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

Thomas Wilke

Abstract

A coherent exposition of the connection of alternating tree automata and modal μ-calculus is given, advocating an automaton model specifically tailored for working with modal μ-calculus. The advantage of the automaton model proposed is that it can deal with arbitrary branching in a very natural way. It is really equivalent to the modal μ-calculus with respect to expressive power, just as the one proposed by Janin and Walukiewicz, but simpler. The main focus is on the model checking and the satisfiability problem for μ-calculus. Both problems are solved by reductions to corresponding problems on alternating tree automata, namely to the acceptance and the (non-)emptiness problem, respectively. These problems, in turn, are solved using parity games.

Résumé

On donne une présentation cohérente du lien entre les automates d'arbres alternants et le *mu*-calcul modal, grâce à un modèle d'automate spécialement adapté au *mu*-calcul. L'avantage du modèle d'automate proposé est qu'il peut prendre en compte des branchements d'ordre arbitraire de manière très naturelle. Il a un pouvoir d'expression équivalent à celui du *mu*-calcul, tout comme celui proposé par Janin et Walukiewicz. mais il est plus simple. L'accent est principalement mis sur la vérification et le problème de la satisfiabilité du *mu*-calcul. Ces deux problèmes sont résolus par réduction aux problèmes correspondants sur les automates d'arbres alternants, à savoir l'acceptance et le problème du vide respectivement. Ces problèmes sont à leur tour résolus en utilisant des jeux à parié.

1 Introduction

Since Dexter Kozen's seminal paper in 1983, [12], modal μ-calculus has received ever growing interest, mainly for two reasons: (1) its mathematical theory (model theory) is very rich; (2) modal μ-calculus is well suited for specifying properties of transition systems. The interest was further stimulated by Ken McMillan's observation, [15], that specifications in modal μ-calculus can be checked efficiently for finite transition systems 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.

Modal μ-calculus is best investigated using an automata-theoretic approach. In fact, most of the deep results on modal μ-calculus (such as the completeness of Kozen's axiom scheme, [12, 26, 27], the exact complexity of the satisfiability problem, [6], and the strictness of the fixed point alternation hierarchy, [1, 2, 3, 4, 13]) have been (can be) obtained using automata theory.

The full beauty of the connection between modal μ-calculus and (alternating) tree automata was first revealed in the fundamental study [19] by Damian Niwiński. While in [19], 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. A coherent exposition of the connection of alternating tree automata and modal μ-calculus is given, advocating an automaton model specifically tailored for working with modal μ-calculus. The advantage of the automaton model proposed is that it can deal with arbitrary branching in a very natural way. It is really equivalent to the modal μ-calculus with respect to expressive power, just as the one used in [9], but simpler.

The main focus is on the model checking and the satisfiability problem. The approach is modular in that the two problems are first reduced to corresponding problems on alternating tree automata, the acceptance and the nonemptiness problem, respectively. Then these two problems are in turn reduced to the winner problem for parity games. The reduction from the acceptance problem is straightforward while the reduction from the nonemptiness problem is quite involved and makes use of Safra's fundamental determinization result, [22, 17].

This paper resulted from an invited talk given at the Journées Montoises, Marne-la-Vallée, March 2000. It has improved a lot through numerous discussions I had with the participants of the GI Dagstuhl workshop on Automata, Logic and Infinite Games, held in Wadern in February, 2001.

Ordinals. Von Neumann's convention on ordinals is used, in particular, when n denotes a natural number, then $n = \{0, \dots, n-1\}$. Following this convention, the set of natural numbers is denoted by ω.

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 $f\colon M \to \omega$ be any (possibly partial) function with an infinite domain and a finite image. Then the set of all elements occurring infinitely often in the image, the set given by $\{n \in N \mid |f^{-1}(n)| = \infty\}$, is a nonempty finite set of natural numbers and we denote its maximum by $\sup(f)$. Often, we will consider functions f that result from composing several other functions; it will be convenient to write gh for the composition of g and h, defined by $gh(x) = h(g(x))$.

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. The root of a tree $\boldsymbol{T}$ will be denoted by $\rho^{\boldsymbol{T}}$.

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

2 From Modal μ-Calculus to Alternating Tree Automata

Our first goal is to understand the semantics of modal μ-calculus in terms of alternating tree automata. We will 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 and a correctness proof of this conversion.

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 multi-modal case. An extension to backward modalities as treated in [25] is not too 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

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

where

- W, the *universe* of $\boldsymbol{K}$, is a set of *worlds,*
- $A \subseteq W \times W$ is an *accessibility relation,* and
- $\kappa \colon Q \to 2^W$ 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 $\boldsymbol{K}$ 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 an 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.

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 (that is, $\neg q$ does not occur), 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 $\mathrm{free}(\varphi)$.

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

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

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 set is partitioned into two sets according to their prefix: 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:

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

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

The definition of the semantics of the fixed point operators 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 $\kappa^{\boldsymbol{K}}[q \mapsto W]$ itself 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.

We also have 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 $\{W_\lambda\}_{\lambda \in \mathrm{On}}$ 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, where approximation starts with $W_0 = W^{\boldsymbol{K}}$.

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

Example 1. Consider the following simple formula, $\varphi_0 = \mu q_0(\Box q_0)$. In every Kripke structure $\boldsymbol{K}$, $||\varphi_0||_{\boldsymbol{K}}$ is the set of all worlds of $\boldsymbol{K}$ where no infinite path starts.

Next, consider the formula $\varphi_1 = \mu q_1(q_0 \vee \Diamond q_1)$. For every Kripke structure $\boldsymbol{K}$, $||\varphi_1||_{\boldsymbol{K}}$ is the set of all worlds of $\boldsymbol{K}$ from which, in a finite number of steps, possible 0, a world is reachable where q_0 holds. Similarly, $\varphi_1' = \Diamond \varphi_1$ defines the property that in a finite number of steps, but at least 1, a world is reachable where q_0 holds. Thus, $\varphi_2 = \nu q_2(\varphi_1'[q_2 \wedge q_0/q_0])$ defines the property that there exists a path where at infinitely many positions q_0 holds.

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 $||\varphi||$. For every connective or operator of L_μ, we will define a corresponding operation on Kripke queries.

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,

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

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,

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

for every L_μ formula φ.

Finally, we consider 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

Besides its length, the most important characteristic of a modal μ-calculus formula is its fixed point alternation depth, that is, the number of alternations between least and greatest fixed point operators. 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, [7]. It gives rise to a coarser hierarchy. The one we use here defines an even coarser hierarchy and was introduced by Damian Niwiński, [18]. His definition turned out to be the most useful one in the sense that it yields the best complexity bounds.

We denote the relation "is proper subformula of" by $<$. Let φ be an L_μ formula. An *alternating μ-chain* in φ of length l is a sequence

$$\varphi \geq \mu q_0 \psi_0 > \nu q_1 \psi_1 > \mu q_2 \psi_2 > \cdots > \mu/\nu q_{l-1} \psi_{l-1} \tag{25}$$

where, for every $i < l-1$, the variable q_i occurs free in every formula ψ with $\psi_i \geq \psi \geq \psi_{i+1}$. The maximum length of an alternating μ-chain in φ is denoted by $m^\mu(\varphi)$. Symmetrically, ν-chains and $m^\nu(\varphi)$ are defined.

The *alternation depth* of an L_μ formula φ is the maximum of $m^\mu(\varphi)$ and $m^\nu(\varphi)$ and is denoted by $\alpha(\varphi)$.

Example 2. Let φ_0, φ_1, φ_1', and φ_2 be the L_μ formulas from Example 1. Then $\alpha(\varphi_0) = \alpha(\varphi_1) = \alpha(\varphi_1') = 1$. On the other hand, $\alpha(\varphi_2) = 2$.

A more interesting example is the formula $\varphi_3 = \mu q_1(\nu q_1(q_0 \wedge \Diamond q_1) \vee \Box q_1)$. This formula defines the property that on all paths eventually a world is reached where an infinite path starts on which q_0 holds generally. The alternation depth of φ_3 is 1.

Alternatively, the alternation depth of a formula can be defined using the notion of the graph of an L_μ formula.

The *graph* of an L_μ formula φ, denoted $\boldsymbol{G}(\varphi)$, is the directed graph whose vertices are the subformulas of φ and where the edges starting in a vertex ψ are determined as follows.

