Alternating Tree Automata, Parity Games, and Modal μ-Calculus

Thomas Wilke*

Christian-Albrechts-Universität zu Kiel
Institut für Informatik und Praktische Mathematik
phone: +49 431 880-7511, fax: +49 431 880-7614
email: wilke@ti.informatik.uni-kiel.de
RCS id string: master.tex,v 8.1 2000/11/20 09:18:13 wilke Exp

Contents

*This paper resulted from an invited talk given at the Journées Montoises, Marne-la-Vallée, March 2000.

1 Introduction

Since Dexter Kozen's seminal paper in 1983, [10], modal μ-calculus has received ever growing interest, mainly for two reasons: its mathematical theory (model theory) is very rich; it is well suited for specifying properties of transition systems. The interest was further stimulated by Ken McMillan's observation, [13], that in the case of finite transition systems specifications in modal μ-calculus can be checked efficiently when the state space and the transition relation are represented "symbolically" (by ordered binary decision diagrams). This is the basis for numerous industrial-strength model checkers.

Given that modal μ-calculus is exactly as expressive as monadic second-order logic (over trees), [17, 5], it should come as no surprise that it is best investigated with an automata-theoretic approach. In fact, it is even more amenable to an automata-theoretic treatment than monadic second-order logic, for the connection between modal μ-calculus and the appropriate automaton model—alternating tree automata—is much tighter.

In this paper I give a coherent exposition of the connection of alternating tree automata and modal μ-calculus, advocating an automaton model specifically tailored for working with modal μ-calculus and slightly modified compared to other models used in the literature.

The main focus is on the model-checking and the satisfiability problem for modal μ-calculus. The approach taken is modular: both problems are first translated into problems for alternating tree automata (the "word" problem and the nonemptiness problem), then these automata problems are solved independently, by reductions to appropriate problems for parity games. The reduction of the

emptiness problem involves a new form of parity games, so-called parity games with extended winning condition.

It will be shown how the winner problem for parity games with extended winning conditions can be reduced to the winner problem for ordinary parity games. This reduction is the place where Safra's fundamental determinization result, [18], comes into the picture.

The full beauty of the connection between modal μ-calculus and (alternating) tree automata was first revealed in the fundamental study [15] by Damian Niwiński: there is a correspondence between modal μ-calculus formulas and alternating tree automata that respects the alternation depth respectively the Rabin index. (This result is also the basis for André Arnold's elegant proof of Bradfield's and Lenzi's alternation hierarchy result, [1, 2, 3, 11].) While in [15], a very general approach, applying to a wide range of lattices, is taken, here the definitions are designed to work particularly well with modal μ-calculus.

Ordinals. Von Neumann's convention on ordinals is used. In particular, when n denotes a natural number, then $n = \{0, \ldots, n-1\}$. Also, ω denotes the set of natural numbers.

Sequences. A *finite sequence* over some set M is a function $n \to M$ where n is a natural number; an *infinite sequence* over some set M is a function $\omega \to M$. *Sequence* means finite or infinite sequence. The *length* of a sequence u is denoted by $|u|$. (Observe that $|u|$ coincides with $\mathrm{dom}(u)$, for von Neumann's convention is used.) When u is a finite nonempty sequence, then $u(|u|-1)$ denotes its last element; for simplicity in notation, we will also write $u(*)$ instead.

Let $f : M \to N$ be any partial function. The set of elements from N occurring infinitely often in the range of f is denoted by $\mathrm{inf}(u)$, that is, $\mathrm{inf}(f) = \{n \in N \mid |f^{-1}(n)| = \infty\}$.

Graphs. In this paper, the term graph means directed graph. That is, a *graph* is a pair (V, E) where V is an arbitrary set of vertices and E is an arbitrary subset of $V \times V$. The successors of a vertex v in a graph $\boldsymbol{G}$ will be denoted by $\mathrm{Scs}_{\boldsymbol{G}}(v)$. So if $\boldsymbol{G} = (V, E)$, then $\mathrm{Scs}_{\boldsymbol{G}}(v) = \{v' \in V \mid (v, v') \in E\}$.

A vertex v is a *dead end* of a graph $\boldsymbol{G} = (V, E)$ if $\mathrm{Scs}_{\boldsymbol{G}}(v) = \emptyset$. A *path* through a graph $\boldsymbol{G}$ is a sequence π over V satisfying $(\pi(i), \pi(i+1)) \in E$ for every i with $i + 1 < |\pi|$.

Let M be an arbitrary set. An *M-vertex-labeled graph* is a tuple (V, E, λ) where (V, E) is an ordinary graph and $\lambda V \to M$ is a so-called labeling function.

Structures. As usual, mathematical objects like graphs, trees, automata, etc. will be defined as fixed length tuples with certain components, just as a graph is a pair (V, E). To refer to the individual components of a structure denoted $\boldsymbol{S}$, the superscript $^{\boldsymbol{S}}$ is used. For instance, the vertex set of a graph $\boldsymbol{G}$ is denoted by $V^{\boldsymbol{G}}$.

Trees. In this paper, the term tree means directed tree. A *branch* of a tree $\boldsymbol{T}$ is a maximum path through $\boldsymbol{T}$ starting in the root. So a branch is either a finite path starting in the root and ending in a dead end or an infinite path starting in the root.

We will use the following operations on trees. Assume $\boldsymbol{T}$ is an arbitrary tree. Let V' be a subset of $V^{\boldsymbol{T}}$ not containing the root of $\boldsymbol{T}$. Then $\boldsymbol{T} - V'$ is the tree obtained from $\boldsymbol{T}$ by removing all vertices in V' from $\boldsymbol{T}$ and all their descendants. Let v be some vertex in $\boldsymbol{T}$. Then $\boldsymbol{T}{\downarrow}v$ is the subtree of $\boldsymbol{T}$ rooted at v. Let U be a set of pairs $(v, \boldsymbol{T'})$ where v is a vertex of $\boldsymbol{T}$ and $\boldsymbol{T}'$ a tree. Then $\boldsymbol{T} \cdot U$ is the tree that is obtained from $\boldsymbol{T}$ by adding, for every $(v, \boldsymbol{T'}) \in U$, a copy of $\boldsymbol{T'}$ to $\boldsymbol{T}$ at v in such a way that an edge from v to the root of $\boldsymbol{T}'$ is inserted. If necessary, the vertices of the trees $\boldsymbol{T'}$ are renamed.

We will use the following lemma about trees.

Lemma 1 *Let $\boldsymbol{T} = (V, E)$ be a tree and $U \subset V$. For every ordinal λ, define U_λ as follows.*

- *$U_0 = \emptyset$.*
- *A vertex v belongs to $U_{\lambda+1}$ if $v \in U$ and all descendants of v (excluding v) belong to $\bigcup_{\lambda' \leq \lambda} U_{\lambda'} \cup (V \setminus U)$.*
- *$U_\lambda = \bigcup_{\lambda' < \lambda} U_{\lambda'}$ for every limit ordinal λ.*

Then the following are equivalent for every $v \in U$.

- *There exists some ordinal λ such that $v \in U_\lambda$.*
- *Every branch of $\boldsymbol{T}{\downarrow}v$ contains only a finite number of elements from U.*

Projections. When $t = (t_0, \ldots, t_{n-1})$ is a tuple and $i < n$, then $pr_i(t)$ denotes the i-th component of t, that is, $pr_i(t) = t_i$.

2 From Modal μ-Calculus to Alternating Tree Automata

Our first goal is to prove that each modal μ-calculus formula can be converted into an equivalent alternating tree automaton. In the first subsection, the basics on modal μ-calculus are recalled and basic notation is explained. In the second subsection, the model of alternating tree automaton used in this paper is introduced.

The third subsection presents the desired conversion from modal μ-calculus to alternating tree automata.

2.1 Modal μ-Calculus

Modal μ-calculus is modal logic augmented by operators for least and greatest fixed points. For simplicity, this paper only deals with the unimodal case, but all the definitions, results, and proofs given here extend canonically to the multimodal case. An extension to backward modalities as treated in [21] is not difficult either.

2.1.1 Kripke Structures

Modal μ-calculus—just as modal logic—is a logic to express properties of Kripke structures.

A Kripke structure is a directed graph together with an interpretation (or assignment) of propositional variables in each vertex of the graph. Once and for all, we fix a countably infinite supply Q of propositional variables. Formally, a *Kripke structure* is a tuple

$$\boldsymbol{K} = (W^{\boldsymbol{K}}, A^{\boldsymbol{K}}, \kappa^{\boldsymbol{K}}) \tag{1}$$

where

- $W^{\boldsymbol{K}}$, the *universe* of $\boldsymbol{K}$, is a set of *worlds*,
- $A^{\boldsymbol{K}} \subseteq W^{\boldsymbol{K}} \times W^{\boldsymbol{K}}$ is an *accessibility relation*, and
- $\kappa^{\boldsymbol{K}} \colon Q \to 2^{W^{\boldsymbol{K}}}$ is an *interpretation* of the propositional variables, which assigns to each propositional variable the set of worlds where it holds true.

The class of all Kripke structures is denoted by $\mathfrak{K}$.

A *pointed Kripke structure* is a pair $(\boldsymbol{K}, w)$ where $\boldsymbol{K}$ is a Kripke structure and w a world of it, which is called the *distinguished world* of $(\boldsymbol{K}, w)$. The class of all pointed Kripke structures is denoted by $\mathfrak{P}$.

A Kripke structure or pointed Kripke structure is *finite* if $W^{\boldsymbol{K}}$ is finite and $\kappa^{\boldsymbol{K}}(q) = \emptyset$ for almost all q. Such a Kripke structure can easily be encoded as a finite string and can thus serve as input for decision procedures. This will be important in the next section.

A *Kripke query*[1] is a class of pointed Kripke structures, that is, a subclass of $\mathfrak{P}$. There is a natural one-to-one correspondence between Kripke queries and mappings assigning to each Kripke structure $\boldsymbol{K}$ a subset of $W^{\boldsymbol{K}}$, as explained in the next paragraph.

[1] The term "query" is (should be) reminiscent of the terminology used in finite model theory.

With every Kripke query $\mathfrak{Q}$, one associates the mapping

$$\boldsymbol{K} \mapsto \{w \in W^{\boldsymbol{K}} \mid (\boldsymbol{K}, w) \in \mathfrak{Q}\} \ ; \tag{2}$$

conversely, with every mapping $\mathfrak{Q}\colon \boldsymbol{K} \mapsto \mathfrak{Q}(\boldsymbol{K})$ where $\mathfrak{Q}(\boldsymbol{K}) \subseteq W^{\boldsymbol{K}}$, one associates the Kripke query

$$\{(\boldsymbol{K}, w) \in \mathfrak{P} \mid w \in \mathfrak{Q}(\boldsymbol{K})\} \ . \tag{3}$$

For notational convenience, we will not make a distinction between a Kripke query (a subclasses of $\mathfrak{P}$) and its "mapping view" as explained above (in (2)). In particular, a Kripke query may be defined as a mapping which assigns to every Kripke structure $\boldsymbol{K}$ a subset of $W^{\boldsymbol{K}}$. Also, when $\mathfrak{Q}$ denotes a Kripke query, we may write $\mathfrak{Q}(\boldsymbol{K})$ for $\{w \in W^{\boldsymbol{K}} \mid (\boldsymbol{K}, w) \in \mathfrak{Q}\}$.

2.1.2 Syntax

As stated above, modal μ-calculus is a unimodal logic augmented by least and greatest fixed points operators.

The formulas of modal μ-calculus are built from the constant symbols $\bot$ and $\top$, the symbols from Q and their negations, using disjunction and conjunction, the modalities $\Box$ and $\Diamond$, and operators for least and greatest fixed points, μ and ν, with some minor restriction on the use of the fixed point operators.[2] Formally, the set of all L_μ formulas is defined inductively as follows.

- The symbols $\bot$ and $\top$ are L_μ formulas.
- For every $q \in Q$, q and $\neg q$ are L_μ formulas.
- If φ and ψ are L_μ formulas, then $\varphi \vee \psi$ and $\varphi \wedge \psi$ are L_μ formulas.
- If φ is an L_μ formula, then $\Box\varphi$ and $\Diamond\varphi$ are L_μ formulas.
- If $q \in Q$ and φ is an L_μ formula where q occurs only positive, then $\mu q\varphi$ and $\nu q\varphi$ are L_μ formulas.

The restriction on the application of the least and greatest fixed point operator expressed in the last rule above is imposed to justify the terminology: this restriction ensures that the argument of a fixed point operator can be viewed as a monotone function and that a fixed point actually exists (for details, see below).

Fixed point operators are viewed as quantifiers, and the standard terminology and notation used with quantifiers is adopted. For instance, the set of all propositional variables occurring free in an L_μ formula φ is denoted by $\mathrm{free}(\varphi)$.

A *fixed point formula* is an L_μ formula where a fixed point operator is the outermost connective, that is, a fixed point formula is an L_μ formula of the form $\eta q\psi$ where $\eta = \mu$ or $\eta = \nu$. The set of all fixed point formulas is denoted F_η. This

[2] In this paper, no distinction is made between symbols for propositional constants and propositional variables.

set is partitioned into two sets according to the type of the outermost operator: the set of all fixed point formulas starting with μ is denoted F_μ, while the set of all fixed point formulas starting with ν is denoted F_ν. We also speak of μ- and ν-formulas, respectively.

