ϕ is non-alternating, i.e. $\phi \in \Delta_2$. □

Notice some easy properties of formulae in the normal form.

Lemma 5.2.4 Let ϕ, ϕ' be formulae in banan-form. Then $\neg\phi$, $\bigcirc\phi$, $\phi \wedge \phi'$ and $\phi[\phi'/z]$ are in banan-form.

Proof: Straightforward. □

Let us then show that the whole language μTL is semantically equiexpressive with its fragment of formulae in banan-form. This is rather easy on the basis of the observations we made about the expressivity of various types of μTL-formulae when relating the mu-calculi and automata to each other. Remember first that in Section 3.5 we noticed that the full μTL is equiexpressive with its non-alternating fragment Δ_2. Therefore it suffices to show that every non-alternating formula can be pushed to the banan-form.

Lemma 5.2.5 For any non-alternating μTL-formula ϕ, there exists a μTL-formula ϕ' in banan-form such that $\models \phi \Leftrightarrow \phi'$

Proof: By Prop. 2.2.36 we can assume without loss of generality that ϕ is guarded and in pnf. The claim is done by induction on the syntactic fixpoint alternation classes (see Def. 2.2.21)
Induction basis: If $\phi \in \Pi_0^{stx} = \Gamma_0^{stx}$, choosing $\phi' = \phi$ fulfils the claim.
Induction step: Take any $\phi \in \Delta_2$ such that $\phi \in \Sigma_{n+1}^{stx}$. This means that ϕ can be written as $\phi = \psi[\phi_1/z_1, \ldots, \phi_m/z_m]$ for some $\psi \in \Sigma_1$ and $\phi_1, \ldots, \phi_m$ in $(\Sigma_n^{stx} \cup \Pi_n^{stx}) \cap \Delta_2$, By induction assumption, there are formulae $\phi_1', \ldots, \phi_m'$ in banan-form such that $\models \phi_i \Leftrightarrow \phi_i'$ for all $1 \leq i \leq m$. Since $\psi \in \Sigma_1$, i.e. ψ does not contain any maximal fixpoints, by Lemma 3.3.3 we know that there exists a restricted formula $\psi' \in \Sigma_1$ such that $\models \psi \Leftrightarrow \psi'$. It is clear that such a ψ' is also in banan-form. By Lemma 5.2.4, $\psi'[\phi_1'/z_1, \ldots, \phi_m'/z_m]$ is in banan-form. Since

$$\models \psi[\phi_1/z_1, \ldots, \phi_m/z_m] \Leftrightarrow \psi'[\phi_1'/z_1, \ldots, \phi_m'/z_m]$$

choosing $\phi' = \psi'[\phi_1'/z_1, \ldots, \phi_m'/z_m]$ fulfils the claim for ϕ. Consequently, the claim holds for the class Σ_{n+1}^{stx}.

Take then any $\phi \in \Delta_2$ such that $\phi \in \Pi_{n+1}^{stx}$, and define $\phi' = \text{pnf}(\neg\phi)$. It is easy to see that $\phi' \in \Delta_2$ and $\phi' \in \Sigma_{n+1}^{stx}$. By the above there is a ϕ'' in banan-form such that $\models \phi' \Leftrightarrow \phi''$, implying $\models \phi \Leftrightarrow \neg\phi' \Leftrightarrow \neg\phi''$. Furthermore, by Lemma 5.2.4, $\neg\phi''$ is in banan-form, so the claim holds for ϕ, and therefore for the class Π^{stx}_{n+1}. □

Now the expressive equivalence of the whole μTL and the fragment of formulae in banan-form is immediate.

Lemma 5.2.6 For any μTL-formula ϕ, there exists a μTL-formula ϕ' in banan-form such that $\models \phi \Leftrightarrow \phi'$.

Proof: Take any μTL-formula ϕ. By Corollary 3.5.3 there exists an equivalent non-alternating formula, and by Lemma 5.2.5 an equivalent formula in banan-form. □

It needs to be pointed out that this lemma does not imply that for every ϕ there is a ϕ' in banan-form such that $\models \sigma z.\phi \Leftrightarrow \sigma z.\phi'$: although $\sigma z.\phi$ would be well-defined, i.e. although z would occur only positively in ϕ, this does not necessarily hold of ϕ'.

5.3 Completeness

For the completeness proof, let us state a technical lemma first.

Lemma 5.3.1 [**Substitution**] For any μTL-formulae ϕ and ψ, if $\vdash \phi$, then $\vdash \phi[\psi/z]$.

Proof: Induction on the length of the proof of $\vdash \phi$. □

Let us show first that the axiomatisation is complete in the class of all μ-aconjunctive formulae. This follows easily from Kozen's results for the modal mu-calculus [55]. However, as the result is extended slightly in Lemma 5.3.10, a proof of it is sketched down here, as well. The formulation of the proof presented here is due to Stirling.

Definition 5.3.2 A bundled tableau $T = (i_0, \Gamma_0, d_0)(i_1, \Gamma_1, d_1)\ldots$ for a μTL-formula ϕ is *consistent* iff $\bigwedge \Gamma_j[d_j]$ is consistent for every $j \in \mathbb{N}$, i.e. iff not $\vdash \neg \bigwedge \Gamma_j[d_j]$ for any $j \in \mathbb{N}$. □

Lemma 5.3.3 Let ϕ be a guarded μTL-formula in pnf. If there is a proper bundled tableau T for ϕ such that T is consistent, then ϕ is satisfiable.

Proof: It is clear that if a bundled tableau T is consistent (in the sense of Def. 5.3.2) then it is also propositionally consistent (in the sense of Def. 2.2.38). The claim follows then immediately from Prop. 2.2.40. □

name	application	name	application
$\boldsymbol{\sigma'}$	$\dfrac{i,\quad \Gamma \cup \{\sigma z.\phi\},\quad d,\quad d^s}{i,\quad \Gamma \cup \{u\},\quad d',\quad d^{s'}}$ **1**	$\boldsymbol{U\nu}$	$\dfrac{i,\quad \Gamma \cup \{u\},\quad d,\quad d^s}{i,\quad \Gamma \cup \{\phi[u/z]\},\quad d,\quad d^s}$ **2**
$\boldsymbol{U\mu}$	$\dfrac{i,\quad \Gamma \cup \{u\},\quad d,\quad d^s}{i,\quad \Gamma \cup \{\phi[u/z]\},\quad d,\quad d^{s'}}$ **3**		
Note:	**1**: u does not appear in d, $d' = d \cdot (u, \sigma z.\phi)$, $d'_s = d_s \cdot (u, \sigma z.\phi)$. **2**: $d(u) = \nu z.\phi$ **3**: if $d = (u_1, \sigma z_1.\phi_1) \dots (u_n, \sigma z_n.\phi_n)$, $u = u_m$, $d(u_m) = \mu z.\phi$ and $d^s(u_m) = \mu z.(\phi \wedge \alpha)$, then $d^{s\prime}(u_i) = d^s(u_i)$ for $1 \leq i < m$, $d^{s\prime}(u_i) = d(u_i)$ for $m < i \leq n$, and $d^{s\prime}(u_m) = \mu z.(\phi \wedge \alpha \wedge \neg \bigwedge \Gamma[d^x])$ where $d^x(u_i) = d^s(u_i)$ for $1 \leq i \leq m$, $d^x(u_i) = d(u_i)$ for $m < i \leq n$		

Table 5.1: Strong tableau rules for μTL

If we look at the bundled tableau rules (Table 2.2 on page 37), it is easy to see that for any tableau element (i, Γ, d), if $\bigwedge \Gamma[d]$ is consistent, then some tableau rule can be applied to (i, Γ, d) to yield an element (i', Γ', d') so that $\bigwedge \Gamma'[d']$ is consistent. Therefore, it is easy to construct a consistent bundled tableau T for any consistent formula ϕ. However, there is nothing in this construction to guarantee that the resulting T would be proper as well as consistent. To this purpose we use a technique similar to Kozen's [55] for strengthening minimal fixpoints, based on the following lemma.

Lemma 5.3.4 Let ϕ and ψ be μTL-formulae and z a variable which does not occur free in ψ. If

$$\psi \wedge \mu z.\phi \text{ is consistent,}$$

then

$$\psi \wedge \phi[\mu z.(\phi \wedge \neg\psi)/z] \text{ is consistent.}$$

Proof: If $\phi[\mu z.(\phi \wedge \neg\psi)/z] \wedge \psi$ is inconsistent, $\vdash \phi[\mu z.(\phi \wedge \neg\psi)/z] \Rightarrow \neg\psi$, hence $\vdash \phi[\mu z.(\phi \wedge \neg\psi)/z] \Rightarrow \mu z.(\phi \wedge \neg\psi)$. By fixpoint induction rule we have then $\vdash \mu z.\phi \Rightarrow \mu z.(\phi \wedge \neg\psi)$, implying $\vdash \mu z.\phi \Rightarrow \neg\psi$, i.e. $\mu z.\phi \wedge \psi$ is inconsistent. $\square$

Definition 5.3.5 Let ϕ be a μTL-formula in pnf. A *strong tableau* T for ϕ is an infinite sequence $T = (i_0, \Gamma_0, d_0, d^s_0)(i_1, \Gamma_1, d_1, d^s_1) \dots$ where

- every (i_j, Γ_j, d_j) is as in Def. 2.2.37, and d^s_j is a definition list such that if $d_j = (u_1, \sigma z_1.\phi_1) \dots (u_n, \sigma z_n.\phi_n)$, $d^s_j = (u_1, \sigma z_1.\phi_1 \wedge \alpha_1) \dots (u_n, \sigma z_n.\phi_n \wedge \alpha_n)$ for some formulae α_i (possibly $\top$),

- every $(i_{j+1}, \Gamma_{j+1}, d_{j+1}, d^s_{j+1})$ is derived from $(i_j, \Gamma_j, d_j, d^s_j)$ by one of the rules $\vee L$, $\vee R$, $\wedge$ or $\bigcirc$, which are as in Table 2.2 (page 37) or by σ', $U\nu$ or $U\mu$ in Table 5.1, and
- $(i_0, \Gamma_0, d_0, d^s_0) = (0, \{\phi\}, \epsilon, \epsilon)$.

A strong tableau T being *proper* is defined as for bundled tableaux in Def. 2.2.38. T is *consistent* iff for every $j \in \mathbb{N}$, $\bigwedge \Gamma_j[d^s_j]$ is consistent. $\square$

Lemma 5.3.6 Let ϕ be a μ-aconjunctive formula in pnf, and T a strong tableau or a bundled tableau for ϕ. For every $j \in \mathbb{N}$ and $u \in \mathcal{U}$ such that $d_j(u) = \mu z.\phi'$ for some z and ϕ', the constant u is active in at most one formula $\psi \in \Gamma_j$.

Proof: The claim holds trivially for the first element of T. All the tableau rules except $\wedge$ clearly preserve the validity of the claim, and $\wedge$ preserves it thanks to the μ-aconjunctivity of ϕ. $\square$

Lemma 5.3.7 Let ϕ be a μ-aconjunctive μTL-formula in pnf. If ϕ is consistent, there is a consistent strong tableau T for ϕ.

Proof: Let (i, Γ, d, d^s) be an element of a strong tableau such that $\bigwedge \Gamma[d_s]$ is consistent. It is easy to see that if any of rules $\wedge$, $\bigcirc$, σ' or $U\nu$ can be applied to it, then for the resulting element $(i', \Gamma', d', d^{s\prime})$, $\bigwedge \Gamma'[d^{s\prime}]$ is consistent. By Lemmas 5.3.4 and 5.3.6 the same holds for the $U\mu$ rule. If $\vee L$ and $\vee R$ rules can be applied to (i, Γ, d, d^s), then at least one of them yields a $(i', \Gamma', d', d^{s\prime})$ such that $\bigwedge \Gamma'[d^{s\prime}]$ is consistent. As some rule is always applicable, this means that we can construct a consistent strong tableau T for ϕ, starting from the element $(0, \{\phi\}, \epsilon, \epsilon)$. $\square$

Lemma 5.3.8 Let ϕ be a guarded μTL-formula in pnf, and T a strong tableau for ϕ. If T is consistent, then T is proper.

Proof: Assume that $T = (i_0, \Gamma_0, d_0, d^s_0) \ldots$ is not proper, and take the smallest $m \in \mathbb{N}$ such that for some $k \in \mathbb{N}$ and $n \geq m$, $d_k = (u_1, \sigma z_1.\phi_1) \ldots (u_n, \sigma z_n.\phi_n)$, $d_k(u_m) = \mu z_m.\phi_m$, and $u_m \in \Gamma_j$ for infinitely many j. For every $j \in \mathbb{N}$ define $\Gamma'_j = \{\psi \in \Gamma_j \mid u_m \text{ not active in } \psi\}$.

As ϕ is guarded, there is an infinite sequence of indices $j_1, j_2 \ldots$ such that the $U\mu$-rule is applied to u_m at point $j_h - 1$ of T for every $h \in N$. By Lemma 5.3.6 this means that for all $h \in N$, $\Gamma_{j_h - 1} = \Gamma'_{j_h - 1} \cup \{u_m\}$ and $\Gamma'_{j_h - 1} = \Gamma'_{j_h}$, implying $\vdash \bigwedge \Gamma'_{j_h - 1}[d^s_{j_h - 1}] \Rightarrow \bigwedge \Gamma'_{j_h}[d^s_{j_h}]$ and $\vdash u_m[d^s_{j_h}] \Rightarrow \neg \bigwedge \Gamma'_{j_h}[d^s_{j_h}]$. By the choice of m we can assume without loss of generality that for every $m' < m$, if $d_k(u_{m'}) = \mu z_{m'}.\phi_{m'}$ then $u_{m'} \notin \Gamma_j$ for all $j \geq j_1$, meaning that the $U\mu$-rule is not applied to any of $u_1, \ldots, u_{m-1}$ at any point $j \geq j_1$. Remembering the above, this

name	application
$\boldsymbol{U\nu\chi}$	$\dfrac{i,\ \ \Gamma\cup\{u\},\qquad d,\ \ d^s}{i,\ \ \Gamma\cup\{u,\bar{\chi}[u/x]\},\ \ d,\ \ d^s}$ **1**
$\bigcirc$	$\dfrac{i,\qquad \Gamma\cup\{\bigcirc\phi_1,\ldots,\bigcirc\phi_k\},\ \ d,\ \ d^s}{i+1,\ \ \{\phi_1,\ldots,\phi_k\},\qquad d,\ \ d^s}$ **2**
Note:	**1**: $d(u)=\nu x.\chi$ and $U\nu\chi$ has not been applied after previous $\bigcirc$-point **2:** $\Gamma\subseteq\mathcal{Z}\cup\{\neg z\mid z\in\mathcal{Z}\}\cup U_\chi\cup\{\neg u\mid u\in U_\chi\}$ where $U_\chi=\{u\in U\mid d(u)=\nu x.\chi\}$ and for every $u\in U_\chi\cap\Gamma$ the $U\nu\chi$-rule has been applied to u after previous $\bigcirc$-point.

Table 5.2: Modified strong tableau rules

implies that $\vdash u_m[d^s_j]\Rightarrow\neg\bigwedge\Gamma'_{j_h}[d^s_{j_h}]$ for all $j\geq j_h$ and all $h\in N$. Furthermore, $\Gamma'_{j_h}[d^s_{j_h}]\subseteq\mathrm{cl}(\Gamma_{j_1}[d^s_{j_1}])$ for all $h\in N$.

Since $\mathrm{cl}(\Gamma_{j_1}[d^s_{j_1}])$, the closure of $\Gamma_{j_1}[d^s_{j_1}]$ (see Def. 2.2.12), is finite, there are some $h<l$ such that $\Gamma'_{j_h}[d^s_{j_h}]=\Gamma'_{j_l}[d^s_{j_l}]$. But then

$$\vdash\bigwedge\Gamma_{j_l-1}[d^s_{j_l-1}]\Rightarrow(u_m[d^s_{j_l-1}]\wedge\bigwedge\Gamma'_{j_l-1}[d^s_{j_l-1}])\Rightarrow(\neg\bigwedge\Gamma'_{j_h}[d^s_{j_h}]\wedge\bigwedge\Gamma'_{j_l}[d^s_{j_l}])\Rightarrow\bot$$

implying that T is not consistent. □

Proposition 5.3.9 Let ϕ be a guarded μ-aconjunctive μTL-formula. If ϕ is consistent, then ϕ is satisfiable.

Proof: As $\vdash\phi\Leftrightarrow\mathrm{pnf}(\phi)$, we can assume that ϕ is in pnf. If ϕ is consistent, by Lemmas 5.3.7 and 5.3.8, there is a proper consistent strong tableau and therefore a proper consistent bundled tableau T for ϕ. By Lemma 5.3.3 this means that ϕ is satisfiable. □

For technical reasons we need a slight extension of the previous result. This is caused by the fact that the transformation of formulae to the banan-form does not necessarily preserve the positivity of free variables in a formula.

Lemma 5.3.10 Let ψ and χ be formulae such that

- $\psi\wedge\nu x.\chi$ is well-formed and consistent,
- ψ is guarded, μ-aconjunctive and in pnf, and
- there exists a guarded μ-aconjunctive formula $\bar{\chi}$ in pnf such that $\vdash\chi\Leftrightarrow\bar{\chi}$.

Then $\psi\wedge\nu x.\chi$ is satisfiable.

Proof: Let us modify slightly the rules for a strong tableau by adding a new $U\nu\chi$-rule and modifying the $\bigcirc$-rule as in Table 5.2, and by requiring that the

$U\nu$-rule is not applied to a constant u such that $d(u) = \nu x.\chi$. Notice that as x does not necessarily occur only positively in $\bar{\chi}$, we can have negated occurrences of a constant u corresponding to $\nu x.\chi$ in a tableau.

Since $\vdash \chi \Leftrightarrow \bar{\chi}$ implies $\vdash \nu x.\chi \Leftrightarrow \chi[\nu x.\chi/x] \Leftrightarrow \bar{\chi}[\nu x.\chi/x]$ by Lemma 5.3.1, the $U\nu\chi$-rule preserves consistency. As in Lemma 5.3.7, the consistency of $\psi \wedge \nu x.\chi$ implies then the existence of a consistent strong tableau (with the modified rules) $T = (i_1, \Gamma_1, d_1, d_1^s) \ldots$ for $\psi \wedge \nu x.\chi$. As in Lemma 5.3.8 the consistency of T means that it is proper, as well.

Define a model M on the basis of T as in the proof of Lemma 5.3.3, and define a set $W \subseteq \mathbb{N}$ by $W = \{k \in \mathbb{N} \mid \exists j \in \mathbb{N}, u \in \mathcal{U} : i_j = k, u \in \Gamma_j \text{ and } d_j(u) = \nu x.\chi\}$. For every $s \in W$, we can read from T a proper bundled tableau witnessing $M[W/x], s \models \bar{\chi}$ by Lemma 2.2.39. Since $\vdash \chi \Leftrightarrow \bar{\chi}$, this implies by Theorem 5.1.2 that $M[W/x], s \models \chi$ for all $s \in W$. As $0 \in W$, this implies $M \models \nu x.\chi$. From T we can also read a proper tableau witnessing $M \models \psi$. Consequently, $M \models \psi \wedge \nu x.\chi$, as required. □

The following lemma is the heart of the completeness proof. Essentially it shows that we can lift the expressive equivalence of the full μTL and the fragment of formulae in banan-form from the level of semantics to the level of provability.

Lemma 5.3.11 For any μTL-formula ϕ, there exists a formula ϕ' in banan-form such that $\vdash \phi \Leftrightarrow \phi'$.

Proof: We show the claim by induction on the structure of the formula ϕ. Without loss of generality we can assume that ϕ is written using just the $\wedge, \neg, \bigcirc$ and μ-operators.

Induction basis: For an atomic ϕ, choosing $\phi' = \phi$ clearly fulfils the claim.

Induction step for $\wedge$, $\neg$, $\bigcirc$: Suppose that for ϕ_1, ϕ_2 we have ϕ_1', ϕ_2' in banan-form such that $\vdash \phi_1 \Leftrightarrow \phi_1'$ and $\vdash \phi_2 \Leftrightarrow \phi_2'$. By Lemma 5.2.4 $\phi_1' \wedge \phi_2'$, $\neg\phi_1'$ and $\bigcirc\phi_1'$ are in banan-form and clearly $\vdash \phi_1 \wedge \phi_2 \Leftrightarrow \phi_1' \wedge \phi_2'$, $\vdash \neg\phi_1 \Leftrightarrow \neg\phi_1'$ and $\vdash \bigcirc\phi_1 \Leftrightarrow \bigcirc\phi_1'$.