- If $\psi = \bot$, $\psi = \top$, $\psi = \neg q$, or $\psi = q$ and the occurrence of q is free in φ, then ψ has no outgoing edge.
- If $\psi = q$ and the occurrence of q is bound in ψ, say $\eta q \chi$ binds q, then ψ has an edge to $\eta q \chi$.
- If $\psi = \chi \vee \chi'$ or $\psi = \chi \wedge \chi'$, then ψ has an edge to χ and to χ'.
- If $\psi = \Diamond \chi$, $\psi = \Box \chi$, $\psi = \mu q \chi$, or $\psi = \mu q \chi$, then ψ has an edge to χ.

Strictly speaking, one has to distinguish between a subformula and its occurrence. In the definition of $\boldsymbol{G}(\varphi)$, we always mean the latter. This makes no difference when we assume φ is in normal form as explained in Subsection 2.2.2.

The alternation depth of a formula φ can now be determined as follows. If no variable is bound in φ, then $\alpha(\varphi) = 0$. If $\varphi = \eta q \psi$ with q free in ψ and ψ without fixed point operators, then $\alpha(\varphi) = 1$. In all other cases, suppose the alternation depth of all proper subformulas of φ has already been determined. Let M be the maximum of all these values. If φ is a fixed point formula of the form $\eta q \psi$ and there exists a fixed point formula $\eta' q' \psi'$ of alternation depth M in the strongly connected component of φ in $\boldsymbol{G}(\varphi)$ and $\eta \neq \eta'$ holds, then $\alpha(\varphi) = M + 1$. In all other cases, $\alpha(\varphi) = M$.

This also shows:

Remark 1. *Let φ be an L_μ formula and Ψ a strongly connected subset of $\boldsymbol{G}(\varphi)$. Then the following is true.*

1. *There exists a fixed point formula $\psi = \eta q \chi$ of alternation depth > 0 which is maximal in Ψ (w. r. t. the subformula order).*
2. *Let $\psi' = \eta' q' \chi'$ be any other fixed point formula in Ψ. Then $\alpha(\psi) \geq \alpha(\psi')$ and, moreover, the inequality is strict if $\eta \neq \eta'$.*

2.2 Alternating Tree Automata

Alternating tree automata are used to define Kripke queries, and we will later see that they can define every Kripke query which is definable by a modal μ-calculus formula. The converse is true as well, but will not be proved in this paper. For details, see [19].

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. Namely, every state will be assigned a priority and an infinite branch of the computation will be accepting if the maximum priority occurring infinitely often is even; a computation tree will be accepting if each of its infinite branches is accepting.

Example 3. A perfect example is the following automaton, $\boldsymbol{A_0}$, which accepts a pointed Kripke structure $(\boldsymbol{K}, w_I)$ if all paths starting in w_I are finite. The automaton has only one state, s, which is assigned priority 1. So, according to what was said above, it allows no infinite computation paths. The transition function is defined by $\delta(s) = \Box s$, which determines that if a copy of the automaton reaches a world w (in state s), then this copy splits up and every successor of w gets a copy in state s. Clearly, the only situation where no infinite computation paths emerge is when there is no infinite path starting in w_I. In other words, this automaton is equivalent to the formula φ_0 from Example 1.

Next, we construct an alternating tree automaton $\boldsymbol{A_1}$ which accepts a pointed Kripke structure $(\boldsymbol{K}, w_I)$ if from w_I a world is reachable where q_0 holds. The only difference with $\boldsymbol{A_0}$ is that the transition function is defined by $\delta(s) = q_0 \vee \Diamond s$. This is read as follows. If a copy of the automaton is in some world w (in state s), the automaton may stop if q_0 holds or a copy of the automaton is sent to a successor and goes into state s. Since s is assigned priority 1 and we require that on every infinite computation path the maximum priority occurring infinitely often be even, no infinite computation path is allowed. In other words, $\boldsymbol{A_1}$ is equivalent to φ_1 from Example 1.

Finally, we construct an automaton $\boldsymbol{A_2}$, which is equivalent to formula φ_2 from Example 1. The automaton $\boldsymbol{A_2}$ has two states, s and $\bar{s}$, where s has priority 2 and $\bar{s}$ has priority 1. Then an infinite branch of the computation is accepted if s occurs infinitely often. The transition function of $\boldsymbol{A_2}$ is designed in such a way that it simply propagates whether q_0 holds or not in the current world to one of the successors, that is, the transition function is given by $\delta(s) = \delta(\bar{s}) = (q_0 \wedge \Diamond s) \vee (\neg q_0 \wedge \Diamond \bar{s})$. Clearly, this automaton works.

2.2.2 Formal Definition

Formally, an *alternating tree automaton* is a tuple

$$\boldsymbol{A} = (S, s_I, \delta, \Omega) \tag{26}$$

where

- S is a finite set of *states,*
- $s_I \in S$ is an *initial state,*
- δ is a *transition function* as specified below, and
- $\Omega \colon S \to \omega$ is a *priority function,* which assigns a *priority* to each state.

The transition function δ maps every state to a transition condition over S where the set of all *transition conditions* over S is defined by:

- 0 and 1 are transition conditions over S,
- q and $\neg q$ are transition conditions over S, for every $q \in Q$,
- s, $\Box s$, and $\Diamond s$ are transition conditions over S, for every $s \in S$,
- $s \wedge s'$ and $s \vee s'$ are transition conditions over S, for $s, s' \in S$.

Example 4. Clearly, automaton $\boldsymbol{A_0}$ from Example 3 is an alternating tree automaton in the above sense. Strictly speaking, this is not true for the two other automata from Example 3, for their transition conditions are too complicated. By introducing additional states, however, this can be taken care of.

For instance, to turn $\boldsymbol{A_1}$ into an equivalent alternating tree automaton according to the above definition one would add new states s_l and s_r with priority 1 and define the transition function by $\delta(s) = s_l \vee s_r$, $\delta(s_l) = q_0$, and $\delta(s_r) = \Diamond s$.

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

$$\boldsymbol{R} = (V^{\boldsymbol{R}}, E^{\boldsymbol{R}}, \lambda^{\boldsymbol{R}}) \tag{27}$$

such that $\rho^{\boldsymbol{R}}$ is labelled (w_I, s_I) and for every vertex v with label (w, s) the following conditions are satisfied.

- $\delta(s) \neq 0$.
- If $\delta(s) = q$, then $w \in \kappa^{\boldsymbol{K}}(q)$, and if $\delta(s) = \neg q$, then $w \notin \kappa^{\boldsymbol{K}}(q)$.
- If $\delta(s) = s'$, then there exists $v' \in \mathrm{Scs}_{\boldsymbol{R}}(v)$ such that $\lambda(v') = (w, s')$.
- If $\delta(s) = \Diamond s'$, then there exists $v' \in \mathrm{Scs}_{\boldsymbol{R}}(v)$ such that $s^{\boldsymbol{R}}(v') = s'$ and $w^{\boldsymbol{R}}(v') \in \mathrm{Scs}_{\boldsymbol{K}}(w)$.
- If $\delta(s) = \Box s'$, then for every $w' \in \mathrm{Scs}_{\boldsymbol{K}}(w)$ there exists $v' \in \mathrm{Scs}_{\boldsymbol{R}}(v)$ such that $\lambda(v') = (w', s')$.
- If $\delta(s) = s' \vee s''$, then there exists $v' \in \mathrm{Scs}_{\boldsymbol{R}}(v)$ such that $\lambda(v') = (w, s')$ or $\lambda(v') = (w, s'')$.
- If $\delta(s) = s' \wedge s''$, then there exist $v', v'' \in \mathrm{Scs}_{\boldsymbol{R}}(v)$ such that $\lambda(v') = (w, s')$ and $\lambda(v'') = (w, s'')$.

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

An infinite branch π of $\boldsymbol{R}$ is *accepting* if $\sup(\pi\lambda\pi_1\Omega)$ is even. (Recall that $\lambda\pi_1$ denotes the the state component of the labeling.) The 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 labelings of the vertices and applying the priority function Ω; we require that the maximum natural number occurring infinitely often is even.

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 $||A||$, contains all pointed Kripke structures accepted by $\boldsymbol{A}$. An L_μ formula is *equivalent* to an alternating tree automaton if the formulas defines the query that the automaton recognizes.