2.1.3 Substitution

Assume $\varphi, \psi_0, \ldots, \psi_{l-1}$ are L_μ formulas and $q_0, \ldots, q_{l-1}$ are distinct predicate symbols whose free occurrences in φ are positive. Then

$$\varphi[\psi_0/q_0, \ldots, \psi_{l-1}/q_{l-1}] \tag{4}$$

denotes the L_μ formula that is obtained from φ by substituting in parallel each free occurrence of q_i by ψ_i. As with predicate logic, a renaming of the bound variables in φ takes place. Note that it is necessary to require that the free occurrences of the q_i's in φ are positive because otherwise the resulting syntactic object will not be an L_μ formula.

2.1.4 Semantics

The formulas of modal μ-calculus are interpreted in Kripke structures. Inductively, it is defined to which set of worlds an arbitrary L_μ formula is evaluated in a given Kripke structure. More precisely, for every Kripke structure $\boldsymbol{K}$ and every L_μ formula φ a set $||\varphi||_{\boldsymbol{K}} \subseteq W^{\boldsymbol{K}}$ is defined.

The semantics of the atomic formulas is determined by

$$||\bot||_{\boldsymbol{K}} = \emptyset \ , \qquad ||\top||_{\boldsymbol{K}} = W^{\boldsymbol{K}} \ , \tag{5}$$

$$||q||_{\boldsymbol{K}} = \kappa^{\boldsymbol{K}}(q) \ , \qquad ||\neg q||_{\boldsymbol{K}} = W^{\boldsymbol{K}} \setminus \kappa^{\boldsymbol{K}}(q) \ . \tag{6}$$

Disjunction and conjunction are interpreted as union and intersection, respectively,

$$||\varphi_0 \vee \varphi_1||_{\boldsymbol{K}} = ||\varphi_0||_{\boldsymbol{K}} \cup ||\varphi_1||_{\boldsymbol{K}} \ , \tag{7}$$

$$||\varphi_0 \wedge \varphi_1||_{\boldsymbol{K}} = ||\varphi_0||_{\boldsymbol{K}} \cap ||\varphi_1||_{\boldsymbol{K}} \ . \tag{8}$$

The modal operators are interpreted in the usual way:

$$||\Box\varphi||_{\boldsymbol{K}} = \{w \in W^{\boldsymbol{K}} \mid \mathrm{Scs}_{\boldsymbol{K}}(w) \subseteq ||\varphi||_{\boldsymbol{K}}\} \ , \tag{9}$$

$$||\Diamond\varphi||_{\boldsymbol{K}} = \{w \in W^{\boldsymbol{K}} \mid \mathrm{Scs}_{\boldsymbol{K}}(w) \cap ||\varphi||_{\boldsymbol{K}} \neq \emptyset\} \ . \tag{10}$$

Finally, we have to explain the semantics of the fixed point operators. This needs a little preparation. When $\boldsymbol{K}$ is a Kripke structure, q is a propositional variable, and $W \subseteq W^{\boldsymbol{K}}$, then $\boldsymbol{K}[q \mapsto W]$ denotes the Kripke structure defined by

$$\boldsymbol{K}[q \mapsto W] = (W^{\boldsymbol{K}}, A^{\boldsymbol{K}}, \kappa^{\boldsymbol{K}}[q \mapsto W])$$

where $\kappa^{\boldsymbol{K}}[q \mapsto W]$ is given by

$$\kappa^{\boldsymbol{K}}[q \mapsto W](q') = \begin{cases} W & \text{if } q' = q, \\ \kappa^{\boldsymbol{K}}(q') & \text{if } q' \neq q, \end{cases} \tag{11}$$

that is, $\kappa^{\boldsymbol{K}}[q \mapsto W]$ is identical to $\kappa^{\boldsymbol{K}}$ except at q where its value is W.

The semantics of the fixed point operators is now defined by

$$||\mu q \varphi||_{\boldsymbol{K}} = \bigcap \{W \subseteq W^{\boldsymbol{K}} \mid ||\varphi||_{\boldsymbol{K}[q \mapsto W]} \subseteq W\} \ , \tag{12}$$

$$||\nu q \varphi||_{\boldsymbol{K}} = \bigcup \{W \subseteq W^{\boldsymbol{K}} \mid ||\varphi||_{\boldsymbol{K}[q \mapsto W]} \supseteq W\} \ . \tag{13}$$

Let φ be an arbitrary L_μ formula and $\boldsymbol{K}$ a Kripke structure. Then it is easy to see that $W \mapsto ||\varphi||_{\boldsymbol{K}[q \mapsto W]}$ is a monotone function on $2^{W^{\boldsymbol{K}}}$. Therefore, this function has a least and a greatest fixed point (with respect to set inclusion). By the Knaster/Tarski Theorem these fixed points are identical with the sets denoted by the right-hand sides of (12) and (13), respectively.

The Knaster/Tarski Theorem also gives the following characterization. Let f be the above function on $2^{W^{\boldsymbol{K}}}$. For every ordinal λ, let $W_\lambda \subseteq W^{\boldsymbol{K}}$ be defined by

- $W_0 = \emptyset$,
- $W_{\lambda+1} = f(W_\lambda)$,
- $W_\lambda = \bigcup_{\lambda' < \lambda} W_{\lambda'}$ for every limit ordinal.

The set W_λ is called the λ-*approximant* of the least fixed point of f. Then $\{W_\lambda\}_\lambda$ is a monotone sequence that at some point becomes stationary and reaches the least fixed point of f. A symmetric statement holds for the greatest fixed point of f.

Given a pointed Kripke structure K, we will write $(\boldsymbol{K}, w) \models \varphi$ for $w \in ||\varphi||_{\boldsymbol{K}}$.

2.1.5 Query-Based Semantics

It will be useful to have a different view of the semantics of L_μ. Clearly, when φ is an L_μ formula, then $\boldsymbol{K} \mapsto ||\varphi||_{\boldsymbol{K}}$ is a Kripke query, which we denote by $||\varphi||$. It will be convenient to have operators on Kripke queries at hand which correspond to the connectives and operators of L_μ.

Disjunction and conjunction are easy to deal with. Clearly,

$$||\varphi \vee \psi|| = ||\varphi|| \cup ||\psi|| \ , \tag{14}$$

$$||\varphi \wedge \psi|| = ||\varphi|| \cap ||\psi|| \ , \tag{15}$$

for arbitrary L_μ formulas φ and ψ.

For the modalities $\Diamond$ and $\Box$ we define two new operators on Kripke queries. For every Kripke query $\mathfrak{Q}$, we define

$$\Diamond\mathfrak{Q}\colon \boldsymbol{K} \mapsto \{w \in W^{\boldsymbol{K}} \mid \mathrm{Scs}_{\boldsymbol{K}}(w) \cap \mathfrak{Q}(\boldsymbol{K}) \neq \emptyset\} \ , \tag{16}$$

$$\Box\mathfrak{Q}\colon \boldsymbol{K} \mapsto \{w \in W^{\boldsymbol{K}} \mid \mathrm{Scs}_{\boldsymbol{K}}(w) \subseteq \mathfrak{Q}(\boldsymbol{K})\} \ . \tag{17}$$

Then,

$$||\Diamond\varphi|| = \Diamond||\varphi|| \ , \qquad ||\Box\varphi|| = \Box||\varphi|| \ , \tag{18}$$

for every L_μ formula φ.

Similarly, we define operators on Kripke queries corresponding to the two fixed point operators. For every Kripke query $\mathfrak{Q}$ and propositional variable q we define Kripke queries $\mu q\mathfrak{Q}$ and $\nu q\mathfrak{Q}$ by

$$\mu q\mathfrak{Q}\colon \boldsymbol{K} \mapsto \bigcap\{W \subseteq W^{\boldsymbol{K}} \mid \mathfrak{Q}(\boldsymbol{K}[q \mapsto W]) \subseteq W\} \ , \tag{19}$$

$$\nu q\mathfrak{Q}\colon \boldsymbol{K} \mapsto \bigcup\{W \subseteq W^{\boldsymbol{K}} \mid \mathfrak{Q}(\boldsymbol{K}[q \mapsto W]) \supseteq W\} \ . \tag{20}$$

Then,

$$||\mu q\varphi|| = \mu q||\varphi|| \ , \qquad ||\nu q\varphi|| = \nu q||\varphi|| \ , \tag{21}$$

for every L_μ formula φ.

To conclude, let's look at substitution, even though substitution is neither part of the definition of the syntax of L_μ nor involved in the definition of its semantics. But it will turn out to be useful to have a counterpart to substitution on the query side. It will be enough to consider substitutions with one variable only.

Assume $\mathfrak{Q}$ and $\mathfrak{Q}'$ are Kripke queries and q is a propositional variable. The query

$$\mathfrak{Q}[q \mapsto \mathfrak{Q}'] \tag{22}$$

is defined by

$$\boldsymbol{K} \mapsto \mathfrak{Q}(\boldsymbol{K}[q \mapsto \mathfrak{Q}'(\boldsymbol{K})]) \ . \tag{23}$$

Using a straightforward induction, one can now prove that if q is a propositional variable, φ an L_μ formula positive in q, and $\psi \in L_\mu$, then

$$||\varphi[\psi/q]|| = ||\varphi||[q \mapsto ||\psi||] \ . \tag{24}$$

This is the analogue of the substitution principle in predicate logic. Note that it is necessary to require that φ be positive in q because otherwise the substitution is not defined.

2.1.6 Fixed Point Alternation

There are several ways to define an appropriate concept of fixed point alternation. The simplest one is to count syntactic alternations between least and greatest fixed point operators. A more involved one, which was tailored specifically for the purposes of efficient model-checking, was introduced by Emerson and Lei, [6]. It gives rise to a coarser hierarchy. The one that we will use here gives an even coarser hierarchy and was introduced by Damian Niwiński, [14]. His definition turned out to be the most useful one, and we use it here as well. There are actually two ways of defining Niwiński's hierarchy.

First approach. For every natural number i, sets Σ_i and Π_i of L_μ formulas are defined. Σ_0 and Π_0 contain all modal L_μ formulas, that is, all formulas that do not contain any fixed point operator.

The class Σ_{i+1} is the smallest class of formulas which contains Π_i and is closed under substitution (see Subsection 2.1.3) and application of the least fixed point operator. Symmetrically, Π_{i+1} is the smallest class of formulas which contains Σ_i and is closed under substitution and application of the greatest fixed point operator.

Second approach, see [16]. First, we define for every fixed point formula φ its *alternation number,* denoted $\alpha_0(\varphi)$, by induction on the length of φ.

For brevity in notation, we use $<$ to denote the relation "is subformula of". Assume $\varphi = \eta q \psi$ is a fixed point formula. Let X_φ be the set of all fixed point subformulas χ of ψ where $q \in \mathrm{free}(\chi)$ and $\mathrm{free}(\chi) \subseteq \mathrm{free}(\chi')$ for every subformula χ' of ψ which is a superformula of χ. That is, X_φ contains a formula χ if

$$\chi \leq \psi \wedge \chi \in F_\eta \wedge q \in \mathrm{free}(\chi) \wedge \forall \chi' (\chi < \chi' \leq \psi \rightarrow \mathrm{free}(\chi) \subseteq \mathrm{free}(\chi')) \ . \tag{25}$$

Let

$$m_\mu = \max(\{\alpha_0(\chi) \mid \chi \in X_\varphi \cap F_\mu\} \cup \{0\}) \ , \tag{26}$$

$$m_\nu = \max(\{\alpha_0(\chi) \mid \chi \in X_\varphi \cap F_\nu\} \cup \{0\}) \ . \tag{27}$$

Finally, $\alpha_0(\varphi)$ is defined by

$$\alpha_0(\varphi) = \begin{cases} \max(m_\mu, m_\nu + 1) & \text{if } \eta = \mu, \\ \max(m_\nu, m_\mu + 1) & \text{if } \eta = \nu. \end{cases} \tag{28}$$

The *alternation depth* of a formula φ, denoted $\alpha(\varphi)$, is the maximum of the alternation numbers of its fixed point subformulas or else 0 if no fixed point operator occurs, i. e.,

$$\alpha(\varphi) = \max(\{\alpha_0(\psi) \mid \psi \leq \varphi \wedge \psi \in F_\eta\} \cup \{0\}) \ . \tag{29}$$

The relation of this notion of alternation depth with the classes Σ_i and Π_i defined above is as follows.

Remark 1 [16] *Let k be an arbitrary natural number and φ an arbitrary L_μ formula.*

1. *The alternation depth of φ is $\leq k$ iff $\varphi = \psi[\psi_0/q_0, \ldots, \psi_{l_1}/q_{l-1}]$ for some formula $\psi \in \Sigma_0$ and formulas $\psi_i \in \Sigma_k \cup \Pi_k$.*
2. *The formula φ belongs to Σ_k iff $\alpha(\varphi) \leq k$ and there is no ν-subformula ψ of φ with $\alpha_0(\psi) = k$.*
3. *The formula φ belongs to Π_k iff $\alpha(\varphi) \leq k$ and there is no μ-subformula ψ of φ with $\alpha_0(\psi) = k$.*

2.2 Alternating Tree Automata

Alternating tree automata are used to define Kripke queries.

2.2.1 Informal Description

Alternating tree automata are finite-state devices designed to accept or reject pointed Kripke structures. The computation of an alternating tree automaton on a pointed Kripke structure proceeds in rounds. At the beginning of every round there are several copies of the alternating tree automaton in different worlds of the Kripke structure, each of them in its own state; some worlds might be occupied by many copies, others might not accommodate a single one. During a round, each copy splits up in several new copies, which are sent to neighbored worlds and change their states, all this done according to the transition function. Initially, there is only one copy of the alternating tree automaton; it resides in the distinguished world of the pointed Kripke structure and starts in the initial state of the alternating tree automaton. To determine acceptance or rejectance of a computation of an alternating tree automaton on a pointed Kripke structure the entire computation tree is inspected; acceptance is then defined via path conditions for the infinite branches of the computation tree.