Induction step for μ: Suppose that for ϕ we have a ϕ' in banan-form such that $\vdash \phi \Leftrightarrow \phi'$. By Lemma 5.2.6, there exists a formula ψ in banan-form such that $\models \mu z.\phi \Leftrightarrow \psi$. If we have $\vdash \mu z.\phi \Leftrightarrow \psi$, the induction step is satisfied, as ψ is in banan-form. Suppose then that $\nvdash \mu z.\phi \Leftrightarrow \psi$. This means that either

1 $\nvdash \mu z.\phi \Rightarrow \psi$, or

2 $\nvdash \psi \Rightarrow \mu z.\phi$

In case 1 we must have $\nvdash \phi[\psi/z] \Rightarrow \psi$, as otherwise $\vdash \mu z.\phi \Rightarrow \psi$ could be derived by the fixpoint induction rule.[1] This means that $\phi[\psi/z] \wedge \neg\psi$ is consistent. As $\vdash \phi \Leftrightarrow \phi'$, by Lemma 5.3.1 $\vdash \phi[\psi/z] \Leftrightarrow \phi'[\psi/z]$, implying that $\phi'[\psi/z] \wedge \neg\psi$ is consistent. Since ϕ' and ψ are in banan-form, by Lemma 5.2.4 $\phi'[\psi/z] \wedge \neg\psi$ is in banan-form, hence guarded and μ-aconjunctive. As it is consistent, by Prop. 5.3.9 it is satisfiable, i.e. there are M and s such that $M, s \models \phi'[\psi/z] \wedge \neg\psi$.

Since $\vdash \phi[\psi/z] \Leftrightarrow \phi'[\psi/z]$, by Theorem 5.1.2 $\models \phi[\psi/z] \Leftrightarrow \phi'[\psi/z]$, which implies $M, s \models \phi[\psi/z] \wedge \neg\psi$. By the choice of ψ, we know that $\models \psi \Leftrightarrow \mu z.\phi$, which implies $\models \phi[\psi/z] \Leftrightarrow \phi[\mu z.\phi/z]$, i.e. $\models \phi[\psi/z] \Leftrightarrow \mu z.\phi$. Consequently, $M, s \models \mu z.\phi \wedge \neg\psi$. But this contradicts $\models \mu z.\phi \Leftrightarrow \psi$, meaning that case 1 cannot hold.

In case 2, $\psi \wedge \neg\mu z.\phi = \psi \wedge \nu z.\neg\phi[\neg z/z]$ is consistent. As $\vdash \phi \Leftrightarrow \phi'$, by Lemma 5.3.1 $\vdash \neg\phi[\neg z/z] \Leftrightarrow \neg\phi'[\neg z/z]$. Since ϕ' is in banan-form, by Lemma 5.2.4 $\neg\phi'[\neg z/z]$ is in banan-form, hence guarded and μ-aconjunctive. But then by Lemma 5.3.10 $\psi \wedge \nu z.\neg\phi[\neg z/z] = \psi \wedge \neg\mu z.\phi$ is satisfiable, contradicting $\models \mu z.\phi \Leftrightarrow \psi$, i.e. case 2 cannot hold either.

Consequently, $\vdash \mu z.\phi' \Leftrightarrow \psi$, which concludes the induction step. □

Based on this lemma, the completeness of the axiomatisation follows easily.

Theorem 5.3.12 If a μTL-formula ϕ is consistent, then ϕ is satisfiable.

Proof: Immediate from Lemma 5.3.11, Proposition 5.3.9, and Theorem 5.1.2. □

Corollary 5.3.13 [**Completeness for μTL**] For any μTL-formula ϕ, if $\models \phi$ then $\vdash \phi$.

Proof: If $\models \phi$, then $\neg\phi$ is not satisfiable, therefore not consistent, implying $\vdash \neg\neg\phi$, i.e. $\vdash \phi$, by Theorem 5.3.12. □

5.4 Discussion

The completeness proof for μTL above uses in an essential way the fact that the whole μTL and its non-alternating fragment are equiexpressive; the transformation of formulae to banan-form in the semantic level depends on the ability to first transform any formula to an equivalent non-alternating one. However, with the exception of this point, the rest of the proof does not take advantage of the linear

[1]For the record, this in retrospect very natural observation had escaped us, leading to a more restricted and cumbersome solution, until we saw it used in passing in Igor Walukiewicz's work [101].

nature of the language. This allows us to use the same proof, modus modendi, to show the completeness of the axiomatisation also for the non-alternating fragment of the modal mu-calculus μKn.

Definition 5.4.1 We say that a μKn-formula ϕ is *provable* and write $\vdash \phi$, iff it is derivable in the following deductive system.

Axiom schemas:

ax1 All propositional tautologies

ax2 $\textcircled{i}(\phi \Rightarrow \psi) \Rightarrow (\textcircled{i}\phi \Rightarrow \textcircled{i}\psi)$

ax3 $\textcircled{i}\phi \Leftrightarrow \neg\textcircled{i}\neg\phi$

ax4 $\phi[\mu z.\phi/z] \Rightarrow \mu z.\phi$

Rules of inference:

modus ponens: from ϕ and $\phi \Rightarrow \psi$ infer ψ

necessitation: from ϕ infer $\textcircled{i}\phi$

fixpoint induction: from $\phi[\psi/z] \Rightarrow \psi$ infer $\mu z.\phi \Rightarrow \psi$ □

Theorem 5.4.2 For any non-alternating μKn-formula ϕ, $\models \phi$ iff $\vdash \phi$.

Proof: With obvious modifications the claim is shown in the same way as for μTL, except that Lemma 5.2.6 becomes vacuously the same as Lemma 5.2.5, and no reference to Cor. 3.5.3 is needed. In fact, the extension of Prop. 5.3.9 in Lemma 5.3.10 becomes unnecessary, since the translation of Lemma 5.2.5 preserves the positivity of free variables in a formula. □

However, this is is how far the result goes; there appears to be no way to extend it naturally to the full μKn. In fact, the axiomatisation of the modal mu-calculus used to be a longstanding open problem, until in 1995 Walukiewicz presented a completeness proof for what is essentially the axiomatisation above.

Theorem 5.4.3 For any μKn-formula ϕ, $\models \phi$ iff $\vdash \phi$.

Proof: See [101]. □

Like the proof for μTL here, Walukiewicz's proof is based on transforming formulae inductively to a particular normal form, called the *disjunctive* form. This is effectively the same form we call strongly aconjunctive here. Walukiewicz's proof naturally carries over from the modal mu-calculus to the linear one, as well, and is more general in this sense. It also deals with the extra complications caused by using the modalities $< a >$ and $[a]$, *for some/all a-successors*, instead of the indexed ones $\textcircled{i}$. However, the proof involves a fairly complex argument using games between tableaux and a priority technique to create a winning strategy in a game. In this respect the easy negatability of formulae in the bi-aconjunctive non-alternating normal form, a property the disjunctive normal form lacks, makes the approach here rather more straightforward. Bar one observation that was used in passing in Walukiewicz's work and has been adopted here to give a more elegant solution, the proof presented above was discovered independently.

In addition to the axiomatisations above for the fixpoint-based languages μTL and μKn, several axiomatisations for the other main class of logical formalisms in the current work, the quantifier-based languages, are also known. For the linear case, a complete axiomatisation for *S1S* was described by Siefkes already in 1970 [81]. More recently, Kesten and Pnueli showed the completeness of an axiomatisation for $\exists TL$ [53]. Both of these completeness proofs work by inductively transforming formulae to a normal form corresponding to Büchi automata on strings. For the branching case, on the other hand, the situation is less well developed. As far as we know, the only result concerning axiomatisations of quantifier-based branching formalisms is an axiomatisation of the weak language *WS2S* by Siefkes in [82], In particular, we are not aware of any complete axiomatisations for the strong branching languages *SnS* or $\exists Kn$.

Chapter 6

Axiomatising path quantifiers

In this chapter we depart slightly from the previous framework and examine an extension of the linear time mu-calculus μTL with *path quantifiers*. When we extended μTL to a branching formalism in Section 2.3, the step from μTL to μKn took place by replacing the single nexttime operator $\bigcirc$ with indexed operators $\textcircled{i}$. Path quantifiers $\overset{\rightarrow}{\exists}\phi$ and $\overset{\rightarrow}{\forall}\phi$, *for some path* ϕ and *for all paths* ϕ are another way of extending μTL to a formalism capable of describing branching properties. This way of extending a linear-time formalism to a branching one is used in various temporal logics; probably the best example is the full computation tree logic CTL^* [28, 29], which extends the standard linear-time temporal logic TL with path quantifiers. We call the formalism consisting of μTL and path quantifiers here the *extended computation tree logic* $\overset{\rightarrow}{\exists}\mu TL$. Expressively equivalent formulations of extended computation tree logic using linear time operators corresponding to ω-regular expressions and various types of finite automata on infinite strings are discussed in [96, 92, 20]. Requiring an infinite family of temporal operators, these formulations are syntactically less elegant than the fixpoint-based $\overset{\rightarrow}{\exists}\mu TL$, which only requires the single *nexttime* temporal operator. At the basis of all these extensions of the branching time logic CTL^* are extensions of the underlying linear time logic TL, either by automata-based temporal operators [103, 99], or by fixpoints as in μTL.

In the current chapter the axiomatisation of the linear-time mu-calculus μTL is extended to an axiomatisation of $\overset{\rightarrow}{\exists}\mu TL$. In general, it has turned out to be difficult to axiomatise branching-time logics with path quantifiers, even though natural axiom systems for the underlying linear-time formalisms would be known. The main reason for this is the interaction of path quantifiers with other operators of the logic, which means that it is generally not enough to simply add the obvious quantifier rules to an axiomatisation of the underlying linear time logic. This holds also for the extended computation tree logic $\overset{\rightarrow}{\exists}\mu TL$, so extending the axiomatisation of μTL to $\tilde{\exists}\mu TL$ is a non-trivial task.

Temporal logics with path quantifiers are often interpreted over generalised branching structures, where all infinite branches through a structure do not necessarily count as paths for the purpose of path quantification. There are several different classes of such structures for interpreting branching time logics. The most common and computationally natural class of models are the *R-generable* structures [24], basically normal transition systems where every maximal sequence of pairwise connected states counts as a path. More general classes of models in which not all such sequences are considered paths for the purposes of path quantification arise e.g. from fairness considerations. Although the notion of what counts as a path can in principle be arbitrary, the set of paths is usually required to fulfil some regular properties, such as suffix, fusion or limit closure. These three requirements together correspond to R-generability in the sense that a formula of a branching time logic is valid in all R-generable models iff it is valid in all suffix, fusion and limit closed models [24].

Different classes of models correspond to different notions of universal validity, and therefore to different axiomatisation problems. An axiomatisation of the computation tree logic CTL^* that is complete with respect to all suffix closed models is presented in [83], and it is shown that this can be extended to an axiomatisation that is complete with respect to all suffix and fusion closed models by adding the axiom $\tilde{\forall}\bigcirc\phi \Rightarrow \bigcirc\tilde{\forall}\phi$. However, the problem of completely axiomatising CTL^* for R-generable models, i.e. capturing limit closure by axioms, has been an open problem for some while, stated e.g. in [35, 25, 83]. The best that is known is an axiomatisation for CTL, a restricted sublogic of CTL^*, where limit closure is characterised by the axiom schema $\vdash \tilde{\forall}\mathbf{G}(\phi \Rightarrow \tilde{\exists}\bigcirc\phi) \Rightarrow (\phi \Rightarrow \tilde{\exists}\mathbf{G}\phi)$ [30, 25].

We present next a solution to this axiomatisation problem with respect to R-generable structures for the extended computation tree logic $\tilde{\exists}\mu TL$. To characterise limit closure, we introduce a new inference rule, the *$\tilde{\exists}\nu$-induction*. The completeness proof is is based on transforming formulae to a strongly aconjunctive deterministic normal form that corresponds to first recurrence automata on infinite strings.

An intriguing aspect in this completeness proof is that the ability to transform a formula to the deterministic form requires the power given by arbitrary alternation of fixpoints. Therefore, the approach is not directly applicable for the formulation of extended computation tree logic with ω-regular expressions, although this is semantically equiexpressive with $\tilde{\exists}\mu TL$. The same holds also for CTL^*, so the axiomatisation problem for it remains open.

6.1 Preliminaries

6.1.1 Path quantifiers

In Section 2.3 we extended the linear-time mu-calculus μTL, describing properties of sequences, to a formalism describing properties of trees by changing the set of basic modalities, i.e. replacing the single nexttime operator $\bigcirc$ with indexed operators $\textcircled{\scriptsize i}$. Another approach is to keep the nexttime-operator as it is, interpreting the logic primarily over paths, but to add to the logic an operator which makes it possible to switch the path a formula is interpreted over and to quantify over paths. In such a language we can expres properties like *on every path a holds someitmes and on some path b holds constantly.*

Originally the approach of path quantification was used to extend the standard linear-time temporal logic TL to a branching-time formalism. Probably the most widely known of these branching-time logics is the full computation tree logic CTL^* [29]. Some examples of properties which we can express in CTL^* using the path quantifiers $\vec{\exists}\phi$ and $\vec{\forall}\phi$, *for some path* ϕ and *for all paths* ϕ are

- $\vec{\exists}\mathbf{G}a$, *on some path always a*,
- $\vec{\forall}\mathbf{F}a$, *on all paths sometimes a*,
- $\vec{\forall}\mathbf{FG}a$, *on every path almost always a*, and
- $\vec{\forall}\mathbf{F}\vec{\forall}\mathbf{G}a$, *on every path there is a point such that for every path from that point onwards always a.*

The last two formulae correspond to the μKn-formulae $\mu z.\nu x.(a \vee \Box z) \wedge \Box x$ and $\mu z.(\nu x.a \wedge \Box x) \vee \Box z$, respectively. For discussion on the difference of these, see page 52.

In the following we examine the language obtained from the linear-time mu-calculus μTL is a similar fashion. Let us first define this formally.

Definition 6.1.1 The formulae of the *extended computation tree logic* $\vec{\exists}\mu TL$ are defined by the abstract syntax:

$$\phi ::= z \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \bigcirc\phi \mid \vec{\exists}\phi \mid \mu z.\phi$$

where z varies over $\mathcal{Z}$. In $\mu z.\phi$, z is required to be *bindable* in ϕ, that is:

- z only occurs positively in ϕ, and
- z does not occur in the scope of a path quantifier $\vec{\exists}$ in ϕ.

The derived operator $\vec{\forall}$ stands for $\vec{\forall}\phi = \neg\vec{\exists}\neg\phi$, and the symbol $\vec{\Pi}$ refers to to both $\vec{\forall}$ and $\vec{\exists}$. $\Box$

Let us also introduce some technical terminology.

Definition 6.1.2 A $\overset{\rightsquigarrow}{\exists}\mu TL$-formula ϕ is a *basic state formula* iff either ϕ is atomic, or $\phi = \overset{\rightsquigarrow}{\forall}\phi'$ or $\phi = \neg\overset{\rightsquigarrow}{\forall}\phi'$ for some ϕ'. If ϕ is a boolean composition of basic state formulae, ϕ is a *state formula*, otherwise ϕ is a *proper path formula*. A formula without any path quantifiers $\overset{\rightsquigarrow}{\Pi}$ is a *pure path formula*.

The concept of the *positive normal form* for $\overset{\rightsquigarrow}{\exists}\mu TL$-formulae is analogous to Def. 2.2.20, with the exception that formulae may also contain $\overset{\rightsquigarrow}{\exists}$ and $\overset{\rightsquigarrow}{\forall}$ operators, and the following additional laws are used:

$$\begin{aligned} \neg\overset{\rightsquigarrow}{\exists}\phi &= \overset{\rightsquigarrow}{\forall}\neg\phi \\ \neg\overset{\rightsquigarrow}{\forall}\phi &= \overset{\rightsquigarrow}{\exists}\neg\phi \end{aligned}$$

The *path quantifier depth* of ϕ, denoted by $d_{pq}(\phi)$, is the level of nesting of path quantifiers in ϕ,

$$d_{pq}(\phi) = \max\{n \in \mathbb{N} \mid \exists\phi_1, \dots \phi_n : \phi \trianglelefteq \overset{\rightsquigarrow}{\Pi}\phi_1 \lhd \dots \lhd \overset{\rightsquigarrow}{\Pi}\phi_n\}$$

□

Definition 6.1.3 Let $M \in \mathcal{M}_{\leq\omega}$ be a branching model and $p \in \mathrm{paths}(M)$ a path of M. The set of points on path p of M satisfying a $\overset{\rightsquigarrow}{\exists}\mu TL$-formula ϕ, denoted by $\|\phi\|_{M,p}$, is defined by

$$\|\phi\|_{M,p} = \|\phi\|_{M,p,M[p]}$$

where $\|\phi\|_{M,p,M'}$ denotes the auxiliary concept of the set of points of linear model M' corresponding to the path p of M. This is defined inductively as follows:

$$\begin{aligned} \|z\|_{M,p,M'} &= \{i \in \mathrm{st}(M') \mid z \in M'(i)\} \\ \|\neg\phi\|_{M,p,M'} &= \mathrm{st}(M') \setminus \|\phi\|_{M,p,M'} \\ \|\phi \wedge \phi'\|_{M,p,M'} &= \|\phi\|_{M,p,M'} \cap \|\phi'\|_{M,p,M'} \\ \|\bigcirc\phi\|_{M,p,M'} &= \{i \in \mathrm{st}(M') \mid i+1 \in \|\phi\|_{M,p,M'}\} \\ \|\overset{\rightsquigarrow}{\exists}\phi\|_{M,p,M'} &= \{s \in \mathrm{st}(M') \mid \exists p' \in \mathrm{paths}(M, p(s)) : 0 \in \|\phi\|_{M,p',M[p']}\} \\ \|\mu z.\phi\|_{M,p,M'} &= \bigcap\{W \subseteq \mathrm{st}(M') \mid \|\phi\|_{M,p,M'[W/z]} \subseteq W\} \end{aligned}$$

Here the notation $M[p]$ is defined in Def. 2.1.5, $\mathrm{paths}(M,p)$ in Def. 2.1.3, and $M'[W/z]$ in Def. 2.2.4.

We say that ϕ *is true at point s of path p of model M* and write $M, p, s \models \phi$ iff $s \in \|\phi\|_{M,p}$. We write $M, p \models \phi$ iff $M, p, 0 \models \phi$. We say that ϕ *is true at state s of model M* and write $M, s \models \phi$ iff $M, p \models \phi$ for all paths $p \in \mathrm{paths}(M, s)$. Furthermore, we write $M \models \phi$ iff $M, s \models \phi$ for all states $s \in \mathrm{st}(M)$, and $\models \phi$ iff $M \models \phi$ for all models M. As before, a formula ϕ is *satisfiable* iff $M, p \models \phi$ for some model M and path p. □

For some examples of $\tilde{\vec{\exists}}\mu TL$-formulae, the property *a is true everywhere* can be expressed by $\tilde{\vec{\forall}}\nu z.a \wedge \bigcirc z$, and the property *on some path a holds in every even state* by $\tilde{\vec{\exists}}\nu z.a \wedge \bigcirc\bigcirc z$. The CTL^*-formula $\tilde{\vec{\forall}}\mathbf{F}\tilde{\vec{\forall}}\mathbf{G}a$ can be expressed by $\tilde{\vec{\forall}}\mu z.(\tilde{\vec{\forall}}\nu x.a \wedge \bigcirc x) \vee \bigcirc z$, and the CTL^*-formula $\tilde{\vec{\forall}}\mathbf{FG}a$ by $\tilde{\vec{\forall}}\mu z.(\nu x.a \wedge \bigcirc x) \vee \bigcirc z$.

6.1.2 Extended models

In the truth definition for $\tilde{\vec{\exists}}\mu TL$-formulae above, every path through the model tree counts for the purposes of path quantification. However, we can also take a more general approach, where only a subset of paths are considered for quantification. Such more general models become useful for example when some infinite execution sequences of a program are not considered valid because of fairness considerations. Also various viewpoints about the relation of time and chance can be modelled naturally using such models.

Definition 6.1.4 An *extended branching model* is a pair $\hat{M} = (M, P)$, where

- $M \in \mathcal{M}_{\leq\omega}$ is a branching model, and
- $P \subseteq \text{paths}(M)$ is a set of paths of M, such that for every state $s \in \text{st}(M)$ and every child s' of s, there is some path $p \in P$ and point $i \in \mathbb{N}$ such that $s = p(i)$ and $s' = p(i+1)$.