2.2.4 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. 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 $\delta(s)$. Let $\mathfrak{C}^{\boldsymbol{A}}$ be the set of all strongly connected components of the transition graph of $\boldsymbol{A}$. For every $C \in \mathfrak{C}^{\boldsymbol{A}}$, let

$$m_C^{\boldsymbol{A}} = |\{\Omega^{\boldsymbol{A}}(s) \mid s \in C\}| \tag{28}$$

denote the number of priorities used in C. The *index* of $\boldsymbol{A}$, denoted $\operatorname{ind}(\boldsymbol{A})$, is the maximum of all these values, that is,

$$\operatorname{ind}(\boldsymbol{A}) = \max(\{m_C^{\boldsymbol{A}} \mid C \in \mathfrak{C}^{\boldsymbol{A}}\} \cup \{0\}) \ . \tag{29}$$

2.2.5 Partial Priority Functions and Complex Transition Conditions

From now on we will also allow partial priority functions with alternating tree automata. The only thing we require of an alternating tree automaton $\boldsymbol{A}$ with a partial priority function is that on every infinite path through $\boldsymbol{G}(\boldsymbol{A})$ there must be a state with a priority assigned. This also implies that on every infinite path a state with a priority assigned occurs infinitely often.

Equivalently, the restriction of $\boldsymbol{G}(\boldsymbol{A})$ to the states with no priority must not have a strongly connected component. By convention, a partial priority function is extended to a total priority function by assigning to each state with no priority the minimum priority assigned in the strongly connected component of $\boldsymbol{G}(\boldsymbol{A})$ the state belongs to. Further, all states that do not belong to a strongly connected component get priority 0. Note that this does not change the index of the automaton.

Acceptance for automata with partial priority functions can equivalently be defined as follows. A run is accepting if on every infinite branch infinitely many states with a priority occur and the maximum priority occurring infinitely often is even.

The transition conditions defined above are simpler than the ones used in Example 1. The more restrictive definition of transition condition makes most of the proofs easier and does not go along with a loss in expressiveness. In some situations however, for instance, if one wants to show that a certain Kripke query is recognizable by an alternating tree automaton, one would want to use a richer syntax.

The most general option is to allow any modal formula over Q and S as a transition condition. (An adaption of the semantics is straightforward.) It is in fact easy to see that even such a rich syntax does not lead to a more powerful automaton model: every such automaton can be turned into an automaton according to our definition by introducing additional states, namely a new state for every subformula of a complex transition condition, and adapting the transition condition appropriately, see Example 4.

So, henceforth, we will—without loss of generality—allow complex transition conditions.

2.3 The Translation

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 Informal Description

Let φ be an arbitrary L_μ formula. An alternating tree automaton can check that a given pointed Kripke structure $(\boldsymbol{K}, w)$ satisfies φ as follows. It simply unwinds the formula, that is, the automaton will have a state $\langle\psi\rangle$ for each subformula ψ of φ, and the transition condition for a state reflects the outermost connective of the corresponding subformula. For instance, if $\psi = \chi_0 \wedge \chi_1$, then $\delta(\langle\psi\rangle) = \langle\chi_0\rangle \wedge \langle\chi_1\rangle$. The most interesting case is when $\psi = q$. If the occurrence of q in φ is free, then $\delta(\langle q\rangle) = q$, that is, the automaton simply checks if q holds in the current world. But if the occurrence is bound, then it is bound by a fixed point operator, say χ is the corresponding subformula, and the automaton will expand the variable q in that it replaces state $\langle q\rangle$ by $\langle\chi\rangle$. The intuition behind this is that it starts a recursion on χ — it expands the fixed point.

Clearly, least and greatest fixed points have to be dealt with differently. For a greatest fixed point it is ok if it is expanded infinitely often, but for a least fixed point only a finite number of expansions should be allowed. If fixed point subformulas are nested and expanded over and over again, the outermost fixed point is the most relevant one. To account for this, the priority function is chosen appropriately.

Example 5. Consider φ_1 from Example 1 again. The automaton corresponding to φ_1 will have the following states: $\langle\mu q_1(q_0 \vee \Diamond q_1)\rangle$, $\langle q_0 \vee \Diamond q_1\rangle$, $\langle q_0\rangle$, $\langle\Diamond q_1\rangle$, and $\langle q_1\rangle$. The initial state is $\langle\mu q_1(q_0 \vee \Diamond q_1)\rangle$ and all states get priority 0 except for $\langle\mu q_1(q_0 \vee \Diamond q_1)\rangle$, which gets priority 1. The transition function is given by

$$\delta(\langle\mu q_1(q_0 \vee \Diamond q_1)\rangle) = \langle q_0 \vee \Diamond q_1\rangle \ , \tag{30}$$
$$\delta(\langle q_0 \vee \Diamond q_1\rangle) = \langle q_0\rangle \vee \langle\Diamond q_1\rangle \ , \tag{31}$$
$$\delta(\langle q_0\rangle) = q_0 \ , \tag{32}$$
$$\delta(\langle\Diamond q_1\rangle) = \Diamond\langle q_1\rangle \ , \tag{33}$$
$$\delta(\langle q_1\rangle) = \langle\mu q_1(q_0 \vee \Diamond q_1)\rangle \ . \tag{34}$$

From this transition table, we see that $\langle q_0 \vee \Diamond q_1\rangle$, $\langle q_0\rangle$, and $\langle q_1\rangle$ are not really necessary in the sense that we can take shortcuts. In fact, we can shorten the definition of the transition function:

$$\delta(\langle\mu q_1(q_0 \vee \Diamond q_1)\rangle) = \langle q_0\rangle \vee \langle\Diamond q_1\rangle \ , \tag{35}$$
$$\delta(\langle q_0\rangle) = q_0 \ , \tag{36}$$
$$\delta(\langle\Diamond q_1\rangle) = \Diamond\langle\mu q_1(q_0 \vee \Diamond q_1)\rangle \ . \tag{37}$$

This automaton is exactly the same automaton (modulo renaming of the states) as the one in Example 4.

The details of the general construction follow.

2.3.2 Formal Definition

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.

We first define a normal form for L_μ formulas. 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 and alternation depth. 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

$$\boldsymbol{A}(\varphi) = (S, s_I, \delta, \Omega) \tag{38}$$

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 $s_I = \langle\varphi\rangle$,
- the transition function is defined by

$$\delta(\langle\bot\rangle) = 0 \ , \qquad \delta(\langle\top\rangle) = 1 \ , \tag{39}$$

$$\delta(\langle q\rangle) = \begin{cases} q & \text{if } q \in \text{free}(\varphi), \\ \langle\varphi_q\rangle & \text{if } q \notin \text{free}(\varphi), \end{cases} \qquad \delta(\langle\neg q\rangle) = \neg q \ , \tag{40}$$

$$\delta(\langle\psi \wedge \chi\rangle) = \langle\psi\rangle \wedge \langle\chi\rangle \ , \qquad \delta(\langle\psi \vee \chi\rangle) = \langle\psi\rangle \vee \langle\chi\rangle \ , \tag{41}$$

$$\delta(\langle\Diamond\psi\rangle) = \Diamond\langle\psi\rangle \ , \qquad \delta(\langle\Box\psi\rangle) = \Box\langle\psi\rangle \ , \tag{42}$$

$$\delta(\langle\mu q\psi\rangle) = \langle\psi\rangle \ , \qquad \delta(\langle\nu q\psi\rangle) = \langle\psi\rangle \ , \tag{43}$$

- for every $\psi \in F_\mu$ with $\alpha(\psi) > 0$, $\Omega(\langle\psi\rangle) = 2\lceil\alpha(\psi)/2\rceil - 1$, and
- for every $\psi \in F_\nu$ with $\alpha(\psi) > 0$, $\Omega(\langle\psi\rangle) = 2\lfloor\alpha(\psi)/2\rfloor$.

This concludes the definition of $\boldsymbol{A}(\varphi)$.

From Remark 1 it follows:

Remark 2. *Let φ be an L_μ formula and Ψ a set of subformulas of φ such that $\{\langle\psi\rangle \mid \psi \in \Psi\}$ is strongly connected in the transition graph of $\boldsymbol{A}(\varphi)$. Then the following is true.*

1. *There exists a fixed point formula ψ of alternation depth > 0 which is maximal in Ψ (w. r. t. the subformula order).*
2. *For every element $\chi \in \Psi$, we have $\Omega(\psi) \geq \Omega(\chi)$.*

Another important observation is the following.