2.2.2 Formal Definition

Formally, an *alternating tree automaton* is a tuple

$$\boldsymbol{A} = (S^{\boldsymbol{A}}, s_I^{\boldsymbol{A}}, \delta^{\boldsymbol{A}}, \Omega^{\boldsymbol{A}}) \tag{30}$$

where

- $S^{\boldsymbol{A}}$ is a finite set of *states*,
- $s_I^{\boldsymbol{A}} \in S$ is an *initial state*,
- $\delta^{\boldsymbol{A}}$ is a *transition function* as specified below, and
- $\Omega^{\boldsymbol{A}} \colon S \to \mathbf{N}$ is a partial *priority function*, which assigns a *priority* to some states.

The transition function $\delta^{\boldsymbol{A}}$ maps every state to a transition condition over S; the set of all *transition conditions* over S is defined by:

- 0 and 1 are transition conditions over S,
- for every $q \in Q$, q and $\neg q$ are transition conditions over S,
- for every $s \in S$, s, $\Box s$, and $\Diamond s$ are transition conditions over S,
- if φ and ψ are transition conditions over S, then $\varphi \wedge \psi$ and $\varphi \vee \psi$ are transition conditions over S.

The *transition graph* of $\boldsymbol{A}$, denoted $\boldsymbol{G}(\boldsymbol{A})$, is the directed graph with vertex set S and with an edge from a state s to a state s' if s' occurs in $\delta(s)$. The priority function $\Omega^{\boldsymbol{A}}$ must have the property that $\inf(\pi) \cap \mathrm{dom}(\Omega^{\boldsymbol{A}}) \neq \emptyset$ holds for every infinite path π through $\boldsymbol{G}(\boldsymbol{A})$ starting with s_I.

2.2.3 Runs

The computational behavior of alternating tree automata is explained using the notion of a run. Assume $\boldsymbol{A}$ is an alternating tree automaton and $(\boldsymbol{K}, w_I)$ a pointed Kripke structure. A *run* of $\boldsymbol{A}$ on $(\boldsymbol{K}, w_I)$ is a $(W \times S)$-vertex-labeled tree

$$\boldsymbol{R} = (V^{\boldsymbol{R}}, E^{\boldsymbol{R}}, \lambda^{\boldsymbol{R}}) \tag{31}$$

satisfying the "initial condition" and "local consistency" as described further below. To phrase these conditions concisely, we need some more definitions.

For simplicity in notation, we will write $w^{\boldsymbol{R}}(v)$ and $s^{\boldsymbol{R}}(v)$ for the first and second component of $\lambda^{\boldsymbol{R}}(v)$, respectively. For every vertex v of $\boldsymbol{R}$, we define what it means for a transition condition τ over S to hold in v, denoted $\boldsymbol{K}, \boldsymbol{R}, v \models \tau$. This definition is by induction on the structure of τ:

- $\boldsymbol{K}, \boldsymbol{R}, v \not\models 0$ and $\boldsymbol{K}, \boldsymbol{R}, v \models 1$,
- $\boldsymbol{K}, \boldsymbol{R}, v \models q$ if $q \in \kappa^{\boldsymbol{K}}(w^{\boldsymbol{R}}(v))$ and $\boldsymbol{K}, \boldsymbol{R}, v \models \neg q$ if $q \notin \kappa^{\boldsymbol{K}}(w^{\boldsymbol{R}}(v))$,
- $\boldsymbol{K}, \boldsymbol{R}, v \models \varphi \vee \psi$ if $\boldsymbol{K}, \boldsymbol{R}, v \models \varphi$ or $\boldsymbol{K}, \boldsymbol{R}, v \models \psi$,
- $\boldsymbol{K}, \boldsymbol{R}, v \models \varphi \wedge \psi$ if $\boldsymbol{K}, \boldsymbol{R}, v \models \varphi$ and $\boldsymbol{K}, \boldsymbol{R}, v \models \psi$,

– $\boldsymbol{K}, \boldsymbol{R}, v \models s$ if there exists $v' \in \text{Scs}_{\boldsymbol{R}}(v)$ such that $\lambda^{\boldsymbol{R}}(v') = (w^{\boldsymbol{R}}(v), s)$,
– $\boldsymbol{K}, \boldsymbol{R}, v \models \Diamond s$ if there exists $v' \in \text{Scs}_{\boldsymbol{R}}(v)$ such that $s^{\boldsymbol{R}}(v') = s$ and $w^{\boldsymbol{R}}(v') \in \text{Scs}_{\boldsymbol{K}}(w^{\boldsymbol{R}}(v))$, and
– $\boldsymbol{K}, \boldsymbol{R}, v \models \Box s$ if for every $w \in \text{Scs}_{\boldsymbol{K}}(w^{\boldsymbol{R}}(v))$ there exists $v' \in \text{Scs}_{\boldsymbol{R}}(v)$ such that $\lambda^{\boldsymbol{R}}(v') = (w, s)$.

Note that the above satisfaction relation is local in the sense that only vertices in the neighborhood of v are inspected.

The two additional conditions that are required of a run now read as follows.

– *Initial condition.* Let v_0 be the root of $\boldsymbol{R}$. Then

$$\lambda^{\boldsymbol{R}}(v_0) = (w_I, s_I) \ . \tag{32}$$

– *Local consistency.* For every $v \in V^{\boldsymbol{R}}$,

$$\boldsymbol{K}, \boldsymbol{R}, v \models \delta(s^{\boldsymbol{R}}(v)) \ . \tag{33}$$

So locally an alternating tree automaton can inspect the labeling of the world it sits in; depending on its current state it can stay in this world and change its state, it can proceed to any or all successors of this world and change its state, and it can combine these options.

2.2.4 Acceptance

A run $\boldsymbol{R}$ is accepting if the state labeling of every infinite branch through $\boldsymbol{R}$ satisfies the parity acceptance condition determined by $\Omega^{\boldsymbol{A}}$. This is formalized as follows.

Let $f\colon M \to N$ and $\Omega\colon N \to \mathbf{N}$ be partial functions. The function f is *even* [*odd*] with respect to Ω if $\inf(\Omega \circ f)$ is nonempty and bounded and $\max(\inf(\Omega \circ f))$ is even [odd].

Let $\boldsymbol{R}$ be a run of an alternating tree automaton $\boldsymbol{A}$. An infinite branch π of $\boldsymbol{R}$ is *accepting* if $s^{\boldsymbol{R}} \circ \pi$ is even with respect to $\Omega^{\boldsymbol{A}}$. A run $\boldsymbol{R}$ is *accepting,* if every infinite branch through $\boldsymbol{R}$ is accepting.

In other words, for every infinite branch π we consider the sequence of natural numbers that is obtained from π by extracting the state component of the labellings of the vertices and applying the valuation function $\Omega^{\boldsymbol{A}}$; we require that the maximum natural number occurring infinitely often is even.

Note that for every [accepting] run $\boldsymbol{R}$ there exists a minimal [accepting] run which is a rooted subtree of $\boldsymbol{R}$. Moreover, every minimal run has the property that $\inf(\Omega^{\boldsymbol{A}} \circ s^{\boldsymbol{R}} \circ \pi)$ is nonempty for every branch π through $\boldsymbol{R}$.

A pointed Kripke structure is *accepted* by $\boldsymbol{A}$ if there exists an accepting run of $\boldsymbol{A}$ on the Kripke structure. The *query recognized* by $\boldsymbol{A}$, denoted $||\boldsymbol{A}||$, contains all pointed Kripke structures accepted by $\boldsymbol{A}$.

2.2.5 Index

Similar to the notion of alternation depth for L_μ formulas we now define the notion of index of an alternating tree automaton.

Let $\boldsymbol{A}$ be an alternating tree automaton and $\mathfrak{C}^{\boldsymbol{A}}$ the set of all strongly connected components of the transition graph of $\boldsymbol{A}$ reachable from $s_I^{\boldsymbol{A}}$. For every $C \in \mathfrak{C}^{\boldsymbol{A}}$, let

$$m_C^{\boldsymbol{A}} = \max\{\Omega^{\boldsymbol{A}}(s) - \Omega^{\boldsymbol{A}}(s') \mid s, s' \in C\} \ . \tag{34}$$

The *index* of $\boldsymbol{A}$, denoted $\mathrm{ind}(\boldsymbol{A})$, is defined by

$$\mathrm{ind}(\boldsymbol{A}) = \max(\{m_C + 1^{\boldsymbol{A}} \mid C \in \mathfrak{C}^{\boldsymbol{A}}\} \cup \{0\}) \ . \tag{35}$$

2.3 From μ-Calculus to Alternating Tree Automata

As we will see, it is straightforward to construct for every L_μ formula an alternating tree automaton that recognizes the exact query that the formula defines. It is, however, more complicated to prove the correctness of the construction.

2.3.1 The Conversion

Given a formula φ, we define an alternating tree automaton $\boldsymbol{A}_\varphi$ as follows. The subformulas of φ build the states of $\boldsymbol{A}_\varphi$; the formula φ itself is the initial state, the transition function reflects the structure of the formula, and the priority function reflects the alternation structure of the formula. This is explained in detail in what follows.

An L_μ formula is in *normal form* if every propositional variable q is only quantified at most once and if in this case all occurrences of q are in the scope of this quantification. Clearly, every formula is equivalent to a formula in normal form of the same size. So, henceforth, we will—without loss of generality—assume that all formulas are in normal form.

Given an L_μ formula φ in normal form and a propositional variable q occurring in φ, exactly one of the following two conditions is true.

1. Every occurrence of q in φ is free.
2. Every occurrence of q in φ is quantified by the same fixed point operator, that is, it is bound in the same subformula $\eta q \psi$.

In the second case, we denote the formula $\eta q \psi$ by φ_q.

Let φ be an L_μ formula in normal form. The alternating tree automaton $\boldsymbol{A}_\varphi$ is defined by

$$\boldsymbol{A}_\varphi = (S, s^I, \delta, \Omega) \tag{36}$$

where

– S is the set which contains for each subformula ψ of φ (including φ itself) a state denoted $\langle\psi\rangle$,
– the initial state is given by $s^I = \langle\varphi\rangle$, and
– the transition function is defined by

$$\delta(\langle 0\rangle) = 0 \ , \qquad \delta(\langle 1\rangle) = 1 \ , \tag{37}$$

$$\delta(\langle q\rangle) = \begin{cases} q & \text{if } q \in \mathrm{free}(\varphi), \\ \langle\varphi_q\rangle & \text{if } q \notin \mathrm{free}(\varphi), \end{cases} \qquad \delta(\langle\neg q\rangle) = \neg q \ , \tag{38}$$

$$\delta(\langle\psi \wedge \chi\rangle) = \langle\psi\rangle \wedge \langle\chi\rangle \ , \qquad \delta(\langle\psi \vee \chi\rangle) = \langle\psi\rangle \vee \langle\chi\rangle \ , \tag{39}$$

$$\delta(\langle\Diamond\psi\rangle) = \Diamond\langle\psi\rangle \ , \qquad \delta(\langle\Box\psi\rangle) = \Box\langle\psi\rangle \ , \tag{40}$$

$$\delta(\langle\mu q\psi\rangle) = \langle\psi\rangle \ , \qquad \delta(\langle\nu q\psi\rangle) = \langle\psi\rangle \ , \tag{41}$$

– the priority function is defined as follows.

The priority function Ω is defined very much like α_0. The only difference is that μ-formulas get odd values and ν-formulas get even values. Also, we want the valuation to start with 0 if possible.

– For every formula $\psi \in F_\mu$, the priority $\Omega(\langle\psi\rangle)$ is the smallest odd number greater or equal to $\alpha_0(\psi) - 1$.
– For every formula $\psi \in F_\nu$, the priority $\Omega(\langle\psi\rangle)$ is the smallest even number greater or equal to $\alpha_0(\psi) - 1$.

Clearly, Ω will meet the requirement specified earlier, because in every infinite path through the transition graph of $\boldsymbol{A}_\varphi$ at least one fixed point formula occurs infinitely often.

The main theorem of this section is the following.

Theorem 1 *Let φ be an arbitrary L_μ formula. Then*

$$||\varphi|| = ||\boldsymbol{A}_\varphi|| \ . \tag{42}$$

2.3.2 Proof of Correctness

The proof of the above theorem goes by induction on the alternation depth of φ. As a preparation, we prove a series of lemmas, which show how the operators and connectives of L_μ can be modeled by automata.

Disjunction and Conjunction. We start with disjunction and conjunction.