The set of states of a path p of an extended model $\hat{M}$ fulfilling the formula ϕ, denoted by $\|\phi\|_{\hat{M},p}$, is defined as before in Def. 6.1.3, except for

$$\|\tilde{\vec{\exists}}\phi\|_{\hat{M},p,M'} = \{s \in \text{st}(M') \mid \exists p' \in \text{paths}(M, p(s)) \cap P : 0 \in \|\phi\|_{\hat{M},p',M[p']}\}$$

□

The technical restriction for the set P of paths in the previous definition essentially states that a model has no 'superfluous' states which would not be reachable form their parent by any path.

In principle the set of paths in an extended model can be chosen completely arbitrarily, as long as the relevent technical restrictions are observed. However, it is often the case that this completely general case does not correspond to the intuitions underlying the definition of a model, and some constraints must be imposed on the structure of a model for it to make sense. The following definition formulates some commonly used constraints.

Definition 6.1.5 Let $\hat{M} = (M, P)$ be an extended branching model. We say that the set of paths P is

- *R-generated* iff $P = \text{paths}(M)$,

- *suffix closed* iff for every $p = p(0)p(1)p(2)\ldots \in P$, also $p(1)p(2)\ldots \in P$,
- *fusion closed* iff for every $p, p' \in P$, if $p(i) = p'(j)$, then $p(0)p(1)\ldots p(i-1)p'(j)p'(j+1)\ldots \in P$, and
- *limit closed* iff for every $p \in \text{paths}(M)$, if for every $i \in \mathbb{N}$, the string $p(0)p(1)\ldots p(i)$ is a prefix of some $p' \in P$, then $p \in P$. □

In modelling computation, it is hard to see how suffix closure could fail to hold. Fusion closure corresponds intuitively to the idea that how the execution of a program proceeds from a given state depends entirely on the state itself and not on how it has been reached. Limit closure is a continuity property.

The way extended models are defined above differs slightly from e.g. that used in [24]. The usual starting point is more general; the primary components of a model are an arbitrary (often countable) set of states S, and an arbitrary collection P of infinite sequences of elements of S serving as paths. In these structures the concept of R-generability says that it is possible to choose a transition relation R on the set of states so that the set of infinite sequences naturally generated by this transition system is precisely the set P of paths. However, as we assume uniformly in the current work that all models are tree-like and that there is an underlying parent-child transition relation, the only way an extended model (M, P) can be R-generated is when $P = \text{paths}(M)$.

The condition of being R-generated corresponds to suffix, fusion and limit closure in the following sense.

Proposition 6.1.6 Let $\hat{M} = (M, P)$ be an extended branching model. The set of paths P is R-generated iff it is suffix, fusion and limit-closed.

Proof: From left to right, this is [24, Thm 3.1]. From right to left the proof is as that of [24, Thm 3.3]. In the present framework this direction of the proof requires the technical side-condition in the definition of an extended model above which states that every parent-child pair occurs in some path. □

The axiom system decribed below aims at characterising universal validity with respect to the normal notion of models. As these are precisely the R-generated extended models, we can view the axiomatisation problem as capturing suffix, fusion and limit-closure by axioms.

Corollary 6.1.7 For any $\overset{\leftarrow}{\exists}\mu TL$-formula ϕ, we have $M \models \phi$ for all models $M \in \mathcal{M}_{\leq\omega}$ iff $\hat{M} \models \phi$ for all extended models $\hat{M} = (M, P)$ such that $M \in \mathcal{M}_{\leq\omega}$ and P is suffix, fusion and limit-closed. □

6.2 Axiomatisation

The axiomatisation of $\overrightarrow{\exists}\mu TL$ below consists of four components: an axiomatisation of the linear time mu-calculus μTL, i.e. the language of pure path formulae, some obvious quantification rules for the path quantifiers, an axiom corresponding to fusion closure, and an inference rule reflecting limit closure. The main novelty is the last of these. This inference rule characterising limit closure is of the form:

from $\psi \Rightarrow \overrightarrow{\exists}\phi[\psi/z]$ infer $\psi \Rightarrow \overrightarrow{\exists}\nu z.\phi$

This rule, however, is not sound for all formulae ϕ. Let us therefore first characterise semantically a class of formulae, those *bounded* by z, for which the rule is sound, and then show that syntactically strongly aconjunctive formulae enjoy this semantic property.

Definition 6.2.1 Let ϕ be a pure path formula of $\overrightarrow{\exists}\mu TL$, i.e. a formula of the linear-time mu-calculus μTL, and assume that z occurs only positively in ϕ. We say that ϕ is *bounded by* z iff for all models M and all paths p of M such that $M, p \models \phi$, either:

1 $M[\emptyset/z], p \models \phi$, or

2 there is some point $n \in \mathbb{N}$ such that $M, p(n) \models z$, and $M[\{p(n)\}/z], p' \models \phi$ for all paths p' of M such that $p'(0)\dots p'(n) = p(0)\dots p(n)$. □

Intuitively, the above states that if $M, p \models \phi$, either ϕ would hold of p even if z was not true anywhere along p at all, or there is some particular point n along p such that z is true in state $p(n)$, and if we assume that z was true only at $p(n)$, ϕ would still hold of p, and not only of p but of every other path that follows p up to the point $p(n)$, as well. This expresses the idea that ϕ being true of p only depends on what p is like up to $p(n)$.

Lemma 6.2.2 Let ϕ be a pure path formula and ψ any formula of $\overrightarrow{\exists}\mu TL$, and z a variable such that z is bindable in ϕ and ϕ is bounded by z. Let $\hat{M}$ be an extended model $\hat{M} = (M, P)$ such that P is R-generable. If $\hat{M} \models \psi \Rightarrow \overrightarrow{\exists}\phi[\psi/z]$ then $\hat{M} \models \psi \Rightarrow \overrightarrow{\exists}\nu z.\phi$.

Proof: Assume that $\hat{M} \models \psi \Rightarrow \overrightarrow{\exists}\phi[\psi/z]$ and take any path $p \in P$ for which $\hat{M}, p \models \psi$. We want to show that $\hat{M}, p \models \overrightarrow{\exists}\nu z.\phi$.

Define inductively a finite or infinite sequence of pairs $(p^0, k^0)\dots(p^i, k^i)\dots$ where each $p^i \in P$, $k^i \in \mathbb{N}$, such that $p^0(0) = p(0)$ and for all elements of the sequence

- $k^i = 0$ iff (p^i, k^i) is the last element in the sequence and then $\hat{M}[\{p^i(0)\}/z], p^i \models \phi$, and otherwise
- $p^{i+1}(0) = p^i(k^i)$, $\hat{M}, p^i[k^i \ldots] \models \psi$, and $\hat{M}[\{p^i(k^i)\}/z], p' \models \phi$ for all paths $p' \in P$ such that $p'(0)\ldots p'(k^i) = p^i(0)\ldots p^i(k^i)$

Define first a pair (p^{-1}, k^{-1}) that is not a part of the actual sequence by $p^{-1} = p$ and $k^{-1} = 1$. Notice that $\hat{M}, p^{-1}[k^{-1}\ldots] \models \psi$. Assume then that we have p^i, k^i such that $\hat{M}, p^i[k^i\ldots] \models \psi$. As $\hat{M} \models \psi \Rightarrow \overset{\rightarrow}{\exists}\phi[\psi/z]$, we have $\hat{M}, p' \models \phi[\psi/z]$ for some $p' \in \text{paths}(M, p^i(k^i)) \cap P$. Define $p^{i+1} = p'$. Clearly $p^{i+1}(0) = p^i(k^i)$. Since no free variable in ψ is bound by a fixpoint operator in $\phi[\psi/z]$, and since z does not occur is scope of a path quantifier in ϕ, $\hat{M}[W/z], p' \models \phi$ where $W = \|\psi\|_{M,p'}$ As ϕ is bounded by z, one of the following holds:

- $\hat{M}[W/z][\emptyset/z], p' \models \phi$, i.e. $\hat{M}[\emptyset/z], p' \models \phi$. Since z occurs only positively in ϕ, this implies $\hat{M}[\{p(0)'\}/z], p' \models \phi$. Define $k^{i+1} = 0$, and let this be the last element.
- There exists some $n \in \mathbb{N}$ such that
  - $\hat{M}[W/z], p'(n) \models z$ and
  - $\hat{M}[W/z][\{p'(n)\}/z], p'' \models \phi$, i.e. $\hat{M}[\{p'(n)\}/z], p'' \models \phi$, for all paths p'' of $\hat{M}$ such that $p''(0)\ldots p''(n) = p'(0)\ldots p'(n)$.

  Define $k^{i+1} = n$, and if $n = 0$ let this be the last element. If $n = 0$, $\hat{M}[\{p(0)'\}/z], p' \models \phi$, as required. If $n > 0$, $\hat{M}[W/z], p'(n) \models z$ implies $p'(n) \in W$, and by the definition of W we have $\hat{M}, p'[n\ldots] \models \psi$, i.e. $\hat{M}, p^{i+1}[k^{i+1}\ldots] \models \psi$.

If the sequence is finite and (p^n, k^n) is its last element, define a path p' by

$$p' = p^0[0\ldots(k^1-1)] \cdot p^1[0\ldots(k^2-1)] \cdot \ldots p^{n-1}[0\ldots(k^{n-1}-1)] \cdot p^n$$

Since the set P of paths is suffix and fusion closed, we have $p' \in P$. If the sequence is infinite, define

$$p' = p^0[0\ldots(k^1-1)] \cdot p^1[0\ldots(k^2-1)] \cdot \ldots$$

Since the set P of paths is suffix, fusion and limit closed, we have $p' \in P$. Define W as the set of points $W = \{0, k^0, k^0+k^1, k^0+k^1+k^2, \ldots\}$. Since z occurs only positively in ϕ, the claims above mean that $W \subseteq \|\phi\|_{M,p',M[p'][W/z]}$. By Lemma 2.2.9, this implies $M, p' \models \nu z.\phi$, and further $M, p \models \overset{\rightarrow}{\exists}\nu z.\phi$, as $p(0) = p'(0)$. □

Let us then show that strong aconjunctivity implies boundedness.

Lemma 6.2.3 Let ϕ be a pure path formula of $\overset{\rightarrow}{\exists}\mu TL$, i.e. a μTL-formula, and z a variable such that z is bindable in ϕ and $\nu z.\phi$ is strongly aconjunctive. Then ϕ is bounded by z.

Proof: Notice first that by the transformation of Proposition 2.2.36, we can transform any ϕ fulfilling the requirements above to an equivalnet guarded formula ϕ' also fulfilling the requirements. Therefore, we can assume without loss of generality that ϕ is guarded.

To show that ϕ is bounded by z, let us take any model $M \in \mathcal{M}_{\leq\omega}$ and path p of M such that $M, p \models \phi$, and show that one of the cases of Def. 6.2.1 must hold. Define first a linear model M' by $M' = M[p]$. Since $M, p \models \phi$ and ϕ is a μTL-formula. we have $M' \models \phi$. By Prop. 2.2.39 there is a proper bundled tableau $T = (i_0, \Gamma_0, d_0)(i_1, \Gamma_1, d_1)\dots$ for ϕ agreeing with M'. Since $\nu z.\phi$ is well-formed and strongly aconjunctive, z occurs only positively in T and for every point j of T there is at most one $\psi \in \Gamma_j[d_j]$ such that ψ either contains z or the $\bigcirc$-operator.

If the tableau T agrees with $M'[\emptyset/z]$, then it witnesses $M'[\emptyset/z] \models \phi$ and $M[\emptyset/z], p \models \phi$, fulfilling the first case in the definition of boundedness, Def. 6.2.1.

If T does not agree with $M'[\emptyset/z]$, there is some $\bigcirc$-opint m of T and atomic formula $\psi \in \Gamma_m[d_m]$ such that $M', i_m \models \psi$ but $M'[\emptyset/z], i_m \not\models \psi$. Take the smallest such m. Since the atomic formula ψ must contain z and z occurs only positively in T, we must have $\psi = z$. Since ψ contains z, no other formula in $\Gamma_m[d_m]$ can contain z or the $\bigcirc$-operator, which means that $\Gamma_{m+1} = \emptyset$. Since m is the first point in T where T does not agree with $M'[\emptyset/z]$, it is then easy to see that T agrees with $M'[\{i_m\}/z]$. Furthermore, since T does not pay any attention to the states of any model beyond point i_m, T agrees with any model M'' for which $M''(i) = M'[\{i_m\}/z](i)$ for all $i \leq m$. In particular T agrees with $(M[\{p(i_m)\}/z])[p']$ for every path p' of M such that $p'(0)\dots p'(i_m) = p(0)\dots p(i_m)$. Consequently, $M[\{p(i_m)\}/z], p' \models \phi$ for all such paths p', fulfilling the second case in the definition of boundedness. □

We still need to extend the concept of strong aconjunctivity from linear-time mu-calculus to $\vec{\exists}\mu TL$.

Definition 6.2.4 A $\vec{\exists}\mu TL$-formula ϕ in pnf is *strongly aconjunctive* iff ϕ can be written in the form $\phi = \phi'[\vec{\Pi}\psi_1/x_1, \dots, \vec{\Pi}\psi_k/x_2]$ where ϕ' is a pure path formula, i.e. a μTL-formula, which is strongly aconjunctive. A formula ϕ is *strongly co-aconjunctive* iff $\neg\phi$ is strongly aconjunctive. □

Lemma 6.2.5 Let ϕ and ψ be arbitrary $\vec{\exists}\mu TL$-formulae and z a variable such that z is bindable in ϕ and $\nu z.\phi$ is strongly aconjunctive. Let $M \in \mathcal{M}_{\leq\omega}$ be a model such that $M \models \psi \Rightarrow \vec{\exists}\phi[\psi/z]$. Then $M \models \psi \Rightarrow \vec{\exists}\nu z.\phi$.

Proof: We can write $\nu z.\phi$ in the form $\nu z.\phi = \nu z.\phi'[\vec{\Pi}\chi_1/x_1, \dots, \vec{\Pi}\chi_k/x_k]$, where $\nu z.\phi'$ is a strongly aconjunctive pure path formula, x_i fresh variables which do not occur in $\nu z.\phi$ or ψ, and χ_i formulae which do not contain z. Define a model M' by $M' = M[W_1/x_1]\dots[W_k/x_k]$ where each $W_i = \{s \in \mathrm{st}(M) \mid M, s \models \vec{\Pi}\chi_i\}$. As $M \models \psi \Rightarrow \vec{\exists}\phi[\psi/z]$, we have $M' \models \psi \Rightarrow \vec{\exists}\phi'[\psi/z]$. Since $\nu z.\phi'$ is strongly aconjunctive, by Lemma 6.2.3 ϕ' is bounded by z. By Lemma 6.2.2 then $M' \models \psi \Rightarrow \vec{\exists}\nu z.\phi'$, implying $M \models \psi \Rightarrow \vec{\exists}\nu z.\phi$. □

Let us now formulate the axiomatisation.

Definition 6.2.6 We say that a $\vec{\exists}\mu TL$-formula ϕ is *provable* and write $\vdash \phi$ iff it is derivable in the following deductive system.

Axiom schemas:

ax1 all propositional tautologies

ax2 $\bigcirc(\phi \Rightarrow \psi) \Rightarrow (\bigcirc\phi \Rightarrow \bigcirc\psi)$

ax3 $\bigcirc\phi \Leftrightarrow \neg\bigcirc\neg\phi$

ax4 $\nu z.\phi \Rightarrow \phi[\nu z.\phi/z]$

ax5 $\vec{\forall}(\phi \Rightarrow \psi) \Rightarrow (\vec{\forall}\phi \Rightarrow \vec{\forall}\psi)$

ax6 $\vec{\forall}\phi \Rightarrow \phi$

ax7 $\phi \Rightarrow \vec{\forall}\phi$, where ϕ is a state formula

ax8 $\vec{\forall}\bigcirc\phi \Rightarrow \bigcirc\vec{\forall}\phi$

Rules of inference:

modus ponens:	from ϕ and $\phi \Rightarrow \psi$ infer ψ
$\bigcirc$-necessitation:	from ϕ infer $\bigcirc\phi$
ν-induction:	from $\psi \Rightarrow \phi[\psi/z]$ infer $\psi \Rightarrow \nu z.\phi$
$\vec{\forall}$-necessitation:	from ϕ infer $\vec{\forall}\phi$
$\vec{\exists}\nu$-induction:	from $\psi \Rightarrow \vec{\exists}\phi[\psi/z]$ infer $\psi \Rightarrow \vec{\exists}\nu z.\phi$, where $\nu z.\phi$ is strongly aconjunctive

□

Axioms ax1-ax4 and modus ponens, $\bigcirc$-necessitation and ν-induction rules of inference correspond directly to the axiomatisation of μTL in Def. 5.1.1; only the fixpoint induction rule has been here formulated as the dual of the rule in Def. 5.1.1 to stress the similarity between ν-induction and $\vec{\exists}\nu$-induction. Axioms ax5-ax7 express obvious properties of path quantification, and axiom ax8 reflects fusion closure [83]. Finally, $\vec{\exists}\nu$-induction inference rule characterises the limit closure of path sets. There is no particular axiom corresponding to suffix closure. However, from the other axioms and rules we can derive the schema $\bigcirc\vec{\forall}\phi \Rightarrow \bigcirc\phi$, which is not universally valid in non-suffix-closed structures.

Let us introduce the duals of the fixpoint induction rules as derived rules:

μ-induction:	from $\phi[\psi/z] \Rightarrow \psi$ infer $\mu z.\phi \Rightarrow \psi$

$\overrightarrow{\forall}\mu$-induction: from $\overrightarrow{\forall}\phi[\psi/z] \Rightarrow \psi$ infer $\overrightarrow{\forall}\mu z.\phi \Rightarrow \psi$, where $\mu z.\phi$ is strongly co-aconjunctive

Theorem 6.2.7 [**Soundness for** $\overrightarrow{\exists}\mu TL$] For any $\overrightarrow{\exists}\mu TL$-formula ϕ, if $\vdash \phi$ then $\models \phi$.

Proof: The $\overrightarrow{\exists}\nu$-induction rule is sound by Lemma 6.2.5, and all other axioms and rules are obvious. □

Since the axiomatisation of $\overrightarrow{\exists}\mu TL$ contains the earlier axiomatisation of μTL, we know immediately that it is complete for all formulae which belong to both languages.

Lemma 6.2.8 Let ϕ be a $\overrightarrow{\exists}\mu TL$-formula which is a pure path formula. If $\models \phi$ then $\vdash \phi$.

Proof: Immediate from Corollary 5.3.13. □

It is also easy to see that the substitution lemma holds with respect to $\overrightarrow{\exists}\mu TL$, as well.

Lemma 6.2.9 For any $\overrightarrow{\exists}\mu TL$-formulae ϕ and ψ, if $\vdash \phi$ then $\vdash \phi[\psi/z]$.

Proof: Induction on the length of the proof of $\vdash \phi$. □

6.3 Normal form

The completeness proof for the axiomatisation is based on transforming formulae to a normal form, the *full sad-form.*

Definition 6.3.1 Let ϕ be a pure path formula of $\overrightarrow{\exists}\mu TL$, i.e. a formula of the linear-time mu-calculus μTL. We say that ϕ is in the *strongly aconjunctive deterministic form* (abbreviated *sad-form*) iff

- ϕ is guarded,
- ϕ is strongly aconjunctive, and
- ϕ is deterministic.

An arbitrary $\overrightarrow{\exists}\mu TL$-formula ϕ is in sad-form iff ϕ can be written in the form $\phi = \phi'[\overrightarrow{\Pi}\chi_1/x_1, \ldots, \overrightarrow{\Pi}\chi_k/x_k]$, where ϕ' is a pure path formula which is in sad-form. An arbitrary $\overrightarrow{\exists}\mu TL$-formula ϕ is in *full sad-form* iff ϕ can be written in the form $\phi = \phi'[\overrightarrow{\Pi}\chi_1/x_1, \ldots, \overrightarrow{\Pi}\chi_k/x_k]$, where ϕ' is a pure path formula which is in sad-form, and each χ_i is a formula in full sad-form. □

On the basis of the discussion on determinisation in Section 3.5, we already know how to transform linear-time mu-calculus formulae to the sad-form.

Lemma 6.3.2 For every μTL-formula ϕ, there exists a μTL-formula ψ in sad-form such that $\models \phi \Leftrightarrow \psi$.

Proof: By Corollary 3.5.13, we know that for every μTL-formula ϕ there exists an equivalent restricted deterministic formula ψ. By definition being restricted implies guardedness and strong aconjuunctivity, so this ψ is in sad-form. $\Box$

Furthermore, by the completeness result for μTL, we also know that this can be done provably.

Lemma 6.3.3 For every μTL-formula ϕ, there exists a μTL-formula ψ in sad-form such that $\vdash \phi \Leftrightarrow \psi$.

Proof: Immediate from Lemmas 6.3.2 and 6.2.8. $\Box$

Let us show then that by applying this transformation, we can inductively transform to all $\vec{\exists}\mu TL$-formulae to the full sad-form.

