Journal of Applied Non-Classical Logics
Vol. 00, No. 00, Month 201X, 1–18

RESEARCH ARTICLE

Deciding the Unguarded Modal μ-Calculus

Oliver Friedmann[a] and Martin Lange[b]

[a] Dept. of Computer Science, University of Munich, Germany
oliver.friedmann@googlemail.com
[b] Dept. of Elect. Eng. and Computer Science, University of Kassel, Germany
martin.lange@uni-kassel.de

(. . .)

The modal μ-calculus extends basic modal logic with second-order quantification in terms of arbitrarily nested fixpoint operators. Its satisfiability problem is EXPTIME-complete. Decision procedures for the modal μ-calculus are not easy to obtain though since the arbitrary nesting of fixpoint constructs requires some combinatorial arguments for showing the well-foundedness of least fixpoint unfoldings. The tableau-based decision procedures so far also make assumptions on the unfoldings of fixpoint formulas, e.g. explicitly require formulas to be in guarded normal form. In this paper we present a tableau calculus for deciding satisfiability of arbitrary, i.e. not necessarily guarded μ-calculus formulas. It is based on a new unfolding rule for greatest fixpoint formulas which allows unguarded formulas to be handled without an explicit transformation into guarded form, thus avoiding a (seemingly) exponential blow-up in formula size. We prove soundness and completeness of the calculus, and compare it empirically to using guarded transformation instead. The new unfolding rule can also be used to handle nested star operators in PDL formulas correctly.

Keywords: satisfiability checking; fixpoints; tableaux; automata

1. Introduction

1.1 *Deciding the Modal μ-Calculus*

The modal μ-calculus $\mathcal{L}_\mu$ as introduced by Kozen (Kozen, 1983) is a fundamental modal fixpoint logic. It is expressively equivalent to the bisimulation-invariant fragment of monadic second-order logic (Janin & Walukiewicz, 1996) and can therefore express all bisimulation-invariant properties of Kripke structures that can be defined using finite automata or any other machinery with at most regular expressive power. Consequently, there are embeddings of temporal logics like CTL and CTL* into $\mathcal{L}_\mu$ (Emerson, 1990; Dam, 1994), as well as of dynamic logics like PDL (Kozen, 1983), even when extended with certain extras (Emerson & Lei, 1986).

Decidability of $\mathcal{L}_\mu$ can be established (Kozen & Parikh, 1983) by observing that its semantics can be expressed in monadic second-order logic which is known to be decidable due to Rabin's famous result from 1969 (Rabin, 1969). This, however, only gives a non-elementary upper complexity bound. The easy embedding of PDL yields a lower bound of deterministic exponential time, also known by the time of $\mathcal{L}_\mu$'s invention (Fischer & Ladner, 1979).

Closing this gap has taken some time and effort. Emerson and Streett showed decidability in deterministic triple exponential time (Streett & Emerson, 1984). Their procedure reduces the satisfiability problem to the problem of testing a finite tree automaton for emptiness. This finite tree automaton is obtained as the product of two automata: the first, called *local* automaton, accepts all locally-consistent Hintikka-tree structures for the input formula. It guesses, simply speaking, which subformulas of the input formula are satisfied at which part of a model and ensures that this guess is consistent with the Boolean semantics of the subformula; for instance, a conjunction is only satisfied if both conjuncts are satisfied, etc.

A second automaton, called *global* automaton, is needed which checks for well-foundedness of the unfolding relation for least fixpoint constructs. This problem is characteristic for satisfiability checking procedures for logics with fixpoint constructs. The product of these two accepts exactly the Hintikka tree models of the original formula which is sufficient for deciding satisfiability. Later, Emerson and Jutla have improved the involved automata-theoretic constructions to obtain ExpTime-completeness of this problem (Emerson & Jutla, 2000).

There are also tableau-based decision procedure for (fragments) of $\mathcal{L}_\mu$. Kozen gave a tableau calculus in the introductory paper but could only prove soundness and completeness for the so-called aconjunctive fragment (Kozen, 1983). This has been extended by Walukiewicz to the so-called weak aconjunctive fragment (Walukiewicz, 2000) in the context of finding a complete axiom system for $\mathcal{L}_\mu$. The differences between tableau-based satisfiability checking and a proof system for validity are, however, merely a matter of taste in this setting. The property of being aconjunctive implies that any least fixpoint construct can only regenerate in a foreseeable way through a sequence of Hintikka sets which eliminates a large part of the difficulty in deciding well-foundedness of the unfolding relation. Bradfield and Stirling wrote *"it is an open question whether the tableau technique can be made to work directly for all formulae"* (Bradfield & Stirling, 2001). A tableau calculus which also works for non-aconjunctive formulas has recently been presented by Jungteerapanich (Jungteerapanich, 2009).

Most tableau-based decision procedures still impose a restriction on the syntax of formulas. They only work for formulas in guarded form which intuitively ensures that every infinite sequence of Hintikka sets corresponds to an infinite sequence of states in a Kripke model. Guardedness synchronizes all subformulas in a tableau node via the usual rule for modalities which strips modal operators at the top of formulas and, hence, corresponds semantically to the visiting of new states in a model. When applying rules to unguarded formulas in an arbitrary order, it is possible to leave infinite unfoldings of least fixpoint formulas undetected by continuously unfolding a greatest fixpoint construct.

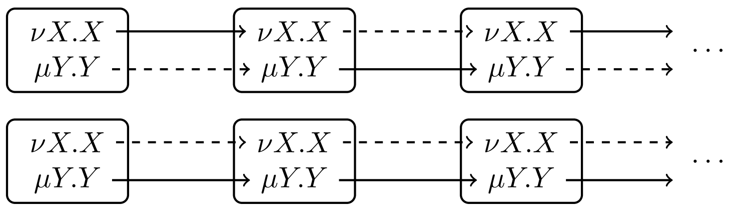
Example. Typical tableau rules for fixpoint formulas replace a formula of the form $\sigma X.\varphi$ by $\varphi[\sigma X.\varphi/X]$, i.e. its *unfolding* which is obtained by replacing every (free) occurrence of the recurrence variable X with the entire formula in the body of this fixpoint definition. Now consider the set $\{\nu X.X, \mu Y.Y\}$ representing the conjunction of the greatest and the least fixpoint of the identity function. These two formulas are unguarded because X and Y do not occur under the scope of modal operators in their defining fixpoint formulas. Moreover, logically, the greatest fixpoint of the identity function is *true* and the least one is ff. Thus, this set is unsatisfiable.

A more mechanical proof that does not rely on the knowledge of equivalences between $\nu X.X$ and *true* etc. would have to detect unsatisfiability by seeing that $\mu Y.Y$ would get unfolded infinitely often. Note that both formulas equal their unfoldings, thus a tableau for this set is just the infinite sequence

$$\boxed{\begin{array}{c}\nu X.X\\ \mu Y.Y\end{array}} \quad \boxed{\begin{array}{c}\nu X.X\\ \mu Y.Y\end{array}} \quad \boxed{\begin{array}{c}\nu X.X\\ \mu Y.Y\end{array}} \qquad \cdots$$

If tableau rules can be applied arbitrarily then it is possible to create this sequence in various ways, for instance by unfolding $\nu X.X$ and $\mu Y.Y$ in a turn-based fashion.

However, it can also be created by only unfolding $\nu X.X$ and never touching $\mu Y.Y$. The difference can be depicted as follows where a dashed line indicates that the target formula has been created from the source one by unfolding; a continuous line indicates that it has just been carried over as a non-principal formula in this rule application.



While these two ways create the same tableau branch, they are fundamentally different in that the first way contains an infinite unfolding of a least fixpoint construct which may cause unsatisfiability whereas the second one does not.

Such a phenomaenon does not occur with guarded formulas. Guardedness ensures that no fixed point formula can be unfolded twice without unfolding all other fixpoint formula that are present in the current tableau node. The reason simply is the rule for modalities which acts globally on all formulas in the current tableau node whereas all other rules only replace a single formula and leave the others untouched.

Unguarded formulas may not occur often in natural specifications formulated in the modal μ-calculus. However, unguardedness naturally occurs when translating formulas of PDL into $\mathcal{L}_\mu$ (Mateescu, 2002). The same holds for PSL, an extension of the linear-time temporal logic LTL with the purpose of defining an expressive standardised program specification language. It can be translated into the linear-time μ-calculus (Lange, 2007), a variant of the $\mathcal{L}_\mu$ that is interpreted over linear flows of time only (Banieqbal & Barringer, 1989). Unguardedness is, in both cases, a result of translating regular expressions with nested Kleene star operators into fixpoint expressions. Similarly, unguarded $\mathcal{L}_\mu$ formulas occur when translating any kind of finite-state automaton with ϵ-transitions into $\mathcal{L}_\mu$ (Dam, 1992; Kaivola, 1997; Emerson & Jutla, 1991). Last but not least, there are also examples of explicitly given unguarded formulas that express interesting properties. For instance, Berwanger showed that two variables suffice to express the winning region in parity games (Berwanger, 2003). This heavily relies on unguarded occurrences of variables. A fully guarded variant is know as the Walukiewicz formula (Walukiewicz, 1996), and that needs as many variables as there are different priorities in the parity game. Thus, Berwanger's formula expresses winning regions in arbitrary games, Walukiewicz's formulas only express them in games of a fixed number of priorities. Unguardedness seems to be necessary for this boost in generality.