Lemma 2 *Let $\boldsymbol{A}_0$ and $\boldsymbol{A}_1$ be alternating tree automata with disjoint state sets. Further, let s_I be some new state and $\boldsymbol{A}_\cup$ and $\boldsymbol{A}_\cap$ be defined by*

$$\boldsymbol{A}_\cup = (S^0 \cup S^1 \cup \{s_I\}, s_I, \delta^0 \cup \delta^1 \cup \{(s_I, s_I^0 \vee s_I^1)\}, \Omega^0 \cup \Omega^1) \ , \tag{43}$$

$$\boldsymbol{A}_\cap = (S^0 \cup S^1 \cup \{s_I\}, s_I, \delta^0 \cup \delta^1 \cup \{(s_I, s_I^0 \wedge s_I^1)\}, \Omega^0 \cup \Omega^1) \ . \tag{44}$$

Here, S^0 stands for $S^{\boldsymbol{A}_0}$, S^1 stands for $S^{\boldsymbol{A}_1}$, and so on. Then

$$||\boldsymbol{A}_\cup|| = ||\boldsymbol{A_0}|| \cup ||\boldsymbol{A_1}|| \ , \qquad ||\boldsymbol{A}_\cap|| = ||\boldsymbol{A_0}|| \cap ||\boldsymbol{A_1}|| \ . \tag{45}$$

Proof. We only prove the claim for $\cup$; the proof for $\cap$ is similar. First, assume $(\boldsymbol{K}, w)$ is accepted by $\boldsymbol{A}_\cup$. Let $\boldsymbol{R}$ be a minimal accepting run of $\boldsymbol{A}_\cup$ on $(\boldsymbol{K}, w)$. Then, by definition of the transition function of $\boldsymbol{A}_\cup$, the root of $\boldsymbol{R}$ has exactly one successor v, which is labeled with (w, s_I^0) or (w, s_I^1). Clearly, $\boldsymbol{R}{\downarrow}v$ (the subtree of $\boldsymbol{R}$ rooted at v) is an accepting run of $\boldsymbol{A}_0$ or $\boldsymbol{A}_1$. So $(\boldsymbol{K}, w) \in ||\boldsymbol{A_0}|| \cup ||\boldsymbol{A_1}||$.

For the converse containment, assume $\boldsymbol{R}$ is an accepting run of $\boldsymbol{A}_0$ or $\boldsymbol{A}_1$ on some pointed Kripke structure $(\boldsymbol{K}, w)$. Consider the following tree. It has a root v labeled (w, s_I) with exactly on successor v' and the subtree rooted at v' is identical with $\boldsymbol{R}$. Then this tree is an accepting run of $\boldsymbol{A}_\cup$ on $(\boldsymbol{K}, w)$. Hence, $(\boldsymbol{K}, w) \in ||\boldsymbol{A}_\cup||$. □

The requirement that the state sets of $\boldsymbol{A}_0$ and $\boldsymbol{A}_1$ be disjoint is very strong. Clearly, the following would be enough. The transition functions of $\boldsymbol{A}_0$ and $\boldsymbol{A}_1$ as well as their priority functions agree on every joint state. So, overlapping state spaces can be tolerated to a certain extent.

Modal operators. Next, we consider the modal operators.

Lemma 3 *Let $\boldsymbol{A}$ be an alternating tree automaton. Further, let s_I be some new state and $\boldsymbol{A}_\Diamond$ and $\boldsymbol{A}_\Box$ defined by*

$$\boldsymbol{A}_\Diamond = (S^{\boldsymbol{A}} \cup \{s_I\}, s_I, \delta^{\boldsymbol{A}} \cup \{(s_I, \Diamond s_I^{\boldsymbol{A}})\}, \Omega^{\boldsymbol{A}}) \ , \tag{46}$$

$$\boldsymbol{A}_\Box = (S^{\boldsymbol{A}} \cup \{s_I\}, s_I, \delta^{\boldsymbol{A}} \cup \{(s_I, \Box s_I^{\boldsymbol{A}})\}, \Omega^{\boldsymbol{A}}) \ . \tag{47}$$

Then

$$||\boldsymbol{A}_\Diamond|| = \Diamond||\boldsymbol{A}|| \ , \qquad ||\boldsymbol{A}_\Box|| = \Box||\boldsymbol{A}|| \ . \tag{48}$$

Proof. We only prove the claim for $\Diamond$; the proof for $\Box$ is analogous.

Let $(\boldsymbol{K}, w)$ be an arbitrary pointed Kripke structure. First, assume $(\boldsymbol{K}, w)$ is accepted by $\boldsymbol{A}_\Diamond$. Let $\boldsymbol{R}$ be a minimal accepting run of $\boldsymbol{A}_\Diamond$ on $(\boldsymbol{K}, w)$. Then, by definition of the transition function of $\boldsymbol{A}_\Diamond$, the root of $\boldsymbol{R}$ has exactly one successor v with $s^{\boldsymbol{R}}(v) = s_I^{\boldsymbol{A}}$ and $w^{\boldsymbol{R}}(v) \in \mathrm{Scs}_{\boldsymbol{K}}(w)$. Clearly, the subtree of $\boldsymbol{R}$ rooted at this vertex is an accepting run of $\boldsymbol{A}$ on $(\boldsymbol{K}, s^{\boldsymbol{R}}(w))$. So $(\boldsymbol{K}, w) \in \Diamond||\boldsymbol{A}||$.

For the converse containment, assume $(\boldsymbol{K}, w) \in \Diamond||\boldsymbol{A}||$. Then there exists $w' \in \mathrm{Scs}_{\boldsymbol{K}}(w)$ such that $(\boldsymbol{K}, w) \in ||\boldsymbol{A}||$. Let $\boldsymbol{R}$ be an accepting run of $\boldsymbol{A}$ on $(\boldsymbol{K}, w')$. Consider the following tree. It has a root v labeled (w, s_I) with exactly one successor v' and the subtree rooted at v' is identical with $\boldsymbol{R}$. Then this tree is an accepting run of $\boldsymbol{A}_\Diamond$ on $(\boldsymbol{K}, w)$. Thus, $(\boldsymbol{K}, w) \in \boldsymbol{A}_\Diamond$. □

Composition. An alternating tree automaton $\boldsymbol{A}$ is *positive* in the propositional variable q if $\neg q$ does not occur in any transition condition $\delta^{\boldsymbol{A}}(s)$. Observe that whenever $\boldsymbol{A}$ is such an alternating tree automaton, then

$$||\boldsymbol{A}||(\boldsymbol{K}[q \mapsto W]) \subseteq ||\boldsymbol{A}||(\boldsymbol{K}[q \mapsto W']) \tag{49}$$

for every Kripke structure $\boldsymbol{K}$ and sets W, W' with $W \subseteq W' \subseteq W^{\boldsymbol{K}}$.

Let q be an arbitrary propositional variable, $\boldsymbol{A}$ an alternating tree positive in q, and $\boldsymbol{A}'$ an arbitrary alternating tree automaton. We define the alternating tree automaton $\boldsymbol{A}[\boldsymbol{A}'/q]$ by

$$\boldsymbol{A}[\boldsymbol{A}'/q] = (S^{\boldsymbol{A}} \cup S^{\boldsymbol{A}'}, s_I^{\boldsymbol{A}}, \delta, \Omega^{\boldsymbol{A}} \cup \Omega^{\boldsymbol{A}'}) \tag{50}$$

where the state sets are made disjoint beforehand and where δ is obtained from $\delta^{\boldsymbol{A}} \cup \delta^{\boldsymbol{A}'}$ by replacing q in every transition condition $\delta^{\boldsymbol{A}}(s)$ by $s_I^{\boldsymbol{A}'}$. In fact, just as above it is not really necessary to make the state sets disjoint—it suffices to make them disjoint where the transition functions or the priority functions do not agree.

Lemma 4 *Let q be a propositional variable, $\boldsymbol{A}$ an alternating tree automaton positive in q, and $\boldsymbol{A}'$ an arbitrary alternating tree automaton. Then*

$$||\boldsymbol{A}[\boldsymbol{A}'/\boldsymbol{q}]|| = ||\boldsymbol{A}||\,[q \mapsto ||\boldsymbol{A}'||] \quad . \tag{51}$$

Proof. Let $(\boldsymbol{K}, w)$ be a pointed Kripke structure and assume $(\boldsymbol{K}, w)$ is accepted by $\boldsymbol{A}[\boldsymbol{A}'/q]$. Let $\boldsymbol{R}$ be a minimal accepting run of this alternating tree automaton on $(\boldsymbol{K}, w)$. Let V' be the set of all vertices v of $\boldsymbol{R}$ with $s^{\boldsymbol{R}}(v) = s_I^{\boldsymbol{A}'}$ and $W' = \{w^{\boldsymbol{R}}(v) \mid v \in V'\}$.

Let $\boldsymbol{T} = \boldsymbol{R} - V'$. By construction, this tree is an accepting run of $\boldsymbol{A}$ on the pointed Kripke structure $(\boldsymbol{K}[q \mapsto V'], w)$. Since $\boldsymbol{A}$ is positive in q, it is enough to show that $W' \subseteq ||\boldsymbol{A}'||(\boldsymbol{K})$. But this is trivial, as for every $v \in V'$, the tree $\boldsymbol{R}{\downarrow}v$ is an accepting run of $\boldsymbol{A}'$ on $(\boldsymbol{K}, w^{\boldsymbol{K}}(v))$.

For the converse, assume $(\boldsymbol{K}, w) \in ||\boldsymbol{A}||\,[q \mapsto ||\boldsymbol{A}'||]$. Then $(\boldsymbol{K}[q \mapsto W], w) \in ||\boldsymbol{A}||$ where $W = \{w' \in W^{\boldsymbol{K}} \mid (\boldsymbol{K}, w') \in ||\boldsymbol{A}'||\}$. For every $w' \in W$, let $\boldsymbol{R}_{w'}$ be a minimal accepting run of $\boldsymbol{A}'$ on $(\boldsymbol{K}, w')$. Further, let $\boldsymbol{R}$ be a minimal accepting run of $\boldsymbol{A}$ on $(\boldsymbol{K}[q \mapsto W], w)$.

Consider the tree $\boldsymbol{R}' = \boldsymbol{R} \cdot \{(v, \boldsymbol{R}_{w^{\boldsymbol{K}}(v)}) \mid w^{\boldsymbol{K}}(v) \in W\}$. Clearly, this tree is an accepting run of $\boldsymbol{A}[\boldsymbol{A}'/q]$ on $(\boldsymbol{K}, w)$. □

Fixed point operators. Finally, we model least and greatest fixed point operators. Let q be an arbitrary propositional variable and $\boldsymbol{A}$ an alternating tree automaton positive in q. We want to construct an alternating tree automaton that

recognizes $\mu q||\boldsymbol{A}||$. This only difficult part in the construction is the definition of the priority function. This needs some preparation.

Let $\boldsymbol{G}(\boldsymbol{A}) = (S^{\boldsymbol{A}}, E)$ be the transition graph of $\boldsymbol{A}$. Consider the graph

$$\boldsymbol{G} = (S^{\boldsymbol{A}} \cup \{s_I\}, E \cup \{(s_I, s_I^{\boldsymbol{A}})\} \cup E') \tag{52}$$

where s_I is a new state and E' contains an edge (s, s_I) for every s where q occurs in $\delta(s)$. Let C be the strongly connected component of $\boldsymbol{G}$ the state s_I belongs to and $m = \max(\{\Omega^{\boldsymbol{A}}(s) \mid s \in C \cap \mathrm{dom}(\Omega^{\boldsymbol{A}})\})$. Further, let m^μ and m^ν be an arbitrary odd respectively even number greater or equal to m.

The alternating tree automata $\boldsymbol{A}_{\mu q}$ and $\boldsymbol{A}_{\nu q}$ are defined by

$$\boldsymbol{A}_{\mu q} = (S^{\boldsymbol{A}} \cup \{s_I\}, s_I, \delta, \Omega^{\boldsymbol{A}} \cup \{(s_I, m^\mu)\}) \ , \tag{53}$$

$$\boldsymbol{A}_{\nu q} = (S^{\boldsymbol{A}} \cup \{s_I\}, s_I, \delta, \Omega^{\boldsymbol{A}} \cup \{(s_I, m^\nu)\}) \ , \tag{54}$$

where δ is as $\delta^{\boldsymbol{A}} \cup \{(s_I, s_I^{\boldsymbol{A}})\}$ except that every occurrence of q is replaced by s_I.

Lemma 5 *Let q be a propositional variable and $\boldsymbol{A}$ an alternating tree automaton positive in q. Then*

$$||\boldsymbol{A}_{\boldsymbol{\mu q}}|| = \mu q||\boldsymbol{A}|| \ , \qquad ||\boldsymbol{A}_{\boldsymbol{\nu q}}|| = \nu q||\boldsymbol{A}|| \ . \tag{55}$$

Proof. Let $\boldsymbol{K}$ be an arbitrary Kripke structure. Let $f\colon 2^{W^{\boldsymbol{K}}} \to 2^{W^{\boldsymbol{K}}}$ be defined by

$$f\colon W \mapsto ||\boldsymbol{A}||(\boldsymbol{K}[q \mapsto W]) \ . \tag{56}$$

We want to show that $||\boldsymbol{A}_{\boldsymbol{\mu q}}||(\boldsymbol{K})$ and $||\boldsymbol{A}_{\boldsymbol{\nu q}}||(\boldsymbol{K})$ are, respectively, the least and greatest fixed point of f. We denote these fixed points by W_μ and W_ν, respectively.

We first show that W_μ and W_ν are fixed points of f and consider only μ; the argument is similar for ν. Clearly, (accepting) runs of $\boldsymbol{A}_{\mu q}$ and $\boldsymbol{A}[\boldsymbol{A}_{\mu q}/q]$ are in a natural one-to-one correspondence. Thus,

$$W_\mu = ||\boldsymbol{A}_{\boldsymbol{\mu q}}||(\boldsymbol{K}) \tag{57}$$

$$= ||\boldsymbol{A}[\boldsymbol{A}_{\boldsymbol{\mu q}}/\boldsymbol{q}]||(\boldsymbol{K}) \tag{58}$$

$$= ||\boldsymbol{A}||(\boldsymbol{K}\,[q \mapsto ||\boldsymbol{A}_{\boldsymbol{\mu q}}||(\boldsymbol{K})]) \tag{59}$$

$$= ||\boldsymbol{A}||(\boldsymbol{K}\,[q \mapsto W_\mu]) \tag{60}$$

$$= f(W_\mu) \ , \tag{61}$$

where (57) and (60) use the definition of W_μ, (58) is due to the above observation, (59) is Lemma 4, and (61) is just the definition of f.