Proposition 6.3.4 For every $\vec{\exists}\mu TL$-formula ϕ there is a formula ϕ' in full sad-form such that $\vdash \phi \Leftrightarrow \phi'$.

Proof: Let us show by induction on n that for every $n \in \mathbb{N}$, the claim holds for all ϕ such that $d_{pq}(\phi) \leq n$. Notice first that if $d_{pq}(\phi) = 0$, then ϕ is a pure path formula, and the claim holds by Lemma 6.3.3.

Assume then that the claim holds for n, and take any ϕ such that $d_{pq}(\phi) = n+1$. We can write ϕ in the form $\phi = \psi[\vec{\forall}\chi_1/x_1, \ldots, \vec{\forall}\chi_m/x_m]$, where ψ is a pure path formula and $d_{pq}(\chi_i) \leq n$ for every χ_i. Since ψ is a pure path formula, by induction assumption there is a ψ' in full sad-form such that $\vdash \psi \Leftrightarrow \psi'$.

Let then $\bar{x}_1 \ldots \bar{x}_m$ be fresh variables, and denote by ψ'' the formula obtained by replacing each $\neg x_i$ by $\bar{x}_i$ in ψ'. Then $\psi' = \psi''[\neg x_1/\bar{x}_1, \ldots, \neg x_m/\bar{x}_m]$ and every x_i and $\bar{x}_i$ occurs only positively in ψ''. Since for every i, $d_{pq}(\neg\chi_i) = d_{pq}(\chi_i) \leq n$, by induction assumption there are formulae χ_i' and $\bar{\chi}_i'$ in full sad-form such that $\vdash \chi_i \Leftrightarrow \chi_i'$ and $\vdash \neg\chi_i \Leftrightarrow \bar{\chi}_i'$. Define

$$\phi' = \psi''[\vec{\forall}\chi_1'/x_1, \ldots, \vec{\forall}\chi_m'/x_m, \vec{\exists}\bar{\chi}_1'/\bar{x}_1, \ldots, \vec{\exists}\bar{\chi}_m'/\bar{x}_m]$$

Then

$$\vdash \phi \quad = \quad \psi[\vec{\forall}\chi_1/x_1, \ldots, \vec{\forall}\chi_m/x_m]$$

$$\begin{aligned}
&\Leftrightarrow\ \psi'[\overrightarrow{\forall}\chi_1/x_1,\ldots,\overrightarrow{\forall}\chi_m/x_m]\\
&=\ \psi''[\overrightarrow{\forall}\chi_1/x_1,\ldots,\overrightarrow{\forall}\chi_m/x_m,\neg\overrightarrow{\forall}\chi_1/\bar{x}_1,\ldots,\neg\overrightarrow{\forall}\chi_m/\bar{x}_m]\\
&\Leftrightarrow\ \psi''[\overrightarrow{\forall}\chi_1'/x_1,\ldots,\overrightarrow{\forall}\chi_m'/x_m,\overrightarrow{\exists}\bar{\chi}_1'/\bar{x}_1,\ldots,\overrightarrow{\exists}\bar{\chi}_m'/\bar{x}_m]\\
&=\ \phi'
\end{aligned}$$

Moreover, since x_i, $\bar{x}_i$ occur in ψ'' only positively and χ_i', $\bar{\chi}_i'$ are in full sad-form, ϕ' is in full sad-form. □

6.4 Completeness

In this section we show the completeness of the axiom system with respect to formulae in the full sad-form. As by the results of previous section any formula can be provably transformed into this form, the completeness of the axiomatisation for whole $\overrightarrow{\exists}\mu TL$ follows immediately. Let us first describe a tableau construction for formulae in full sad-form, made possible by special properties that this normal form enjoys, shown in Lemmas 6.4.1 and 6.4.2.

Lemma 6.4.1 Let ϕ and ϕ' be $\overrightarrow{\exists}\mu TL$-formulae such that $\phi \vee \phi'$ is in sad-form. Then $\vdash \overrightarrow{\forall}(\phi \vee \phi') \Leftrightarrow (\overrightarrow{\forall}\phi) \vee (\overrightarrow{\forall}\phi')$.

Proof: Assume first that ϕ and ϕ' are pure path formulae. Since $\phi \vee \phi'$ is deterministic, by definition there exists a propositional formula ψ which is a deterministic choice for $\phi \vee \phi'$, i.e. $\models \phi \Rightarrow \psi$ and $\models \phi' \Rightarrow \neg\psi$. Since ϕ, ϕ' and ψ are pure path formulae, by Lemma 6.2.8 this implies $\vdash \phi \Rightarrow \psi$ and $\vdash \phi' \Rightarrow \neg\psi$. As ψ is a state formula, we have $\vdash (\overrightarrow{\forall}\psi) \vee (\overrightarrow{\forall}\neg\psi)$. As $\vdash \phi \Rightarrow \psi$ and $\vdash \phi' \Rightarrow \neg\psi$, then

$$\begin{aligned}
\vdash \overrightarrow{\forall}(\phi \vee \phi') &\Leftrightarrow (\overrightarrow{\forall}(\psi) \wedge \overrightarrow{\forall}(\phi \vee \phi') \vee \overrightarrow{\forall}(\neg\psi) \wedge \overrightarrow{\forall}(\phi \vee \phi'))\\
&\Leftrightarrow (\overrightarrow{\forall}(\psi \wedge (\phi \vee \phi')) \vee \overrightarrow{\forall}(\neg\psi \wedge (\phi \vee \phi')))\\
&\Leftrightarrow (\overrightarrow{\forall}\phi \vee \overrightarrow{\forall}\phi')
\end{aligned}$$

Let then ϕ and ϕ' be any $\overrightarrow{\exists}\mu TL$-formulae such that $\phi \vee \phi'$ is in sad-form. By definition $\phi \vee \phi'$ can be written in the form $\phi \vee \phi' = (\psi \vee \psi')[\overrightarrow{\Pi}\chi_1/x_1,\ldots\overrightarrow{\Pi}\chi_k/x_k]$ where $\psi \vee \psi'$ is a pure path formula in the sad-form. By the above, we know that $\vdash \overrightarrow{\forall}(\psi \vee \psi') \Leftrightarrow (\overrightarrow{\forall}\psi) \vee (\overrightarrow{\forall}\psi')$. But by Lemma 6.2.9 this implies

$$\vdash (\overrightarrow{\forall}(\psi \vee \psi') \Leftrightarrow (\overrightarrow{\forall}\psi) \vee (\overrightarrow{\forall}\psi'))[\overrightarrow{\Pi}\chi_1/x_1,\ldots\overrightarrow{\Pi}\chi_k/x_k]$$

i.e. $\vdash \overrightarrow{\forall}(\phi \vee \phi') \Leftrightarrow (\overrightarrow{\forall}\phi) \vee (\overrightarrow{\forall}\phi')$. □

name	application	name	application
$\vee\boldsymbol{L}$	$\dfrac{s,\ \ \Gamma\cup\{\phi\vee\phi'\},\ \ d}{s,\ \ \Gamma\cup\{\phi\},\ \ d}$	$\vee\boldsymbol{R}$	$\dfrac{s,\ \ \Gamma\cup\{\phi\vee\phi'\},\ \ d}{s,\ \ \Gamma\cup\{\phi'\},\ \ d}$
$\vec{\Pi}\vee\boldsymbol{L}$	$\dfrac{s,\ \ \Gamma\cup\{\vec{\Pi}(\phi\vee\phi')\},\ \ d}{s,\ \ \Gamma\cup\{\vec{\Pi}\phi\},\ \ d}$	$\vec{\Pi}\vee\boldsymbol{R}$	$\dfrac{s,\ \ \Gamma\cup\{\vec{\Pi}(\phi\vee\phi')\},\ \ d}{s,\ \ \Gamma\cup\{\vec{\Pi}\phi'\},\ \ d}$
$\wedge$	$\dfrac{s,\ \ \Gamma\cup\{\phi\wedge\phi'\},\ \ d}{s,\ \ \Gamma\cup\{\phi,\phi'\},\ \ d}$		
$\vec{\Pi}\wedge\boldsymbol{L}$	$\dfrac{s,\ \ \Gamma\cup\{\vec{\Pi}(\phi\wedge\psi)\},\ \ d}{s,\ \ \Gamma\cup\{(\vec{\Pi}\phi),\psi\},\ \ d}\ \ \mathbf{1}$	$\vec{\Pi}\wedge\boldsymbol{R}$	$\dfrac{s,\ \ \Gamma\cup\{\vec{\Pi}(\psi\wedge\phi)\},\ \ d}{s,\ \ \Gamma\cup\{\psi,(\vec{\Pi}\phi)\},\ \ d}\ \ \mathbf{1}$
$\vec{\Pi}\boldsymbol{\sigma}$	$\dfrac{s,\ \ \Gamma\cup\{\vec{\Pi}\sigma z.\phi\},\ \ d}{s,\ \ \Gamma\cup\{\vec{\Pi}u\},\ \ d\cdot(u,\sigma z.\phi)}\ \ \mathbf{2}$	$\vec{\Pi}\boldsymbol{U}$	$\dfrac{s,\ \ \Gamma\cup\{\vec{\Pi}u\},\ \ d}{s,\ \ \Gamma\cup\{\vec{\Pi}\phi[u/z]\},\ \ d}\ \ \mathbf{3}$
state	$\dfrac{s,\ \ \Gamma\cup\{\vec{\Pi}\psi\},\ \ d}{s,\ \ \Gamma\cup\{\psi\},\ \ d}\ \ \mathbf{1}$		
$\bigcirc$	$\dfrac{s,\ \ \Gamma_{\mathcal{Z}}\cup\{\vec{\forall}\bigcirc\phi_1,\ldots,\vec{\forall}\bigcirc\phi_n\}\cup\{\vec{\exists}\bigcirc\psi_1,\ldots,\vec{\exists}\bigcirc\psi_m\},\ \ d}{s\cdot 0,\Gamma_{\vec{\forall}},d\qquad s\cdot 1,\Gamma_{\vec{\forall}}\cup\{\vec{\exists}\psi_1\},d\quad\ldots\quad s\cdot m,\Gamma_{\vec{\forall}}\cup\{\vec{\exists}\psi_m\},d}\ \ \mathbf{4}$		
Note:	**1:** $\psi[d]$ is a state formula, **2:** u does not appear in d, **3:** $d(u)=\sigma z.\phi$ **4:** $\Gamma_{\mathcal{Z}}\subseteq\mathcal{Z}\cup\{\neg z\mid z\in\mathcal{Z}\}$ and $\Gamma_{\vec{\forall}}=\{\vec{\forall}\phi_1,\ldots,\vec{\forall}\phi_n\}$		

Table 6.1: Tableau rules for $\vec{\exists}\mu TL$-formulae in full sad-form

Lemma 6.4.2 If $\phi\wedge\phi'$ is strongly aconjunctive, then either

- ϕ is a state formula and $\vdash\vec{\Pi}(\phi\wedge\phi')\Leftrightarrow\phi\wedge(\vec{\Pi}\phi')$, or
- ϕ' is a state formula and $\vdash\vec{\Pi}(\phi\wedge\phi')\Leftrightarrow(\vec{\Pi}\phi)\wedge\phi'$.

Proof: It is clear from the definition of strong aconjunctivity that either ϕ or ϕ' is a state formula. Suppose ϕ is. Then $\vdash\phi\Leftrightarrow\vec{\forall}\phi\Leftrightarrow\vec{\exists}\phi$, implying

$$\vdash\vec{\forall}(\phi\wedge\phi')\Leftrightarrow(\vec{\forall}\phi)\wedge(\vec{\forall}\phi')\Leftrightarrow\phi\wedge(\vec{\forall}\phi')$$
$$\vdash\vec{\exists}(\phi\wedge\phi')\Rightarrow(\vec{\exists}\phi)\wedge(\vec{\exists}\phi')\Leftrightarrow\phi\wedge(\vec{\exists}\phi')$$
$$\vdash\phi\wedge(\vec{\exists}\phi')\Leftrightarrow(\vec{\forall}\phi)\wedge(\vec{\exists}\phi')\Rightarrow\vec{\exists}(\phi\wedge\phi')$$

□

Definition 6.4.3 Let ϕ be a state formula in full sad-form. A *tableau* T for ϕ is an infinite tree such that

- every node t of T is labelled with a triple (s_t,Γ_t,d_t) where $s_t\in\mathbb{N}^*$, Γ_t is a finite set of extended formulae in pnf, d_t a definition list such that $\Gamma_t[d_t]$ is a set of state formulae,
- the children of a node t of T are derived by using one of the rules in Table 6.1, and

- the root of T is labelled with $(\epsilon, \{\phi\}, \epsilon)$.

A node t of T is a $\bigcirc$-*node* iff the $\bigcirc$-rule is applied at t. For every node t of T and every child t' of t, the rule applied at node t induces a dependency relation $\rightarrow \subseteq \Gamma_t \times \Gamma_{t'}$ as in Def. 2.2.38, except for:

- if the rule is $\bigcirc$, then $\overset{\rightsquigarrow}{\forall}\bigcirc\phi \rightarrow \overset{\rightsquigarrow}{\forall}\phi$ for every formula of the form $\overset{\rightsquigarrow}{\forall}\bigcirc\phi \in \Gamma_t$, and $\overset{\rightsquigarrow}{\exists}\bigcirc\phi \rightarrow \overset{\rightsquigarrow}{\exists}\phi$ for every formula of the form $\overset{\rightsquigarrow}{\exists}\phi \in \Gamma_{t'}$.

A tableau T is *proper* iff there is no infinite path p of T, node t along p, and definition constant u such that $d_t(u) = \mu z.\phi$ for some ϕ, and $\overset{\rightsquigarrow}{\Pi}u \in \Gamma_{t'}$ for infinitely many nodes t' along p. A tableau T is *consistent* iff for every node t of T, the formula $\bigwedge \Gamma_t[d_t]$ is consistent. □

Lemma 6.4.4 Let ϕ be a state formula in full sad-form. If there is a proper consistent tableau T for ϕ, then ϕ is satisfiable.

Proof: A consistent tableau T naturally induces a model M. We can show that $M, s_t \models \psi$ for all $\psi \in \Gamma_t[d_t]$ and all nodes t of T, by using induction on the path quantifier depth of ψ and by reading linear bundled tableaux from T and applying Proposition 2.2.39. □

To construct a proper tableau for a consistent formula, we use a technique similar to Lemma 5.3.4 for strengthening minimal fixpoints. The method is based on the fact that if $(\overset{\rightsquigarrow}{\Pi}\mu z.\phi) \wedge \psi$ is consistent, then $(\overset{\rightsquigarrow}{\Pi}\phi[\mu z.(\phi \wedge \neg\psi)/z]) \wedge \psi$ is consistent. For $\overset{\rightsquigarrow}{\Pi} = \overset{\rightsquigarrow}{\exists}$ this holds for all $\mu z.\phi$, but for $\overset{\rightsquigarrow}{\Pi} = \overset{\rightsquigarrow}{\forall}$ the special properties of sad-form are required.

Lemma 6.4.5 Let $\mu z.\phi$ and ψ be $\overset{\rightsquigarrow}{\exists}\mu TL$-formulae such that $\mu z.\phi$ is strongly co-aconjunctive, and ψ is a state formula without free occurrences of z. If

$$(\overset{\rightsquigarrow}{\forall}\mu z.\phi) \wedge \psi \text{ is consistent,}$$

then

$$(\overset{\rightsquigarrow}{\forall}\phi[\mu z.(\phi \wedge \neg\psi)/z]) \wedge \psi \text{ is consistent.}$$

Proof: Assume that $(\overset{\rightsquigarrow}{\forall}\phi[\mu z.(\phi \wedge \neg\psi)/z]) \wedge \psi$ is inconsistent, which implies $\vdash (\overset{\rightsquigarrow}{\forall}\phi[\mu z.(\phi \wedge \neg\psi)/z]) \Rightarrow \neg\psi$. As ψ is a state formula, $\vdash \neg\psi \Leftrightarrow \overset{\rightsquigarrow}{\forall}\neg\psi$, and

$$\begin{aligned}
\vdash (\overset{\rightsquigarrow}{\forall}\phi[\mu z.(\phi \wedge \neg\psi)/z]) &\Rightarrow (\overset{\rightsquigarrow}{\forall}\phi[\mu z.(\phi \wedge \neg\psi)/z]) \wedge (\overset{\rightsquigarrow}{\forall}\neg\psi) \\
&\Rightarrow (\overset{\rightsquigarrow}{\forall}(\phi \wedge \neg\psi)[\mu z.(\phi \wedge \neg\psi)/z]) \\
&\Rightarrow (\overset{\rightsquigarrow}{\forall}\mu z.(\phi \wedge \neg\psi))
\end{aligned}$$

Since $\vdash (\overset{\rightsquigarrow}{\forall}\mu z.(\phi\wedge\neg\psi)) \Rightarrow \mu z.(\phi\wedge\neg\psi)$, then $\vdash (\overset{\rightsquigarrow}{\forall}\phi[\mu z.(\phi\wedge\neg\psi)/z]) \Rightarrow \mu z.(\phi\wedge\neg\psi)$. As $\mu z.\phi$ is strongly co-aconjunctive, this implies by the $\overset{\rightsquigarrow}{\forall}\mu$-induction rule that $\vdash (\overset{\rightsquigarrow}{\forall}\mu z.\phi) \Rightarrow \mu z.(\phi \wedge \neg\psi)$. As $\vdash \mu z.(\phi \wedge \neg\psi) \Rightarrow \neg\psi$, then $\vdash (\overset{\rightsquigarrow}{\forall}\mu z.\phi) \Rightarrow \neg\psi$, i.e. $(\overset{\rightsquigarrow}{\forall}\mu z.\phi) \wedge \psi$ is inconsistent. □

Lemma 6.4.6 If $\mu z.\phi$ is in the sad-form, there is a formula ϕ' such that

- $\vdash \phi \Leftrightarrow \phi'$,
- z is bindable in ϕ' and
- $\mu z.\phi'$ is strongly co-aconjunctive.

Proof: Assume first that $\mu z.\phi$ is a pure path formula. Define inductively a formula $\overline{\psi}$ for every subformula ψ of $\mu z.\phi$ by:

$$\begin{array}{rcl}
\overline{\psi} & = & \neg\psi \text{ for } \psi \in \mathcal{Z} \cup \{\bot, \top\} \\
\overline{\neg\psi} & = & \psi \\
\overline{\psi \wedge \psi'} & = & \overline{\psi} \vee \overline{\psi'} \\
\overline{\mu x.\psi} & = & \nu x.\overline{\psi}[\neg x/x] \\
\overline{\nu x.\psi} & = & \mu x.\overline{\psi}[\neg x/x] \\
\overline{\bigcirc\psi} & = & \bigcirc\overline{\psi} \\
\overline{\psi \vee \psi'} & = & (\gamma \wedge \overline{\psi}) \vee (\neg\gamma \wedge \overline{\psi'})
\end{array}$$

where γ is a deterministic choice for $\psi \vee \psi'$ not containing any variable bound by a fixpoint in $\mu z.\phi$.

Define then $\phi' = \neg\overline{\phi}$. It is easy to see that $\models \overline{\phi} \Leftrightarrow \neg\phi$, i.e. $\models \phi' \Leftrightarrow \phi$. By the completeness for pure path formulae (Lemma 6.2.8), this implies $\vdash \phi' \Leftrightarrow \phi$. Since z occurs only positively in ϕ, it does so in ϕ' as well, and z is bindable in ϕ'.

It is easy to see that $\nu z.\overline{\phi}[\neg z/z]$ is strongly aconjunctive, and that consequently $\mu z.\phi'$ is strongly co-aconjunctive.