Remark 3. *Let φ be an L_μ formula. Then $\alpha(\varphi) = \operatorname{ind}(\boldsymbol{A}(\varphi))$.*

The main theorem of this section is:

Theorem 1. *Let φ be an arbitrary L_μ formula. Then φ and $\boldsymbol{A}(\varphi)$ are equivalent, that is,*

$$||\varphi|| = ||\boldsymbol{A}(\varphi)|| \ . \tag{44}$$

2.3.3 Proof of Correctness

The proof of the above theorem goes by induction on the size 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. In fact, the constructions discussed are exactly as in the translation described above.

Disjunction and Conjunction. We start with disjunction and conjunction. Let $\boldsymbol{A}$ and $\boldsymbol{A}'$ be alternating tree automata. Further, let s_I be some new state and m the maximum of $\Omega^{\boldsymbol{A}}(s_I^{\boldsymbol{A}})$ and $\Omega^{\boldsymbol{A}'}(s_I^{\boldsymbol{A}'})$. Then $\boldsymbol{A} + \boldsymbol{A}'$ and $\boldsymbol{A} \cdot \boldsymbol{A}'$ are defined by

$$\boldsymbol{A} + \boldsymbol{A}' = (S^{\boldsymbol{A}} \cup S^{\boldsymbol{A}'} \cup \{s_I\}, s_I, \delta^{\boldsymbol{A}} \cup \delta^{\boldsymbol{A}'} \cup \{(s_I, s_I^{\boldsymbol{A}} \vee s_I^{\boldsymbol{A}'})\}, \Omega^{\boldsymbol{A}} \cup \Omega^{\boldsymbol{A}'}) \ , \tag{45}$$
$$\boldsymbol{A} \cdot \boldsymbol{A}' = (S^{\boldsymbol{A}} \cup S^{\boldsymbol{A}'} \cup \{s_I\}, s_I, \delta^{\boldsymbol{A}} \cup \delta^{\boldsymbol{A}'} \cup \{(s_I, s_I^{\boldsymbol{A}} \wedge s_I^{\boldsymbol{A}'})\}, \Omega^{\boldsymbol{A}} \cup \Omega^{\boldsymbol{A}'}) \ , \tag{46}$$

where $S^{\boldsymbol{A}}$ and $S^{\boldsymbol{A}'}$ are made disjoint beforehand.

The first thing we observe is the following. If φ and ψ are L_μ formulas, then $\boldsymbol{A}(\varphi \vee \psi) = \boldsymbol{A}(\varphi) + \boldsymbol{A}(\psi)$, and, similarly, $\boldsymbol{A}(\varphi \wedge \psi) = \boldsymbol{A}(\varphi) \cdot \boldsymbol{A}(\psi)$ (up to isomorphism).

In addition, we have:

Lemma 2. *Let $\boldsymbol{A}$ and $\boldsymbol{A}'$ be alternating tree automata. Then*

$$||\boldsymbol{A} + \boldsymbol{A}'|| = ||\boldsymbol{A}|| \cup ||\boldsymbol{A}'|| \ , \qquad ||\boldsymbol{A} \cdot \boldsymbol{A}'|| = ||\boldsymbol{A}|| \cap ||\boldsymbol{A}'|| \ . \tag{47}$$

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

For the converse containment, assume $\boldsymbol{R}$ is an accepting run of $\boldsymbol{A}$ or $\boldsymbol{A}'$ 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 $\boldsymbol{A}+\boldsymbol{A}'$ on $(\boldsymbol{K}, w)$. Hence, $(\boldsymbol{K}, w) \in ||\boldsymbol{A}+\boldsymbol{A}'||$.■

The requirement that the state sets of $\boldsymbol{A}$ and $\boldsymbol{A}'$ are made disjoint is very strong. Clearly, the following would be enough. The transition functions of $\boldsymbol{A}$ and $\boldsymbol{A}'$ 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. For an arbitrary alternating tree automaton $\boldsymbol{A}$, we set

$$\Diamond \boldsymbol{A} = (S^{\boldsymbol{A}} \cup \{s_I\}, s_I, \delta^{\boldsymbol{A}} \cup \{(s_I, \Diamond s_I^{\boldsymbol{A}})\}, \Omega^{\boldsymbol{A}}) \ , \tag{48}$$

$$\Box \boldsymbol{A} = (S^{\boldsymbol{A}} \cup \{s_I\}, s_I, \delta^{\boldsymbol{A}} \cup \{(s_I, \Box s_I^{\boldsymbol{A}})\}, \Omega^{\boldsymbol{A}}) \ , \tag{49}$$

where s_I is some new state.

Similar to above, we first note that if φ is an L_μ formula, then $\Diamond \boldsymbol{A}(\varphi) = \boldsymbol{A}(\Diamond\varphi)$ and $\Box \boldsymbol{A}(\varphi) = \boldsymbol{A}(\Box\varphi)$. In addition to this, we have:

Lemma 3. *Let $\boldsymbol{A}$ be an alternating tree automaton. Then*

$$||\Diamond \boldsymbol{A}|| = \Diamond ||\boldsymbol{A}|| \ , \qquad\qquad ||\Box \boldsymbol{A}|| = \Box ||\boldsymbol{A}|| \ . \tag{50}$$

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 $\Diamond \boldsymbol{A}$. Let $\boldsymbol{R}$ be a minimal accepting run of $\Diamond \boldsymbol{A}$ on $(\boldsymbol{K}, w)$. Then, by definition of the transition function of $\Diamond \boldsymbol{A}$, 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 $(\boldsymbol{K}, w^{\boldsymbol{R}}(v))$. 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 $\Diamond \boldsymbol{A}$ on $(\boldsymbol{K}, w)$. Thus, $(\boldsymbol{K}, w) \in ||\Diamond \boldsymbol{A}||$. ∎

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

$$||\boldsymbol{A}||(\boldsymbol{K}[q \mapsto W]) \subseteq ||\boldsymbol{A}||(\boldsymbol{K}[q \mapsto W']), \tag{51}$$

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

$$\boldsymbol{A}[\boldsymbol{A}'/q] = (S^{\boldsymbol{A}} \cup S^{\boldsymbol{A}'}, s_I^{\boldsymbol{A}}, \delta, \Omega^{\boldsymbol{A}} \cup \Omega^{\boldsymbol{A}'}) \tag{52}$$

where the state sets are made disjoint beforehand and δ is obtained from $\delta^{\boldsymbol{A}} \cup \delta^{\boldsymbol{A}'}$ by replacing q in every transition condition $\delta^{\boldsymbol{A}}(s)$ by $s_I^{\boldsymbol{A}'}$. 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.

Here, we have:

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*

$$||\boldsymbol{A}[\boldsymbol{A}'/q]|| = ||\boldsymbol{A}|| \, [q \mapsto ||\boldsymbol{A}'||] \ . \tag{53}$$

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 $(\boldsymbol{K}, w)$. ■

The observant reader might have noticed that I have not made the following claim (in analogy to the other operations on automata that we have considered before): if φ and ψ are L_μ formulas and q a variable occurring positive in φ, then $\boldsymbol{A}(\varphi[\psi/q]) = \boldsymbol{A}(\varphi)[\boldsymbol{A}(\psi)/q]$ (up to isomorphism). The reason is this is not true; the priority function would have to be defined in a more sophisticated way. But we don't have to make this effort, for the above lemma will not be used directly in the proof of Theorem 1, only in the proof of the lemma below about the fixed point operators.

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 alternating tree automata that recognize $\mu q||\boldsymbol{A}||$ and $\nu q||\boldsymbol{A}||$, respectively. The only difficult part in the construction is the definition of the priority function.

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

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

$$\nu q\boldsymbol{A} = (S^{\boldsymbol{A}} \cup \{s_I\}, s_I, \delta, \Omega^{\boldsymbol{A}} \cup \{(s_I, m^\nu)\}) \ , \tag{55}$$

where s_I is a new state, δ is as $\delta^{\boldsymbol{A}} \cup \{(s_I, s_I^{\boldsymbol{A}})\}$ except that every occurrence of q is replaced by s_I, and m^μ and m^ν are yet to be defined.

Let M be the maximum priority in $\boldsymbol{A}$. If s_I does not belong to a strongly connected component of $\boldsymbol{G}(\boldsymbol{A})$, then $m^\mu = m^\nu = 0$. If it belongs to a strongly connected component of $\boldsymbol{G}(\boldsymbol{A})$ and this component contains a state with priority M, then m^μ $[m^\nu]$ is the least odd [even] number $\geq M$. In all other cases, m^μ $[m^\nu]$ is the greatest odd [even] number $\leq M$. The last rule is not well-defined for m^μ if $M = 0$; by convention, in this case, we set $m^\mu = 1$.