So for the rest it is enough to show:

1. Every element of W_μ belongs to some approximant for the least fixed point of f.
2. Whenever $||\boldsymbol{A}||(K[q \mapsto W]) = W$ for some $W \subseteq W^{\boldsymbol{K}}$, then $W \subseteq W_\nu$.

First, assume $w \in W_\mu$. Let $\boldsymbol{R}$ be a minimal accepting run for $\boldsymbol{A}_{\mu q}$ on $(\boldsymbol{K}, w)$ and U the set of all vertices v with $s^{\boldsymbol{R}}(v) = s_I$. For every ordinal λ, let U_λ be defined as explained in Lemma 1. Since $\boldsymbol{R}$ is assumed to be accepting, only a finite number of elements from U occur on every branch through $\boldsymbol{R}$. Lemma 1 then implies that for every $v \in U$ there exists an ordinal λ_v such that $v \in U_\lambda$. Using transfinite induction, it is easy to show that for every ordinal λ and every $v \in U_\lambda$, $w^{\boldsymbol{R}}(v)$ is in the λ-approximant of f from below. In particular, w is in the λ_v-approximant of f from below where v is the root of $\boldsymbol{R}$.

Second, suppose W is a fixed point of f. Just as above, we can argue that for every $w \in W$ there exists an accepting run of $\boldsymbol{A}$ on $(\boldsymbol{K}[q \mapsto W], w)$. Pick such an accepting run for every $w \in W$ and denote it by $\boldsymbol{R}_w$. Further, assume all the $\boldsymbol{R}_w$'s are minimal.

Fix an arbitrary $w \in W$. We define a sequence $\boldsymbol{T}_0, \boldsymbol{T}_1, \ldots$ of trees where each tree $\boldsymbol{T}_i$ is a subgraph of $\boldsymbol{T}_{i+1}$. The limit of this sequence, which we denote by $\boldsymbol{T}$, will be an accepting run of $\boldsymbol{A}_\nu$ on $(\boldsymbol{K}, w)$.

The inductive definition of the $\boldsymbol{T}_i$'s is as follows. First, for every $w' \in W$ let $\boldsymbol{R}'_{w'}$ be the tree that results from $\boldsymbol{R}_{w'}$ by adding a new root labeled (w', s_I). Second, let $\boldsymbol{T}_0 = \boldsymbol{R}'_w$. Third, assume $\boldsymbol{T}_i$ has already been defined and let $B_i = \{v \in V^{\boldsymbol{T}_i} \mid w^{\boldsymbol{T}_i}(v) \in W\}$. Then $\boldsymbol{T}_{i+1}$ is defined by

$$\boldsymbol{T}_{i+1} = \boldsymbol{T}_i \cdot \{(v, \boldsymbol{R}'_{w^{\boldsymbol{T}_i}(v)}) \mid v \in B_i\} \ . \tag{62}$$

Clearly, $\boldsymbol{T}$ is a run of $\boldsymbol{A}_\nu$ on $(\boldsymbol{K}, w)$. We only need to show that it is accepting.

Assume π is an infinite branch of $\boldsymbol{T}$. We distinguish two cases. First, suppose π is a branch of some tree $\boldsymbol{T}_i$. Then, just as before, there is some $w' \in W$ such that a suffix of π is an infinite branch of $\boldsymbol{R}_{w'}$ and is therefore accepting. Second, suppose π is not a branch of any $\boldsymbol{T}_i$. Then s_I occurs infinitely often on π, but $\Omega^{\boldsymbol{A}}(s_I)$ is the maximum priority and even. So π is an accepting branch. □

Inductive argument. As stated above, the proof of Theorem 1 goes by induction.

Clearly, $||\boldsymbol{A}_{\boldsymbol{\varphi}}|| = ||\varphi||$ whenever φ is of the form $\bot$, $\top$, q, or $\neg q$. If φ is a composite formula, we distinguish several cases according to the outermost connective or operator.

If the outermost connective is disjunction or conjunction, the claim follows from Lemma 2. Similarly, if the outermost operator is a modal operator, the claim follows from Lemma 3. The only interesting case is when the outermost connec-

tive is a fixed point operator, say $\varphi = \mu q \psi$. Then $\boldsymbol{A}_\varphi = \boldsymbol{A}_{\psi_{\boldsymbol{\mu q}}}$. So the claim follows from Lemma 5. The case where φ is of the form $\nu q \psi$ is analogous. □

3 Model-Checking

In this section, we look at a first application of the main theorem of the last section: we investigate the complexity of the model checking problem for L_μ. This is the following problem.

MODELCHECKING: given a finite pointed Kripke structure $(\boldsymbol{K}, w)$ and an L_μ formula φ, determine whether or not $(\boldsymbol{K}, w) \models \varphi$.

Now that we know that for every L_μ formula there exists an equivalent alternating tree automaton, the model checking problem can be reduced to the "word" problem for alternating tree automata, which is the following problem.

ACCEPTS: given a finite pointed Kripke structure $(\boldsymbol{K}, w)$ and an alternating tree automaton $\boldsymbol{A}$, determine whether $\boldsymbol{A}$ accepts $(\boldsymbol{K}, w)$.

The word problem for alternating tree automata itself will be reduced to the winner problem for parity games.

In the first subsection, the fundamentals about parity games are briefly recalled. In the second subsection, the reduction from the model checking problem for the modal μ-calculus to the winner problem for parity games is described. This also yields the desired upper bound for the complexity of the model checking problem for modal μ-calculus.

3.1 Parity Games

Parity games are a special form of two-player infinite games on graphs.

3.1.1 Informal Description

A parity game is played by two players, the male Player 0 and the female Player 1. It is played on a game board which shows circles, Player 0's locations, and boxes, Player 1's locations. The circles and boxes are connected by arrows. One of the locations is a distinguished initial location, and every location is assigned a number from a finite set of natural numbers, its priority.

A parity game is played using a pebble, which during a play of the game is moved by the players from location to location along the arrows on the game board.

A play of a game proceeds in rounds. At the beginning of each round, the pebble is in some location, the current location. In the first round, the current location is the initial location. The rules for playing a round are as follows. If the

current location is a dead end, then the play is over and the player who owns the locations looses. If the current location is no dead end the player who owns the current location moves the pebble from this location along an arrow to another location and thereby completes the round. (Note that there is no restriction on the arrows. So it can very well be that there are self arrows and the new current location will be the old current location and the same player goes again in the next round.)

If a play does not stop after a finite number of rounds an infinite number of locations is visited during the course of the play and the play can be viewed as an infinite sequence of locations. In this case, the winner is determined as follows. One considers (the bounded) sequence of natural numbers that is obtained by replacing every location of the above sequence by its priority. Player 0 wins if the maximum number occurring infinitely often in this sequence is even, else Player 1 wins.

3.1.2 Formal Definition

Formally, a *parity game* is a tuple

$$\boldsymbol{P} = (L^{\boldsymbol{P}}, L_0^{\boldsymbol{P}}, L_1^{\boldsymbol{P}}, l_I^{\boldsymbol{P}}, M^{\boldsymbol{P}}, \Omega^{\boldsymbol{P}}) \tag{63}$$

where

- $L^{\boldsymbol{P}}$ is a set of *locations*,
- $L_0^{\boldsymbol{P}}$ and $L_1^{\boldsymbol{P}}$ partition $L^{\boldsymbol{P}}$ into *Player 0's and Player 1's locations*, respectively,
- $l_I^{\boldsymbol{P}} \in L_0^{\boldsymbol{P}} \cup L_1^{\boldsymbol{P}}$ is an *initial location*,
- $M^{\boldsymbol{P}} \subseteq (L_0^{\boldsymbol{P}} \cup L_1^{\boldsymbol{P}}) \times (L_0^{\boldsymbol{P}} \cup L_1^{\boldsymbol{P}})$ is a set of *moves*, and
- $\Omega^{\boldsymbol{P}} \colon L_0^{\boldsymbol{P}} \cup L_1^{\boldsymbol{P}} \to \mathbf{N}$ is a partial *priority function* assigning to some locations a priority.

It is required that there exists a natural number b such that $\Omega^{\boldsymbol{P}}(l) < b$ holds for all $l \in L_0^{\boldsymbol{P}} \cup L_1^{\boldsymbol{P}}$, and there is a requirement on the domain of $\Omega^{\boldsymbol{P}}$, which is stated further below.

Let $\boldsymbol{P}$ be a game as above. Clearly, the ordered pair $(L^{\boldsymbol{P}}, M^{\boldsymbol{P}})$ is a directed graph, which is denoted $\boldsymbol{G_P}$ and called the *game graph* of $\boldsymbol{P}$.

A *partial play* of $\boldsymbol{P}$ is a path through $\boldsymbol{G_P}$ starting with $l_I^{\boldsymbol{P}}$. A *play* of $\boldsymbol{P}$ is a maximum path through $\boldsymbol{G_P}$ starting with $l_I^{\boldsymbol{P}}$. A play p is *winning* for Player 0 if it is infinite and even with respect to $\Omega^{\boldsymbol{P}}$ or if it is finite and $p(*) \in L_1^{\boldsymbol{P}}$. Symmetrically, a play is winning for Player 1 if it is infinite and odd with respect to $\Omega^{\boldsymbol{P}}$ or if it is finite and $p(*) \in L_0^{\boldsymbol{P}}$.

The domain of the priority function $\Omega^{\boldsymbol{P}}$ must have the property that $\{i \mid p(i) \in \mathrm{dom}(\Omega^{\boldsymbol{P}})\}$ is infinite for every infinite play p. This is very similar to the requirement for the priority function of alternating tree automata of previous section.

A Player 0 wins a game (as opposed to a play) if he has a way to move such that regardless of how his opponent moves he wins each of the resulting plays. This is formalized as follows.

Let $\boldsymbol{P}$ be a parity game. A *strategy tree* for Player 0 in $\boldsymbol{P}$ is a rooted subtree of

- The root of $\boldsymbol{T_P}$ is labeled $l_I^{\boldsymbol{P}}$.
- Every $v \in V^{\boldsymbol{T}}$ with $\lambda^{\boldsymbol{T}}(v) \in L_0^{\boldsymbol{P}}$ has a successor in $\boldsymbol{T}$ labeled with a successor of $\lambda^{\boldsymbol{T}}(v)$ in $\boldsymbol{G_P}$, that is, Player 0 must move when it is his turn.
- Every $v \in V^{\boldsymbol{T}}$ with $\lambda^{\boldsymbol{T}}(v) \in L_1^{\boldsymbol{P}}$ has, for every successor l of $\lambda^{\boldsymbol{T}}(v)$ in $\boldsymbol{G_P}$ a successors in $\boldsymbol{T}$ labeled l, that is, all options for Player 1 have to be taken into account.

A branch v of a strategy tree $\boldsymbol{T}$ is *winning* if its labeling, which is a play, is winning. A strategy tree $\boldsymbol{T}$ for Player 0 is *winning* if every branch through $\boldsymbol{T}$ is winning. Player 0 wins a game $\boldsymbol{P}$ if there exists a winning strategy tree for him. Symmetrically, it is defined what it means for Player 1 to win a play or a game.

3.1.3 Determinacy and Complexity

There are two basic questions about games that we need to answer before we can try to solve the model-checking problem.

1. Does every game have a winner?
2. How difficult is it to determine who wins a finite game (if there is a winner at all)?

The first question has a positive answer, which is usually phrased using the notion of determinacy. A game is said to be *determined* if one of the two players wins it.

Theorem 2 [7] *Every parity game is determined.*

The decision problem of determining the winner of a parity game is formally defined as follows.

WINS: given a finite parity game $\boldsymbol{P}$, determine whether or not Player 0 wins the game $\boldsymbol{P}$.

To describe the actual complexity of solving WINS, we need some more definitions; we need to define a notion of index for a parity game.

The *index* of a finite parity game $\boldsymbol{P}$ is determined as follows, very similar to the index of an alternating tree automaton. Let $\mathfrak{C}^{\boldsymbol{P}}$ be the set of all strongly connected components of the game graph of $\boldsymbol{P}$ reachable from $l_I^{\boldsymbol{P}}$. For every $C \in \mathfrak{C}^{\boldsymbol{P}}$, let $m_C^{\boldsymbol{P}} = \max\{\Omega^{\boldsymbol{P}}(l) - \Omega^{\boldsymbol{P}}(l') \mid l, l' \in C\}$. The *index* of $\boldsymbol{P}$, denoted $\mathrm{ind}(\boldsymbol{P})$, is defined by

$$\mathrm{ind}(\boldsymbol{P}) = \max(\{m_C^{\boldsymbol{P}} \mid C \in \mathfrak{C}^{\boldsymbol{P}}\} \cup \{0\}) + 1 \ . \tag{64}$$

The best known upper bounds for the time complexity of WINS are:

Theorem 3 [8, 9]

1. WINS, *the winner problem for finite parity games, is solvable in time*

$$\mathcal{O}\left(m\left(\frac{2n}{b}\right)^{\lfloor b/2\rfloor}\right) \tag{65}$$

   where m is the number of moves in a given game, n the number of locations, and b its index, that is, $n = |L^{\boldsymbol{P}}|$, $m = |M^{\boldsymbol{P}}|$, and $b = \mathrm{ind}(\boldsymbol{P})$.
2. WINS *is in* ***UP*** $\cap$ ***co-UP***.