Let then $\mu z.\phi$ be an arbitrary $\overset{\rightsquigarrow}{\exists}\mu TL$-formula in sad-form. By definition $\mu z.\phi$ can be written in the form $\mu z.\phi = (\mu z.\psi)[\overset{\rightsquigarrow}{\Pi}\chi_1/x_1, \ldots \overset{\rightsquigarrow}{\Pi}\chi_k/x_k]$ where $\mu z.\psi$ is a pure path formula in the sad-form, and z does not occur in any χ_i. By the above we know that there is a formula ψ' fulfilling the claim of the lemma with respect to $\mu z.\psi$. Define then $\phi' = \psi'[\overset{\rightsquigarrow}{\Pi}\chi_1/x_1, \ldots \overset{\rightsquigarrow}{\Pi}\chi_k/x_k]$. Since $\vdash \psi \Leftrightarrow \psi'$, we have $\vdash \phi \Leftrightarrow \phi'$ by Lemma 6.2.9. It is clear that z is bindable in ϕ' and that $\mu z.\phi'$ is strongly co-aconjunctive. □

Lemma 6.4.7 Let $\mu z.\phi$ be a $\overset{\leadsto}{\exists}\mu TL$-formula in the sad-form, and ψ a state formula without free occurrences of z. If

$$(\overset{\leadsto}{\forall}\mu z.\phi) \wedge \psi \text{ is consistent,}$$

then

$$(\overset{\leadsto}{\forall}\phi[\mu z.(\phi \wedge \neg\psi)/z]) \wedge \psi \text{ is consistent.}$$

Proof: Take the formula ϕ' provided by Lemma 6.4.6. As $\vdash \phi \Leftrightarrow \phi'$, we have $\vdash (\overset{\leadsto}{\forall}\mu z.\phi) \Leftrightarrow (\overset{\leadsto}{\forall}\mu z.\phi')$ and $\vdash (\overset{\leadsto}{\forall}\phi[\mu z.(\phi \wedge \neg\psi)/z]) \Leftrightarrow (\overset{\leadsto}{\forall}\phi'[\mu z.(\phi' \wedge \neg\psi)/z])$.

Therefore, if $(\overset{\leadsto}{\forall}\mu z.\phi) \wedge \psi$ is consistent, $(\overset{\leadsto}{\forall}\mu z.\phi') \wedge \psi$ is consistent, implying by Lemma 6.4.5 that $(\overset{\leadsto}{\forall}\phi'[\mu z.(\phi' \wedge \neg\psi)/z]) \wedge \psi$ is consistent, and further that $(\overset{\leadsto}{\forall}\phi[\mu z.(\phi \wedge \neg\psi)/z]) \wedge \psi$ is consistent. □

Lemma 6.4.8 Let ϕ be a $\overset{\leadsto}{\exists}\mu TL$-formula, and ψ a state formula without free occurrences of z. If

$$(\overset{\leadsto}{\exists}\mu z.\phi) \wedge \psi \text{ is consistent,}$$

then

$$(\overset{\leadsto}{\exists}\phi[\mu z.(\phi \wedge \neg\psi)/z]) \wedge \psi \text{ is consistent.}$$

Proof: If $(\overset{\leadsto}{\exists}\phi[\mu z.(\phi \wedge \neg\psi)/z]) \wedge \psi$ is inconsistent, $\vdash (\overset{\leadsto}{\exists}\phi[\mu z.(\phi \wedge \neg\psi)/z]) \Rightarrow \neg\psi$. Then $\vdash \phi[\mu z.(\phi \wedge \neg\psi)/z] \Rightarrow (\overset{\leadsto}{\exists}\phi[\mu z.(\phi \wedge \neg\psi)/z]) \Rightarrow \neg\psi$, and

$$\vdash \phi[\mu z.(\phi \wedge \neg\psi)/z] \Rightarrow (\phi \wedge \neg\psi)[\mu z.(\phi \wedge \neg\psi)/z] \Rightarrow \mu z.(\phi \wedge \neg\psi)$$

By the μ-ind. rule then $\vdash \mu z.\phi \Rightarrow \mu z.(\phi \wedge \neg\psi)$ and $\vdash (\overset{\leadsto}{\exists}\mu z.\phi) \Rightarrow (\overset{\leadsto}{\exists}\mu z.(\phi \wedge \neg\psi))$. As $\vdash \mu z.(\phi \wedge \neg\psi) \Rightarrow \neg\psi$ and ψ is a state formula,

$$\vdash (\overset{\leadsto}{\exists}\mu z.(\phi \wedge \neg\psi)) \Rightarrow (\overset{\leadsto}{\exists}\neg\psi) \Rightarrow \neg\psi$$

Therefore $\vdash (\overset{\leadsto}{\exists}\mu z.\phi) \Rightarrow \neg\psi$, and $(\overset{\leadsto}{\exists}\mu z.\phi) \wedge \psi$ is inconsistent. □

Definition 6.4.9 Let ϕ be a state formula in full sad-form. A *strong tableau* T for ϕ is an infinite tree, every node t of which is labelled with a 4-tuple $(s_t, \Gamma_t, d_t, d_t^s)$ where

- (s_t, Γ_t, d_t) is a tableau node label as defined in Def. 6.4.3, and d_t^s is a definition list such that if
$$d_t = (u_0, \sigma z_0.\phi_0) \ldots (u_n, \sigma z_n.\phi_n)$$
then
$$d_t^s = (u_0, \sigma z_0.\phi_0 \wedge \alpha_0) \ldots (u_n, \sigma z_n.\phi_n \wedge \alpha_n)$$
for some state formulae α_i (possibly $\top$),

name	application
$\widetilde{\vec{\Pi}}\boldsymbol{\sigma'}$	$\dfrac{s,\ \ \Gamma \cup \{\widetilde{\vec{\Pi}}\sigma z.\phi\},\ \ d,\ \ d^s}{s,\ \ \Gamma \cup \{\widetilde{\vec{\Pi}}u\},\ \ d',\ \ d^{s\prime}}$ **1**
$\widetilde{\vec{\Pi}}\boldsymbol{U\nu}$	$\dfrac{s,\ \ \Gamma \cup \{\widetilde{\vec{\Pi}}u\},\ \ d,\ \ d^s}{s,\ \ \Gamma \cup \{\widetilde{\vec{\Pi}}\phi[u/z]\},\ \ d,\ \ d^s}$ **2**
$\widetilde{\vec{\Pi}}\boldsymbol{U\mu}$	$\dfrac{s,\ \ \Gamma \cup \{\widetilde{\vec{\Pi}}u\},\ \ d,\ \ d^s}{s,\ \ \Gamma \cup \{\widetilde{\vec{\Pi}}\phi[u/z]\},\ \ d,\ \ d^{s\prime}}$ **3**

Note: **1**: u does not appear in d, $d' = d \cdot (u, \sigma z.\phi)$, $d'_s = d_s \cdot (u, \sigma z.\phi)$.
2: $d(u) = \nu z.\phi$
3: if $d = (u_0, \sigma z_0.\phi_0) \dots (u_n, \sigma z_n.\phi_n)$, $u = u_m$,
$d(u_m) = \mu z.\phi$ and $d^s(u_m) = \mu z.(\phi \wedge \alpha)$, then
$d^{s\prime}(u_i) = d^s(u_i)$ for $0 \leq i < m$, $d^{s\prime}(u_i) = d(u_i)$ for $m < i \leq n$, and
$d^{s\prime}(u_m) = \mu z.(\phi \wedge \alpha \wedge \alpha')$ where $\alpha' = \mathrm{pnf}(\widetilde{\vec{\forall}}\neg \bigwedge \Gamma[d^x])$ and
$d^x(u_i) = d^s(u_i)$ for $0 \leq i \leq m$, and $d^x(u_i) = d(u_i)$ for $m < i \leq n$

Table 6.2: Strong tableau rules for $\widetilde{\vec{\exists}}\mu TL$

- the children of a node t are derived by the rules $\vee L$, $\vee R$, $\widetilde{\vec{\Pi}}\vee L$, $\widetilde{\vec{\Pi}}\vee R$, $\wedge$, $\widetilde{\vec{\Pi}}\wedge L$, $\widetilde{\vec{\Pi}}\wedge R$, state or $\bigcirc$, which are as in Table 6.1, with the extra definition list passed on unchanged, or by the rules $\widetilde{\vec{\Pi}}\sigma'$, $\widetilde{\vec{\Pi}}U\nu$ or $\widetilde{\vec{\Pi}}U\mu$ in Table 6.2, and
- the root of T is labelled with $(\epsilon, \{\phi\}, \epsilon, \epsilon)$.

A strong tableau T being *proper* is defined as in Def. 6.4.3. A strong tableau T is *consistent* iff for every node t of T, the formula $\bigwedge \Gamma_t[d^s_t]$ is consistent. □

Lemma 6.4.10 Let T be a (strong) tableau for a $\widetilde{\vec{\exists}}\mu TL$-formula ϕ. For every node t of T every definition constant u is active in at most one formula $\gamma \in \Gamma_t$.

Proof: Let us see first that for every node t of T and every constant u, if u occurs in $\phi \in \Gamma_t$, then ϕ is of the form $\phi = \widetilde{\vec{\Pi}}\phi'$, and u does not occur in the scope of a path quantifier in ϕ'. This holds for the root of T, and clearly all rules except state, $\widetilde{\vec{\Pi}}\wedge L$ and $\widetilde{\vec{\Pi}}\wedge R$ preserve the property. If state-rule is applied to $\widetilde{\vec{\Pi}}\psi$ at t, $\psi[d_t]$ is a state formula, and as for every u $d(u) = \sigma z.\phi$ for some ϕ, no u can occur in ψ outside path quantifiers. As the claim holds for t, no u can occur in ψ in the scope of a path quantifier, either. The cases for $\widetilde{\vec{\Pi}}\wedge L$ and $\widetilde{\vec{\Pi}}\wedge R$ are analogous.

The main claim of the lemma holds for the root of T, and clearly all rules except $\wedge$, $\widetilde{\vec{\Pi}}\wedge L$ and $\widetilde{\vec{\Pi}}\wedge R$ preserve the property. By the above, no u can occur in $\phi \wedge \phi' \in \Gamma_t$, so $\wedge$-rule preserves the claim. For $\widetilde{\vec{\Pi}}\wedge L$ and $\widetilde{\vec{\Pi}}\wedge R$, as seen above, no u can occur in the ψ for which $\psi[d_t]$ is a state formula, so these rules preserve the property, as well. □

Lemma 6.4.11 Let T be a strong tableau for a $\tilde{\vec{\exists}}\mu TL$-formula. If T is consistent, then T is proper.

Proof: Take a strong tableau T which is not proper. Then there is an infinite path p of T, a node t along p and a minimal constant u such that $\tilde{\vec{\Pi}}u \in \Gamma_{p(i)}$ for infinitely many $i \in \mathbb{N}$. Let $(s_0, \Gamma_0, d_0, d_0^s)(s_1, \Gamma_1, d_1, d_1^s)\ldots$ be the sequence of labels along p and let $u = u_m$. Assume that m is the smallest index such that u_m is a minimal constant and $\tilde{\vec{\Pi}}u_m \in \Gamma_{p(i)}$ for infinitely many $i \in \mathbb{N}$. Then there is some bound $n \in \mathbb{N}$ such that for all $i \geq n$ and all minimal constants u_j for which $j < m$, $\tilde{\vec{\Pi}}u_j \notin \Gamma_i$. This implies $d_i^s(u_j) = d_n^s(u_j)$ for all $0 \leq j < m$ and $i \geq n$.

For every $i \geq n$, define a definition list d_i^x by: $d_i^x(u_j) = d_i^s(u_j)$ for every $0 \leq j \leq m$, and $d_i^x(u_j) = d_i(u_j)$ for every $j > m$. Let d_i^y be a definition list like d_i^x, except for $d_m^y(u_j) = d_i(u_j)$. For a set Γ of state formulae in pnf, define $\text{cl}(\Gamma)$, the closure of Γ, as the minimal set which contains Γ and fulfils the following requirements: if $\tilde{\vec{\Pi}}(\phi \wedge \phi') \in \text{cl}(\Gamma)$ or $\tilde{\vec{\Pi}}(\phi \vee \phi') \in \text{cl}(\Gamma)$, then $\tilde{\vec{\Pi}}\phi, \tilde{\vec{\Pi}}\phi' \in \text{cl}(\Gamma)$, if $\phi \wedge \phi' \in \text{cl}(\Gamma)$ or $\phi \vee \phi' \in \text{cl}(\Gamma)$, then $\phi, \phi' \in \text{cl}(\Gamma)$, if $\tilde{\vec{\Pi}}\sigma z.\phi \in \text{cl}(\Gamma)$ then $\tilde{\vec{\Pi}}\phi[\sigma z.\phi/z] \in \text{cl}(\Gamma)$, if $\tilde{\vec{\Pi}}\bigcirc\phi \in \text{cl}(\Gamma)$ then $\tilde{\vec{\Pi}}\phi \in \text{cl}(\Gamma)$, and if $\tilde{\vec{\Pi}}\phi \in \text{cl}(\Gamma)$ and ϕ is a state formula, then $\phi \in \text{cl}(\Gamma)$. Define $X = \text{cl}(\Gamma_n[d_n^y])$. Notice that X is clearly finite. As $\tilde{\vec{\Pi}}U\mu$-rule is not applied to any of $u_0, \ldots, u_{m-1}$, $\Gamma_i[d_i^y] \subseteq X$ for all $i \geq n$.

Let then $n \leq i_0 \leq i_1 \leq \ldots$ be an infinite sequence of indices such that for every $j \in \mathbb{N}$, $\tilde{\vec{\Pi}}u_m \in \Gamma_{i_j}$ and the $\tilde{\vec{\Pi}}U\mu$-rule is applied to $\tilde{\vec{\Pi}}u_m$ to obtain Γ_{i_j+1} from Γ_{i_j}. For each $j \in \mathbb{N}$, define $\Gamma'_{i_j} = \Gamma_{i_j} \setminus \{\tilde{\vec{\Pi}}u_m\}$. By Lemma 5.3.6, u_m is not active in any formula in Γ'_{i_j}. Therefore, $\Gamma'_{i_j}[d_{i_j}^x] = \Gamma'_{i_j}[d_{i_j}^y] \subseteq X$ for all $j \in \mathbb{N}$.

Since X is finite there must be some points k and l in the sequence $i_0, i_1, \ldots$ such that $k < l$ and $\Gamma'_k[d_k^x] = \Gamma'_l[d_l^x]$. As the $\tilde{\vec{\Pi}}U\mu$-rule is applied to $\tilde{\vec{\Pi}}u_m$ at k and as the $\tilde{\vec{\Pi}}U\mu$-rule is not applied to any of $u_0, \ldots, u_{m-1}$ between k and l, this means that $\vdash (\tilde{\vec{\Pi}}u_m[d_l^s]) \Rightarrow (\neg \bigwedge \Gamma'_k[d_k^x]) \Leftrightarrow (\neg \bigwedge \Gamma'_l[d_l^x])$. But then

$$\begin{aligned}
\vdash (\bigwedge \Gamma_l[d_l^s]) &\Leftrightarrow (\bigwedge \Gamma'_l[d_l^s] \wedge (\tilde{\vec{\Pi}}u_m[d_l^s])) \\
&\Rightarrow (\bigwedge \Gamma'_l[d_l^x] \wedge (\tilde{\vec{\Pi}}u_m[d_l^s])) \\
&\Rightarrow (\bigwedge \Gamma'_l[d_l^x] \wedge (\neg \bigwedge \Gamma'_l[d_l^x])) \\
&\Leftrightarrow \bot
\end{aligned}$$

meaning that T is not consistent. □

Lemma 6.4.12 Let ϕ be a state formula in full sad-form, and let T be a strong tableau for ϕ. For every node t of T and every formula $\psi \in \Gamma_t$, $\psi[d_t]$ is in full sad-form, and if ψ is of the form $\psi = \tilde{\vec{\Pi}}\psi'$, then $\psi'[d_t^s]$ is in sad-form.

Proof: Showing that for every node t of T and every formula $\psi \in \Gamma_t$, $\psi[d_t]$ is in full sad-form is done inductively on T. The root of T clearly fulfils the claim.

Every rule preserves the validity of the claim, since ϕ is in full sad-form iff $\tilde{\vec{\Pi}}\phi$ is in full sad-form iff $\bigcirc\phi$ is in full sad-form, if $\phi \wedge \phi'$ or $\phi \vee \phi'$ is in full sad-form, then so are ϕ and ϕ', and if $\sigma z.\phi$ is in full sad-form, then so is $\phi[\sigma z.\phi/z]$.

Take then any $\tilde{\vec{\Pi}}\psi' \in \Gamma_t$. As by the above $\tilde{\vec{\Pi}}\psi'[d_t]$ is in full sad-form, $\psi'[d_t]$ is in sad-form. For every t and u, if $d_t(u) = \sigma z.\phi$, $d_t^s(u) = \sigma z.(\phi \wedge \alpha)$ where $\alpha = \bigwedge \mathrm{pnf}(\tilde{\vec{\forall}}\alpha_i)$ for some α_i. It is then easy to see that replacing $d_t(u)$ by $d_t^s(u)$ in $\psi'[d_t]$ does not affect the fact that the formula is in sad-form, i.e. $\psi'[d_t^s]$ is in sad-form. □

Lemma 6.4.13 Let ϕ be a state formula in full sad-form. If ϕ is consistent, then there is a consistent strong tableau for ϕ.

Proof: Let us show that given any node t of a strong tableau T such that $\bigwedge \Gamma_t[d_t^s]$ is consistent, there is some rule that can be applied to node t so that for every resulting child t', $\bigwedge \Gamma_{t'}[d_{t'}^s]$ is consistent. As ϕ is consistent, we can then build a strong tableau T for ϕ inductively, starting from $(\epsilon, \{\phi\}, \epsilon, \epsilon)$ and always applying some rule for which the resulting children are consistent.

Take any label $(s_t, \Gamma_t, d_t, d_t^s)$ such that $\bigwedge \Gamma_t[d_t^s]$ is consistent. By Lemma 6.4.12 we know that for all $\tilde{\vec{\Pi}}\psi \in \Gamma_t$, $\psi[d_t^s]$ is in sad-form.

Assume that the $\bigcirc$-rule can be applied at t. Then for all the children t' of t, $\bigwedge \Gamma_{t'}[d_{t'}^s]$ is consistent, since $\vdash \tilde{\vec{\forall}}\bigcirc\phi \Leftrightarrow \tilde{\vec{\forall}}\bigcirc\tilde{\vec{\forall}}\phi$, $\vdash \tilde{\vec{\exists}}\bigcirc\phi \Leftrightarrow \tilde{\vec{\exists}}\bigcirc\tilde{\vec{\exists}}\phi$ and $\vdash (\tilde{\vec{\forall}}\bigcirc\tilde{\vec{\forall}}\phi) \wedge (\tilde{\vec{\exists}}\bigcirc\tilde{\vec{\exists}}\phi') \Rightarrow (\tilde{\vec{\exists}}\bigcirc(\tilde{\vec{\forall}}\phi \wedge \tilde{\vec{\exists}}\phi'))$.

If the $\bigcirc$-rule cannot be applied at t, there is either some $\psi \in \Gamma_t$ which is not a basic state formula, or there is some $\tilde{\vec{\Pi}}\psi \in \Gamma_t$ such that ψ is not of the form $\bigcirc\psi'$. In the first case either $\psi = \psi_1 \vee \psi_2$, and $\vee L$ or $\vee R$ can be applied to ψ to yield a consistent child, or $\psi = \psi_1 \wedge \psi_2$, and $\wedge$ can be applied to yield a consistent child.

Assume then that there is some $\tilde{\vec{\Pi}}\psi \in \Gamma_t$, $\psi \neq \bigcirc\psi'$ for any ψ'. If $\tilde{\vec{\Pi}}\psi$ is of the form that the state, $\tilde{\vec{\Pi}}\sigma'$ or $\tilde{\vec{\Pi}}U\nu$-rules can be applied to it, it is obvious that the resulting child is consistent.

If $\tilde{\vec{\Pi}}\psi = \tilde{\vec{\exists}}(\psi_1 \vee \psi_2)$, one of $\tilde{\vec{\Pi}}\vee L$ or $\tilde{\vec{\Pi}}\vee R$ can be applied to yield a consistent child, as $\vdash \tilde{\vec{\exists}}(\psi_1[d_t^s] \vee \psi_2[d_t^s]) \Leftrightarrow (\tilde{\vec{\exists}}\psi_1[d_t^s] \vee \tilde{\vec{\exists}}\psi_2[d_t^s])$. If $\tilde{\vec{\Pi}}\psi = \tilde{\vec{\forall}}(\psi_1 \vee \psi_2)$, one of $\tilde{\vec{\Pi}}\vee L$ or $\tilde{\vec{\Pi}}\vee R$ can be applied to yield a consistent child, as $\psi_1[d_t^s] \vee \psi_2[d_t^s]$ being in sad-form implies by Lemma 6.4.1 that $\vdash \tilde{\vec{\forall}}(\psi_1[d_t^s] \vee \psi_2[d_t^s]) \Leftrightarrow (\tilde{\vec{\forall}}\psi_1[d_t^s] \vee \tilde{\vec{\forall}}\psi_2[d_t^s])$.

If $\tilde{\vec{\Pi}}\psi = \tilde{\vec{\Pi}}(\psi_1 \wedge \psi_2)$, one of $\tilde{\vec{\Pi}}\wedge L$ or $\tilde{\vec{\Pi}}\wedge R$ can be applied to yield a consistent child, as $\psi_1[d_t^s] \wedge \psi_2[d_t^s]$ being in sad-form implies by Lemma 6.4.2 that either $\psi_1[d_t^s]$ is a state formula and $\vdash \tilde{\vec{\Pi}}(\psi_1[d_t^s] \wedge \psi_2[d_t^s]) \Leftrightarrow (\psi_1[d_t^s] \wedge (\tilde{\vec{\Pi}}\psi_2[d_t^s]))$, or vice versa.