We first note that $\boldsymbol{A}(\mu q\varphi) = \mu q\boldsymbol{A}(\varphi)$ and $\boldsymbol{A}(\nu q\varphi) = \nu q\boldsymbol{A}(\varphi)$ (up to isomorphism).

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

$$||\mu q\boldsymbol{A}|| = \mu q||\boldsymbol{A}|| \ , \qquad ||\nu q\boldsymbol{A}|| = \nu q||\boldsymbol{A}|| \ . \tag{56}$$

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

$$f\colon W \mapsto ||\boldsymbol{A}||(\boldsymbol{K}[q \mapsto W]) \ . \tag{57}$$

We want to show that $||\mu q\boldsymbol{A}||(\boldsymbol{K})$ and $||\nu q\boldsymbol{A}||(\boldsymbol{K})$ 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 $\mu q\boldsymbol{A}$ and $\boldsymbol{A}[\mu q\boldsymbol{A}/q]$ are in a natural one-to-one correspondence. Thus,

$$\begin{aligned} W_\mu &= ||\mu q\boldsymbol{A}||(\boldsymbol{K}) && (58)\\ &= ||\boldsymbol{A}[\mu q\boldsymbol{A}/q]||(\boldsymbol{K}) && (59)\\ &= ||\boldsymbol{A}||(\boldsymbol{K}\,[q \mapsto ||\mu q\boldsymbol{A}||(\boldsymbol{K})]) && (60)\\ &= ||\boldsymbol{A}||(\boldsymbol{K}\,[q \mapsto W_\mu]) && (61)\\ &= f(W_\mu) \ , && (62) \end{aligned}$$

where (58) and (61) use the definition of W_μ, (59) is due to the above observation, (60) is Lemma 4, and (62) 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 $\mu q\boldsymbol{A}$ 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 $\boldsymbol{T}_0, \boldsymbol{T}_1, \dots$ 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 $\nu q\boldsymbol{A}$ 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

$$\boldsymbol{T}_{i+1} = \boldsymbol{T}_i \cdot \{(v, \boldsymbol{R}'_{w^{\boldsymbol{T}_i}(v)}) \mid v \in B_i\} \ . \tag{63}$$

Clearly, $\boldsymbol{T}$ is a run of $\nu q\boldsymbol{A}$ 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 occurring in a strongly connected component of the transition graph of $\nu q \boldsymbol{A}$ and $\Omega^{\boldsymbol{A}}(s_I)$ is even. So π is an accepting branch. ■

Proof of Theorem 1. As stated above, the proof of Theorem 1 goes by induction. Clearly, $||\boldsymbol{A}(\varphi)|| = ||\varphi||$ 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. Finally, if the outermost connective is a fixed point operator, the claim follows from Lemma 5. ■

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 acceptance 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 acceptance 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

$$\boldsymbol{P} = (L_0, L_1, l_I, M, \Omega) \tag{64}$$

where

- L_0 and L_1 are disjoint sets, the sets of *Player 0's* and *Player 1's locations,* respectively,
- $l_I \in L_0 \cup L_1$ is an *initial location,*
- $M \subseteq (L_0 \cup L_1) \times (L_0 \cup L_1)$ is a set of *moves,* and
- $\Omega\colon (L_0 \cup L_1) \to \omega$ is a *priority function* with a finite range.

The set of all locations, $L_0 \cup L_1$, is denoted by L. Clearly, the ordered pair (L, M) is a directed graph, which is denoted $\boldsymbol{G}(\boldsymbol{P})$ and called the *game graph* of $\boldsymbol{P}$.

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

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.

A *strategy tree* for Player 0 in $\boldsymbol{P}$ is a tree $\boldsymbol{T}$ satisfying the following conditions.

- The root of $\boldsymbol{T}$ is labeled l_I.
- Every $v \in V^{\boldsymbol{T}}$ with $\lambda^{\boldsymbol{T}}(v) \in L_0$ has a successor in $\boldsymbol{T}$ labeled with a successor of $\lambda^{\boldsymbol{T}}(v)$ in $\boldsymbol{G}(\boldsymbol{P})$, that is, Player 0 must move when it is his turn.

- Every $v \in V^{\boldsymbol{T}}$ with $\lambda^{\boldsymbol{T}}(v) \in L_1$ has, for every successor l of $\lambda^{\boldsymbol{T}}(v)$ in $\boldsymbol{G}(\boldsymbol{P})$ 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.

Example 6. First, suppose $\boldsymbol{P}$ is a game where $\Omega^P(l) = 0$ for every $l \in L^{\boldsymbol{P}}$. Then, clearly, Player 0 wins $\boldsymbol{P}$ if and only if he has a way to avoid to run into a dead end where it is his move.

Second, suppose $\Omega^P(l) = 1$ for every $l \in L^{\boldsymbol{P}}$. Then Player 0 wins $\boldsymbol{P}$ if and only if he has a way to force the game into a dead end where it is Player 1's move.

Next, suppose $\Omega^P(l) \leq 1$ and that there are no dead ends. Then Player 0 wins if and only if he can play in such a way that only a finite number of locations with priority 1 are visited.

Example 7. Let $\boldsymbol{P}$ be an arbitrary parity game with $\Omega^{\boldsymbol{P}} \colon L^{\boldsymbol{P}} \to \{0, 1, \dots, b-1\}$. Let $(\boldsymbol{K}, w_I)$ be the pointed Kripke structure where $W^{\boldsymbol{K}} = L^{\boldsymbol{P}}$, $A^{\boldsymbol{K}} = M^{\boldsymbol{P}}$, $w_I = l_I$, $\kappa(q) = L_0^{\boldsymbol{P}}$ and, for every $i < b$, $\kappa(q_i) = \{l \in L^{\boldsymbol{P}} \mid \Omega^{\boldsymbol{P}}(l) = i\}$. That is, we view $\boldsymbol{P}$ as a pointed Kripke structure.

It is now easy to design an alternating tree automaton $\boldsymbol{A}_b$ which accepts $(\boldsymbol{K}, w_I)$ if and only if Player 0 wins $\boldsymbol{P}$. The automaton $\boldsymbol{A}_b$ has a state s_i with priority i for every $i < b$. The transition function is given by

$$\delta(s_j) = \bigvee_{i<b} (q \wedge q_i \wedge \Diamond s_i) \vee \bigvee_{i<b} (\neg q \wedge q_i \wedge \Box s_i) \enspace , \tag{65}$$

for every $j < n$. So $\boldsymbol{A}_b$ simply propagates the priorities of the locations in exactly the right way: if it is Player 0's move, the priority is propagated to one successor, if it is Player 1's move, the priority is propagated to all successors.

Example 8. [6] The previous example also helps us to come up with an L_μ formula φ_b that holds in $(\boldsymbol{K}, w_I)$ if and only if Player 0 wins $\boldsymbol{P}$. We simply set

$$\varphi_b = \mu/\nu s_{b-1} \nu/\mu s_{b-2} \mu/\nu s_{b-3} \dots \nu s_0 \left(\bigvee_{i<b} (q \wedge q_i \wedge \Diamond s_i) \vee \bigvee_{i<b} (\neg q \wedge q_i \wedge \Box s_i) \right) \tag{66}$$

where the sequence of fixed point operators starts with μ if $b-1$ is odd and ν otherwise. To see that this formula is correct one constructs the automaton $\boldsymbol{A}_{\varphi_b}$ and compares it with $\boldsymbol{A}_b$ as constructed in the previous example.

3.1.3 Complexity

There is a basic question about parity games that we need to answer before we can try to solve the model-checking problem: How difficult is it to determine whether or not Player 0 wins a finite parity game?

Formally, this problem is 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 $m_C^{\boldsymbol{P}} = \max\{\Omega^{\boldsymbol{P}}(l) \mid l \in C\}$. The *index* of $\boldsymbol{P}$, denoted $\operatorname{ind}(\boldsymbol{P})$, is the maximum of these numbers, that is,

$$\operatorname{ind}(\boldsymbol{P}) = \max(\{m_C^{\boldsymbol{P}} \mid C \in \mathfrak{C}^{\boldsymbol{P}}\} \cup \{0\}) \ . \tag{67}$$

The best known upper bounds for the time complexity of WINS are listed in the following theorem.

Theorem 2. [10, 11]

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

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

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 $b = \operatorname{ind}(\boldsymbol{P})$.