WINS is easily seen to be **P**-hard.

3.2 Reduction of the Word Problem

The objective of this subsection is to solve ACCEPTS as specified earlier. This is done by a reduction from ACCEPTS to WINS.

Given an alternating tree automaton $\boldsymbol{A}$ and a pointed Kripke structure $(\boldsymbol{K}, w_I)$ we want to construct a game $\boldsymbol{P}(\boldsymbol{A}, \boldsymbol{K}, w_I)$ that Player 0 wins if and only if $\boldsymbol{A}$ accepts $(\boldsymbol{K}, w_I)$. The basic idea is that the choices Player 0 makes correspond to the choices $\boldsymbol{A}$ has to make when in a transition condition it has to satisfy a disjunction or a $\Diamond$ requirement. Symmetrically, the moves for Player 1 correspond to conjunctions and $\Box$ requirements. Remember that a winning strategy for Player 0 has to make sure that whatever Player 1 does in a play, it will be a win for Player 0.

Formally, the *game $\boldsymbol{P}(\boldsymbol{A}, \boldsymbol{K}, w_I)$ associated with $\boldsymbol{A}$ and $(\boldsymbol{K}, w_I)$* is defined by

$$\boldsymbol{P}(\boldsymbol{A}, \boldsymbol{K}, w_I) = (W^{\boldsymbol{K}} \times T, L_0, L_1, (w_I^{\boldsymbol{K}}, s_I^{\boldsymbol{A}}), M, \Omega) \tag{66}$$

where the individual components are as follows.

The set T is the set of all subformulas occurring in some formula $\delta^{\boldsymbol{A}}(s)$ augmented by all states from $\boldsymbol{A}$. (So T is a finite set of transition conditions.) The set L_0 is the set of all pairs (w, τ) where τ is of the form 0, q with $q \notin \kappa^{\boldsymbol{K}}(w)$, $\neg q$ with $q \in \kappa^{\boldsymbol{K}}(w)$, $\tau_l \vee \tau_r$, or $\Diamond\tau'$. This also determines L_1. A pair $((w, \tau), (w', \tau'))$ is a move, that is, an element of M, if one of the following conditions holds.

- $\tau = s$, $w = w'$, and $\tau' = \delta^{\boldsymbol{A}}(s)$.
- $\tau = \tau_l \vee \tau_r$ or $\tau = \tau_l \wedge \tau_r$, $w = w'$, and $\tau' = \tau_l$ or $\tau' = \tau_r$.
- $\tau = \Box s$ or $\tau = \Diamond s$, $\tau' = s$, and $w' \in \mathrm{Scs}_{\boldsymbol{K}}(w)$.

Finally, the priority function Ω maps (w, τ) to $\Omega^{\boldsymbol{A}}(\tau)$ if $\tau \in S$ and $\Omega^{\boldsymbol{A}}(\tau)$ is defined.

The desired theorem is the following.

Theorem 4 *Let $(\boldsymbol{K}, w)$ be a pointed Kripke structure and $\boldsymbol{A}$ an alternating tree automaton. The alternating tree automaton $\boldsymbol{A}$ accepts $(\boldsymbol{K}, w)$ if and only if Player 0 has a winning strategy in the game $\boldsymbol{P}(\boldsymbol{K}, \boldsymbol{A}, w)$.*

Proof. We will convert accepting runs into winning strategy trees for Player 0 and vice versa.

As a preparation, we extend our notion of a run so that runs also provide information why they are locally consistent. Formally, a $(W \times T)$-labeled tree $\boldsymbol{R}$ is an *extended run* of an alternating tree automaton $\boldsymbol{A}$ on a pointed Kripke structure $(\boldsymbol{K}, w_I)$ if the root is labeled $(w_I, s_I^{\boldsymbol{A}})$ and for every vertex v labeled (w, τ) the following holds true, analogous to local consistency defined earlier.

- If $\tau = \varphi \vee \psi$, then v has one successor, labeled with (w, φ) or (w, ψ).
- If $\tau = \varphi \wedge \psi$, then v has a successor labeled (w, φ) and a successor labeled (w, ψ).
- If $\tau = s$, then v has a successor labeled $(w, \delta^{\boldsymbol{A}}(s))$.
- If $\tau = \Diamond s$, then v has a successor labeled (w', s) for some $w' \in \mathrm{Scs}_{\boldsymbol{K}}(w)$.
- If $\tau = \Box s$, then v has a successor labeled (w', s) for every $w' \in \mathrm{Scs}_{\boldsymbol{K}}(w)$.

An extended run $\boldsymbol{R}$ is accepting if $pr_1 \circ \lambda^{\boldsymbol{R}} \circ \pi$ is even with respect to $\Omega^{\boldsymbol{A}}$ for every infinite branch π through $\boldsymbol{R}$.

Clearly, every [accepting] run can be extended to an [accepting] extended run, and every [accepting] extended run can be reduced to an [accepting] run.

We can now turn to the proof of the theorem. Assume $\boldsymbol{A}$ accepts $(\boldsymbol{K}, w_I)$ and let $\boldsymbol{P} = \boldsymbol{P}(\boldsymbol{A}, \boldsymbol{K}, w_I)$ be defined as above. Since $\boldsymbol{A}$ accepts $(\boldsymbol{K}, w_I)$, there exists an accepting extended run of $\boldsymbol{A}$ on $(\boldsymbol{K}, w_I)$. Let $\boldsymbol{R}$ be a minimal such run. We show that the tree $\boldsymbol{R}$ is a winning strategy tree for Player 0.

Let's first show that $\boldsymbol{R}$ is a strategy tree at all. Assume (w, τ) is the label of some vertex v. If $v \in L_0^{\boldsymbol{P}}$, then

- $\tau = 0$,
- $\tau = q$ and $q \notin \kappa^{\boldsymbol{K}}(w)$,
- $\tau = \neg q$ and $q \in \kappa^{\boldsymbol{K}}(w)$,
- $\tau = \tau_l \vee \tau_r$, or
- $\tau = \Diamond \tau'$.

The first three cases cannot occur, as $\boldsymbol{R}$ is assumed to be an extended run and therefore satisfies the modified rules about local consistency as stated above. In the other two cases, the same rules guarantee that v has a successor in $\boldsymbol{R}$. If $v \in L_1^{\boldsymbol{G}}$, then

- $\tau = 1$,
- $\tau = q$ and $q \in \kappa^{\boldsymbol{K}}(w)$,
- $\tau = \neg q$ and $q \notin \kappa^{\boldsymbol{K}}(w)$,
- $\tau = \tau_l \wedge \tau_r$, or

– $\tau = \Box\tau'$.

In all cases, the modified rules about local consistency guarantee that all successors of v in $\boldsymbol{G_P}$ are also successors of v in $\boldsymbol{R}$.

It remains to show that every branch of $\boldsymbol{R}$ is winning. First, let v be a finite branch through $\boldsymbol{R}$ and assume v ends in a vertex labeled (w, τ). Then, because $\boldsymbol{R}$ is a run,

– $\tau = q$ and $q \in \kappa^{\boldsymbol{K}}(w)$,
– $\tau = \neg q$ and $q \notin \kappa^{\boldsymbol{K}}(w)$, or
– $\tau = 1$.

All such vertices are dead ends in the game graph of $\boldsymbol{P}(\boldsymbol{A}, \boldsymbol{K}, w)$ and belong to Player 1. So all these branches are winning for Player 0. Next, let π be an infinite branch through $\boldsymbol{R}$. Then, because $\boldsymbol{R}$ is an accepting run, $pr_1 \circ \lambda^{\boldsymbol{R}} \circ \pi$ is even with respect to $\Omega^{\boldsymbol{A}}$. Thus, π is winning for Player 0. So $\boldsymbol{R}$ is a winning strategy tree for Player 0.

For the converse, assume $\boldsymbol{T}$ is a winning strategy tree for Player 0 in the game $\boldsymbol{P}(\boldsymbol{A}, \boldsymbol{K}, w)$. Similar to above, it is easy to see that $\boldsymbol{T}$ is an accepting extended run for $\boldsymbol{A}$ on $(\boldsymbol{K}, w_I)$. □

In view of Theorem 3, this yields the following about the complexity of the word problem for alternating tree automata.

Theorem 5 *1.* ACCEPTS, *the word problem for alternating tree automata, is solvable in time*

$$\mathcal{O}\left(kl\left(\frac{2kn}{b}\right)^{\lfloor b/2 \rfloor}\right) \tag{67}$$

where k is the number of worlds of the Kripke structure, l is the size of the accessibility relation, n is the number of subformulas of transition conditions, and b is the maximum value in the priority function.

2. ACCEPTS *is in* ***UP*** $\cap$ ***co-UP***.

As a consequence, we obtain the following complexity bounds on the model checking problem for modal μ-calculus.

Theorem 6 ([8, 19, 12]) *1.* MODELCHECKING, *the model-checking problem for modal μ-calculus, is solvable in time*

$$\mathcal{O}\left(kl\left(\frac{2kn}{b}\right)^{\lfloor b/2 \rfloor}\right) \tag{68}$$

where k is the number of worlds of the Kripke structure, l is the size of the accessibility relation, n is the number of subformulas of the formula, and b is the alternation depth.

2. MODELCHECKING *is in* ***UP*** $\cap$ ***co-UP***.

4 Satisfiability

In this section, we consider another application of the main theorem of Section 2: we investigate the complexity of the satisfiability problem, which is the following problem.

SATISFIABILITY: given an L_μ formula φ, determine whether or not there exists a pointed Kripke structure $(\boldsymbol{K}, w)$ such that $(\boldsymbol{K}, w) \models \varphi$.

We will attack this problem just as the model-checking problem. Since we know that every L_μ formula φ is equivalent to the alternating tree automaton $\boldsymbol{A}_\varphi$, we only need to check whether or not $\boldsymbol{A}_\varphi$ accepts some pointed Kripke structure, that is, we can reduce SATISFIABILITY to the following problem.

NONEMPTINESS: given an alternating tree automaton $\boldsymbol{A}$, determine whether or not $\boldsymbol{A}$ recognizes some pointed Kripke structure.

This problem is reducible to the winner problem for parity games, which we already know from the previous section. A direct reduction is, however, quite complicated. To make the reduction more transparent an intermediate problem is introduced, namely the winner problem for parity games with extended winning condition (extended parity games). The emptiness problem can then be reduced to the winner problem for extended parity games, which, in turn, can be reduced to the winner problem for ordinary parity games.

In the first subsection, it is shown—using a deep theorem from game theory—that we can use a much less complicated object than a run to describe acceptance of an alternating tree automaton. This will later help us to solve the emptiness problem for alternating tree automata. The second subsection is meant as a preparation for the third subsection, where extended parity games are introduced and studied. The fourth subsection presents the reduction from the emptiness problem for alternating tree automata to the winner problem for extended parity games and thus completes the construction.

4.1 Memoryless Strategies and Accepting Witnesses

A game won by Player 0 is called memoryless if the moves Player 0 has to make in order to win the game are independent of the history of the game, that is, if Player 0 does not need to remember anything about the past of a play in order to be able to take the right decision. This can be easily described formally by looking at strategy trees. A *memoryless strategy tree* for Player 0 is a *strategy tree* $\boldsymbol{T}$ satisfying the following additional condition. Whenever $v, v' \in V^{\boldsymbol{T}}$ are such that $\lambda^{\boldsymbol{T}}(v) = \lambda^{\boldsymbol{T}}(v') \in L_0$, then there is a bijection between $\mathrm{Scs}_{\boldsymbol{T}}(v')$ and $\mathrm{Scs}_{\boldsymbol{T}}(v')$ which preserves the labeling. This obviously implies that if $\lambda^{\boldsymbol{T}}(v) = \lambda^{\boldsymbol{T}}(v') \in L_0$, then $\boldsymbol{T}{\downarrow}v$ and $\boldsymbol{T}{\downarrow}v'$ are isomorphic.

Theorem 7 [5] *For the winner of a parity game, there exists a memoryless winning strategy.* □

Next, we define the notion of an accepting witness. Let $\boldsymbol{A}$ be an alternating tree automaton and $(\boldsymbol{K}, w_I)$ a pointed Kripke structure. An *accepting witness* for $\boldsymbol{A}$ and $(\boldsymbol{K}, w_I)$ is a graph

$$\boldsymbol{W} = (V^{\boldsymbol{W}}, E^{\boldsymbol{W}}) \tag{69}$$

with $V \subseteq W \times S$ such that

- $(w, s_I) \in V$,
- every vertex is reachable from (w_I, s_I),
- $(V, E), (w, s) \models \delta(s)$ for every $(w, s) \in V$,
- $pr_1 \circ \pi$ is even with respect to $\Omega^{\boldsymbol{A}}$ for every infinite path π through $\boldsymbol{W}$ starting in (w_I, s_I).

In the above, $\models$ is defined exactly as in Subsection 2.2.3 where it is assumed that a vertex (w, s) is labelled with (w, s).

From Theorem 7 we can now conclude.