Finally, assume that $\tilde{\vec{\Pi}}\psi = \tilde{\vec{\Pi}}u$ and the $\tilde{\vec{\Pi}}U\mu$-rule can be applied to $\tilde{\vec{\Pi}}u$. Let $\Gamma = \Gamma_t \setminus \{\tilde{\vec{\Pi}}u\}$, $d^s = d_t^s$ and let $u = u_m$, z, ϕ, α, α', d^x and $d^{s\prime}$ be as in the definition of the $\widetilde{\vec{\Pi}}U\mu$-rule. Clearly $\vdash (\bigwedge(\Gamma \cup \{\widetilde{\vec{\Pi}}u\})[d^s]) \Rightarrow (\bigwedge(\Gamma \cup \{\widetilde{\vec{\Pi}}u\})[d^x])$, implying that $\bigwedge(\Gamma \cup \{\widetilde{\vec{\Pi}}u\})[d^x]$ is consistent, i.e. that

$$(\bigwedge \Gamma[d^x]) \wedge (\widetilde{\vec{\Pi}}u[d^x]) \;=\; (\bigwedge \Gamma[d^x]) \wedge \widetilde{\vec{\Pi}}(\mu z.\phi \wedge \alpha)[d^y]$$

is consistent, where $d^y(u_i) = d^x(u_i) = d^s(u_i)$ for $1 \leq i < m$ and $d^y(u_i)$ is undefined otherwise.

As $\widetilde{\vec{\Pi}}(\mu z.\phi \wedge \alpha)[d^y] \in \Gamma_t[d^s_t]$, $\mu z.(\phi \wedge \alpha)[d^y]$ is in sad-form. By Lemmas 6.4.8 and 6.4.7, this implies that $(\bigwedge \Gamma[d^x]) \wedge \widetilde{\vec{\Pi}}((\phi \wedge \alpha)[\mu z.(\phi \wedge \alpha \wedge \alpha')/z])[d^y]$ and therefore $(\bigwedge \Gamma[d^x]) \wedge \widetilde{\vec{\Pi}}(\phi[\mu z.(\phi \wedge \alpha \wedge \alpha')/z])[d^y] = (\bigwedge \Gamma[d^x]) \wedge \widetilde{\vec{\Pi}}(\phi[u/z])[d^{s\prime}]$ is consistent. Since by Lemma 6.4.10 $u = u_m$ is active in at most one formula in Γ_t and it is clearly active in $\widetilde{\vec{\Pi}}u$, it is not active in any formula in Γ. Consequently, $\vdash (\bigwedge \Gamma[d^x]) \Leftrightarrow (\bigwedge \Gamma[d^{s\prime}])$, meaning that

$$(\bigwedge \Gamma[d^{s\prime}]) \wedge \widetilde{\vec{\Pi}}(\phi[u/z])[d^{s\prime}] \;=\; \bigwedge(\Gamma \cup \{\widetilde{\vec{\Pi}}\phi[u/z]\})[d^{s\prime}]$$

is consistent. □

Proposition 6.4.14 If ϕ is a consistent state formula in full sad-form, then ϕ is satisfiable.

Proof: Direct from Lemmas 6.4.4, 6.4.11 and 6.4.13. □

Theorem 6.4.15 If ϕ is consistent, ϕ is satisfiable.

Proof: Take any consistent ϕ. By Proposition 6.3.4 there is a ϕ' is full sad-form such that $\vdash \phi \Leftrightarrow \phi'$, implying that ϕ' is consistent. As $\vdash \phi' \Rightarrow \widetilde{\vec{\exists}}\phi'$, $\widetilde{\vec{\exists}}\phi'$ is consistent. Since $\widetilde{\vec{\exists}}\phi'$ is consistent and in full sad-form, by Proposition 6.4.14 $\widetilde{\vec{\exists}}\phi'$ is satisfiable, implying that ϕ' is satisfiable, and as $\vdash \phi \Leftrightarrow \phi'$ implies $\models \phi \Leftrightarrow \phi'$ by the soundness theorem 6.2.7, ϕ is satisfiable. □

Corollary 6.4.16 [**Completeness for** $\widetilde{\vec{\exists}}\mu TL$] For any $\widetilde{\vec{\exists}}\mu TL$-formula ϕ, if $\models \phi$ then $\vdash \phi$.

Proof: If $\models \phi$, then $\neg\phi$ is not satisfiable, therefore not consistent, implying $\vdash \neg\neg\phi$, i.e. $\vdash \phi$, by Theorem 6.4.15. □

6.5 Discussion

In the sections above we described an axiomatisation of $\widetilde{\vec{\exists}}\mu TL$ and proved its completeness with respect to the class of 'normal' models, or suffix, fusion and limit closed extended models. Related to this, there are three open questions, two of which we judge to be easy and one hard. The two easier questions are about axiomatising $\vec{\exists}\mu TL$ with respect to the class of models which are suffix and fusion closed but not necessarily limit closed, and axiomatising it with respect to the class of models which are suffix closed but not necessarily fusion or limit closed. In this respect we conjecture the following.

Conjecture 6.5.1 Let ϕ be a $\vec{\exists}\mu TL$-formula. Then $\vdash \phi$ (without the $\vec{\exists}\nu$-induction rule) iff $\hat{M} \models \phi$ for all extended models $\hat{M} = (M, P)$ for which P is suffix and fusion closed. $\square$

Conjecture 6.5.2 Let ϕ be a $\vec{\exists}\mu TL$-formula. Then $\vdash \phi$ (without the axiom ax8 and the $\vec{\exists}\nu$-induction rule) iff $\hat{M} \models \phi$ for all extended models $\hat{M} = (M, P)$ for which P is suffix closed. $\square$

The hard open problem is naturally that of completely axiomatising the usual extended computation tree logic CTL^* with respect to the suffix, fusion and limit-closed models. This may appear paradoxical, since CTL^* is after all a sublogic of $\vec{\exists}\mu TL$. However, there are two reasons which prevent a direct transferral of the current completeness proof to CTL^*. First, the principle of $\vec{\exists}\nu$-induction cannot be expressed directly in the more restricted language of CTL^*, and secondly, when CTL^*-formulae are transferred to the deterministic normal form, the result is not necessarily in CTL^* any more.

However, we believe that the current work outlines one potential way of attacking the completeness problem for CTL^*. First, the presence of the $\vec{\exists}\nu$-induction rule here leads us to believe that some similar proof principle, allowing us to join infinitely many finite path segments to a single path, will be needed for CTL^*, as well. One possible candidate is the axiom schema

$$\vdash \vec{\forall}\mathbf{G}(\phi \Rightarrow \vec{\exists}\mathbf{F}\phi) \Rightarrow (\phi \Rightarrow \vec{\exists}\mathbf{G}\mathbf{F}\phi)$$

for state formulae ϕ, although it is not clear whether this is sufficiently strong. Secondly, we believe that it should be possible to recast the current proof in a form where the explicit transformation to deterministic normal form is not required, but this is done implicitly in the process of building the model for a consistent formula. If this can be done, then the inability to express the deterministic formulae in CTL^* would not necessarily collapse the proof. However, at the current stage this is still speculation and the axiomatisation problem for CTL^* remains open.

Chapter 7

Conclusion

The main theme of the current work has been an examination of the relations between automata and fixpoint calculi, and the application of these relations to help us understand automata via fixpoint calculi and fixpoint calculi via automata. In Chapter 3 we saw that many of the known results relating fixpoint and quantifier-based second-order calculi and automata to each other could be obtained uniformly from the notions of first recurrence automata and fixpoint constructions for ordinary automata. In Chapter 4 we presented a tableau decision system for linear time and modal mu-calculi, noticing that this could also be viewed as a transformation to an automaton-like strongly aconjunctive form. Then, in Chapters 5 and 6 we reached the most important new contribution of the thesis, proving the completeness of two axiomatisations using normal forms inspired by different kinds of automata. Let us look now briefly at some issues worth further study.

During the course of the work we identified and pointed out several open axiomatisation problems. The more substantial ones are the question of axiomatising the strong second-order languages SnS or $\exists Kn$, and that of axiomatising the full computation tree logic CTL^* with respect to the suffix, fusion and limit closed models. Problems that would appear to be somewhat more tractable include the axiomatisation problem for the weak second-order language $\overset{\mathrm{w}}{\exists} Kn$, and the axiomatisation problems for $\overset{\leadsto}{\exists}\mu TL$ with respect to the classes of suffix closed or suffix and fusion closed models.

In Chapter 3 we described fixpoint constructions for ordinary Büchi and Rabin automata, but not for the third main class of automata in the current work, the first recurrence automata. Such constructions for ordinary first recurrence automata, i.e. restricted mu-calculus formulae, can naturally be derived as instances of the more general translation of Chapter 4. However, we believe that spelling out the fixpoint constructions for FR-automata explicitly may clarify the relation of fixpoints and strong aconjunctivity, and also that this may lead to an alternative account of Walukiewicz's completeness proof for the modal mu-calculus.

A different research direction is to look more closely at the computational aspects of the work and the efficiency of the translations and decision methods. In Section 3.2.4 we described the decision procedure for ordinary first recurrence automata, working in linear time, but noticed that the requirement of the tree-like structure of these automata causes an exponential penalty when translating ordinary Rabin or Büchi automata to them. This is an example of a more general 'unnecessary' exponential gap in conciseness between automata and formulae, caused by the fact that syntactically a formula is always a tree, whereas an automaton does not need to be one. In Section 3.2.4 we discussed briefly the possibility of avoiding this problem by relaxing the structural requirements for first recurrence automata so that they become more general hierarchical graphs instead of trees. These generalised first recurrence automata, which have a natural correspondence with mu-calculus formulae with simultaneous vectorial fixpoints $\nu(x_0, \ldots, x_k).(\phi_0, \ldots, \phi_k)$ and $\mu(x_0, \ldots, x_k).(\phi_0, \ldots, \phi_k)$, would appear to be a very interesting research area.

In Chapter 4 we described an elementary decision procedure for linear-time and modal mu-calculi. However, the main point of emphasis there was not efficiency and the derived complexity bounds are not optimal. Nevertheless, since the tableaux are based on the same principles as the most efficient known automata-theoretic decision procedures, we believe that with sufficient care on the choice of constant naming, it should be possible to achieve the same level of computational complexity.

Another more practical issue related to the tableau decision system is the choice of modalities in the modal mu-calculus. Here we have used the framework of models with a fixed branching degree and an ordering on children, and the modalities *for the i-th child.* However, if we assume, for example, that a model reflects the execution of a concurrent system, this framework is artificial and overly restrictive. It would be more natural to consider trees of varying degrees of branching, and the modal operators *for all children* and *for some child.* We believe that the tableau decision system should extend smoothly to such a model and language. This idea can naturally be lifted to the whole endeavour of relating automata and fixpoint calculi, by trying to characterise a notion of automata corresponding to formulae with such modalities. Especially if we try to keep the important link between ordinary automata and second-order quantification as in Section 3.3, it is not immediately clear what the most appropriate notion of automata would be.

On a more general note, the study of mu-calculi has recently taken several steps forward quite rapidly; the longstanding open problem of providing a complete axiomatisation and that of the the non-collapse of the alternation depth hierarchy have been settled. However, this does not mean that all fundamental questions related to mu-calculi would have been answered. In particular the question of the existence of a polynomial-time model-checking algorithm is still very much open.

Bibliography

[1] Arnold, A.: Logical definability of fixed points, in *Theoretical Computer Science*, vol. 61, 1988, pp. 289-297

[2] Arnold, A.: An initial semantics for the μ-calculus on trees and Rabin's complementation lemma, in *Theoretical Computer Science*, vol. 148, 1995, pp. 121-132

[3] Arnold, A. & Niwiński, D.: Fixed point characterisation of Büchi automata on infinite trees, in *Journal of Inf. Proc. and Cybernetics (EIK)*, vol. 8-9, 1990, pp. 451-459

[4] Arnold, A. & Niwiński, D.: Fixed point characterisation of weak monadic logic definable sets of trees, in Nivat, M. & Podelski, A. (eds.): *Tree Automata and Languages*, Elsevier, 1992, pp. 159-188

[5] de Bakker, J. W. & de Roever, W. P.: A calculus for recursive program schemes, in Nivat, M. (ed.): *Automata, Languages and Programming, Proceedings*, North-Holland, 1973, pp. 167-196

[6] Banieqbal, B. & Barringer, H.: Temporal logic with fixed points, in *Temporal Logic in Specification*, LNCS vol. 398, Springer-Verlag, 1989, pp. 62-74

[7] Barringer, H. & Kuiper, R. & Pnueli, A.: Now you may compose temporal logic specifications, in *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, 1984, pp. 51-63

[8] Barringer, H. & Kuiper, R. & Pnueli, A.: A compositional temporal approach to a CSP-like language, in *Formal Models in Programming*, Elsevier, 1985, pp. 207-227

[9] Barringer, H. & Kuiper, R. & Pnueli, A.: A really abstract concurrent model and its temporal logic, in *Conference Record of the 13th Annual ACM Symposium on Principles of Programming Languages*, 1986, pp. 173-183

[10] Bernholtz, O. & Vardi, M. Y. & Wolper, P.: An automata-theoretic approach to branching time model checking, in *Computer Aided Verification, 6th International Workshop, CAV'94, Proceedings*, LNCS vol. 818, 1994, pp. 142-155

[11] Bradfield, J.: *Verifying Temporal Properties of Systems with Applications to Petri Nets*, PhD thesis CST-83-91, University of Edinburgh, Department of Computer Science, 1991

[12] Bradfield, J.: The modal mu-calculus alternation hierarchy is strict, in *CONCUR'96: Concurrency Theory, 7th International Conference, Proceedings*, LNCS vol. 1119, Springer-Verlag, 1996, pp. 233-246

[13] Bradfield, J. & Stirling, C.: Verifying temporal properties of processes, in *CONCUR'90: Concurrency Theory, Proceedings*, LNCS vol. 458, Springer-Verlag, 1990, pp. 115-125

[14] Bradfield, J. & Stirling, C.: Local model checking for infinite state spaces, in *Theoretical Computer Science*, vol. 96, 1992, pp. 157-174

[15] Büchi, J. R.: On a decision method in restricted second-order arithmetics, in *Proceedings of the 1960 International Congress on Logic, Methodology and Philosophy of Science*, Stanford University Press, 1962, pp. 1-12

[16] Büchi, J. R.: Using determinacy of games to eliminate quantifiers, in *Fundamentals of Computation Theory*, LNCS vol. 56, 1977, pp. 367-378

[17] Büchi, J. R.: State-strategies for games in $F_{\sigma\delta} \cap G_{\delta\sigma}$, in *Journal of Symbolic Logic*, vol. 48, 1983, pp. 1171-1198

[18] Burgess, J. P.: Basic tense logic, in Gabbay, D. & Guenthner, F. (eds.): *Handbook of Philosophical Logic*, vol. II, Reidel, 1984, pp. 89-113

[19] Chandra, A. K. & Kozen, D. C. & Stockmeyer, L. J.: Alternation, in *Journal of the ACM*, vol. 28, 1981, pp. 114-133

[20] Clarke, E. & Grumberg, O. & Kurshan, R. P.: A synthesis of two approaches for verifying finite state concurrent systems, in *Logic at Botik'89, Symposium on Logical Foundations of Computer Science*, LNCS vol. 363, Springer-Verlag, 1989, pp. 81-90

[21] Chellas, B. F.: *Modal Logic: an Introduction*, Cambridge University Press, 1980

[22] Dam, M.: Fixpoints of Büchi automata, in *Foundations of Software Technology and Theoretical Computer Science, 12th Conference, Proceedings*, LNCS vol. 652, Springer-Verlag, 1992, pp. 39-50

[23] Doner, J.E.: Decidability of the weak second-order theory of two successors, in *Notices of the American Mathematical Society*, vol. 12, 1965, p. 819

[24] Emerson, E. A.: Alternative semantics for temporal logics, in *Theoretical Computer Science*, vol. 26, 1983, pp. 121-130

[25] Emerson, E. A.: Temporal and modal logic, in van Leeuwen, J. (ed.): *Handbook of Theoretical Computer Science*, Elsevier/North-Holland, 1990, pp. 997-1072

[26] Emerson, E. A.: Automated temporal reasoning about reactive systems, in *Logics for Concurrency, Structure versus Automata*, LNCS vol. 1043, Springer-Verlag, 1996, pp. 41-101

[27] Emerson, E. A. & Clarke, E. M.: Characterising correctness properties of parallel programs using fixpoints, in *Automata, Languages and Programming, 7th International Colloquium, ICALP'80, Proceedings*, LNCS vol. 85, Springer-Verlag, 1980, pp. 169-181

[28] Emerson, E. A. & Halpern, J. Y.: "Sometimes" and "not never" revisited: on branching versus linear time, in *Conference Record of the 10th Annual ACM Symposium on Principles of Programming Languages*, 1983, pp. 127-140

[29] Emerson, E. A. & Halpern, J. Y.: Deciding full branching time logic, in *Information and Control*, vol. 61, 1984, pp. 175-201

[30] Emerson, E. A. & Halpern, J. Y.: Decision procedures and expressiveness in the temporal logic of branching time, in *Journal of Computer And System Sciences*, vol. 30, 1985, pp. 1-24

[31] Emerson, E. A. & Jutla, C. S.: Complexity of tree automata and modal logics of programs, in *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, 1988, pp. 328-337

[32] Emerson, E. A. & Jutla, C. S.: On simultaneously determinizing and complementing ω-automata, in *Proceedings of the Fourth Annual IEEE Symposium on Logic in Computer Science*, 1989, pp. 333-342

[33] Emerson, E. A. & Jutla, C. S.: Tree automata, mu-calculus, and determinacy, in *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science*, 1991, pp. 368-377

[34] Emerson, E. A. & Lei, C. L.: Efficient model-checking in fragments of the propositional mu-calculus, in *Proceedings of the First IEEE Symposium on Logic in Computer Science*, 1986, pp. 267-278

[35] Emerson, E. A. & Sistla, A. P.: Deciding full branching time logic, in *Information and Control*, vol. 61, 1984, pp. 175-201

[36] Gabbay, D. & Pnueli, A. & Shelah, S. & Stavi, J.: On the temporal analysis of fairness, in *Conference Record the 7th Annual ACM Symposium on Principles of Programming Languages*, 1980, pp. 163-173

[37] Gécseg, F. & Steinby, M.: *Tree Automata*, Akadémiai Kiadó, Budapest, 1984

[38] Gries, D.: *The Science of Programming*, Springer-Verlag, 1981

[39] Gurevich, Y. & Harrington, L.: Trees, automata and games, in *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, 1982, pp. 60-65

[40] Hennessy, M. & Milner, R.: Algebraic laws for nondeterminism and concurrency, in *Journal of the ACM*, vol. 32, 1985, pp. 137-162

[41] Hitchcock, P. & Park, D.: Induction rules and termination proofs, in Nivat, M. (ed.): *Automata, Languages and Programming, Proceedings*, North-Holland, 1973, pp. 225-251

[42] Hodges, W.: *Model Theory*, Encyclopedia of Mathematics and its Applications, vol. 42, Cambridge University Press, 1993

[43] Hoare, C. A. R.: An axiomatic basis for computer programming, in *Communications of the ACM*, vol. 12, 1969, pp. 576-580

[44] Hughes, G. E. & Cresswell, M. J.: *An Introduction to Modal Logic*, 2nd ed., Routledge, 1972

[45] Hüttel, H.: *SnS* can be modally characterised, in *Theoretical Computer Science*, vol. 74, 1990, pp. 239-248

[46] Jutla, C. S.: *Automata on Infinite Objects and Modal Logics of Programs*, PhD thesis, University of Texas at Austin, 1990

[47] Kaivola, R.: On modal mu-calculus and Büchi tree automata, in *Information Processing Letters*, vol. 54, 1995, pp. 17-22, an extended version has also appeared as report ECS-LFCS-94-293, Laboratory for Foundations of Computer Science, University of Edinburgh, 1994, 24 p.