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

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

3.2 Reduction of the Acceptance 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. Recall 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

$$\boldsymbol{P}(\boldsymbol{A}, \boldsymbol{K}, w_I) = (L_0, L_1, (w_I^{\boldsymbol{K}}, s_I^{\boldsymbol{A}}), M, \Omega) \tag{69}$$

where the individual components are as follows. The set L_0 is the set of all pairs (w, s) where $\delta(s)$ is of the form 0, q with $q \notin \kappa^{\boldsymbol{K}}(w)$, $\neg q$ with $q \in \kappa^{\boldsymbol{K}}(w)$, $s' \vee s''$, or $\Diamond s'$. This also determines L_1. The successors of a location (w, s) are determined by the following rules.

- If $\delta(s) \in \{0, 1\}$ or $\delta(s) = q$ or $\delta(s) = \neg q$ for some $q \in Q$, then (w, s) has no successors.
- If $\delta(s) = s'$, then (w, s) has only one successor, namely (w, s').

- If $\delta(s) = s' \vee s''$ or $\delta(s) = s' \wedge s''$, then (w, s) has two successors, namely (w, s') and (w, s'').
- If $\delta(s) = \Diamond s'$ or $\delta(s) = \Box s'$, then (w, s) has a successor (w', s') for every $w' \in \mathrm{Scs}_{\boldsymbol{K}}(w)$.

Finally, the priority function Ω maps (w, s) to $\Omega^{\boldsymbol{A}}(s)$.

The desired theorem is the following.

Theorem 3. *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. Just observe that accepting runs of $\boldsymbol{A}$ on $(\boldsymbol{K}, w)$ and winning strategy trees for Player 0 in $\boldsymbol{P}(\boldsymbol{K}, \boldsymbol{A}, w)$ are identical. ■

In view of Theorem 2, this yields the following about the complexity of the word problem for alternating tree automata. Observe that $\mathrm{ind}(\boldsymbol{P}(\boldsymbol{K}, \boldsymbol{A}, w)) = \mathrm{ind}(\boldsymbol{A})$.

Theorem 4.
1. Accepts, *the word problem for alternating tree automata, is solvable in time*
$$\mathcal{O}\left(ln\left(\frac{2kn}{b}\right)^{\lfloor b/2 \rfloor}\right) \tag{70}$$
where k is the number of worlds of the Kripke structure, l is the size of the accessibility relation, n is the number of states of the automaton, and b is the index of the automaton.
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 5. [10, 23, 14]
1. ModelChecking, *the model-checking problem for modal μ-calculus, is solvable in time*
$$\mathcal{O}\left(ln\left(\frac{2kn}{b}\right)^{\lfloor b/2 \rfloor}\right) \tag{71}$$
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 the second 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 reduce Satisfiability to the following problem.

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

We solve this problem using parity games. The superficial analogy is the following. Solving the nonemptiness problem for alternating tree automata amounts to finding a tree that is accepted. Solving the winner problem for a parity game amounts to finding a winning strategy tree. So in both cases we need to find a tree. A direct reduction to the winner problem is, however, quite complicated. Therefore, we will proceed in several steps.

4.1 Memoryless Strategies and Tree-Like Witnesses

In order to prove that an alternating tree automaton accepts some pointed Kripke structure it is enough to exhibit a pointed Kripke structure together with an accepting run for it. Eventually, we would like to combine the two into a single object, more precisely, into a single tree. The first two steps in this direction are carried out in this subsection.

First, we show that if there exists an accepting run of an alternating tree automaton on some pointed Kripke structure there also exists an accepting run which is structurally very similar to the pointed Kripke structure in question. In fact, we use a deep theorem from the theory of parity games to show that there always exists an accepting run that can be thought of as a directed graph put on top of the Kripke structure. This leads to the notion of a tree-like prewitness for nonemptiness.

Second, we show that if there exists a tree-like prewitness at all, then there exists one which is not too arbitrary in the following sense. When we consider a world of the underlying Kripke structure together with its successors and the corresponding subgraph of the accepting run put on top of it, then this structure can be chosen from a small (exponential in the size of the automaton) set. This leads to the notion of a tree-like witness for nonemptiness.

For the rest of this chapter, we fix an alternating tree automaton $\boldsymbol{A} = (S, s_I, \delta, \Omega)$. By Γ, we denote the set of all propositional variables occurring in the transition conditions of $\boldsymbol{A}$. Only these are relevant to the behavior of the automaton.

4.1.1 Memoryless Strategies

We start with the aforementioned result from the theory of parity games. 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 6. [**6, 16**] *For the winner of any parity game, there exists a memoryless winning strategy tree.*

4.1.2 Kripke Trees and Tree-like Prewitnesses

Next, we define the notions of a Kripke tree and a tree-like prewitness. A *Kripke tree* is a pointed Kripke structure $(\boldsymbol{K}, w)$ where $(W^{\boldsymbol{K}}, A^{\boldsymbol{K}})$ is a tree with root w. For notational convenience, we will omit w and denote it by $\rho^{\boldsymbol{K}}$.

A *tree-like prewitness* for (the nonemptiness of) $\boldsymbol{A}$ is a pair $(\boldsymbol{K}, \boldsymbol{W})$ consisting of a Kripke tree $\boldsymbol{K}$ and a graph $\boldsymbol{W} = (V, E)$ satisfying the following conditions.

(A) The vertex set V is a subset of $W^{\boldsymbol{K}} \times S^{\boldsymbol{A}}$.
(B) For every edge $((w, s), (w', s')) \in E$ we have $w' \in \mathrm{Scs}_{\boldsymbol{K}}(w)$ or $w = w'$.
(C) Every $v \in V$ is reachable from $(\rho^{\boldsymbol{K}}, s_I^{\boldsymbol{A}})$, which belongs to V.
(D) In every $(w, s) \in V$, the transition condition $\delta(s)$ is satisfied. (For instance, if $\delta(s) = \Diamond s'$, then there must exist $w' \in \mathrm{Scs}_{\boldsymbol{K}}(w)$ such that $(w', s') \in V$ and $((w, s), (w', s')) \in E$.) This will be denoted by $(w, s) \models \delta(s)$.
(E) For every infinite path π through $\boldsymbol{W}$, the number $\sup(p\,\pi_1\,\Omega^{\boldsymbol{A}})$ is even.

From Theorem 6 we can now conclude:

Proposition 1. *The automaton $\boldsymbol{A}$ accepts some pointed Kripke structure if and only if there is a tree-like prewitness for $\boldsymbol{A}$.*

Proof. First, note that if $\boldsymbol{A}$ accepts some pointed Kripke structure it also accepts a Kripke tree—just unravel any pointed Kripke structure it accepts. Now, reconsider the proof of Theorem 3. Assume a Kripke tree $(\boldsymbol{K}, w)$ is accepted by $\boldsymbol{A}$. Then there exists a winning strategy tree for Player 0 in $\boldsymbol{P}(\boldsymbol{A}, \boldsymbol{K}, w)$. By Theorem 6, there exists a memoryless winning strategy tree for Player 0 in this game. Consider the run that corresponds to this strategy tree. It can be viewed as an accepting witness. One only needs to identify all vertices with the same labeling and replace each vertex by its label. ∎

4.1.3 Tree-like Witnesses

Reconsider the definition of a tree-like prewitness; (D) says:

$$(w, s) \models \delta(s) \qquad \text{for every } (w, s) \in V\ . \tag{72}$$

In this subsection, we will rephrase this in graph-theoretic terms, and using this new formulation, we will be able to simplify the notion of a tree-like prewitness as indicated above.

Let $\Gamma' \subseteq \Gamma$, $S_\triangle \subseteq S$, $S_\Box \subseteq S$, and $S_\Diamond \subseteq S$. We define a (complex) transition condition by

$$\tau(\Gamma', S_\triangle, S_\Box, S_\Diamond) = \bigwedge_{q \in \Gamma'} q \wedge \bigwedge_{q \in \Gamma \setminus \Gamma'} \neg q \wedge \bigwedge_{s \in S_\triangle} s \wedge \bigwedge_{s \in S_\Diamond} \Diamond s \wedge \bigwedge_{s \in S_\Box} \Box s\ . \tag{73}$$

It is now important to observe that every transition condition of $\boldsymbol{A}$ is equivalent to a disjunction of transition conditions of this form. We will soon see what this implies.