A naïve solution is to impose a fairness constraint on the application of unfolding rules. This is equivalent to storing subformulas in a tableau node as a list rather than a set which increases the complexity slightly, namely there would be $\mathcal{O}(n!)$ rather than $\mathcal{O}(2^n)$ many different tableau nodes.

1.2 *Using Guarded Transformation*

It is known that every formula can be transformed into an equivalent guarded one. Such constructions are presented in several places in the literature, either without an explicit analysis of the incurring blow-up which is easily seen to be exponential (Banieqbal & Barringer, 1989; Walukiewicz, 2000), or stating that the blow-up is polynomial, for instance quadratic (Mateescu, 2002) or even just linear (Kupferman, Vardi, & Wolper, 2000). While the transformations in the latter two are correct, their analyzes are both flawed as recently observed (Bruse, Friedmann, & Lange, 2013) and the translations are in fact exponential.

Many decision procedures for the modal μ-calculus explicitly require formulas to be in guarded form (Kozen, 1983; Walukiewicz, 2000; Jungteerapanich, 2009). Thus, the acclaimed results regarding the complexity of deciding the modal μ-calculus with these procedures need to be restricted to the guarded fragment, or one has to factor in an exponential blow-up when using guarded transformation. For instance, Jungteerapanich's tableaux thus only yield a nondterministic doubly exponential time decision procedure for the entire μ-calculus.

1.3 *Deciding Unguarded Formulas*

In this paper we present a tableau-based decision procedure for the full modal μ-calculus in unrestricted form. The requirement for guardedness is eliminated using a special unfolding rule for greatest fixpoint formulas. Intuitively, unfolding of greatest fixpoint constructs leads to two subgoals: one containing this unfolding, the other one not containing it. We prove soundness and completeness of this calculus and show how to obtain a decision procedure from it. This uses some automata-theoretic machinery similar to the use of the global automata in the approaches of Emerson et al.

The paper provides the following benefits: it presents a novel approach of dealing with unguarded fixpoint formulas inside a tableau calculus. This may be applicable to other logics with similar syntactic facets (like nested Kleene stars in PDL for instance). With the required pre-transformation into guarded form, Jungteerapanich's tableaux only lead to a nondeterministic double exponential time algorithm. The decision procedure derived from the tableaux presented here runs in deterministic single exponential time. This even marginally beats the worst-case runtime of the automata-theoretic procedure. Finally, the tableaux presented here are used in what seems to be the first attempt at implementing a decision procedure for $\mathcal{L}_\mu$, realized in the tool MLSolver (Friedmann & Lange, 2010).

The paper is organised as follows. Sect. 2 recalls the modal μ-calculus and necessary technicalities. Sect. 3 presents the tableaux calculus and proves them to be sound and complete with respect to satisfiability. Sect. 5 shows how to obtain a complexity-theoretically optimal decision procedure for the modal μ-calculus from these tableaux. Sect. 6 presents some experimental results showing that transformation into guarded form is worth being avoided.

2. The Modal μ-Calculus

Transition Systems. A *labeled transition system* (LTS) over a set of *action names* Σ and a set of *atomic propositions* $\mathcal{P}$ is a tuple $\mathcal{T} = (S, \rightarrow, \ell)$ where S is a set of *states*, $\rightarrow \subseteq S \times \Sigma \times S$ defines a set of *transitions* between states that are labeled with action names, and $\ell : S \rightarrow 2^{\mathcal{P}}$ labels each state with a set of atomic propositions that are true in this state.

Syntax. Let Σ and $\mathcal{P}$ be as above and $\mathcal{V}$ be a set of variables. Formulas of the modal μ-calculus $\mathcal{L}_\mu$ in positive normal form are given as follows.

$$\varphi \;::=\; q \mid \overline{q} \mid X \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle a \rangle \varphi \mid [a]\varphi \mid \mu X.\varphi \mid \nu X.\varphi$$

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

The operators μ and ν act as *binders* for the variables in a formula. A *free occurrence* of a variable X is therefore one that does not occur under the scope of such a binder. We assume all formulas φ to be *well-named* in the sense that each variable is bound at most once. We will write σ for either μ or ν.

We write $\varphi[\psi/X]$ to denote the formula that results from φ by replacing every free occurrence of the variable X in it with the formula ψ.

2.0.0.1 Fischer-Ladner Closure.. The Fischer-Ladner closure of a formula φ is the least set $Cl(\varphi)$ that contains φ and satisfies the following.

- If $\psi_1 \wedge \psi_2 \in Cl(\varphi)$ or $\psi_1 \vee \psi_2 \in Cl(\varphi)$ then $\psi_1, \psi_2 \in Cl(\varphi)$.
- If $\langle a \rangle \psi \in Cl(\varphi)$ or $[a]\psi \in Cl(\varphi)$ then $\psi \in Cl(\varphi)$.
- If $\sigma X.\psi \in Cl(\varphi)$ then $\psi[\sigma X.\psi/X] \in Cl(\varphi)$.

It is a standard exercise to show that $|Cl(\varphi)|$ is linear in the syntactic length of φ. We therefore define $|\varphi| := |Cl(\varphi)|$.

Fixpoint Nestings. Let φ be fixed and take two fixpoint formulas $\sigma X.\psi, \sigma' Y.\psi' \in Cl(\varphi)$. The latter *depends on* the former if this X occurs freely inside of ψ'. Let $\succ_\varphi$ be the reflexive-transitive closure of this dependency order. The *alternation depth* of φ, $ad(\varphi)$, is the maximal length of a $\succeq_\varphi$-chain s.t. adjacent formulas in this chain are of different fixpoint type μ or ν.

Semantics. Formulas of $\mathcal{L}_\mu$ are interpreted in states s of an LTS $\mathcal{T} = (S, \rightarrow, \ell)$ which we assume fixed for the moment. Let $\rho : \mathcal{V} \rightarrow 2^{\mathcal{S}}$ be an environment used to interpret free variables. We write $\rho[X \mapsto T]$ to denote the environment which maps X to T and behaves like ρ on all other arguments. The semantics is given as a function mapping a formula to the set of states that it is true in w.r.t. the environment.

$$
\begin{array}{ll}
[\![q]\!]_\rho & = \{s \in S \mid q \in \ell(s)\} \\
[\![\overline{q}]\!]_\rho & = \{s \in S \mid q \notin \ell(s)\} \\
[\![X]\!]_\rho & = \rho(X) \\
[\![\varphi \vee \psi]\!]_\rho & = [\![\varphi]\!]_\rho \cup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho & = [\![\varphi]\!]_\rho \cap [\![\psi]\!]_\rho \\
[\![\langle a \rangle \varphi]\!]_\rho & = \{s \in S \mid \exists t \in [\![\varphi]\!]_\rho \text{ with } s \xrightarrow{a} t\} \\
[\![[a]\varphi]\!]_\rho & = \{s \in S \mid \forall t \in \mathcal{S} : \text{ if } s \xrightarrow{a} t \text{ then } t \in [\![\varphi]\!]_\rho\} \\
[\![\mu X.\varphi]\!]_\rho & = \bigcap \{T \subseteq S \mid [\![\varphi]\!]_{\rho[X \mapsto T]} \subseteq T\} \\
[\![\nu X.\varphi]\!]_\rho & = \bigcup \{T \subseteq S \mid T \subseteq [\![\varphi]\!]_{\rho[X \mapsto T]}\}
\end{array}
$$

Two formulas φ and ψ are equivalent, written $\varphi \equiv \psi$, iff for all LTS and all environments ρ we have $[\![\varphi]\!]_\rho = [\![\psi]\!]_\rho$. We may also write $s \models_\rho \varphi$ instead of $s \in [\![\varphi]\!]_\rho$.

Guarded Form. A formula φ is *guarded w.r.t. a variable* X iff every occurrence of X that is bound by some $\sigma X.\psi$ is in the scope of a modal operator $\langle a \rangle$ or $[a]$ within ψ. A formula φ is *guarded* iff φ is guarded w.r.t. every bound variable.

Proposition 1 ((Banieqbal & Barringer, 1989; Walukiewicz, 2000; Kupferman et al., 2000; Mateescu, 2002))**.** *For every $\varphi \in \mathcal{L}_\mu$ there is a guarded φ' s.t. $\varphi' \equiv \varphi$, $|\varphi'| = 2^{\mathcal{O}(|\varphi|)}$, and $ad(\varphi') = ad(\varphi)$.*

$$(\mathrm{Or})\ \frac{\varphi_0 \vee \varphi_1, \Phi}{\varphi_i, \Phi} \qquad (\mathrm{And})\ \frac{\varphi_0 \wedge \varphi_1, \Phi}{\varphi_0, \varphi_1, \Phi} \qquad (\mathrm{FP}_\mu)\ \frac{\mu X.\varphi, \Phi}{\varphi[\mu X.\varphi/X], \Phi}$$

$$(\mathrm{FP}^U_\nu)\ \frac{\nu X.\varphi, \Phi}{\Phi \qquad \varphi[\nu X.\varphi/X], \Phi} \qquad (\mathrm{FP}^G_\nu)\ \frac{\nu X.\varphi, \Phi}{\varphi[\nu X.\varphi/X], \Phi}\ X \text{ guarded in } \varphi$$

$$(\mathrm{Mod})\ \frac{\langle a_1\rangle\varphi_1, \dots, \langle a_n\rangle\varphi_n, [b_1]\psi_1, \dots, [b_m]\psi_m, q_1, \dots, q_k, \overline{p_1}, \dots, \overline{p_h}}{\varphi_1, \{\psi_i \mid a_1 = b_i\} \quad \varphi_2, \{\psi_i \mid a_2 = b_i\} \quad \dots \quad \varphi_n, \{\psi_i \mid a_m = b_i\}}\ \forall i, j. q_i \neq p_j$$

Figure 1. The tableaux rules for $\mathcal{L}_\mu$ satisfiability.