[48] Kaivola, R.: A simple decision method for the linear-time mu-calculus, in *Structures in Concurrency Theory, Proceedings of the International Workshop on Structures in Concurrency Theory (STRICT), Berlin, May 1995*, Workshops in Computing series, Springer-Verlag, 1995, pp. 190-204

[49] Kaivola, R.: Axiomatising linear time mu-calculus, in *CONCUR'95: Concurrency Theory, 6th International Conference, Proceedings*, LNCS vol. 962, Springer-Verlag, 1995, pp. 423-437

[50] Kaivola, R.: Axiomatising extended computation tree logic, in *Trees in Algebra and Programming – CAAP'96, Proceedings*, LNCS vol. 1059, Springer-Verlag, 1996, pp. 87-101, also to appear in *Theoretical Computer Science*, 1997

[51] Kaivola, R.: Fixpoints for Rabin tree automata make complementation easy in *Automata, Languages and Programming, 23rd International Colloquium, ICALP'96, Proceedings*, LNCS vol. 1099, Springer-Verlag, 1996, pp. 312-323

[52] Kamp, H.: *Tense Logic and the Theory of Linear Order*, Ph.D. Dissertation, UCLA, Los Angeles, 1968

[53] Kesten, Y. & Pnueli, A.: A complete proof system for $QPTL$, in *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science*, 1995, pp. 2-12

[54] Kleene, S. C.: *Introduction to Metamathematics*, North-Holland, 1952

[55] Kozen, D.: Results on the propositional μ-calculus, in *Theoretical Computer Science*, vol. 27, 1983, pp. 333-354

[56] Kozen, D. & Parikh, R.: A decision procedure for the propositional μ-calculus, in *Logics of Programs: Workshop, Carnegie Mellon University, 1983*, LNCS vol. 164, Springer-Verlag, 1984, pp. 313-325

[57] Läuchli, H.: A decision procedure for the weak second-order theory of linear order, in Schutte, K. (ed.): *Contributions to Mathematical Logic*, North-Holland, 1968, pp. 189-197

[58] Lenzi, G.: A hierarchy theorem for the μ-calculus, in *Automata, Languages and Programming, 23rd International Colloquium, ICALP'96, Proceedings*, LNCS vol. 1099, Springer-Verlag, 1996, pp. 87-109

[59] Lichtenstein, O.: *Decidability, Completeness, and Extensions of Linear Time Temporal Logic*, PhD thesis, The Weizmann Institute of Science, Rehovot, Israel, 1991

[60] Lichtenstein, O. & Pnueli, A. & Zuck, L.: The glory of the past, in *Proceedings of Workshop on Logics of Programs*, LNCS vol. 193, Springer-Verlag, 1985, pp. 97-107

[61] Lindsay, P. A.: On alternating ω-automata, in *Journal of Computer and System Sciences*, vol. 36, 1988, pp. 16-24

[62] Loeckx, J. & Sieber, K., in collab. with Stansifer, R.D.: *The Foundations of Program Verification, 2nd Edition*, Wiley-Teubner Series in Computer Science, 1987

[63] McNaughton, R.: Testing and generating infinite sequences by a finite automaton, in *Information and Control*, vol. 9, 1966, pp. 521-530

[64] Miyano, S. & Hayashi, T.: Alternating finite automata on ω-words, in *Theoretical Computer Science*, vol. 32, 1984, pp. 321-330

[65] Mostowski, A. W.: Regular expressions for infinite trees and a standard form of automata, in *Computation Theory, Fifth Symposium, Proceedings*, LNCS vol. 208, Springer-Verlag, 1985, pp. 157-168

[66] Muller, D. E. & Schupp, P. E.: Alternating automata on infinite objects, determinacy and Rabin's theorem, in *Automata on Infinite Words*, LNCS vol. 192, Springer-Verlag, 1985

[67] Muller, D. E. & Schupp, P. E.: Alternating automata on infinite trees, in *Theoretical Computer Science*, vol. 54, 1987, pp. 267-276

[68] Muller, D. E. & Saoudi, A. & Schupp, P.: Alternating automata, the weak monadic theory of the tree, and its complexity, in *Automata, Languages and Programming, 13th International Colloquium, ICALP'86, Proceedings*, LNCS vol. 226, Springer-Verlag, 1986, pp. 275-283

[69] Muller, D. E. & Saoudi, A. & Schupp, P.: Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time, in *Proceedings of the 3rd IEEE Symposium on Logic in Computer Science*, 1988, pp. 422-427

[70] Niwiński, D.: On fixed point clones, in *Automata, Languages and Programming, 13th International Colloquium, ICALP'86, Proceedings*, LNCS vol. 226, Springer-Verlag, 1986, pp. 464-473

[71] Niwiński, D.: Fixed points vs. infinite generation, in *Proceedings of the 3rd IEEE Symposium on Logic in Computer Science*, 1988, pp. 402-409

[72] Park, D.: Fixpoint induction and proofs of program properties, in *Machine Intelligence 5*, Edinburgh University Press, 1969, pp. 59-78

[73] Park, D.: Concurrency and automata on infinite sequences, in *Theoretical Computer Science: 5th GI-conference*, LNCS vol. 104, Springer-Verlag, 1981, pp. 167-183

[74] Pnueli, A.: The temporal logic of programs, in *Proceedings of the 18th Annual IEEE Symposium on Foundations of Computer Science*, 1977, pp. 46-57

[75] Pnueli, A.: Linear and branching structures in the semantics of logics of reactive systems, in *Automata, Languages and Programming, 12th International Colloquium, ICALP'85, Proceedings*, LNCS vol. 194, Springer-Verlag, 1985, pp. 15-32

[76] Rabin, M. O.: Decidability of second order theories and automata on infinite trees, in *Transactions of the Americal Mathematical Society*, vol. 141, 1969, pp. 1-35

[77] Rabin, M. O.: Weakly definable relations and special automata, in Bar-Hillel, Y. (ed.): *Mathematical Logic and Foundations of Set Theory*, North-Holland, 1970, pp. 1-23

[78] Rabin, M. O.: Decidable theories, in Barwise, J. (ed.): *Handbook of Mathematical Logic*, North-Holland, 1977, pp. 595-629

[79] Rescher, N. & Urquhart, A.: *Temporal Logic*, Library of Exact Philosophy, vol. 3, Springer-Verlag, 1971

[80] Safra, S.: On the complexity of ω-automata, in *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, 1988, pp. 319-327

[81] Siefkes, D.: *Büchi's Monadic Second Order Successor Arithmetic, Decidable Theories I*, LNM vol. 120, Springer-Verlag, 1970

[82] Siefkes, D.: An axiom system for the weak monadic second order theory of two successors, in *Israel Journal of Mathematics*, vol. 30, no. 3, 1978, pp. 264-284

[83] Stirling, C.: Modal and temporal logics, in Abramsky, S. & al. (eds.): *Handbook of Logic in Computer Science*, Oxford University Press, 1992, pp. 477-563

[84] Stirling, C.: Local model checking games, in *CONCUR'95: Concurrency Theory, 6th International Conference, Proceedings*, LNCS vol. 962, Springer-Verlag, pp. 1-11

[85] Stirling, C. & Walker, D.: Local model checking in the modal mu-calculus, in *Theoretical Computer Science*, vol. 89, 1991, pp. 161-177

[86] Streett, R. S. & Emerson, E. A.: The propositional mu-calculus is elementary, in *Automata, Languages and Programming, 11th International Colloquium, ICALP'84, Proceedings*, LNCS vol. 172, Springer-Verlag, 1984, pp. 467-472

[87] Streett, R. S. & Emerson, E. A.: An automata theoretic decision procedure for the propositional mu-calculus, in *Information and Computation*, vol. 81, 1989, pp. 249-264

[88] Szalas, A.: Axiomatising fixpoint logics, in *Information Processing Letters*, vol. 41, 1992, pp. 175-180

[89] Takahashi, M.: The greatest fixed-points and rational omega-tree languages, in *Theoretical Computer Science*, vol. 44, 1986, pp. 259-274

[90] Tarski, A.: A lattice-theoretical fixpoint theorem and its applications, in *Pacific Journal of Mathematics*, vol. 5, 1955, pp. 285-309

[91] Thatcher, J. W. & Wright, J. B.: Generalised finite automata with an application to a decision problem of second-order logic, in *Mathematical Systems Theory*, vol. 2, 1968, pp. 57-82

[92] Thomas, W.: Computation tree logic and regular ω-languages, in *Linear Time, Branching Time and Partial Order in Concurrency*, LNCS vol. 354, Springer-Verlag, 1988, pp. 690-713

[93] Thomas, W.: Automata on infinite objects, in van Leeuwen, J. (ed.): *Handbook of Theoretical Computer Science*, vol. 2, Elsevier/North-Holland, 1990, pp. 133-191

[94] Vardi, M. Y.: Verification of concurrent programs: the automata-theoretic framework, in *Proceedings of the 2nd IEEE Symposium on Logic in Computer Science*, 1987, pp. 167-176

[95] Vardi, M. Y.: A temporal fixpoint calculus, in *Conference Record of the 15th Annual ACM Symposium on Principles of Programming Languages*, 1988, pp. 250-259

[96] Vardi, M. Y. & Wolper, P.: Yet another process logic, in *Logics of Programs: Workshop, Carnegie Mellon University, 1983*, LNCS vol. 164, Springer-Verlag, 1984, pp. 501-512

[97] Vardi, M. Y. & Wolper, P.: Automata-theoretic techniques for modal logics of programs, in *Journal of Computer and System Sciences*, vol. 32, 1986, pp. 183-221

[98] Vardi, M. Y. & Wolper, P.: An automata theoretic approach to automatic program verification, in *Proceedings of the First IEEE Symposium on Logic in Computer Science*, 1986, pp. 332-344

[99] Vardi, M. Y. & Wolper, P.: Reasoning about infinite computations, in *Information and Computation*, vol. 115, 1994, pp. 1-37

[100] Walukiewicz, I.: On completeness of the μ-calculus, in *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*, 1993, pp. 136-146

[101] Walukiewicz, I.: Completeness of Kozen's axiomatisation of the propositional μ-calculus, in *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science*, 1995, pp. 14-24

[102] Wolper, P.: *Synthesis of Communicating Processes from Temporal Logic Specifications*, PhD thesis, Stanford University, 1982

[103] Wolper, P.: Temporal logic can be more expressive, in *Information and Control*, vol 56, 1983, pp. 72-99

Index

Appendix A

Technical proofs

This appendix contains full versions of certain proofs that were just asserted or sketched in the main text.

A.1 Ordinary vs. restricted alternating automata

The following lemmas show the equivalence of ordinary and restricted alternating automata for Büchi, Rabin and first recurrence automata.

Lemma 3.1.15. For every ordinary Büchi automaton A there exists a restricted alternating Büchi automaton A', and vice versa, such that $L(A) = L(A')$.

Proof: Let $A = (Q, q_{init}, \Delta, F)$ be an ordinary Büchi automaton on n-branching trees. A corresponding alternating automaton $A' = (Q', q'_{init}, \Delta', l', F')$ can be defined by

$$\begin{aligned}
Q' &= \{(\vee, q) \mid q \in Q\} \;\cup\; \{(\wedge, d) \mid d \in \Delta\} \;\cup\; \{(\textcircled{\scriptsize i}, d) \mid d \in \Delta \text{ and } i \in [n]\} \;\cup \\
&\quad\; \{(z, d) \mid d = (q, Z, \overline{q}) \in \Delta \text{ and } z \in Z\} \\
q'_{init} &= (\vee, q_{init}) \\
\Delta' &= \{((\vee, q), (\wedge, d)) \mid d = (q, Z, \overline{q}) \in \Delta\} \cup \\
&\quad\; \{((\wedge, d), (\textcircled{\scriptsize i}, d)) \mid d \in \Delta \text{ and } i \in [n]\} \cup \\
&\quad\; \{((\wedge, d), (z, d)) \mid d = (q, Z, \overline{q}) \in \Delta \text{ and } z \in Z\} \cup \\
&\quad\; \{((\textcircled{\scriptsize i}, d), (\vee, q')) \mid d = (q, Z, \overline{q}) \in \Delta \text{ and } q' = \overline{q}_i\} \\
l' &: (x, y) \mapsto x \\
F' &= \{(\vee, q) \mid q \in F\}
\end{aligned}$$

It is obvious that $L(A) = L(A')$ and A' is restricted.

Let then $A' = (Q', q'_{init}, \Delta', l', F')$ be a restricted alternating Büchi automaton. We shall construct an equivalent ordinary automaton A from A' by taking the principal states of A' as the states of A, and coding the auxiliary states between principal states of A' into transitions of A. Without loss of generality we can assume that A' does not contain any $\top$ or $\bot$-states, since they can be easily removed from any A' without affecting the language recognised by A'.

Denote the set of principal states of A' by Q'_p, and let q_{acc} be a fresh state. Let us associate with every state $q \in Q'$ which is not a $\vee$-state, a pair $(Z, \overline{q})$ where $Z \subseteq \mathcal{Z} \cup \overline{\mathcal{Z}}$ and $\overline{q} \in (Q'_p \cup \{q_{acc}\})^n$ as follows:

- If $l'(q) = \wedge$ then $Z = \{z \in \mathcal{Z} \cup \overline{\mathcal{Z}} \mid \exists q' \in Q' : (q, q') \in \Delta'$ and $l'(q') = z\}$ and for every $i \in [n]$, if there are $q', q'' \in Q'$ such that $(q, q') \in \Delta'$, $(q', q'') \in \Delta'$ and $l(q') = \textcircled{\scriptsize i}$, then $\overline{q}_i = q''$, and otherwise $\overline{q}_i = q_{acc}$. Notice that due to strong aconjunctivity, if the q'' exists, it is unique.
- If $l'(q) = \textcircled{\scriptsize i}$ then $Z = \emptyset$, and $\overline{q}_i$ is the unique $q' \in Q'$ such that $(q, q') \in \Delta'$, and $\overline{q}_j = q_{acc}$ for all $j \neq i$.
- If $l(q') = z \in \mathcal{Z} \cup \overline{\mathcal{Z}}$, then $Z = \{z\}$ and $\overline{q}_i = q_{acc}$ for all $i \in [n]$.

Define the ordinary Büchi automaton $A = (Q, q_{init}, \Delta, F)$ by

- $Q = Q'_p \cup \{q_{acc}\}$,
- $q_{init} = q'_{init}$,
- $F = F' \cup \{q_{acc}\}$ and
- for every $q \in Q'_p$, $Z \subseteq \mathcal{Z} \cup \overline{\mathcal{Z}}$ and $\overline{q} \in (Q'_p \cup \{q_{acc}\})^n$, we have $(q, Z, \overline{q}) \in \Delta$ iff there is a sequence of states $q_0, \ldots, q_k \in Q'$ such that $k \geq 0$, $q_0 = q$, $l'(q_i) = \vee$ and $(q_i, q_{i+1}) \in \Delta'$ for every $0 \leq i < k$, $l'(q_k) \neq \vee$, and $(Z, \overline{q})$ is the pair associated with state q_k according to the rules above. Furthermore $(q_{acc}, \emptyset, q^n_{acc}) \in \Delta$.

It should be easy to see that $L(A) = L(A')$. □

Lemma 3.1.16. For every ordinary Rabin automaton A there exists a restricted alternating Rabin automaton A', and vice versa, such that $L(A) = L(A')$.

Proof: Given an ordinary Rabin automaton $A = (Q, q_{init}, \Delta, \Omega)$, where $\Omega = ((G_0, R_0) \ldots (G_{m-1}, R_{m-1}))$, we can construct a corresponding restricted alternating Rabin automaton $A' = (Q', q'_{init}, \Delta', l', \Omega')$ precisely as with Büchi automata in the proof of Lemma 3.1.15, except that $\Omega' = ((G'_0, R'_0) \ldots (G'_{m-1}, R'_{m-1}))$ where for every $i \in [m]$, $G'_i = \{(\vee, q) \mid q \in G_i\}$ and $R'_i = \{(\vee, q) \mid q \in R_i\}$.

Given a restricted alternating Rabin automaton $A' = (Q', q'_{init}, \Delta', l', \Omega')$, we can construct a corresponding ordinary Rabin automaton $A = (Q, q_{init}, \Delta, \Omega)$ precisely as with Büchi automata in the proof of Lemma 3.1.15, except that $\Omega = ((G_0, R_0) \ldots (G_{m-1}, R_{m-1}))$ where $G_0 = G'_0 \cup \{q_{acc}\}$, $R_0 = R'_0$, and for every $1 \leq i < m$, $G_i = G'_i$ and $R_i = R'_i$. □

Lemma 3.2.3. For every ordinary first recurrence automaton A there exists a restricted alternating first recurrence automaton A', and vice versa, such that $L(A) = L(A')$.

Proof: The proof is analogous to that of Lemma 3.1.15, where the same claim was made for Büchi automata. However, some extra care is required to guarantee the tree-like structure required of FR-automata.

Let $A = (Q, q_{init}, \Delta, (G, R))$ be an ordinary FR-automaton on n-branching trees. Let us define inductively a finite tree T labelled with pairs (x, y) where $x \in \{\vee, \wedge\} \cup \{ \textcircled{i} \mid i \in [n]\} \cup \mathcal{Z} \cup \overline{\mathcal{Z}}$ and y is either a state $q \in Q$ or a transition $d \in \Delta$:

- the root of T is labelled with $T(\epsilon) = (\vee, q_{init})$,
- if $T(t) = (\vee, q)$ for some $q \in Q$ and t is not a leaf (see below), then t has a child labelled with $(\wedge, d)$ for every transition $d = (q', Z, \overline{q}') \in \Delta$ such that $q = q'$,
- if $T(t) = (\wedge, d)$ for some $d = (q, Z, \overline{q}) \in \Delta$, then t has a child labelled with (z, d) for every $z \in Z$, and a child labelled with $(\textcircled{i}, d)$ for every $i \in [n]$,
- if $T(t) = (\textcircled{i}, d)$ for some $i \in [n]$ and $d = (q, Z, \overline{q}) \in \Delta$, then t has one child labelled with $(\vee, \overline{q}_i)$.

A node t of T is a leaf iff either

1 $T(t) = (z, d)$ for some $z \in \mathcal{Z} \cup \overline{\mathcal{Z}}$, or

2 $T(t) = (\vee, q)$ and there is some proper ancestor t' of t such that $T(t') = T(t)$. In this case we call t' the loop node corresponding to t.

We can then define the automaton A' by $A' = (Q', q'_{init}, \Delta', l', (G', R'))$, where

- $Q' = \{t \in \mathrm{dom}(T) \mid t \text{ is not a leaf of type } 2\}$,
- $q'_{init} = \epsilon$,
- $(t, t') \in \Delta'$ iff $t, t' \in Q'$ and either t' is a child of t, or there is some child t'' of t such that t'' is a type 2 leaf and t' the loop node corresponding to t'',
- $l'(t) = x$ where $T(t) = (x, y)$,
- $G' = \{t \in Q' \mid T(t) = (\vee, q) \text{ and } q \in G\}$.
- $R' = \{t \in Q' \mid T(t) = (\vee, q) \text{ and } q \in R\}$.

It is obvious that $L(A) = L(A')$ and A' is restricted.

Let then $A' = (Q', q'_{init}, \Delta', l', (G', R'))$ be a restricted alternating first recurrence automaton. We shall construct an equivalent ordinary automaton A by taking the principal states of A' as the states of A. Without loss of generality we assume that A' does not contain any $\top$ or $\bot$-states. Define inductively a finite tree T, each node of which is labelled either with a principal state q of A' or a fresh state q_{acc}:

- the root of T is labelled with $T(\epsilon) = q'_{init}$,
- if $T(t) = q$ for a principal state $q \in Q'$, then t has one child labelled with q_{acc} and a child labelled with q' for every principal state $q' \in Q'$ such that $q \prec q'$, i.e. q' is a proper descendant of q, and there is no principal $q'' \in Q'$ such that $q \prec q'' \prec q'$,
- if $T(t) = q_{acc}$, then t is a leaf.

We associate with every state $q \in Q'$ which is not a $\vee$-state, a pair $(Z, \overline{q})$ as in the proof of Lemma 3.1.15. Define now $A = (Q, q_{init}, \Delta, (G, R))$ by $Q = \text{dom}(T)$, $q_{init} = \epsilon$, $G = \{t \in \text{dom}(T) \mid T(t) \in G'\}$, $R = \{t \in \text{dom}(T) \mid T(t) \in R'\}$, and define Δ as follows.

If $T(t) = q_{acc}$, then $(t, \emptyset, t^n) \in \Delta$. If $T(t) = q$ for a principal state $q \in Q'$, then $(t, Z, \overline{t}) \in \Delta$ iff there is a sequence of states $q_0, \ldots, q_k \in Q'$ and a vector $\overline{q}$ such that $k \geq 0$, $T(t) = q_0$, $l'(q_i) = \vee$ and $(q_i, q_{i+1}) \in \Delta'$ for every $0 \leq i < k$, $l'(q_k) \neq \vee$, $(Z, \overline{q})$ is the pair associated with state q_k, and for every $i \in [n]$, if $\overline{q}_i \neq q_{acc}$, then $T(\overline{t}_i) = \overline{q}_i$, and if $\overline{q}_i = q_{acc}$, then $\overline{t}_i$ is the child of t such that $T(\overline{t}_i) = q_{acc}$. □

A.2 Bundled tableaux vs. definition tree tableaux

The following lemmas show that definition tree tableaux characterise satisfiability in the same way as bundled tableaux.