We need some more notation. Let $(\boldsymbol{K}, \boldsymbol{W})$ be as above and $w \in W^{\boldsymbol{K}}$. Further, assume $w' \in \mathrm{Scs}_{\boldsymbol{K}}(w)$. We set

$$\Gamma(w) = \{q \in \Gamma \mid w \in \kappa^{\boldsymbol{K}}(q)\} \ , \tag{74}$$

$$S(w) = \{s \in S^{\boldsymbol{A}} \mid (w, s) \in V^{\boldsymbol{W}}\} \ , \tag{75}$$

$$\boldsymbol{G}(w) = \{(s, s') \mid ((w, s), (w, s')) \in E^{\boldsymbol{W}}\} \ , \tag{76}$$

$$\boldsymbol{G}(w, w') = \{(s, s') \mid ((w, s), (w', s')) \in E^{\boldsymbol{W}}\} \ . \tag{77}$$

So $\Gamma(w)$ is the set of propositions from Γ holding true in w, the set $S(w)$ is the set of states assumed in w, the relation $\boldsymbol{G}(w)$ describes the subgraph of $\boldsymbol{W}$ restricted to the vertices with first component w, and the relation $\boldsymbol{G}(w, w')$ describes the subgraph of $\boldsymbol{W}$ restricted to the edges from vertices with first component w to vertices with first component w'. Observe that $(\boldsymbol{K}, \boldsymbol{W})$ is implicit in the notation.

Assume $(\boldsymbol{K}, \boldsymbol{W})$ is a tree-like prewitness for $\boldsymbol{A}$. Fix $w \in W^{\boldsymbol{K}}$. Consider an arbitrary state $s \in S(w)$. Since $\boldsymbol{W}$ is a tree-like prewitness, we know there exist sets $S_\triangle$, $S_\Box$, $S_\Diamond$ such that $\tau(\Gamma(w), S_\triangle, S_\Box, S_\Diamond)$ implies $\delta(s)$ and

$$(w, s) \models \tau(\Gamma(w), S_\triangle, S_\Box, S_\Diamond) \ . \tag{78}$$

Clearly, this must hold for every $s \in S(w)$, with possibly different sets $S_\triangle$, $S_\Box$, and $S_\Diamond$. This leads to the following definition. An *admissible tile* is a tuple

$$(\Gamma', S', H, H_\Box, H_\Diamond) \tag{79}$$

where $\Gamma' \subseteq \Gamma$, $S' \subseteq S$, $H \subseteq S' \times S'$, and $H_\Box, H_\Diamond \subseteq S' \times S$ such that for every $s \in S'$ the transition condition $\tau(\Gamma', sH, sH_\Box, sH_\Diamond)$ implies $\delta(s)$. Here, sH denotes the set of all s' such there exists an s with $(s, s') \in H$ and similar for $sH_\Box$ and $sH_\Diamond$.

So (D) from the definition of tree-like prewitness can now be rephrased as follows.

(D$_1$) For every $w \in W^{\boldsymbol{K}}$, there exists an admissible tile $(\Gamma(w), S(w), H, H_\Box, H_\Diamond)$ such that $H \subseteq \boldsymbol{G}(w)$, and for every $(s, s') \in H_\Diamond$, there exists $w' \in \mathrm{Scs}_{\boldsymbol{K}}(w)$ with $H_\Box \cup \{(s, s')\} \subseteq \boldsymbol{G}(w, w')$.

We want to keep witnesses as simple as possible. A tree-like prewitness $(\boldsymbol{K}, \boldsymbol{W})$ is called a *tree-like witness* if:

(D$_2$) For every $w \in W$, there exists an admissible tile $(\Gamma(w), S(w), H, H_\Box, H_\Diamond)$ such that $H = \boldsymbol{G}(w)$ and for every $(s, s') \in H_\Diamond$, there exists exactly one $w' \in \mathrm{Scs}_{\boldsymbol{K}}(w)$ with $H_\Box \cup \{(s, s')\} = \boldsymbol{G}(w, w')$.

The difference is that we require equality instead of containment and unique successors.

By replicating subtrees in a given tree-like prewitness and removing unnecessary vertices, we can obviously turn it into a tree-like witness. So, as a consequence of Proposition 1, we note:

Corollary 1. *The automaton $\boldsymbol{A}$ accepts some pointed Kripke structure if and only if there exists a tree-like witness for $\boldsymbol{A}$.*

4.2 Automata on Infinite Words and Tree Witnesses

Recall that we want to view a nonemptiness witness as a winning strategy tree in an appropriate game. The problem with tree-like witnesses is that we require of a tree-like witness $(\boldsymbol{K}, \boldsymbol{W})$ that the parity conditions holds on every infinite path through $\boldsymbol{W}$ rather than on every branch of $\boldsymbol{K}$ (cf. (E)). In this subsection, we show there exists an exponential size ω-automaton that checks on every branch of $\boldsymbol{K}$ that all paths of $\boldsymbol{W}$ put on top of this path satisfy the parity condition. This leads to the notion of a tree witness.

4.2.1 Using Automata on Infinite Words

Let $(\boldsymbol{K}, \boldsymbol{W})$ be a tree-like witness and $\pi = w_0 w_1 \dots$ a branch of $\boldsymbol{W}$. With π, we associate the graph $\boldsymbol{G}(\pi) = (V, E)$ where

$$V = \{(i,s) \mid (w_i, s) \in V^{\boldsymbol{W}}\} \ , \tag{80}$$

$$\begin{aligned} E = \{&((i,s),(i,s')) \mid (s,s') \in \boldsymbol{G}(w_i)\} \\ &\cup \{((i,s),(i+1,s')) \mid (s,s') \in \boldsymbol{G}(w_i, w_{i+1})\} \ . \end{aligned} \tag{81}$$

So $\boldsymbol{G}(\pi)$ "is" the graph put on top of π. Condition (E) from the definition of a tree-like prewitness can now be rephrased as follows.

(E$_1$) For every branch $\pi = w_0 w_1 \dots$ and every infinite path π' through $\boldsymbol{G}(\pi)$ the parity condition of $\boldsymbol{A}$ holds, that is, $\sup(\pi' \pi_1 \Omega)$ is even.

There are two types of infinite paths $(s_0, i_0)(s_1, i_1)(s_2, i_2)\dots$ through a branch π as above:

1. There exists j such that $i_j = i_{j+1} = i_{j+2} = \dots$
2. For every i there exists j with $i_j \geq i$.

Paths of the first type get stuck in some world of π and are easy to deal with. In the following, we will focus on paths of second type, which are called *diverging paths.*

Let π be as above. The graph $\boldsymbol{G}(\pi)$ is determined by the sequence

$$p(\pi) = (\boldsymbol{G}(w_0), \boldsymbol{G}(w_0, w_1))(\boldsymbol{G}(w_1), \boldsymbol{G}(w_1, w_2))(\boldsymbol{G}(w_2), \boldsymbol{G}(w_2, w_3))\dots \tag{82}$$

Such a sequence can be viewed as an ω-word over the alphabet $\Theta = 2^{S \times S} \times 2^{S \times S}$. We will construct an ω-automaton of exponential size that accepts exactly those ω-words over Θ that correspond to graphs where every infinite sequence satisfies the parity condition of $\boldsymbol{A}$. We will then be able to rephrase (E$_1$) more conveniently.

We start with some definitions. Let $g = (B_0, C_0)(B_1, C_1)(B_1, C_1)\dots$ be any ω-word over Θ. The *graph of g,* denoted $\boldsymbol{G}(g)$, is the graph whose edge set is given by

$$\{((i,s),(i,s')) \mid \exists i((s,s') \in B_i)\} \cup \{((i,s),(i+1,s')) \mid \exists i((s,s') \in C_i)\} \ . \tag{83}$$

A *branch* of g is a maximum path through $\boldsymbol{G}(g)$ starting with $(0, s_I)$ and diverging in the above sense. The ω-word g is *even* if $\sup(\pi \pi_1 \Omega)$ is even for every infinite branch π of g.

We next show that there exists an exponential size deterministic parity ω-automaton that accepts all even ω-words over Θ.

Proposition 2. *There exists a deterministic parity ω-automaton $\boldsymbol{C}$ with $2^{\mathcal{O}(|S|^4 \log |S|)}$ states and priorities bounded by $\mathcal{O}(|S|^4)$ that recognizes the set of all even ω-words over Θ.*

Proof. Let n be the number of states of $\boldsymbol{A}$. Without loss of generality, we assume $\Omega\colon S \to \{0, \ldots, n\}$.

We first construct a nondeterministic ω-automaton $\boldsymbol{B}$ recognizing the complement. This automaton will then be transformed into the deterministic ω-automaton we are looking for.