We remark that guarded transformation can increase the number of μ-bound variables in a formula, even exponentially. This measure is used at the end of Sect. 5 in a comparison of different decision procedures.

3. Tableaux for the Modal μ-Calculus

We fix a formula ϑ and present a calculus of infinite tableaux for this particular ϑ. A *pre-tableau* for ϑ is a possibly infinite but finitely-branching tree in which nodes are labeled with subsets of $Sub(\vartheta)$, the set of subformulas of ϑ. The root is labeled with the singleton set containing ϑ, and successors in the tree are being built using the rules in Fig. 1.

The rules for the boolean connectives are straight-forward, and the modal rule (Mod) is also the usual one. Least fixpoint variables are handled using simple unfolding with rule (FP_μ). The handling of greatest fixpoints is different, though. Rule (FP^U_ν) creates two subgoals, one containing the usual unfolding of the fixpoint formula, the other one consisting of the current side formulas only. This rule can be applied to unfold any greatest fixpoint formula. On the other hand, rule (FP^G_ν) is the usual unfolding rule which can only be applied to formulas in which the bound variable is guarded.

Remember the description of what can happen when two fixpoint formulas, in particular a least and a greatest one, occur in a formula set which is to be tested for satisfiability, as explained at the end of Section 1.1. If both fixpoint formulas are being unfolded in the style of rules (FP_μ) and (FP^G_ν), and they contain unguarded occurrences of their fixpoint variables then it is possible to continuously unfold one of them and miss the other. Now there are various cases to consider based on the intuition that an infinite unfolding of a least fixpoint formula is bad (in the sense of an unsatisfied formula in this set) whereas an infinite unfolding of a greatest fixpoint formula is fine (because it signals satisfaction of this formula in the set).

- If both the greatest and the least fixpoint formula are being unfolded infinitely often then the infinite unfolding of the least one witnesses unsatisfiability.
- If the least one is unfolded infinitely often and the greatest one is not because an unguarded occurrence of the least fixpoint variable makes the greatest one "miss its turn" then unsatisfiability is still witnessed by the infinite unfolding of the least fixpoint formula. Thus, unguarded occurrences of variables bound by a least fixpoint quantifier are harmless.
- If the greatest one gets unfolded infinitely often and the least one does not because

it is being missed out due to an unguarded occurrence of the greatest fixpoint variable then this is problematic because the infinite unfolding of the least fixpoint formula that witnesses unsatisfiability is not visible in that case. Hence, unguarded occurrences of greatest fixpoint variables are problematic, and this is where rule (FP_ν^G) comes into play. Apart from the usual subgoal in which the greatest fixpoint variable has been unfolded, it creates a second subgoal in which this formula is missing. Note that if $\nu X.\varphi \wedge \Phi$ is satisfiable then so is Φ. Thus, the additional subgoal Φ does not prevent the tableau to be successful for satisfiable formulas. It simply checks for infinite unfoldings of least fixpoint formulas which could be missed out in the presence of infinite unfoldings of greatest fixpoint formulas via unguarded occurrences of their variables.

Next we will formalise the intuitive notion of infinite unfolding in a tableau branch. A formula ϑ induces the *connection* relation $\rightsquigarrow \subseteq 2^{Sub(\vartheta)} \times Sub(\vartheta) \times 2^{Sub(\vartheta)} \times Sub(\vartheta)$ defined as follows. We have $\Phi, \varphi \rightsquigarrow \Psi, \psi$ iff there is an instance of a rule of Fig. 1 s.t.

- $\varphi \in \Phi$, $\psi \in \Psi$, and
- Φ is the conclusion (on top), Ψ is one of the premisses (below), and
- either φ is not principal in this rule application and $\psi = \varphi$, or φ is a principal formula in Φ and ψ is a replacement of φ.

For example, in rule (And), $\varphi_0 \wedge \varphi_1$ is connected to both φ_0 and φ_1. In rule (Mod), $\Box\psi_j$ is connected to ψ_j in any premiss, literals are not connected to anything, and $\Diamond\varphi_i$ is only connected to φ_i in the i-th premiss; etc.

A *thread* in an infinite pre-tableaux branch $\Phi_0, \Phi_1, \Phi_2, \dots$ is an infinite sequence $\varphi_0, \varphi_1, \varphi_2, \dots$ s.t. $\Phi_i, \varphi_i \rightsquigarrow \Phi_{i+1}, \varphi_{i+1}$ for every $i \in \mathbb{N}$. It is called *active* if the thread's formulas are principal infinitely often.

Note that only the unfolding rules (FP_μ) and (FP_ν^U) do not decrease the size of a principal formula. Hence, each active thread must contain infinitely many formulas of the form $\sigma X.\psi$. A thread is called μ-thread if the greatest (w.r.t. $\succeq_\vartheta$) formula occurring in it is of the form $\mu X.\varphi$. If it is of the form $\nu X.\varphi$ then the thread is a ν-thread. The variable X is called *thread variable*. The following is not hard to see.

Lemma 2. *Every infinite branch in pre-tableau contains at least one active thread and every active thread is either of type μ or ν.*

A *tableau* for ϑ is a pre-tableau s.t. every finite branch ends in a node labeled with $\Box$-formulas and consistent literals only, and every infinite branch does not have an active μ-thread.

Example. Consider $\varphi = \nu X.\,(\langle a\rangle X \vee \mu Y.(X \vee \langle b\rangle Y)) \wedge \nu R.\mu S.\,(([a]S) \vee ([b]R))$ which states that every path consists of a- and b-labelings, every path has infinitely many b's, and that there exists a path with infinitely many a's. This formula is obviously satisfiable.

See Fig. 2 for a tableau witnessing the satisfiability of φ. We write φ_F as an abbreviation for the fixpoint bodies, i.e. $\varphi_S = ([a]S) \vee ([b]R)$, etc.; the tableau has only infinite branches, and every thread is a ν-thread. All threads are marked by the arrow notation.

We conclude this section with a remark on the handling of greatest fixpoint formulas. Many formulas used in applications are naturally guarded. Since the tableau calculus is sound and complete for the entire $\mathcal{L}_\mu$, it can be used for guarded formulas as well. However, handling guarded greatest fixpoint operators with rule (FP_ν^U) may introduce unnecessary subgoals. Rule (FP_ν^G) can therefore be regarded as an optimization. However, it is not a priori clear whether it is advisable to use this optimization. It clearly reduces the number of immediate subgoals in a tableau but these subgoals may be