Lemma 4.2.4. Let T be a definition tree tableau. The following statements are mutually equivalent:

- T is not proper,
- there is a bad direct dependency sequence in T, and
- there is a bad indirect dependency sequence in T.

Proof: Let us show first that the first condition implies the second. Assume that T is not proper and that there is a constant u and a point n of T such that u is not deleted at any point beyond n, and u is contracted at infinitely many points $n = k(0) < k(1) < \ldots$ of T. By clause 5 of Lemma 4.2.3, for every $i \in \mathbb{N}$, there is some $\psi \in \Gamma_{k(i)}$ such that u is active in ψ relative to $d_{k(i)}$, and for every $\psi \in \Gamma_{k(i+1)}$ for which u is active in ϕ, there is some $\phi \in \Gamma_{k(i)}$ and a direct dependency sequence from ϕ at $k(i)$ to ψ at $k(i+1)$ such that u is active in every formula of this dependency sequence and occurs at least once in it. Since the size of the sets Γ_i is bounded by clause 1 of Lemma 4.2.3, by König's lemma we can piece together an infinite direct dependency sequence $\phi_0, \phi_1, \ldots$ from point n of T such that u is active in every formula of the sequence and occurs infinitely often in it.

Since any direct dependency sequence is also an indirect dependency sequence, the second condition obviously implies the third. Let us show then that the third condition implies the first. Assume, on the contrary, that T is proper, but there is an infinite indirect dependency sequence $\phi_0, \phi_1, \ldots$ from some point n of T and a minimal or auxiliary constant u such that u is active in every ϕ_i relative to d_{n+i}, and $\phi_i = u$ for infinitely many $i \in \mathbb{N}$. Since T is proper, there is some point $m \in \mathbb{N}$ such that no ancestor of u (including u) is contracted in T after point m. Take then some $k \geq m - n$ such that $\phi_k = u$ and the $U\mu$ rule is applied to u at point $n + k$ of T. Since u is not contracted anywhere in T after point m, by clause 6 of Lemma 4.2.3 we see then that $u \neq \phi_i$ for all $i \geq k$, contradicting the assumption that $\phi_i = u$ for infinitely many i. □

Lemma 4.2.5. Let $T = (s_0, \Gamma_0, d_0)(s_1, \Gamma_1, d_1) \ldots$ be a proper definition tree tableau, n a point of T and $\phi_0, \phi_1, \ldots$ an infinite direct or indirect dependency sequence from point n of T. Then there is some $m \geq 0$ and constant u such that

- u is active in ϕ_i, relative to d_{n+i} for every $i \geq m$, and
- $\phi_i = u$ for infinitely many $i \geq m$.

Proof: Without loss of generality we can assume that $n = 0$. Let us define inductively a finite sequence $u_0, \ldots, u_k$ of constants and a finite non-decreasing sequence $m(0), \ldots, m(k)$ of points of T such that for all $0 \leq i \leq k$:

1 u_i is defined in $d_{m(i)}$ and $\phi_{m(i)}$ is a descendant of u_i in $d_{m(i)}$,

2 if v is a proper ancestor of u_i in $d_{m(i)}$, then v is not contracted in T at any point $j \geq m(i)$,

3 ϕ_j is not a proper ancestor of u_i (relative to d_j) at any point $j \geq m(i)$,

4 if $i > 0$, then u_{i-1} is defined in $d_{m(i)}$ and $u_i \prec_{d_{m(i)}} u_{i-1}$, and

5 ϕ_j is a descendant of u_k (relative to d_j) at every point $j \geq m(k)$, and $\phi_j = u_k$ for infinitely many points $j \geq m(k)$.

Defining $m(0) = 0$ and $u_0 = u_\epsilon$ (the dummy variable corresponding to the root of the tree induced by any definition tree) fulfils trivially conditions 1–4.

Assume then that we have u_i and $m(i)$ fulfilling conditions 1–4, but not condition 5. Now either

- ϕ_j is a descendant of u_i in d_j, for every $j \geq m(i)$, or
- there is some $j \geq m(i)$ such that ψ_j is a descendant of u_i in d_j, but ϕ_{j+1} is not a descendant of u_i in d_{j+1}.

Assume first that the former case holds. Since T is proper, there is some point $m \geq m(i)$ such that u_i is not contracted anywhere in T after point m. By the assumption that u_i and $m(i)$ do not fulfil condition 5, we can choose m so that

$u_i \neq \phi_j$ for every $j \geq m$, as well. Define now $m(i+1) = m$ and let u_{i+1} be the unique child u' of u_i in d_m such that ϕ_m is a descendant of u' in d_m. Such a child exists, since ϕ_m is a proper descendant of u_i in d_m. It is obvious that u_{i+1} and $m(i+1)$ fulfil condition 1. Conditions 2 and 3 holds by the induction assumption and the choice of m. Since u_{i+1} is a proper descendant of u_i on $d_{m(i+1)}$, $u_{i+1} \prec_{d_{m(i+1)}} u_i$ and condition 4 holds, as well.

Assume then that the latter case holds. Since by condition 2 of the induction assumption we know that no proper ancestor of u_i is contracted at point j of T, and since by assumption ϕ_{j+1} is not a descendant of u_i in d_{j+1}, by clause 4 of Lemma 4.2.3 we see that ϕ_{j+1} must be older than u_i in d_{j+1}. Then, by condition 3 of the induction assumption, ϕ_{j+1} is older than but not an ancestor of u_i in d_{j+1}. Define now $m(i+1) = j+1$, and let u_{i+1} be the oldest ancestor of ϕ_{j+1} in d_{j+1} which is not an ancestor of u_i. It is obvious that u_{i+1} and $m(i+1)$ fulfil condition 1. Conditions 2 and 3 follow from the same conditions in the induction assumption and the fact that every proper ancestor of u_{i+1} in d_{j+1} is also a proper ancestor of u_i. Since u_{i+1} is older than but not an ancestor of u_i in d_{j+1}, $u_{i+1} \prec_{d_{m(i+1)}} u_i$ and condition 4 holds, as well.

So far we have shown that for any u_i and $m(i)$ fulfilling conditions 1–4, either condition 5 holds or we can construct u_{i+1} and $m(i+1)$ fulfilling conditions 1–4. To see that the sequence cannot go on infinitely, we associate with it a descending chain in a well-founded ordering. Define first for every constant u appearing in T a point $\mathrm{nc}(u) \in \mathbb{N} \cup \{\omega\}$ by: $\mathrm{nc}(u)$ is the minimal $i \in \mathbb{N}$ such that no ancestor v of u in d_i (including u itself) is contracted in T after point i, and $\mathrm{nc}(u) = \omega$ if no such i exists. Define a tree d_ω labelled with (not necessarily all) constants u appearing in T by:

- $d_\omega(\epsilon) = u_\epsilon$,
- if $d_\omega(t) = u$ and $\mathrm{nc}(u) = \omega$, then t has no children,
- if $d_\omega(t) = u$ and $\mathrm{nc}(u) = i$, then t has a child labelled with u' for every u' such that u' is a child of u in d_i for some $i \geq \mathrm{nc}(u)$.

The children of any node t of d_ω are ordered in some fixed manner so that if t' and t'' are children of t, $d_\omega(t') = u'$ and $d_\omega(t'') = u''$ and there is some $i \geq \mathrm{nc}(u)$ such that u' is older than u'' in d_i, then t' is also older than t'' in d_ω. The well-definedness of the tree d_ω follows from clause 3 of Lemma 4.2.3.

Since by clause 2 of Lemma 4.2.3 there is a bound on the depth of any d_i, there is also a bound on the depth of d_ω, although it may be infinitely branching. Notice then that in the sequence $u_0, u_1, \ldots$, every u_{i+1} is either a proper descendant of u_i or older than but not an ancestor of u_i. In other words $u_{i+1} \prec_{d_\omega} u_i$. But this is a well-founded ordering, so the sequence $u_0, u_1, \ldots$ cannot be infinite. □

Lemma 4.2.6. Let T be a proper and propositionally consistent bundled tableau for a guarded μTL-formula ϕ in pnf. There exists a proper and propositionally consistent definition tree tableau T' for ϕ such that for any model $M \in \mathcal{M}_{TL}$, if T agrees with M then T' agrees with M.

Proof: Take a proper bundled tableau $T = (s_0, \Gamma_0, d_0)(s_1, \Gamma_1, d_1)\dots$ for formula ϕ. We shall construct a definition tree tableau $T' = (s'_0, \Gamma'_0, d'_0)(s'_1, \Gamma'_1, d'_1)\dots$ for ϕ, together with an infinite non-decreasing sequence of indices $k(0) \leq k(1) \leq \dots$ such that $k(0) = 0$ and for every $i \in \mathbb{N}$, none of the del-ϕ, del-U or contr-U rules are applicable at point $k(i)$. Intuitively, the derivation step $(s_i, \Gamma_i, d_1)(s_{i+1}, \Gamma_{i+1}, d_{i+1})$ of T is mimicked in T' by the fragment $(s'_{k(i)}, \Gamma'_{k(i)}, d'_{k(i)})\dots(s'_{k(i+1)}, \Gamma'_{k(i+1)}, d'_{k(i+1)})$.

There are two main differences between tableaux T and T'. On the one hand, due to the $U\mu$ rule introducing new constants every time a minimal fixpoint is unfolded in T', the tableau T' may have multiple constants corresponding to a single constant of T. On the other hand, as multiple copies of similar formulae are erased by the del-ϕ rule in T', the tableau T' may have fewer formulae than T at the corresponding point and therefore some constants of T do not ever get introduced in T'.

When building the tableau T' we take care to preserve the meaning of constants in the sense that if u is defined in $d'_{k(i)}$ and it is not an auxiliary constant, then u is defined in d_i as well, and u in d_i and u in $d'_{k(i)}$ are similar. For every $i \in \mathbb{N}$ and every $\phi \in \Gamma'_{k(i)}$, define inductively the formula underlying ϕ in Γ_i, denoted by $f_i(\phi)$, by:

- if ϕ contains no auxiliary constant constants of $d'_{k(i)}$, then $f_i(\phi) = \phi$, and
- if ϕ contains an auxiliary constant u of $d'_{k(i)}$, then $f_i(\phi) = f_i(\phi[u'/u])$ where $u' = d'_{k(i)}(u)$.

Notice that for any $\phi, \psi \in \Gamma'_{k(i)}$, if $f_i(\phi) = f_i(\psi)$ then $\phi = \psi$, since otherwise there would be two similar formulae ϕ and ψ in $\Gamma'_{k(i)}$ and the del-ϕ rule would be applicable at point $k(i)$, contrary to the properties of the sequence $k(i)$.

Define first $k(0) = 0$ and $(s'_0, \Gamma'_0, d'_0) = (0, \{\phi\}, \epsilon)$. For every $i \in \mathbb{N}$, we define $k(i+1)$ and the fragment of T' from $k(i)$ to $k(i+1)$ as follows.

If the rule applied at point i of T is the $\bigcirc$-rule, then $k(i+1) = k(i) + 1$ and $(s'_{k(i)+1}, \Gamma'_{k(i)+1}, d'_{k(i)+1})$ is derived from $(s'_{k(i)}, \Gamma'_{k(i)}, d'_{k(i)})$ by applying the $\bigcirc$-rule. If the rule applied at point i of T is not the $\bigcirc$-rule, and it is applied to a formula $\phi \in \Gamma_i$ such that ϕ does not underlie any formula in $\Gamma'_{k(i)}$, i.e. $\phi \neq f_i(\psi)$ for all $\psi \in \Gamma'_{k(i)}$, then $k(i+1) = k(i)$ and the corresponding fragment of T' empty.

If the rule R applied at point i of T is not the $\bigcirc$-rule and it is applied to some $\phi \in \Gamma_i$ such that $\phi = f_i(\psi)$ for some $\psi \in \Gamma'_{k(i)}$, then the fragment

$(s'_{k(i)}, \Gamma'_{k(i)}, d'_{k(i)}) \dots (s'_{k(i+1)}, \Gamma'_{k(i+1)}, d'_{k(i+1)})$ consists of first applying the rule R' corresponding to R to the unique formula $\psi \in \Gamma'_{k(i)}$ for which $f_i(\psi) = \phi$, and then applying the del-ϕ, del-U and contr-U rules until no further applications of these rules are possible. If the rule R is $\vee R$, $\vee L$ or $\wedge$ then the corresponding rule R' is the same rule. If R is σ, then R' is σ and we use the same constant name u in T' as in T. If R is the U-rule, the corresponding rule R' in T' is either $U\nu$ or $U\mu$ depending on whether U is a maximal or minimal constant. In the latter case, when R' is $U\mu$, we choose the new constant name u' in T' so that it is different from any constant name used in T.

Notice that for any $i \in \mathbb{N}$ such that $k(i) < k(i+1)$ and any $\psi \in \Gamma'_{k(i+1)}$, if there is a dependency sequence from ϕ at point $k(i)$ of T' to ψ at point $k(i+1)$, then $f_i(\phi) \rightarrow f_i(\psi)$ relative to the dependencies between Γ_i and Γ_{i+1} in T.

Let us show then that the tableau T' is proper. Assuming the contrary, by Lemma 4.2.4 there is some point n of T', a minimal or auxiliary constant u of d'_n, and a direct dependency sequence ϕ_0, ϕ_1 from point n such that $u = \phi_i$ for infinitely many i. Notice that there must be infinitely many $i \in \mathbb{N}$ such that $\phi_{k(i)-n} = u$.

We can now project this dependency sequence of T' back to a dependency sequence of T. Choose some m such that $k(m) \geq n$. Define a sequence $\psi_0, \psi_1, \dots$ by: $\psi_i = f_i(\phi_{k(m+i)-n})$. By the observations above, it is a dependency sequence of T. Define then v as the constant of d'_n such that v is not an auxiliary constant and either $v = u$ or u is an unfolding of v. It is easy to see that for every $i \geq n$, if $u \in \Gamma_{k(i)}$ then $f_i(u) = v$. Consequently, the minimal fixpoint constant v occurs infinitely often in the dependency sequence $\psi_0, \psi_1, \dots$ and T is not proper, contrary to the assumption.

It is obvious that if T is propositionally consistent, then so is T', and that for any model $M \in \mathcal{M}_{TL}$, if T agrees with M then T' agrees with M. □

Lemma 4.2.7. Let T be a proper and propositionally consistent definition tree tableau for a guarded μTL-formula ϕ in pnf. There exists a proper and propositionally consistent bundled tableau T' for ϕ such that for any model $M \in \mathcal{M}_{TL}$, if T agrees with M then T' agrees with M.

Proof: Take a proper dependency tree tableau $T = (s_0, \Gamma_0, d_0)(s_1, \Gamma_1, d_1) \dots$ for formula ϕ. Let $k(0) < k(1) < \dots$ stand for the unique increasing series of indices such that $k(0) = 0$ and for every $i \in \mathbb{N}$, the rule applied at point $k(i)$ is neither del-ϕ, del-U nor contr-U, and the rule applied any point between $k(i) + 1$ and $k(i+1) - 1$ is either del-ϕ, del-U or contr-U. Without loss of generality we assume that each constant name is introduced at most once in T, i.e. that no constant is re-introduced by the σ or $U\mu$ rules after having been deleted by the del-ϕ rule.

We shall construct a bundled tableau $T' = (s'_0, \Gamma'_0, d'_0)(s'_1, \Gamma'_1, d'_1)\dots$ for ϕ, together with an infinite increasing sequence of indices $l(0) < l(1) < \dots$ such that $l(0) = 0$. Intuitively the fragment of T' from point $l(i)$ to $l(i+1)$ corresponds to the fragment of T from point $k(i)$ to $k(i+1)$.

For every $i \in \mathbb{N}$ and every $\phi \in \Gamma'_{l(i)}$, define the formula underlying ϕ in $\Gamma_{k(i)}$, denoted by $f_i(\phi)$, as the unique formula $\psi \in \Gamma_{k(i)}$ such that ψ is similar to ϕ, relative to $d_{k(i)}$ and $d'_{l(i)}$. The existence of such a ψ is guaranteed by the structure of the tableau T'.

Define first $l(0) = 0$ and $(s'_0, \Gamma'_0, d'_0) = (0, \{\phi\}, \epsilon)$. For every $i \in \mathbb{N}$, we define $l(i+1)$ and the fragment of T' from $l(i)$ to $l(i+1)$ as follows.

If the rule applied at point $k(i)$ of T is the $\bigcirc$-rule, which implies $k(i+1) = k(i)+1$, then $l(i+1) = l(i)+1$ and $(s'_{l(i)+1}, \Gamma'_{l(i)+1}, d'_{l(i)+1})$ is derived from $(s'_{l(i)}, \Gamma'_{l(i)}, d'_{l(i)})$ by applying the $\bigcirc$-rule.

If the rule R applied at point i of T is not the $\bigcirc$-rule and it is applied to formula $\phi \in \Gamma_{k(i)}$, then the fragment of T' from $l(i)$ to $l(i+1)$ consists of applying the corresponding rule R' to every $\psi \in \Gamma'_{l(i)}$ such that $f_i(\psi) = \phi$. If the rule R is $\vee R$, $\vee L$ or $\wedge$ then the corresponding rule R' is the same rule. If R is σ, then R' is σ and we choose the name of the new constant introduced in T' arbitrarily. If R is $U\nu$ or $U\mu$, then R' is the U-rule.

Notice that each application of the rule R' between $l(i)$ and $l(i+1)$ acts on a different formula. Futhermore, the effect of R' on a formula $\phi \in \Gamma'_j$ where $l(i) \leq j < l(i+1)$ is the same as that of R on $f_i(\phi) \in \Gamma_{k(i)}$. Therefore it is easy to see that if there is a dependency sequence from ϕ in $\Gamma'_{l(i)}$ to ψ in $\Gamma'_{l(i+1)}$, then there is a direct or indirect dependency sequence from $f_i(\phi)$ in $\Gamma_{k(i)}$ to $f_{i+1}(\psi)$ in $\Gamma_{k(i+1)}$.

Let us show then that the resulting bundled tableau T' is proper. Assume the contrary. Then there is some minimal constant u of T' and an infinite dependency sequence $\phi_0, \phi_1, \dots$ from point $l(n)$ for some $n \in \mathbb{N}$ such that $\phi_i = u$ for infinitely many $i \in \mathbb{N}$. From the construction of T', it is easy to see that then $\phi_{l(n+i)-l(n)} = u$ for infinitely many $i \in \mathbb{N}$, as well.

Since for every $i \in \mathbb{N}$ there is a dependency sequence from $\phi_{l(n+i)-l(n)} \in \Gamma'_{l(n+i)}$ to $\phi_{l(n+i+1)-l(n)} \in \Gamma'_{l(n+i+1)}$ in T', by the observation above, there is an indirect dependency sequence from $f_{n+i}(\phi_{l(n+i)-l(n)}) \in \Gamma_{k(n+i)}$ to $f_{n+i+1}(\phi_{l(n+i+1)-l(n)}) \in \Gamma_{k(n+i+1)}$ in T. Consequently, there is an infinite indirect dependency sequence $\psi_0, \psi_1, \dots$ from point $k(n)$ of T such that $\psi_{k(n+i)-k(n)} = f_{n+i}(\phi_{l(n+i)-l(n)})$ for every $i \in \mathbb{N}$.

By Lemma 4.2.5 there is some $m \geq 0$ and constant v such that v is active in ψ_i relative to $d_{k(n)+i}$, for every $i \geq m$, and $\psi_i = v$ for infinitely many $i \geq m$. Since T is proper, by Lemma 4.2.4 we know that this v cannot be a minimal or auxiliary constant.

Take then some $j \in \mathbb{N}$ such that $k(n+j) - k(n) \geq m$ and $\psi_{k(n+j)-k(n)} = v$, and denote by u' the maximal constant $u' = \phi_{l(n+j)-l(n)}$. From the fact that v is active in every ψ_i for $i \geq m$ and $\psi_i = v$ for infinitely many $i \geq m$, it is easy to see that u' is active in every ϕ_i for $i \geq l(n+j) - l(n)$ and $\phi_i = u'$ for infinitely many i. But this contradicts the assumption that $\phi_i = u$ for infinitely many i and the minimal constant u.

Finally, it is obvious that if T is propositionally consistent, then so is T', and that for any model $M \in \mathcal{M}_{TL}$, if T agrees with M then T' agrees with M. $\square$