Let $(B, C) \in \Theta$. We write $\Delta(B, C)$ for the set of all triples (s, s', i) where there exists a state $s'' \in S$ and a path from s to s'' in the graph (S, B) with maximum priority j and where $(s'', s') \in C$ and $i = \max(j, \Omega(s'))$.

Consider the nondeterministic parity ω-automaton

$$\boldsymbol{B} = (S \times \{0, \ldots, |S|\}, (s_I, \Omega(s_I)), \Delta, \Omega^{\boldsymbol{B}}) \ ,$$

where Δ is given by

$$\Delta = \{((s, i), (B, C), (s', i')) \mid (s, s', i') \in \Delta(B, C)\} \ , \tag{84}$$

and $\Omega^{\boldsymbol{B}}((s, i)) = i + 1$ for every $s \in S$. Clearly, $\boldsymbol{B}$ recognizes the set of all even ω-words over Θ.

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 $\boldsymbol{B}_3$ dual to $\boldsymbol{B}_2$. 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 ω-words over Θ.

Let $m = n(n+1)$ be the number of states of $\boldsymbol{B}$. The first step is simple and yields an automaton with $\mathcal{O}(m^2)$ states. The second step can be carried out using Safra's construction, [22], and thus yields an automaton with $2^{\mathcal{O}(m^2 \log m)}$ states and $\mathcal{O}(m^2)$ 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, [24], and yields an automaton with a larger number of states but still with $2^{\mathcal{O}(m^2 \log m)}$ many states and priorities bounded by $\mathcal{O}(m^2)$. ∎

Using the automaton $\boldsymbol{C}$ from the previous theorem, we can replace (E_1) by:

(E_2) (a) For every infinite branch π of $\boldsymbol{K}$, the ω-word $p(\pi)$ is accepted by $\boldsymbol{C}$ from above.
(b) There is no $w \in W^{\boldsymbol{K}}$ such that $\boldsymbol{G}(w)$ contains a strongly connected set where the maximum priority is odd.

The first part takes care of diverging paths, the second one of nondiverging paths.

4.2.2 Tree Witnesses

We put everything together what we have obtained thus far. Let R be the set of all admissible tiles $(\Gamma', S', H, H_\Box, H_\Diamond)$ where there exists no strongly connected component in H with a maximum priority which is odd. A *tree witness* is a tree $\boldsymbol{T} = (V, E, \lambda)$ with labels in $R \cup 2^{S \times S}$ such that the following conditions hold.

1. The root of $\boldsymbol{T}$ is labelled with some tile $(\Gamma', S', H, H_\Box, H_\Diamond)$ where $s_I \in S'$.
2. If v is a vertex labelled $C \subseteq S \times S$, then it has a successor v' labelled with an element $(\Gamma', S', H, H_\Box, H_\Diamond)$ such that $\{s' \mid \exists s((s,s') \in C)\} \subseteq S'$.
3. If v is a vertex labelled $(\Gamma', S', H, H_\Box, H_\Diamond)$, then v has a successor labelled $H_\Box \cup \{(s,s')\}$ for every $(s,s') \in H_\Diamond$.
4. For every infinite branch of $\boldsymbol{T}$ labelled $T_0 C_1 T_1 \dots$ with
$$T_i = (\Gamma_i, S_i, H_i, H_{\Box,i}, H_{\Diamond,i}),$$
the ω-automaton $\boldsymbol{C}$ accepts $(H_0, C_0)(H_1, C_1)\dots$

From Corollary 1 and Proposition 2, we can conclude:

Corollary 2. *The automaton* $\boldsymbol{A}$ *accepts some pointed Kripke structure if and only if there exists a tree witness for* $\boldsymbol{A}$.

4.3 Reduction to Parity Games

Corollary 2 enables us to carry out the last step in our reduction from the nonemptiness problem for alternating tree automata, for a tree witness can easily be interpreted as a winning strategy tree in a suitable parity game.

Assume $\boldsymbol{C}$ from above is given as $\boldsymbol{C} = (U, u_I, \delta^{\boldsymbol{C}}, \Omega^{\boldsymbol{C}})$. Consider the parity game
$$\boldsymbol{P} = (2^{S\times S} \times U, R \times U, (\{(s_I, s_I)\}, u_I), M, \Omega^{\boldsymbol{P}}) \tag{85}$$
where M is defined as below and $\Omega^{\boldsymbol{P}}$ is defined by $\Omega^{\boldsymbol{P}}((x,u)) = \Omega^{\boldsymbol{C}}(u)$ for every $x \in 2^{S\times S} \cup R$. The set M contains two types of moves.

1. Let $t = (\Gamma', S', H, H_\Box, H_\Diamond)$ be any element from R and $u \in U$. Further, let $(s,s') \in H_\Diamond$ and $C = H_\Box \cup \{(s,s')\}$. Then there is a move from (t,u) to $(C, \delta^{\boldsymbol{C}}(u, (H, C)))$.
2. Let $C \subseteq S \times S$, $u \in U$, and $(\Gamma', S', H, H_\Box, H_\Diamond) \in R$ with $\{s' \mid \exists s((s,s') \in C)\} \subseteq S'$. Then there is a move from (C,u) to (t,u).

From Corollary 2, we can finally conclude:

Proposition 3. *The automaton* $\boldsymbol{A}$ *accepts some pointed Kripke structure if and only if Player 0 wins the game* $\boldsymbol{P}$.

Proof. We only need to show that there exists a tree witness for $\boldsymbol{A}$ if and only if Player 0 wins the game $\boldsymbol{P}$. This is straightforward, and we only show one direction. Assume $\boldsymbol{T} = (V, E, \lambda)$ is a tree witness for $\boldsymbol{A}$. To construct a winning strategy tree $\boldsymbol{T'} = (V', E', \lambda')$ for $\boldsymbol{P}$, we let $\boldsymbol{C}$ run over $\boldsymbol{T}$. We set $V' = V \cup \{v_I\}$ for some new vertex v_I, and $E' = E \cup \{(v_I, \rho^{\boldsymbol{T}})\}$. The vertex v_I is labelled $(\{(s_I, s_I)\}, u_I)$. The rest of the labeling is determined as follows. Let $(v, v') \in E'$. If $\lambda'(v) = (t, u)$ with $t = (\Gamma', S', H, H_\Box, H_\Diamond)$ and if $\lambda(v') = C$, then $\lambda'(v') = (C, \delta^{\boldsymbol{C}}(u, (H, C)))$. Similarly, if $\lambda'(v) = (C, u)$ and $\lambda(v') = t$, then $\lambda'(v') = (t, u)$. Clearly, in view of the above definition of $\boldsymbol{P}$, the resulting tree is a winning strategy tree for $\boldsymbol{P}$. ■

This gives the desired upper bound for the complexity of the satisfiability problem for L_μ:

Corollary 3. [5]

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 the previous proposition 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 $2^d \times 2^{\mathcal{O}(n^4 \log n)}$ 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^4 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 2. The second claim is an immediate consequence of the first claim in view of Theorem 1. ■

Conclusion

We have seen how the proposed model of alternating tree automata, together with parity games, can be used to understand μ-calculus. It should be noted that many other results concerning the μ-calculus can be obtained and phrased using the same automaton model. For instance, Niwinski's important result, [19], that every Kripke query recognizable by an alternating tree automaton can be defined by a modal μ-calculus formula.

References

[1] André Arnold. The mu-calculus alternation-depth hierarchy is strict on binary trees. *Theoretical Informatics and Applications*, 33:329–339, 1999.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

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

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

[9] David Janin and Igor Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96: Concurrency Theory, 7th International Conference*, volume 1119 of *Lecture Notes in Computer Science*, pages 263–277, Pisa, Italy, 1996.

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

[11] 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.

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

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

[14] 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.

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

[16] Andrzej W. Mostowski. Hierarchies of weak automata and weak monadic formulas. *Theoretical Computer Science*, 83(2):323–335, 1991.

[17] David E. Muller and Paul E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54(2-3):267–276, 1987.

[18] 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.

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

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

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

[22] 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.

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

[24] 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.

[25] 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.

[26] Igor Walukiewicz. Completeness of Kozen's Axiomatization of the Propositional μ-Calculus. In *10th IEEE Symposium on Symposium on Logic in Computer Science*, pages 14–24, San Diego, California, 1995.

[27] Igor Walukiewicz. Completeness of Kozen's axiomatisation of the propositional μ-calculus, 2000. *Information and Computation* 157:142–182, 2000.

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`