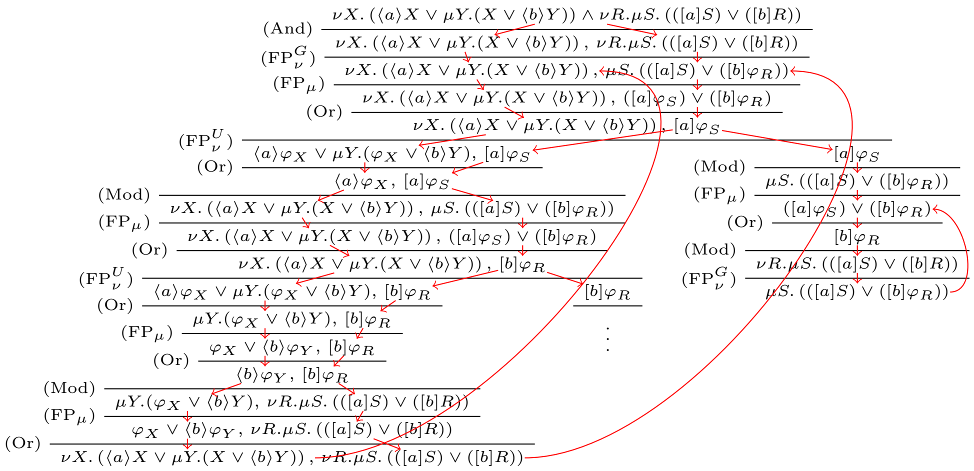

(And) $\nu X.(\langle a\rangle X \vee \mu Y.(X \vee \langle b\rangle Y)) \wedge \nu R.\mu S.(([a]S) \vee ([b]R))$
(FP$^G_\nu$) $\nu X.(\langle a\rangle X \vee \mu Y.(X \vee \langle b\rangle Y))\,,\ \nu R.\mu S.(([a]S) \vee ([b]R))$
(FP$_\mu$) $\nu X.(\langle a\rangle X \vee \mu Y.(X \vee \langle b\rangle Y))\,,\ \mu S.(([a]S) \vee ([b]\varphi_R))$
(Or) $\nu X.(\langle a\rangle X \vee \mu Y.(X \vee \langle b\rangle Y))\,,\ ([a]\varphi_S) \vee ([b]\varphi_R)$
$\nu X.(\langle a\rangle X \vee \mu Y.(X \vee \langle b\rangle Y))\,,\ [a]\varphi_S$
(FP$^U_\nu$) $\langle a\rangle\varphi_X \vee \mu Y.(\varphi_X \vee \langle b\rangle Y),\ [a]\varphi_S$
(Or) $\langle a\rangle\varphi_X,\ [a]\varphi_S$
(Mod) $\nu X.(\langle a\rangle X \vee \mu Y.(X \vee \langle b\rangle Y))\,,\ \mu S.(([a]S) \vee ([b]\varphi_R))$
(FP$_\mu$) $\nu X.(\langle a\rangle X \vee \mu Y.(X \vee \langle b\rangle Y))\,,\ ([a]\varphi_S) \vee ([b]\varphi_R)$
(Or) $\nu X.(\langle a\rangle X \vee \mu Y.(X \vee \langle b\rangle Y))\,,\ [b]\varphi_R$
(FP$^U_\nu$) $\langle a\rangle\varphi_X \vee \mu Y.(\varphi_X \vee \langle b\rangle Y),\ [b]\varphi_R$
(Or) $\mu Y.(\varphi_X \vee \langle b\rangle Y),\ [b]\varphi_R$
(FP$_\mu$) $\varphi_X \vee \langle b\rangle\varphi_Y,\ [b]\varphi_R$
(Or) $\langle b\rangle\varphi_Y,\ [b]\varphi_R$
(Mod) $\mu Y.(\varphi_X \vee \langle b\rangle Y),\ \nu R.\mu S.(([a]S) \vee ([b]R))$
(FP$_\mu$) $\varphi_X \vee \langle b\rangle\varphi_Y,\ \nu R.\mu S.(([a]S) \vee ([b]R))$
(Or) $\nu X.(\langle a\rangle X \vee \mu Y.(X \vee \langle b\rangle Y))\,,\ \nu R.\mu S.(([a]S) \vee ([b]R))$
$[b]\varphi_R$
$\vdots$
$[a]\varphi_S$
(Mod) $\mu S.(([a]S) \vee ([b]\varphi_R))$
(FP$_\mu$) $([a]\varphi_S) \vee ([b]\varphi_R)$
(Or) $[b]\varphi_R$
(Mod) $\nu R.\mu S.(([a]S) \vee ([b]R))$
(FP$^G_\nu$) $\mu S.(([a]S) \vee ([b]\varphi_R))$

Figure 2. A tableau for $\nu X.(\langle a\rangle X \vee \mu Y.(X \vee \langle b\rangle Y)) \wedge \nu R.\mu S.(([a]S) \vee ([b]R))$

present somewhere else anyway in which case it only reduces the number of connections between subgoals. Neither decreases the asymptotic complexity of the calculus.

4. Correctness

The aim of this section is to prove that the tableau calculus presented above is sound and complete for all μ-calculus formulas, not just guarded ones. Like other correctness proofs for μ-calculus machinery, it needs some technicalities and some combinatorics.

4.1 *Approximants and Signatures*

We need *fixpoint approximants* in order to prove absence of any μ-threads in tableaux. Here we introduce them via annotations of fixpoint formulas with ordinal numbers. These annotated fixpoint formulas are interpreted in a way that is different to ordinary fixpoint formulas. Let $\mathcal{T} = (S, \to, \ell)$ be the underlying transition system.

$$\begin{array}{llll}
[\![\mu^0 X.\psi]\!]_\rho & := \emptyset & [\![\nu^0 X.\psi]\!]_\rho & := S \\
[\![\mu^{\alpha+1} X.\psi]\!]_\rho & := [\![\psi]\!]_{\rho[X\mapsto[\![\mu^\alpha X.\psi]\!]_\rho]} & [\![\nu^{\alpha+1} X.\psi]\!]_\rho & := [\![\psi]\!]_{\rho[X\mapsto[\![\nu^\alpha X.\psi]\!]_\rho]} \\
[\![\mu^\lambda X.\psi]\!]_\rho & := \bigcup\limits_{\alpha<\lambda} [\![\mu^\alpha X.\psi]\!]_\rho & [\![\nu^\lambda X.\psi]\!]_\rho & := \bigcap\limits_{\alpha<\lambda} [\![\nu^\alpha X.\psi]\!]_\rho
\end{array}$$

where α is an arbitrary ordinal and λ is a limit ordinal.

A *signature* is an annotation of a formulas fixpoint subformulas with ordinal numbers. We distinguish two types of signatures: a *μ-signature* annotates least fixpoint subformulas, a *ν-signature* annotates greatest fixpoint subformulas. We write φ^ζ to denote the annotation of fixpoint formulas of corresponding type in φ with the values in ζ. Remember that fixpoint subformulas of a formula ϑ are partially ordered by $\succeq$. This extends to a lexicographic and well-founded order of μ- or ν-signatures on ϑ which we will also call $\succeq$.

The following lemma summarizes well-known facts about signatures that will be used in the proofs later on.

Lemma 3. *Let s be a state in a transition system $\mathcal{T}$, $\varphi \in \mathcal{L}_\mu$.*

(1) $s \in [\![\varphi]\!]_\rho$ iff there is a μ-signature ζ s.t. $s \in [\![\varphi^\zeta]\!]_\rho$.

(2) $s \notin [\![\varphi]\!]_\rho$ *iff there is a* ν*-signature* ζ *s.t.* $s \notin [\![\varphi^\zeta]\!]_\rho$.

(3) Let φ' *result from* φ *by replacing some* $\mu X.\psi$ *in it with its unfolding* $\psi[\mu X.\psi/X]$. *Suppose there is a* μ*-signature* ζ *s.t.* $s \in [\![\varphi^\zeta]\!]_\rho$. *Then there is a* μ*-signature* ζ' *with* $\zeta \not\succeq \zeta'$ *and* $s \in [\![\varphi'^{\zeta'}]\!]_\rho$.

(4) Let φ' *result from* φ *by replacing some* $\nu X.\psi$ *in it with its unfolding* $\psi[\nu X.\psi/X]$. *Suppose there is a* ν*-signature* ζ *s.t.* $s \notin [\![\varphi^\zeta]\!]_\rho$. *Then there is a* ν*-signature* ζ' *with* $\zeta \not\succeq \zeta'$ *and* $s \notin [\![\varphi'^{\zeta'}]\!]_\rho$.

4.2 *Soundness*

We represent (pre-)tableaux as pointed directed acyclic graphs $(V, v_0, \prec, \mathcal{M})$ with V being the set of nodes, v_0 being the initial node, $\prec$ being the transition relation and $\mathcal{M}$ being a labeling function that maps each node $v \in V$ to the corresponding sequent.

Let $\mathcal{P} = (V, v_0, \prec, \mathcal{M})$ be a tableau for ϑ. A *ν-strategy for $\mathcal{P}$* is a partial map $\varrho : V \to V$ that is defined on every node v that is the conclusion of the application of the (FP_ν^U)-rule and fulfills for every such v that $v \prec \varrho(v)$.

A branch $v_0, v_1, \ldots$ in $\mathcal{P}$ *conforms* with ϱ iff for every i with v_i being the conclusion of the application of the (FP_ν^U)-rule it holds that $\varrho(v_i) = v_{i+1}$. We say that a node $v \in V$ is *ϱ-reachable* iff v belongs to a ϱ-conforming branch. The set of ϱ-reachable nodes is denoted by V_ϱ. The pair $(\mathcal{P}, \varrho)$ is called *collapsible* if every ϱ-conforming branch in $\mathcal{P}$ is either finite or comprises infinitely many applications of the (Mod)-rule.

Let $(\mathcal{P}, \varrho)$ be collapsible. We define a *lift operation* $l_{(\mathcal{P},\varrho)} : V_\varrho \to V_\varrho$ that maps every node $v \in V_\varrho$ to v if v is a sink or the conclusion of the application of the (Mod)-rule and otherwise to $l_{(\mathcal{P},\varrho)}(w)$ where w is the uniquely defined ϱ-conforming successor of v. As $(\mathcal{P}, \varrho)$ is collapsible, $l_{(\mathcal{P},\varrho)}$ is indeed well-defined.

Every collapsible $(\mathcal{P}, \varrho)$ for a formula ϑ induces an generic interpretation $\mathcal{T}_{(\mathcal{P},\varrho)} = (V_\varrho, \to, \ell)$ with $\ell : v \mapsto \mathcal{M}(v) \cap \mathcal{P}$ and $v \xrightarrow{a} w$ for two nodes $v, w \in V_\varrho$ iff there is an $u \in V$ with $v \prec u$ *connected via an a-label* and $l_{(\mathcal{P},\varrho)}(u) = w$.

Next, we define an annotation that counts for every formula φ in a sequent, how often every ν-bound variable has been unfolded since the last occurrence of the modal rule. This will help us to define a generic ν-strategy that results in collapsible tableaux while ensuring that every potentially relevant unguarded ν-bound variable that occurs in a thread is unfolded at least once.

Let $\mathcal{P} = (V, v_0, \prec, \mathcal{M})$ be a tableau for ϑ. The *ν-variable annotation for $\mathcal{P}$* is a function $\mathcal{A}_\mathcal{P}$ that maps every node $v \in V$ and every formula $\varphi \in \mathcal{M}(v)$ to a set of sets of ν-variables $\mathcal{A}_\mathcal{P}(v, \varphi)$.

We define the function inductively. For the initial v_0, $\mathcal{A}_\mathcal{P}(v_0, \vartheta) = \{\emptyset\}$. Let now $v, u \in V$ with $v \prec u$ and $\mathcal{A}_\mathcal{P}(v, *)$ be already defined. Then

- $\mathcal{A}_\mathcal{P}(u, \varphi) = \{\emptyset\}$ if $\mathcal{M}(v)$ is the conclusion of a (Mod)-application,
- $\mathcal{A}_\mathcal{P}(u, \varphi) = \{U \cup \{X\} \mid (U \setminus \{X\}) \in \mathcal{A}'_\mathcal{P}(u, \varphi)\}$ if $\varphi = \psi[\nu X.\psi/X]$ and $\nu X.\psi$ is principal in $\mathcal{M}(v)$, and
- $\mathcal{A}_\mathcal{P}(u, \varphi) = \mathcal{A}'_\mathcal{P}(u, \varphi)$ otherwise,

where $\mathcal{A}'_\mathcal{P}(u, \varphi) := \bigcup\{\mathcal{A}_\mathcal{P}(v, \psi) \mid (\mathcal{M}(v), \psi) \rightsquigarrow (\mathcal{M}(u), \varphi)\}$.