Corollary 1 *Let $\boldsymbol{A}$ be an alternating tree automaton and $(\boldsymbol{K}, w)$ a pointed Kripke structure. The automaton $\boldsymbol{A}$ accepts $(\boldsymbol{K}, w)$ if and only if there exists an accepting witness for $\boldsymbol{A}$ and $(\boldsymbol{K}, w)$.*

Proof. Reconsider the proof of Theorem 4. Assume $(\boldsymbol{K}, w)$ is pointed Kripke structure accepted by some alternating tree automaton $\boldsymbol{A}$. Then there exists a winning strategy tree for Player 0 in $\boldsymbol{P}(\boldsymbol{A}, \boldsymbol{K}, w)$. By Theorem 7, there exists a memoryless winning strategy tree for Player 0 in this game. Without loss of generality, we can assume this tree is minimal. Consider the run that corresponds to this strategy tree: it can be viewed as an accepting witness. □

In the rest of this subsection, we develop an alternative definition of local consistency for accepting witnesses, which will be useful later.

Let $\boldsymbol{A}$ be an alternating tree automaton and Γ the set of all propositional variables occurring in $\delta^{\boldsymbol{A}}$. Assume $\Gamma' \subseteq \Gamma$, $S_\triangle \subseteq S$, $S_\Box \subseteq S$, and $S_\Diamond \subseteq S$. We define a transition condition by

$$\tau(\Gamma', S_\triangle, S_\Box, S_\Diamond) = \bigwedge_{q \in \Gamma'} q \wedge \bigwedge_{q \in \Gamma \setminus \Gamma'} \neg q \wedge \bigwedge_{s \in S_\triangle} s \wedge \bigwedge_{s \in S_\Diamond} \Diamond s \wedge \bigwedge_{s \in S_\Box} \Box s \ . \tag{70}$$

Assume $\boldsymbol{W}$ is an accepting witness for some Kripke structure $(\boldsymbol{K}, w_I)$. Let w be an arbitrary world in $\boldsymbol{K}$, $\Gamma' = \kappa^{\boldsymbol{W}}(w)$, and s such that $(w, s) \in V^{\boldsymbol{W}}$. Since $\boldsymbol{W}$ is an accepting witness, we know there exist sets $S_\triangle$, $S_\Box$, $S_\Diamond$ such that

$$\tau(\Gamma', S_\triangle, S_\Box, S_\Diamond) \models \delta(s) \tag{71}$$

(that is, $\tau(\Gamma', S_\triangle, S_\Box, S_\Diamond)$ implies $\delta(s)$) and

$$(w, s) \models \tau(\Gamma', S_\triangle, S_\Box, S_\Diamond) \ . \tag{72}$$

This can also be rephrased as follows. Let $S' = \{s \mid (w, s) \in V^{\boldsymbol{W}}\}$. Then there exist functions $f\colon S' \to 2^{S'}$, $f_\Box\colon S' \to 2^S$, and $f_\Diamond\colon S' \to 2^S$ such that

$$\tau(\Gamma', f(s), f_\Box(s), f_\Diamond(s)) \models \delta(s) \tag{73}$$

and

$$(w, s) \models \tau(\Gamma', f(s), f_\Box(s), f_\Diamond(s)) \tag{74}$$

for every $s \in S'$.

4.2 Gadgets

Let S be an arbitrary set and $\widetilde{S}$ a disjoint copy. A *gadget* over S is a subset of $S \times (S \cup \widetilde{S})$ and should be viewed as a small directed graph on S with some edges leading to copies of elements from S. The set of all gadgets over S is denoted by $\mathrm{Gdg}(S)$.

For every gadget $B \in \mathrm{Gdg}(S)$, let

$$\widehat{B} = \{s \mid \exists s'(s, s') \in B\} \cup \{s' \mid \exists s(s, s') \in B\} \ , \tag{75}$$

$$\widetilde{B} = \{s' \mid \exists s(s, \widetilde{s'}) \in B\} \ . \tag{76}$$

Let g be a sequence of gadgets over S. Imagine you collate all gadgets in g in such a way that elements from $\widetilde{S}$ occurring in one gadget are identified with the respective elements of S in the next gadget. The you obtain the *graph of* g, denoted $\boldsymbol{G}(g)$, and formally defined as follows.

The vertex set of $\boldsymbol{G}(g)$ is given by

$$\bigcup_{i<|g|} \{(i, s) \mid s \in \widehat{g(i)}\} \cup \bigcup_{i<|g|} \{(i+1, s) \mid s \in \widetilde{g(i)}\} \ ; \tag{77}$$

the edge set is given by

$$\bigcup_{i<|g|} \{((i, s), (i, s')) \mid (s, s') \in g(i)\} \cup \bigcup_{i<|g|} \{((i, s), (i+1, s')) \mid (s, \widetilde{s'}) \in g(i)\} \ . \tag{78}$$

A *branch* of g is a maximum path through $\boldsymbol{G}(g)$ starting with $(0, s)$ for some $s \in S$. Let $\Omega\colon S \to \mathbf{N}$ be some priority function. The sequence g is *even* if $pr_1 \circ \pi$ is even with respect to Ω for every infinite branch π of g.

We show next that there exists an exponential size deterministic parity ω-automaton that accepts all even sequences of gadgets.

Proposition 1 *Let S be a finite set and $\Omega\colon S \to \mathbf{N}$ some priority function. There exists a deterministic parity ω-automaton $\boldsymbol{C}$ with $2^{\mathcal{O}(|S|^2 \log |S|)}$ states and priorities bounded by $\mathcal{O}(|S|^2)$ that recognizes the set of all even infinite sequences of gadgets over S.*

Proof. We first construct a nondeterministic ω-automaton $\boldsymbol{B}$ accepting the infinite sequences of gadgets that are not even. This automaton will then be transformed into the deterministic ω-automaton we are looking for.

Let B be some gadget. We write Δ_B for the set of all pairs (s, s') such that there exists a path from s to $\widetilde{s}'$ in $(S \cup \widetilde{S}, B)$. Further, we write S_B for set of all states $s \in S$ such that there exists an infinite path through the graph $(S \cup \widetilde{S}, B)$ which starts with s and is odd with respect to Ω.

Consider the nondeterministic parity ω-automaton

$$\boldsymbol{B} = (L^{\boldsymbol{E}}, S \cup \{\top\}, S, \Delta, \Omega^{\boldsymbol{B}})$$

where Δ and $\Omega^{\boldsymbol{B}}$ are given by

$$\begin{aligned}\Delta = &\{(s, B, s') \mid B \in \mathrm{Gdg}(S) \wedge (s, s') \in \Delta_B\} \\ &\cup \{(x, B, \top) \mid B \in \mathrm{Gdg}(S) \wedge x \in S_B \cup \{\top\}\}\end{aligned} \tag{79}$$

and

$$\Omega^{\boldsymbol{B}}(x) = \begin{cases} \Omega(x) + 1 & \text{if } x \in S, \\ 0 & \text{if } x = \top. \end{cases}$$

Clearly, this ω-automaton accepts the set of all infinite sequences of gadgets that are not even with respect to Ω.

The automaton $\boldsymbol{C}$ is constructed as follows.

1. $\boldsymbol{B}$ is converted into an equivalent nondeterministic Büchi automaton $\boldsymbol{B}_1$.
2. $\boldsymbol{B}_1$ is converted into an equivalent deterministic Rabin automaton $\boldsymbol{B}_2$.
3. $\boldsymbol{B}_2$ is transformed into the deterministic Streett automaton dual $\boldsymbol{B}_3$. It recognizes the complement of what $\boldsymbol{B}_2$ recognizes.
4. $\boldsymbol{B}_3$ is converted into an equivalent deterministic parity automaton $\boldsymbol{C}$.

Clearly, $\boldsymbol{C}$ recognizes the set of all even infinite sequences over $L^{\boldsymbol{E}}$.

Let n be the number of states of $\boldsymbol{B}$. The first step is simple and yields an automaton with $\mathcal{O}(n^2)$ states. The second step can be carried out using Safra's construction, [18], and thus yields an automaton with $2^{\mathcal{O}(n^2 \log n)}$ states and $\mathcal{O}(n^2)$ accepting pairs. The third step neither changes the number of states nor the number of pairs. The fourth step can be implemented using Büchi's index appearance record with hit, [20], and yields an automaton with a larger number of states but still with $2^{\mathcal{O}(n^2 \log n)}$ many states and priorities bounded by $\mathcal{O}(n^2)$. □

4.3 Parity Games with Extended Winning Condition

A parity game with an extended winning condition is very similar to an ordinary parity game. The only difference is that Player 1 can follow several options at the same time, and for Player 1 to win it is sufficient that one of the options she follows is winning. More precisely, a move by Player 1 corresponds to choosing a gadget and the sequence of moves of Player 1 will determine an (infinite) sequence of gadgets; Player 1 wins if one of the branches of the corresponding graph is not even.

4.3.1 Formal Definition

A *parity game with extended winning condition (an extended parity game)* over S is a tuple

$$\boldsymbol{E} = (S, L^{\boldsymbol{E}}, L_0^{\boldsymbol{E}}, L_1^{\boldsymbol{E}}, l_I^{\boldsymbol{E}}, M^{\boldsymbol{E}}, \Omega^{\boldsymbol{E}}) \tag{80}$$

where

- $L^{\boldsymbol{P}}$ is a set of *locations*,
- $L_0^{\boldsymbol{P}}$ and $L_1^{\boldsymbol{P}}$ partition $L^{\boldsymbol{P}}$ into *Player 0's and Player 1's locations*, respectively,
- $L_0^{\boldsymbol{E}}$ is a set of gadgets over S,
- $l_I^{\boldsymbol{P}} \in L_0^{\boldsymbol{P}} \cup L_1^{\boldsymbol{P}}$ is an *initial location*,
- $M^{\boldsymbol{P}} \subseteq (L_0^{\boldsymbol{P}} \cup L_1^{\boldsymbol{P}}) \times (L_0^{\boldsymbol{P}} \cup L_1^{\boldsymbol{P}})$ is a set of *moves*, and
- $\Omega^{\boldsymbol{P}} \colon S \to \mathbf{N}$ is a partial *priority function* assigning to some locations a priority.

The domain of $\Omega^{\boldsymbol{E}}$ must have the property that $\inf(\pi) \cap \mathrm{dom}(\Omega^{\boldsymbol{E}})$ is nonempty for any branch π of any sequence of gadgets from $L_0^{\boldsymbol{E}}$.

The notion of a play is the same as in ordinary parity games. Player 0 wins a play p if the subsequence of p consisting of the elements of $L_0^{\boldsymbol{E}}$ is even with respect to $\Omega^{\boldsymbol{E}}$.

4.3.2 Reduction to Ordinary Parity Games

We show how the winner in an extended parity game can be determined by looking at a related parity game. The idea of the reduction is very simple. Since Player 0 only wins a play if all infinite branches through the corresponding graph are winning he uses a deterministic ω-automaton to keep track of all those branches. In fact, he uses the automaton that we already know from Proposition 1.

Let $\boldsymbol{E}$ be an extended parity game and $\Omega \colon S \to \mathbf{N}$ defined by

$$\Omega(s) = \begin{cases} \Omega^{\boldsymbol{E}}(s) & \text{if } s \in \mathrm{dom}(\Omega^{\boldsymbol{E}}), \\ 0 & \text{otherwise.} \end{cases} \tag{81}$$

Assume that $\boldsymbol{C}$ from Proposition 1 is given as

$$\boldsymbol{C} = (L^{\boldsymbol{E}}, Q, q_I, \delta, \Omega^{\boldsymbol{C}}) \ .$$

Using $\boldsymbol{C}$, we can now easily transfer a finite extended parity game $\boldsymbol{E}$ into an ordinary parity game $\boldsymbol{P}(\boldsymbol{E})$:

$$\boldsymbol{P}(\boldsymbol{E}) = (L^{\boldsymbol{E}} \times Q, L_0^{\boldsymbol{E}} \times Q, L_1^{\boldsymbol{E}} \times Q, (l_I^{\boldsymbol{E}}, q_I), M, \Omega) \tag{82}$$

where

$$\begin{aligned} M = \{((l,q),(l',\delta(q,l))) \mid (l,l') \in M \wedge l \in L_0^{\boldsymbol{E}}\} \\ \cup \{((l,q),(l',q)) \mid (l,l') \in M \wedge l \in L_1^{\boldsymbol{E}}\} \end{aligned} \tag{83}$$

and $\Omega\colon (l,q) \mapsto \Omega^{\boldsymbol{C}}(q)$.

Theorem 8 *Let $\boldsymbol{E}$ be a finite extended parity game.*

1. *The number of locations of $\boldsymbol{P}(\boldsymbol{E})$ is $|L^{\boldsymbol{E}}| \times 2^{\mathcal{O}(n^2 \log n)}$ where $n = |S^{\boldsymbol{E}}|$.*
2. *The priority function of $\boldsymbol{P}(\boldsymbol{E})$ is bounded by cn^2 for some constant c.*
3. *Player 0 wins $\boldsymbol{E}$ if and only if Player 0 wins $\boldsymbol{P}(\boldsymbol{E})$.*

Proof. The first two claims follow immediately from Proposition 1.

For the third claim first observe that there is a one-to-one correspondence between minimal strategy trees for $\boldsymbol{E}$ and $\boldsymbol{P}(\boldsymbol{E})$: a strategy tree for $\boldsymbol{E}$ can be obtained from a strategy free for $\boldsymbol{P}(\boldsymbol{E})$ by forgetting the second component of labellings of the vertices; given a strategy tree for $\boldsymbol{E}$, there is exactly one way to extend it to a strategy tree for $\boldsymbol{P}(\boldsymbol{E})$ by adding appropriate states from $\boldsymbol{C}$ to the labellings of the vertices.