Next, we define a *canonic ν-strategy* $\varrho_\mathcal{P}$ for a tableau $\mathcal{P}$ as follows. Let v be a node in $\mathcal{P}$ s.t. $\mathcal{M}(v)$ is the conclusion of an application of the (FP_ν^U)-rule with $\nu X.\psi$ as principal formula and let u be the successor of v discarding the fixpoint body and w be the successor following the fixpoint body. Then $\varrho_\mathcal{P}(v) = w$ if there is an $U \in \mathcal{A}_\mathcal{P}(v, \nu X.\psi)$ with $X \notin U$ and $\varrho_\mathcal{P}(v) = u$ otherwise.

Lemma 4. *Let $\mathcal{P}$ be a tableau for φ. Then $(\mathcal{P}, \varrho_\mathcal{P})$ is collapsible.*

Proof. Assume that $(\mathcal{P}, \varrho_\mathcal{P})$ is not collapsible, hence there is an infinite $\varrho_\mathcal{P}$-conforming branch $v_0, v_1, \dots$ in $\mathcal{P}$ that contains only finitely many applications of the (Mod)-rule. Let $i^* \geq 0$ s.t. $\mathcal{M}(v_i)$ is not the conclusion of the application of a (Mod)-rule for all $i \geq i^*$.

First observe the following fact. Let $i \geq i^*$ and $\varphi \in \mathcal{M}(v_i)$. Let $U \in \mathcal{A}_\mathcal{P}(v_i, \varphi)$. This implies that there is a (prefix of a) thread s going through φ in the node v_i s.t. between i^* and i, we have

- zero unfoldings for fixpoints $\nu X.\psi$ with $X \notin U$, and
- one unfolding for fixpoints $\nu X.\psi$ with $X \in U$.

Let now $t = \varphi_0, \varphi_1, \dots$ be an active thread with thread variable X, existing due to Lemma 2. Due to the fact that $\mathcal{P}$ is a tableau, X must be of type ν.

Let $j_0, j_1, \dots$ be an infinite sequence of ascending numbers with $j_0 \geq i^*$ s.t. $\varphi_{j_k} = \nu X.\psi$ is the principal formula in $\mathcal{M}(v_{j_k})$ for all k.

For every j_k, there is an $U \in \mathcal{A}_\mathcal{P}(v_{j_k}, \nu X.\psi)$ s.t. $X \notin U$ by the canonic ν-strategy. In other words, for every k there is a (prefix of a) thread s_k going through $\nu X.\psi$ in the node v_{j_k} s.t. between i^* and j_k, we have no more than one unfolding per ν-fixpoint.

Now note that by the pigeonhole principle (infinitely many s_k share the same prefixes and they need to split infinitely often), there are infinitely many s_k which are principal between i^* and j_k. By König's Lemma, this implies that there is an active thread s that has no more than one unfolding per ν-fixpoint.

Since s is clearly not a ν-fixpoint, it follows by Lemma 2 that s is a μ-fixpoint. But this is impossible with $\mathcal{P}$ being a tableau. □

Due to the fact that $(\mathcal{P}, \varrho_\mathcal{P})$ is always collapsible, we can define the *canonic interpretation* $\mathcal{T}_\mathcal{P}$ as $\mathcal{T}_{(\mathcal{P}, \varrho_\mathcal{P})}$.

Theorem 5. *A formula ϑ is satisfiable if there is a tableau $\mathcal{P}$ for ϑ. Particularly, $\mathcal{T}_{(\mathcal{P}, \varrho_\mathcal{P})} \models \vartheta$.*

Proof. By contradiction assume that $\mathcal{T}_{(\mathcal{P}, \varrho_\mathcal{P})} \not\models \vartheta$. We extract a branch $v_0, v_1, \dots$ in $\mathcal{P}$, a sequence of formulas $\varphi_0, \varphi_1, \dots$ with $\varphi_i \in \mathcal{M}(v_i)$ for all i and a sequence of ν-signatures $\zeta_0 \succeq \zeta_1 \succeq \dots$ s.t. the following conditions hold for all i.

(1) ζ_i is the least ν-signature s.t. $l_{(\mathcal{P}, \varrho_\mathcal{P})}(v_i) \not\models \varphi_i^{\zeta_i}$
(2) $(\mathcal{M}(v_i), \varphi_i) \rightsquigarrow (\mathcal{M}(v_{i+1}), \varphi_{i+1})$
(3) $\varphi_i = \nu X.*$ principal implies that $\zeta_i \not\succeq \zeta_{i+1}$

In the following construction of signatures, we will simply show that there are signatures fulfilling all properties disregarding being the least one. Then, we simply select the subsequent signature to be the least one fulfilling the first property. Note that this signature then also fulfills all the other properties.

For $i = 0$ let v_0 be the root of $\mathcal{P}$, $\varphi_0 := \vartheta$ and ζ_0 be the smallest ν-signature s.t. $l_{(\mathcal{P}, \varrho_\mathcal{P})}(v_0) \not\models \varphi_0^{\zeta_0}$ which exists due the Lemma 3.

For $i \rightsquigarrow i+1$ we distinguish on the subsequent rule application. Note that is impossible that v_i ends in a sink. If the next rule to be applied is the (Mod)-rule, we distinguish on whether $\varphi_i = \langle a \rangle \varphi_{i+1}$ or $\varphi_i = [a]\varphi_{i+1}$ which are the only possible cases due to the construction of $\mathcal{T}_{(\mathcal{P}, \varrho_\mathcal{P})}$. If $\varphi_i = \langle a \rangle \varphi_{i+1}$ let v_{i+1} be the successor of v_i following φ_{i+1} and note that $l_{(\mathcal{P}, \varrho)}(v_{i+1}) \not\models \varphi_{i+1}^{\zeta_i}$ indeed holds. If $\varphi_i = [a]\varphi_{i+1}$, select $v_i \prec v_{i+1}$ s.t. $l_{(\mathcal{P}, \varrho)}(v_{i+1}) \not\models \varphi_{i+1}^{\zeta_i}$ holds.

Otherwise let v_{i+1} be the unique successor of v_i. Assume that φ_i is principal in the following rule application since otherwise simply set $\varphi_{i+1} := \varphi_i$. Otherwise, if $\varphi_i = \psi_0 \vee \psi_1$ let φ_{i+1} be the unique successor of φ_i and note that all conditions hold. If

$\varphi_i = \psi_0 \wedge \psi_1$ let $j = 0, 1$ s.t. $l_{(\mathcal{P},\varrho_\mathcal{P})}(v_{i+1}) \not\models \psi_j^{\zeta_i}$ and set $\varphi_{i+1} := \psi_j$.

If otherwise $\varphi_i = \sigma X.\psi$ let φ_{i+1} be the unique successor of φ_i and note that due to Lemma 3 there is a signature ζ'_{i+1} s.t. all conditions hold.

We finally need to show that it is impossible that $\varphi_i = \nu X.\psi$ for some ν-bound X whenever the respective (FP^U_ν)-rule application does not follow ψ. By contradiction assume that $\varphi_i = \nu X.\psi$ principal for some ν-bound X and there is no set $U \in \mathcal{A}_\mathcal{P}(v_i, \nu X.\psi)$ with $X \notin U$. By construction of $\mathcal{A}_\mathcal{P}$, this implies that there is some $j < i$ with $l_{(\mathcal{P},\varrho_\mathcal{P})}(v_j) = l_{(\mathcal{P},\varrho_\mathcal{P})}(v_i)$, $v_j = \nu X.\psi$ and $v_{j+1} = \psi[\nu X.\psi/X]$. By construction of the sequence of signatures, it follows that $\zeta_j \succeq\!\!\!\!/\;\; \zeta_{j+1} \succeq \zeta_i$. But this cannot be the case with ζ_j and ζ_i both being the least ν-signature that falsifies X w.r.t. the same state.

As the modal rule is applied infinitely often in the extracted branch, $\varphi_0, \varphi_1, \ldots$ is an active thread. Since $\mathcal{P}$ is a tableau, $\varphi_0, \varphi_1, \ldots$ is a ν-thread.

Let X^* be the outermost variable in $\varphi_0, \varphi_1, \ldots$ that is unfolded infinitely often. Let i^* be arbitrary s.t. there is no variable $Y > X^*$ with $\varphi_j = \sigma Y.*$ for any $j \geq i^*$. Consider the sequence of signatures $\zeta_{i^*} \succeq \zeta_{i^*+1} \succeq \ldots$ and note that we have:

$$\zeta_i \succeq\!\!\!\!/\;\; \zeta_{i+1} \text{ whenever } \varphi_i = \nu X^*.\psi \text{ and } \varphi_{i+1} = \psi[\nu X^*.\psi/X^*]$$

Therefore we have an infinitely descending sequence $\zeta_{i^*}, \zeta_{i^*+1}, \ldots$ which is impossible as ordinals are well-founded. □

4.3 *Completeness*

Theorem 6. *There is a tableau for a fromula ϑ if ϑ is satisfiable.*

Proof. Let ϑ be a closed formula and $\mathcal{T} = (\mathcal{S}, \longrightarrow, \ell)$ be a transition system and $s_0 \in \mathcal{S}$ be a state s.t. $s_0 \models \vartheta$.

We inductively construct a state-labeled pre-tableau as follows. Starting with the labeled sequence $s_0 : \vartheta$, we apply the following rules in an arbitrary but eligible ordering systematically backwards.

$$(\mathrm{Or})\ \frac{s : \varphi_0 \vee \varphi_1, \Phi}{s : \varphi_i, \Phi}\ (*) \qquad\qquad (\mathrm{And})\ \frac{s : \varphi_0 \wedge \varphi_1, \Phi}{s : \varphi_0, \varphi_1, \Phi}$$

$$(\mathrm{FP}_\mu)\ \frac{s : \mu X.\varphi, \Phi}{s : \varphi[\mu X.\varphi/X], \Phi} \qquad\qquad (\mathrm{FP}^U_\nu)\ \frac{s : \nu X.\varphi, \Phi}{s : \Phi \qquad s : \varphi[\nu X.\varphi/X], \Phi}$$

$$(\mathrm{Mod})\ \frac{s : \langle a_1\rangle\varphi_1, \ldots, \langle a_n\rangle\varphi_n, [b_1]\psi_1, \ldots, [b_m]\psi_m, q_1, \ldots, q_k, \overline{p_1}, \ldots, \overline{p_h}}{s_1{:}\varphi_1,\{\psi_i \mid a_1{=}b_i\} \quad s_2{:}\varphi_2,\{\psi_i \mid a_2{=}b_i\} \quad \ldots \quad s_m{:}\varphi_n,\{\psi_i \mid a_m{=}b_i\}}\ (**)$$

with the following side conditions:

- $(*)$: For every μ-signature ζ with $s \models (\varphi_0 \vee \varphi_1)^\zeta$ it holds that $s \models \varphi_i^\zeta$.
- $(**)$: $s \models (\langle a_1\rangle\varphi_1, \ldots, \langle a_n\rangle\varphi_n, [b_1]\psi_1, \ldots, [b_m]\psi_m, q_1, \ldots, q_k, \overline{p_1}, \ldots, \overline{p_h})$ implies for every i that $s \longrightarrow^{a_i} s_i$ and $s_i \models (\varphi_i, \{\psi_j \mid a_i = b_j\})$. Additionally, for every μ-signature ζ and every i it holds that $s \models (\langle a\rangle\varphi_i)^\zeta$ implies $s_i \models \varphi_i^\zeta$.

Consider that this construction indeed yields pre-tableau with each state-labeled sequence $s : \Phi$ satisfying $s \models \Phi$ as well as all side conditions due to Lemma 3. Moreover note that every finite branch ends in a node labeled with $[*]$-formulas and consistent literals only.

By contradiction assume that the pre-tableau is not a tableau, hence there is a labeled branch $s_0 : \Phi_0, s_1 : \Phi_1, \ldots$ (with $\Phi_0 = \{\vartheta\}$) and a μ-thread $t = t_0, t_1, \ldots$ with $t_i \in \Phi_i$ for all i.

We argue as in the soundness proof that this is impossible. □

Putting the soundness (Thm. 5) and completeness (Thm. 6) together yields the following statement.

Theorem 7. *Let $\vartheta \in \mathcal{L}_\mu$. Then ϑ is satisfiable iff there is a tableau for ϑ.*

5. A Decision Procedure Based on Tableaux

A natural question is: can the tableaux of the previous section be used to decide satisfiability for $\mathcal{L}_\mu$? In this section we will show that the answer is positive and compare the resulting procedure with existing ones. The procedure works as follows. We first show that pre-tableau branches without μ-threads can be recognized by a deterministic parity automaton (DPA). The pre-tableaux nodes can be annotated with states of this DPA resulting in a graph equipped with a parity condition. There are two kinds of branching in this graph: existential branching corresponding to choices with rule (OR), and universal branching corresponding to choices between different subgoals. This graph is finite and forms a parity game (McNaughton, 1993). The question whether or not a tableau exists for an input formula reduces to the problem of solving this game.

5.1 *Automata-Theoretic Machinery*

Again, we fix a formula ϑ. It induces an alphabet Σ_ϑ representing transitions from a goal to a subgoal in a rule application. These symbolic alphabet letter should determine a subgoal of a given goal uniquely and succinctly. Clearly this can be done by naming the principal formula, possibly its replacement, as well as the number of the subgoal of which there can at most be $|\vartheta|$ many. It is clearly possible to realize this in an alphabet Σ_ϑ of size $\mathcal{O}(|\vartheta|^3)$.

With an infinite branch $\rho = \Phi_0, \Phi_1, \ldots$ of a pre-tableau for ϑ we associate a word $w_\rho \in \Sigma_\vartheta^\omega$ in the natural way: the i-th letter of w_ρ is the symbol representing the application of the rule between Φ_i and Φ_{i+1}. Let $\mathit{BadBranch}(\vartheta)$ be the set of all words representing an infinite branch in a pre-tableau for ϑ which contains an active μ-thread, i.e. the set of all branches which may not occur in a tableau.

A nondeterministic parity automaton (NPA) is a tuple $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$ where Q is a finite set of states, Σ is the underlying alphabet, $q_0 \in Q$ is a designated starting state, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation as usual, and $\Omega : Q \to \mathbb{N}$ is the priority function. A run $\rho = q_0, q_1, \ldots$ on an infinite word $w \in \Sigma^\omega$ is defined as usual. It is accepting if the highest priority occurring infinitely often in $\Omega(q_0), \Omega(q_1), \ldots$ is even. Let $|\mathcal{A}|$ denote the size of $\mathcal{A}$, measured as its number of states. Its index, $\mathit{idx}(\mathcal{A})$, is the number of distinct priorities assigned to its states. An NPA as above is deterministic (DPA) if $\delta : Q \times \Sigma \to Q$ in effect. A nondeterministic Büchi automaton (NBA) is an NPA as above with $\Omega : Q \to 1, 2$.

Lemma 8. *There is an NPA $\mathcal{B}'_\vartheta$ over Σ_ϑ s.t. $|\mathcal{B}'_\vartheta| \leq 2 \cdot |\vartheta|$, $\mathit{idx}(\mathcal{B}'_\vartheta) \leq \mathit{ad}(\vartheta) + 2$, and $L(\mathcal{B}'_\vartheta) = \mathit{BadBranch}(\vartheta)$.*

Proof. The NPA simply guesses threads by tracing single formulas from $\mathit{Cl}(\vartheta)$ in its state set. Upon reading an input letter it knows whether the next rule application transforms the currently traced subformula or whether it remains the same on that thread. In order to distinguish inactive threads from active threads, the NBA utilizes a bit to indicate that the focussed thread has been unfolded in the last transition. A parity condition that reflects the alternation depth of each formula inside $Cl(\vartheta)$ can then be used in order to recognise $BadBranch(\vartheta)$. □

It is a standard exercise in automata theory to show that every NPA can equivalently be transformed into an NBA with a quadratic blow-up only.

Lemma 9. *There is an NBA $\mathcal{B}_\vartheta$ over Σ_ϑ s.t. $|\mathcal{B}_\vartheta| \leq 2 \cdot |\vartheta| \cdot (ad(\vartheta) + 2)$, and $L(\mathcal{B}_\vartheta) = BadBranch(\vartheta)$.*

As said above, the goal is to create a parity game as a product of all possible pre-tableau nodes with the states of a automaton recognizing branches that do *not* contain a μ-thread. Hence, complementation of the automaton $\mathcal{B}_\vartheta$ is needed. Moreover, this automaton needs to be deterministic to ensure that common prefixes of two different branches can be paired with a single run of the automaton.

Theorem 10 ((Piterman, 2006)). *For every NBA $\mathcal{B}$ with n states there is a DPA $\mathcal{A}$ with $2^{\mathcal{O}(n \log n)}$ states and index $\mathcal{O}(n)$ s.t. $L(\mathcal{A}) = \overline{L(\mathcal{B})}$.*

Combining this theorem with Lemma 9 yields the following. Note that $ad(\vartheta) \leq |\vartheta|$, and that if $k \leq n$ then $\log(nk) \leq 2 \cdot \log n$.

Corollary 11. *Let $\vartheta \in \mathcal{L}_\mu$ with $n := |\vartheta|$ and $k := ad(\vartheta)$. There is a DPA $\mathcal{A}_\vartheta$ over Σ_ϑ s.t. the number of states in $\mathcal{A}_\vartheta$ is bounded by $2^{\mathcal{O}(n \cdot k \cdot \log n)}$, its index is $\mathcal{O}(n \cdot k)$, and $L(\mathcal{B}_\vartheta) = \overline{BadBranch(\vartheta)}$.*

5.2 *Reduction to Parity Game Solving*

The algorithmic solution to the satisfiability problem is provided by a reduction to parity game solving. A parity game is a tuple $G = (V, V_0, V_1, E, v_0, \Omega)$ s.t. (V, E) is a directed graph with total edge relation E and node set partitioned into V_0 and V_1, v_0 is a designated starting node, and $\Omega : V \to \mathbb{N}$ is a priority function. The game is played between two players 0 and 1 who push a token along the edges of a the graph starting in v_0. If the token is on a node in V_i then player i chooses a successor node. An infinite sequence of nodes created in this way is a ply and it is won by player i iff the highest priority seen infinitely often in this sequence is i modulo 2. A winning strategy is as usual a strategy for a player that lets them win every play regardless of the opponent's choices. We write $|G|$ for the number of nodes in the game G, and $idx(G)$ for its index, i.e. number of distinct priorities.