The construction of $\boldsymbol{C}$ now guarantees that the correspondence between strategy trees for $\boldsymbol{E}$ and $\boldsymbol{P}(\boldsymbol{E})$ preserves winning branches. Observe that Ω was defined in such a way that the elements from S that don't get assigned a value through $\Omega^{\boldsymbol{E}}$ cannot interfere, see (81). □

4.4 Reduction of the Emptiness Problem

We can now solve the emptiness problem for alternating tree automata by reducing it to the winner problem for extended parity games. Given an alternating tree automaton $\boldsymbol{A}$, we will construct an extended parity game $\boldsymbol{G}(\boldsymbol{A})$ that Player 0 wins if and only if $\boldsymbol{A}$ accepts some pointed Kripke structure.

The idea behind the construction described below is that Player 0 guesses an accepting witness for some Kripke structure and Player 1 checks that all branches are accepting.

Let $\boldsymbol{A}$ be an arbitrary alternating tree automaton and Γ the set of propositional variables occurring in $\delta^{\boldsymbol{A}}$. The parity game $\boldsymbol{E}(\boldsymbol{A})$ with extended winning condition is defined by

$$\boldsymbol{E}(\boldsymbol{A}) = (S, L, L_0, L_1, l_I, M, \Omega^{\boldsymbol{A}})$$

where L_1 consists of tuples of the form

$$(\Gamma', S', f, f_\Box, f_\Diamond) \tag{84}$$

as in Subsection 4.1 such that $\tau(\Gamma', f(s), f_\Box(s), f_\Diamond(s)) \models \delta(s)$ for every $s \in S'$.

The initial location l_I is defined by

$$l_I = \{(s_I, \widetilde{s_I})\} \ . \tag{85}$$

The set M contains a pair $((\Gamma', S', f, f_\Box, f_\Diamond), B)$ if there exists $s \in S'$ and $s'' \in f_\Diamond(s)$ such that

$$\begin{aligned} B = \{(s, \widetilde{s'}) \mid s \in S' \wedge s' \in f_\Box(s)\} \\ \cup \{(s, s') \mid s \in S' \wedge s' \in f(s)\} \cup \{(s, \widetilde{s''})\} \ . \end{aligned} \tag{86}$$

Further, M contains $(B, (\Gamma', S', f, f_\Box, f_\Diamond))$ if $\widetilde{B} \subseteq S'$.

Theorem 9 *Let $\boldsymbol{A}$ be an arbitrary alternating tree automaton.*

1. *The number of locations of $\boldsymbol{E}(\boldsymbol{A})$ is $2^d \times 2^{\mathcal{O}(n \log n)}$ where d is the number of propositional variables occurring in δ and $n = |S^{\boldsymbol{A}}|$.*
2. *The maximum priority of $\boldsymbol{E}(\boldsymbol{A})$ is the same as in $\boldsymbol{A}$.*
3. *Player 0 wins the extended parity game $\boldsymbol{E}(\boldsymbol{A})$ if and only if $\boldsymbol{A}$ accepts some Kripke structure.*

Proof. The first two claims are trivial. For the third claim, let $\boldsymbol{E} = \boldsymbol{E}(\boldsymbol{A})$.

Let $\boldsymbol{T}$ be a minimal winning strategy tree for Player 0 in $\boldsymbol{E}$. We construct a pointed Kripke structure $(\boldsymbol{K}, w_I)$ and accepting witness for $\boldsymbol{A}$ and $(\boldsymbol{K}, w_I)$.

Observe that because of the minimality of $\boldsymbol{T}$, every vertex of $\boldsymbol{T}$ labelled with an element from L_0 has exactly one successor, that is, there exists exactly one edge starting from such a vertex. The Kripke structure $\boldsymbol{K}$ is obtained from $\boldsymbol{T}$ by simply contracting these edges. Formally, the worlds and the accessibility relation of $\boldsymbol{K}$ are defined by

$$W^{\boldsymbol{K}} = \{v \in V^{\boldsymbol{T}} \mid \lambda^{\boldsymbol{T}}(v) \in L_1\} \ , \tag{87}$$

$$A^{\boldsymbol{K}} = \{(v, v'') \mid \exists v' \in \mathrm{Scs}_{\boldsymbol{T}}(v)(v'' \in \mathrm{Scs}_{\boldsymbol{T}}(v'))\} \ . \tag{88}$$

The labelling of $\boldsymbol{K}$ is defined as follows. If $v \in W^{\boldsymbol{K}}$ (as defined above) and if $\lambda^{\boldsymbol{T}}(v) = (\Gamma', S', f, f_\Box, f_\Diamond)$, then $\kappa^{\boldsymbol{K}}(v) = \Gamma'$. The distinguished world w_I of $(\boldsymbol{K}, w_I)$ is the unique successor of the root of $\boldsymbol{T}$.

An accepting witness $\boldsymbol{W}$ for $(\boldsymbol{K}, w_I)$ can be extracted directly from $\boldsymbol{T}$. First, a pair (v, s) belongs to $V^{\boldsymbol{W}}$ if $\lambda^{\boldsymbol{T}}(v) = (\Gamma', S', f, f_\Box, f_\Diamond)$ and $s \in S'$. Second, the edges of $\boldsymbol{W}$ are determined as follows. Let v, v', and v'' be vertices of $\boldsymbol{T}$ such that $v, v'' \in V^{\boldsymbol{W}}$, $v' \in \mathrm{Scs}_{\boldsymbol{T}}(v)$ and $v'' \in \mathrm{Scs}_{\boldsymbol{T}}(v')$. Assume $\lambda^{\boldsymbol{T}}(v) = (\Gamma', S', f, f_\Box, f_\Diamond)$ and $\lambda^{\boldsymbol{T}}(v') = B$.

- For every $(s, s') \in B$, $E^{\boldsymbol{W}}$ contains $((v, s), (v, s'))$.
- For every $(s, \widetilde{s}') \in B$, $E^{\boldsymbol{W}}$ contains $((v, s), (v'', s'))$.

Clearly, $\boldsymbol{W}$ is an accepting witness for $\boldsymbol{A}$ and $(\boldsymbol{K}, w_I)$.

For the converse, assume $\boldsymbol{A}$ accepts some pointed Kripke structure. Then $\boldsymbol{A}$ accepts some pointed Kripke structure which is a tree. (Unravel the Kripke structure that it accepts.) Let $(\boldsymbol{K}, w_I)$ be such a Kripke structure and $\boldsymbol{W}$ an accepting witness for $\boldsymbol{A}$ and $\boldsymbol{K}$. We will use $\boldsymbol{W}$ to construct a strategy tree $\boldsymbol{T}$ winning for Player 0 in the game $\boldsymbol{E}(\boldsymbol{A})$.

The construction of $\boldsymbol{T}$ goes by induction. Along with $\boldsymbol{T}$ we define a partial function $\beta \colon V^{\boldsymbol{T}} \to V^{\boldsymbol{W}}$ that associates with every vertex $v \in L_0^{\boldsymbol{E}}$ a vertex of $\boldsymbol{W}$.

For the induction base, choose some vertex v_I as the root of $\boldsymbol{T}$. Add a child v to $\boldsymbol{T}$ and set $\beta(v) = w_I$. Recall that (w_I, s_I) must be a vertex of $\boldsymbol{W}$.

For the induction, assume v is a vertex already added to $\boldsymbol{T}$ which has not been labeled yet. Let $w = \beta(v)$, $\Gamma' = \kappa^{\boldsymbol{K}}(w)$, and $S' = \{s \mid (w, s) \in V^{\boldsymbol{W}}\}$. Since $\boldsymbol{W}$ is a witness, there exist f, $f_\Box$, and $f_\Diamond$ such that

$$\tau(\Gamma', f(s), f_\Box(s), f_\Diamond(s)) \models \delta(s) \tag{89}$$

and

$$(w, s) \models \tau(\Gamma', f(s), f_\Box(s), f_\Diamond(s)) \tag{90}$$

for every $s \in S'$. We label v with $(\Gamma', S', f, f_\Box, f_\Diamond)$ and add children to v according to the rules of the game, that is, we add a child v' with label B if there exists $s \in S'$ and $s'' \in f_\Diamond(s)$ such that (86) holds. To each such vertex v', we add exactly one child v'.

Let v' and v'' be two such vertices. Since $\boldsymbol{W}$ is an accepting witness there exists a world $w' \in \boldsymbol{K}$ such that B is a subgraph of the subgraph of $\boldsymbol{W}$ restricted to all vertices of the form (w, s) or (w', s). We set $\beta(v'') = w'$. This completes the description of the inductive step.

Clearly, $\boldsymbol{T}$ is a strategy tree for Player 0. Further, every vertex labelled with a location of Player 0 has a successor. If p is a path through $\boldsymbol{T}$ and π an infinite branch through $\boldsymbol{G}(p)$, then β determines a path π' through $\boldsymbol{W}$ with the same state

labelling. Since $\boldsymbol{W}$ is an accepting witness, we know π' is even, and, hence, π is even. So $\boldsymbol{T}$ is a strategy tree winning for Player 0. □

Corollary 2 [4]

1. NONEMPTINESS, *the nonemptiness problem for alternating tree automata, is in* ***Exp****.*
2. SATISFIABILITY, *the satisfiability problem for modal μ-calculus, is in* ***Exp****.*

Proof. From Theorems 9 and 8 we can conclude that for every alternating tree automaton $\boldsymbol{A}$ one can construct a parity game $\boldsymbol{P}$ with the following properties.

- The number of locations of $\boldsymbol{P}$ is $2^d \times 2^{\mathcal{O}(n^2 \log n)}$ where d is the number of propositional variables occurring in $\delta^{\boldsymbol{A}}$ and $n = |S^{\boldsymbol{A}}|$.
- The priority function of $\boldsymbol{P}$ is bounded by cn^2 for some constant c.
- Player 0 wins $\boldsymbol{P}$ if and only if $\boldsymbol{A}$ accepts some pointed Kripke structure.

Further, $\boldsymbol{P}$ can easily be constructed, that is, in time polynomial in its size. The first claim now follows from Theorem 3. The second claim is an immediate consequence of the first claim in view of Theorem 1. □

References

[1] Julian C. Bradfield. The modal mu-calculus alternation hierarchy is strict. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96: Concurrency Theory, 7th International Conference*, volume 1119 of *LNCS*, pages 232–246, Pisa, Italy, 1996.

[2] Julian C. Bradfield. The modal mu-calculus alternation hierarchy is strict. *Theoretical Computer Science*, 195(2):133–153, 1998.

[3] Julian C. Bradfield. Simplifying the modal mu-calculus alternation hierarchy. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *STACS '98: 15th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1373 of *LNCS*, pages 39–49, Paris, France, 1998.

[4] E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs (extended abstract). In *29th Annual Symposium on Foundations of Computer Science*, pages 328–337, White Plains, New York, 1988.

[5] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy. In *32nd Annual Symposium on Foundations of Computer Science*, pages 368–377, San Juan, Puerto Rico, 1991.

[6] E. Allen Emerson and Chin-Laung Lei. Efficient model checking in fragments of the propositional mu-calculus (extended abstract). In *1st IEEE Symposium on Symposium on Logic in Computer Science*, pages 267–278, Cambridge, Massachusetts, 1986.

[7] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *14th ACM Symposium on the Theory of Computing*, pages 60–65, San Francisco, 1982.

[8] Marcin Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Information Processing Letters*, 68(3):119–124, 1998.

[9] Marcin Jurdziński. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301, Lille, France, 2000.

[10] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[11] Giacomo Lenzi. A hierarchy theorem for the μ-calculus. In F. Meyer auf der Heide and B. Monien, editors, *Automata, Languages and Programming: 23rd Intern. Colloquium, ICALP '96*, volume 1099 of *LNCS*, pages 87–97, Paderborn, Germany, 1996.

[12] David E. Long, Anca Browne, Edmund M. Clarke, Somesh Jha, and Wilfredo R. Marrero. An improved algorithm for the evaluation of fixpoint expressions. In David L. Dill, editor, *Computer Aided Verification, 6th International Conference, CAV '94*, volume 818 of *Lecture Notes in Computer Science*, pages 338–350, Stanford, California, 1994.

[13] Kenneth L. McMillan. *Symbolic Model Checking*. Kluwer, Boston, 1993.

[14] Damian Niwiński. On fixed point clones. In Laurent Kott, editor, *Automata, Languages and Programming: 13th International Colloquium*, volume 226 of *Lecture Notes in Computer Science*, pages 464–473, Rennes, France, 1986.

[15] Damian Niwiński. Fixed point characterization of infinite behavior of finite-state systems. *Theoretical Computer Science*, 189:1–69, 1997.

[16] Damian Niwiński and Helmut Seidl. On distributive fixed-point expressions. *RAIRO Informatique Théorique*, 33(4/5):427–446, 1999.

[17] Michael Ozer Rabin. Decidability of second-order theories and finite automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.

[18] Shmuel Safra. On the complexity of ω-automata. In *29th Annual Symposium on Foundations of Computer Science*, pages 319–327, White Plains, New York, 1988.

[19] Helmut Seidl. Fast and simple nested fixpoints. *Information Processing Letters*, 59(6):303–308, 1996.

[20] W. Thomas. Languages, automata and logic. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, Beyond Words, pages 389–455. Springer-Verlag, Berlin, 1997.

[21] Moshe Y. Vardi. Reasoning about the past with two-way automata. In Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel, editors, *Automata, Languages and Programming: 25th International Colloquium*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641, Aalborg, Denmark, 1998.