Proposition 12. *Let ϑ be a formula with $n := |\vartheta|$ and $k := ad(\vartheta)$. There is a parity game G_ϑ with $|G| \leq 2^{\mathcal{O}(n \cdot k \cdot \log n)}$ and $idx(G) \leq \mathcal{O}(n \cdot k)$, that is won by player 0 iff ϑ is satisfiable.*

Proof. Let $\mathcal{A}_\vartheta = (Q, \Sigma_\vartheta, q_0, \delta, \Omega)$. The nodes of the game G_ϑ are of the form $2^{Cl(\vartheta)} \times Q$; the designated node v_0 is $(\{\vartheta\}, q_0)$. A node $w = (\Psi, q')$ is a successor of $v = (\Phi, q)$ if a uniquely (for (Φ, q)) chosen rule is applied to Φ that yields Ψ as one of its premisses, this rule is represented by $r \in \Sigma_\vartheta$ and $\delta(q, r) = q'$ where δ is the transition function of $\mathcal{A}_\vartheta$. The node ownership in the game is determined by these uniquely chosen rules: player 0 owns nodes in which rule (Or) is applied, while player 1 owns all the other nodes. Finally, the priority of a game node (Φ, q) is simply $\Omega(q)$.

It is not hard to see that winning strategies for player 0 exactly correspond to tableaux for ϑ. Hence, with Thm. 7, player 0 wins node v_0 iff ϑ is satisfiable. □

	Aut	Tab	Game^G	Game
unguardedness welcome	yes	no	no	yes
worst-case runtime	$2^{\mathcal{O}(n^2m^2\log n)}$	2NExpTime	$2^{2^{\mathcal{O}(n)}}$	$2^{\mathcal{O}(n^2k^2\log n)}$
small model property	$2^{\mathcal{O}(nm\log n)}$	$2^{2^{\mathcal{O}(n)}}$	$2^{2^{\mathcal{O}(n)}}$	$2^{\mathcal{O}(nk\log n)}$
branching-degree	n	$2^{\mathcal{O}(n)}$	$2^{\mathcal{O}(n)}$	n
implemented	no	no	yes	yes

Figure 3. Comparison of different decision methods on formulas of size n and alternation depth k and number of least fixpoint variables m.

Proposition 13. *Satisfiability of a $\mathcal{L}_\mu$ formula ϑ with $n := |\vartheta|$ and $k := ad(\vartheta)$ can be decided in time $2^{\mathcal{O}(n^2\cdot k^2\cdot\log n)}$.*

Proof. Follows immediately from Prop. 12 with the fact that the asymptotically best known algorithms for solving parity games run in time $m^{\mathcal{O}(p)}$ where m is the number of nodes and p is the number of priorities in the game (Schewe, 2007). $\square$

The subgame induced by a winning strategy for player 0 is in effect a model for ϑ. This immediately yields a small model property for $\mathcal{L}_\mu$.

Proposition 14. *Let $\vartheta \in \mathcal{L}_\mu$ with $n := |\vartheta|$ and $k := ad(\vartheta)$. If ϑ is satisfiable then it has a model of size $2^{\mathcal{O}(nk\cdot\log n)}$ and branching-degree at most n.*

5.3 *Comparison*

We compare the presented method (Game) to existing methods, namely the automata-theoretic one by Emerson et al. (Aut) (Streett & Emerson, 1989; Emerson & Jutla, 2000) and the purely tableau-based one by Jungteerapanich (Tab) (Jungteerapanich, 2009). Additionally we consider the method which works as described above but uses pre-transformation into guarded form and rule (FP$_\nu^G$) instead (Game^G). The input formula is parameterized by its size n, its alternation-depth k, and the number of distinct μ-bound variables in it m. Note that we always have $k \leq m < n$.

The reasoning behind the run-time and small model property of method Aut are as follows. A formula ϑ of size n with m μ-bound variables can be translated into a Streett automaton of size $2^{\mathcal{O}(n\cdot m\cdot\log n)}$ and $\mathcal{O}(n\cdot m)$ acceptance pairs (Streett & Emerson, 1989).[1] Emptiness of a Streett tree automaton with s states and p pairs can be decided in time $(s\cdot p)^{\mathcal{O}(p)}$ (Emerson & Jutla, 2000), hence the worst-case runtime of $2^{\mathcal{O}(n^2\cdot m^2\log n)}$ observing that $m < n$. This uses an equivalence-preserving reduction from Streett automata of that size to Rabin automata of size s^2 with p pairs (Emerson & Jutla, 2000). Every Rabin tree automaton with e edges and p pairs accepts a tree that is finitely representable with $\mathcal{O}(e)$ nodes (Emerson, 1985). In this case, we have $e = \mathcal{O}(n\cdot m\cdot s^2)$ because a transition to a tuple of size j counts as j edges, and the branching-degrees of these automata are linear in the size of the original formula. Putting this all together, we obtain a small model property of $2^{\mathcal{O}(n\cdot m\cdot\log n)}$.

It is worth mentioning that conceptually, the method presented here is very close to the purely automata-theoretic method Aut. However, separating the local and global consistency checks into pre-tableau rules and automata-theoretic machinery for the thread structure in tableaux yields a cleaner presentation of the method's ingredients. The

[1]Emerson et al. claim that the global automaton used in their construction is of linear size n, but its description shows that it really is of size $n\cdot m$.

slight asymptotic speed-up using the exponent k instead of m where $k \leq m$ is owed to using the more modern concept of parity automata rather than Streett automata.

There is also a notable difference between the method presented here and the automata-based Aut. Both can deal with unguarded formulas in their input. However, here we presented a method that handles unguarded formulas correctly whereas the work behind Aut handles unguardedness with a definition on what an acceptable Hintikka set is. It does not show how such Hintikka sets can be detected algorithmically. A naïve approach boils down to examining all cycles in a directed graph for the occurrence of some edge on this cycle.

The main advantage of Tab is the fact that tableaux in that calculus are finite as opposed to the infinite ones used here. The price to pay for this seems to be the non-optimal complexity bound. It is not clear whether there is also a deterministic algorithm for that calculus and whether it can be made to work on unguarded formulas as well thus losing one exponential in worst-case runtime and small model property.

Finally, we remark that the reduction to parity game seems to be the only one that has been implemented in a tool (Friedmann & Lange, 2010).

6. Experimental Results

In this section we describe hand-crafted benchmarks given by families of unguarded formulas in order to evaluate the differences between the decision method that transforms the formulas into guarded form first (Game^G), and the method that operates directly on arbitrary formulas (Game).

All tests have been carried out on a 64-bit machine with four quad-core Opteron$^{\text{TM}}$ CPUs and 128GB RAM space, using MLSolver (Friedmann & Lange, 2010). The implementation does not (yet) support parallel computations, hence, each test runs on one core only and needed less than 4 GB RAM.

We only present instances of non-negligible running times. On the other hand, the solving of larger instances not presented in the figures anymore has experienced time-outs after one hour, marked †.

Benchmark 1. Consider the following family of formulas φ_n.

$$\varphi_n := \nu X_1.\mu X_2.\ldots\sigma X_n.\left(\bigvee_{i=1}^{n} X_i \vee \langle a\rangle \bigwedge_{i=1}^{n} X_i\right)$$

Each of these formulas is equivalent to $\mathtt{tt}$, hence, trivially satisfiable. Nevertheless, they are chosen because they feature a high degree of unguardedness.

The first table in Fig. 4 presents the experimental data collected from checking each φ_n for satisfiability. The first column states the formulas' indices, the second block of columns shows the formulas' sizes. By φ_n^G we denote the guarded formula that is equivalent to φ_n according to Prop. 1. Note that the transformation into guarded form does increase the formula non-negligibly, and this increase results in a significant loss of performance of the subsequent satisfiability test, as shown by the next two blocks of columns.

The columns labeled $|\mathcal{B}_\varphi|$, $|\mathcal{A}_\varphi|$ and $|G_\varphi|$ show the sizes of the nondeterministic thread-finding automaton $\mathcal{B}_\varphi$ according to Lemma 9, the complemented deterministic automaton $\mathcal{A}_\varphi$ according to Cor. 11, and the resulting parity game G_φ according to Prop. 12 on the input formula φ respectively.

	formula		Game^G				Game			
n	$\|\varphi_n\|$	$\|\varphi_n^G\|$	$\|\mathcal{B}_{\varphi_n^G}\|$	$\|\mathcal{A}_{\varphi_n^G}\|$	$\|G_{\varphi_n^G}\|$	time	$\|\mathcal{B}_{\varphi_n}\|$	$\|\mathcal{A}_{\varphi_n}\|$	$\|G_{\varphi_n}\|$	time
1	6	6	0	1	4	0.00s	0	1	4	0.00s
2	12	19	8	58	109	0.00s	7	62	63	0.00s
3	18	105	43	816	1,917	0.02s	10	96	102	0.00s
4	24	2,177	2,748	387,914	775,815	32.18s	25	524	543	0.01s
5	30	-	-	-	-	†	31	781	800	0.00s

	formula		Game^G				Game			
n	$\|\psi_n\|$	$\|\psi_n^G\|$	$\|\mathcal{B}_{\psi_n^G}\|$	$\|\mathcal{A}_{\psi_n^G}\|$	$\|G_{\psi_n^G}\|$	time	$\|\mathcal{B}_{\psi_n}\|$	$\|\mathcal{A}_{\psi_n}\|$	$\|G_{\psi_n}\|$	time
1	32	40	15	88	147	0.00s	13	73	269	0.00s
2	48	124	70	862	5,519	0.04s	25	237	2,045	0.02s
3	64	496	435	18,393	1,187,885	28.73s	41	621	12,394	0.15s
4	80	-	-	-	-	†	61	1,489	70,014	1.39s

Figure 4. Deciding unguarded formulas in practice.

Benchmark 2. Consider the following families of regular expressions α_n and β_n.

$$\begin{aligned} \alpha_0 &= a^* & \beta_0 &= b^* \\ \alpha_{n+1} &= (\alpha_n \cup b)^* & \beta_{n+1} &= (a \cup \beta_n)^* \end{aligned}$$

For $n \geq 1$ we have $L(\alpha_n) = L(\beta_n)$ because both describe the language Σ^* in a cumbersome way. Thus, the PDL formula $\langle\alpha_n\rangle q \to \langle\beta_n\rangle q$ is valid; equivalently $\langle\alpha_n\rangle q \wedge \neg\langle\beta_n\rangle q$ is unsatisfiable. Translating this family of PDL formulas into the modal μ-calculus yields the family $\psi_n = \psi'_n(q) \wedge \psi''_n(\neg q)$ where

$$\psi'_n(Z) = \begin{cases} \mu X_n.Z \vee \langle b\rangle X_n \vee \psi'_{n-1}(X_n) & \text{if } n > 0 \\ \mu X_0.Z \vee \langle a\rangle X_0 & \text{otherwise} \end{cases}$$

$$\psi''_n(Z) = \begin{cases} \nu Y_n.Z \wedge [a]Y_n \wedge \psi''_{n-1}(Y_n) & \text{if } n > 0 \\ \nu Y_0.Z \wedge [b]Y_0 & \text{otherwise} \end{cases}$$

Note that each of these is alternation-free but, again, features a high degree of unguardedness.

The second table in Fig. 4 presents experimental data collected from checking ψ_n, $n \geq 1$ for unsatisfiability. As before, the table presents the original formula size and the size of an equivalent guarded formula as well as the sizes of the automata and parity games involved in the decision procedure. Again, it is evident that guarded transformation is harmful in this case since the checking of the unguarded original formula scales much better.

7. Conclusion

We have presented a tableau calculus which is sound and complete for the entire modal μ-calculus, not just guarded formulas. The experiments presented in the previous section suggest that deciding unguarded formulas directly is preferable over using an explicit transformation into equivalent guarded formulas. The deal breaker is the unsettled question of whether or not there is an efficient way of performing such a transformation. So far, the best known transformations are exponential in the size of the formula, and it is reasonably unlikely to find a transformation that is polynomial in that size (Bruse et al., 2013).

A closely related issue is what is known as the nested-star problem in PDL. Some decision procedures for PDL – a fragment of the alternation-free modal μ-calculus – have been found to not handle formulas which contain programs with nested stars correctly, for instance the one implemented in the tool `pdl-tableau`[1]. The rule that handles unfoldings of greatest fixpoint formulas for unguarded μ-calculus formulas can be specialised to handle nested stars in decision procedures for PDL satisfiability correctly.

$$\frac{[\alpha^*]\varphi, \Phi}{\Phi \qquad \varphi, [\alpha][\alpha^*]\varphi, \Phi}$$

On the other hand, there is a simpler way of dealing with unguardedness in PDL. One just has to require that for every subprogram of the form α^* occurring in the input, the language of α does not contain the empty word. It is a simple exercise to transform all programs at a linear blow-up in order to meet this requirement. Then, stars can be nested and the effects that cause such decision procedures to fail will not occur anymore.

An interesting question is the following. Is there a polynomial or even just linear guarded transformation into equi-satisfiable formulas? Such a transformation would not necessarily yield equivalent guarded formulas but it would be enough to be used as a pre-processing step in decision procedures for satisfiability or validity of the modal μ-calculus.

Acknowledgment. The European Research Council has provided financial support under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 259267.

References

Banieqbal, B., & Barringer, H. (1989). Temporal logic with fixed points. In *Proc. coll. on temporal logic in specification* (Vol. 398, pp. 62–73). Springer.

Berwanger, D. (2003). Game logic is strong enough for parity games. *Studia Logica*, *75*(2), 205–219.

Bradfield, J., & Stirling, C. (2001). Modal logics and μ-calculi: an introduction. In J. Bergstra, A. Ponse, & S. Smolka (Eds.), *Handbook of Process Algebra.* Elsevier.

Bruse, F., Friedmann, O., & Lange, M. (2013). Guarded transformation for the modal μ-calculus. *CoRR*, *abs/1305.0648*.

Dam, M. (1992). Fixpoints of Büchi automata. In R. Shyamasundar (Ed.), *Proc. 12th conf. on foundations of software technology and theoretical computer science, FST&TCS'92* (Vol. 652, pp. 39–50). Berlin: Springer.

Dam, M. (1994). CTL* and ECTL* as fragments of the modal μ-calculus. *TCS*, *126*(1), 77–96.

Emerson, E. A. (1985). Automata, tableaux and temporal logics. In R. Parikh (Ed.), *Proc. conf. on logic of programs* (Vol. 193, pp. 79–87). Brooklyn, NY: Springer.

Emerson, E. A. (1990). Temporal and modal logic. In J. van Leeuwen (Ed.), *Handbook of theoretical computer science* (Vol. B: Formal Models and Semantics, pp. 996–1072). New York, USA: Elsevier and MIT Press.

[1]`http://www.cs.man.ac.uk/~schmidt/pdl-tableau/`

Emerson, E. A., & Jutla, C. S. (1991). Tree automata, μ-calculus and determinacy. In *Proc. 32nd symp. on foundations of computer science* (pp. 368–377). San Juan, Puerto Rico: IEEE.

Emerson, E. A., & Jutla, C. S. (2000). The complexity of tree automata and logics of programs. *SIAM Journal on Computing*, *29*(1), 132–158.

Emerson, E. A., & Lei, C. L. (1986). Efficient model checking in fragments of the propositional μ–calculus. In *Symposion on logic in computer science* (pp. 267–278). Washington, D.C., USA: IEEE.

Fischer, M. J., & Ladner, R. E. (1979). Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, *18*(2), 194–211.

Friedmann, O., & Lange, M. (2010). A solver for modal fixpoint logics. In *Proc. 6th workshop on methods for modalities, M4M-6* (Vol. 262, pp. 99–111).

Janin, D., & Walukiewicz, I. (1996). On the expressive completeness of the propositional μ-calculus with respect to monadic second order logic. In *Proc. 7th conf. on concurrency theory, CONCUR'96* (Vol. 1119, pp. 263–277). Springer.

Jungteerapanich, N. (2009). A tableau system for the modal μ-calculus. In *Proc. 18th int. conf. on automated reasoning with analytic tableaux and related methods, TABLEAUX'09* (Vol. 5607, pp. 220–234). Springer.

Kaivola, R. (1997). *Using automata to characterise fixed point temporal logics.* Unpublished doctoral dissertation, LFCS, Division of Informatics, The University of Edinburgh. (Tech. Rep. ECS-LFCS-97-356)

Kozen, D. (1983). Results on the propositional μ-calculus. *TCS*, *27*, 333–354.

Kozen, D., & Parikh, R. (1983). A decision procedure for the propositional μ-calculus. In *Proc. workshop on logics of programs* (Vol. 164, pp. 313–325). Springer.

Kupferman, O., Vardi, M. Y., & Wolper, P. (2000). An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, *47*(2), 312–360.

Lange, M. (2007). Linear time logics around PSL: Complexity, expressiveness, and a little bit of succinctness. In *Proc. 18th int. conf. on concurrency theory, CONCUR'07* (Vol. 4703, pp. 90–104). Springer.

Mateescu, R. (2002). Local model-checking of modal mu-calculus on acyclic labeled transition systems. In *Proc. 8th int. conf. on tools and algorithms for the construction and analysis of systems, TACAS'02* (Vol. 2280, pp. 281–295). Springer.

McNaughton, R. (1993). Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, *65*(2), 149–184.

Piterman, N. (2006). From nondeterministic Büchi and Streett automata to deterministic parity automata. In *Proc. 21st symp. on logic in computer science, LICS'06* (pp. 255–264). IEEE Computer Society.

Rabin, M. O. (1969). Decidability of second-order theories and automata on infinite trees. *Trans. of Amer. Math. Soc.*, *141*, 1–35.

Schewe, S. (2007). Solving parity games in big steps. In *Proc. 27th int. conf. on foundations of software technology and theoretical computer science, FSTTCS'07* (Vol. 4855, pp. 449–460). Springer.

Streett, R. S., & Emerson, E. A. (1984). The propositional μ-calculus is elementary. In J. Paredaens (Ed.), *Proc. 11th coll. on automata, languages, and programming, ICALP'84* (Vol. 172, pp. 465–472). Berlin: Springer.

Streett, R. S., & Emerson, E. A. (1989). An automata theoretic decision procedure for the propositional μ-calculus. *Information and Computation*, *81*(3), 249–264.

Walukiewicz, I. (1996). Monadic second order logic on tree-like structures. In *Proc. 13th annual symp. on theoretical aspects of computer science, STACS'96* (Vol. 1046, pp. 401–413). Grenoble, France: Springer.

Walukiewicz, I. (2000). Completeness of Kozen's axiomatisation of the propositional μ-calculus. *Inf. and Comput.*, *157*(1–2), 142–182.