A Survey on Satisfiability Checking for the μ-Calculus through Tree Automata*

Daniel Hausmann and Nir Piterman

Gothenburg University, Gothenburg, Sweden

Abstract. Algorithms for model checking and satisfiability of the modal μ-calculus start by converting formulas to alternating parity tree automata. Thus, model checking is reduced to checking acceptance by tree automata and satisfiability to checking their emptiness. The first reduces directly to the solution of parity games but the second is more complicated.

We review the non-emptiness checking of alternating tree automata by a reduction to solving parity games of a certain structure, so-called *emptiness games*. Since the emptiness problem for alternating tree automata is ExpTime-complete, the size of these games is exponential in the number of states of the input automaton. We show how the construction of the emptiness games combines a (fixed) structural part with (history-)determinization of parity word automata. For tree automata with certain syntactic structures, simpler methods may be used to handle the treatment of the word automata, which then may be asymptotically smaller than in the general case.

These results have direct consequences in satisfiability and validity checking for (various fragments of) the modal μ-calculus.

1 Introduction

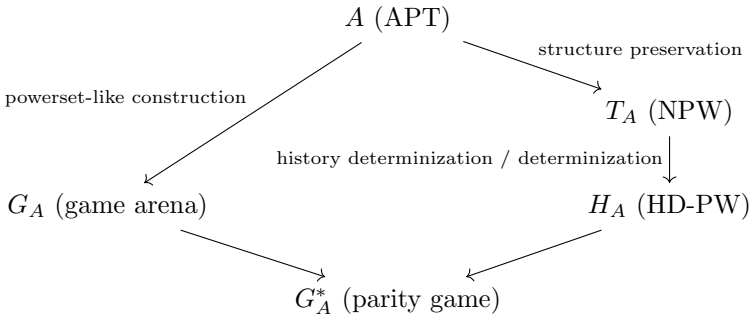
The modal μ-calculus extends modal logic with least and greatest fixpoint operators [16]. The μ-calculus is expressive enough to express many temporal logics, in particular it can capture CTL* and its fragments LTL and CTL [6]. At the same time, the μ-calculus has interesting algorithmic and algebraic properties. For example, the μ-calculus model-checking problem is equivalent to the solution of parity games, a well known (still) open problem attracting much research. This combination led to high interest in the μ-calculus, studying many aspects of the logic.

Here, we are interested in the question of satisfiability of the μ-calculus. The problem is ExpTime-complete and the first algorithms of this complexity were automata based [5]. Much like for other temporal logics, initial treatment of the logic was done through nondeterministic automata [26]. However, later, the richer structure of alternating automata enabled translations that are more natural and direct [20]. This translation defers the complicated handling of the satisfiability problem to standard automata constructions.

* This work is supported by the ERC Consolidator grant D-SynMA (No. 772459).

With this approach, the satisfiability problem for the modal μ-calculus reduces to the non-emptiness problem of alternating parity tree automata [20]. The latter problem is solved either by constructing equivalent nondeterministic parity tree automata [21] or by a direct reduction to two-player perfect information parity games [27].

We revisit the reduction and present it in a way that separates the tree acceptance and the parity acceptance aspects of a parity tree automaton A. The method creates an arena G_A (*strategy arena*), which captures all the decisions made at the same location by A and a nondeterministic parity word automaton T_A (*tracking automaton*) that "accepts" bad branches in run-trees of the original automaton. The original automaton then is non-empty if and only if the combination of G_A with T_A as losing condition is won by the existential player. By using a history deterministic (or fully deterministic) word automaton H_A that accepts the same language as T_A, we construct a parity game G_A^*.



This approach reduces the algorithmic content of non-emptiness checking for alternating automata (and for satisfiability checking in the modal μ-calculus) to a fixed construction of a game arena and (history-)determinization of word automata that depend on the exact structure of the original automaton (μ-calculus formula).

We then show that as the structure of T_A strongly depends on the structure of A, specialized history determinization and determinization constructions lead to complexity results that match bespoke algorithms for different fragments of the μ-calculus. These results are summarized in the following table, where LL stands for limit linear, LD for limit deterministic, N for nondeterministic, HD for history deterministic, D for deterministic, W for weak, B for Büchi, C for co-Büchi, and P for parity. For example, LL-CW is a limit linear co-Büchi word automaton and LD-WT is a limit deterministic weak tree automaton.

		type of T_A	method	type of H_A	size of H_A
Co-Büchi	det.	LL-CW	circle method	DCW	$n^2 \cdot 2^n$
		NCW	Miyano-Hayashi	DCW	3^n
	history-det.	LD-CW	focus method	HD-CW	$n \cdot 2^n$
Büchi	det.	LD-BW	permutation method	DPW	$\mathcal{O}(n!)$
		NBW	Safra-Piterman	DPW	$\mathcal{O}((n!)^2)$
	history-det.	NBW	Henzinger-Piterman	HD-PW	$\mathcal{O}(3^{n^2})$

Going back to the μ-calculus, the following table depicts relations between various syntactic properties of μ-calculus formulas (formally defined later) and automata. Thus, the separated treatment of the arena and the acceptance, and the different constructions for word automata summarize in one framework complexity results relating to various different fragments of the μ-calculus.

property of φ	type of A	type of T_A
limit-linear	LL-WT	LL-CW
alternation-free	AWT	NCW
aconjunctive alternation-free	LD-WT	LD-CW
aconjunctive	LD-PT	LD-BW
unrestricted	APT	NBW

2 The Modal μ-Calculus

We are concerned with satisfiability checking for different syntactical fragments of the branching-time μ-calculus, introduced by Kozen [16].

Syntax. Formulas of the μ-calculus are generated by the following grammar, where At and Var are countable sets of atoms and fixpoint variables, respectively:

$$\varphi, \psi := p \mid \neg p \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \Diamond\varphi \mid \Box\varphi \mid X \mid \xi X.\, \varphi \quad p \in \mathsf{At}, X \in \mathsf{Var}, \xi \in \{\mu, \nu\}$$

Fixpoint operators $\mu X.$ and $\nu X.$ *bind* their variable X, giving rise to standard notions of *bound* and *free variables*; for $\mu X.\, \psi$, free occurrences of X in ψ are *least fixpoint variables* and for $\nu X.\, \psi$, free occurrences of X in ψ are *greatest fixpoint variables*.

The *Fischer-Ladner closure* [16] $\mathsf{FL}(\varphi)$ (or just *closure*) of a closed formula φ is the least set of formulas that contains φ and is closed under taking subformulas for non-fixpoint operators and under unfolding for fixpoint operators; e.g. $\mu X.\, \psi \in \mathsf{FL}(\varphi)$ implies $\psi[X \mapsto \mu X.\, \psi] \in \mathsf{FL}(\varphi)$, where $\psi[X \mapsto \mu X.\, \psi]$ is the formula that is obtained from ψ by replacing every free occurrence of the variable X in ψ by the formula $\mu X.\, \psi$. We have $\mathsf{FL}(\varphi) \leq |\varphi|$ where $|\varphi|$ is the number of operators that are required to write φ (that is, the number of nodes in the syntax tree of φ).

A formula φ is *clean*, if all fixpoint variables are bound at most once in it. Then we denote *the* fixpoint formula $\xi X.\, \psi$ that binds a variable X in φ by $\theta_\varphi(X)$. While transforming an arbitrary formula to a clean formula (by renaming bound variables accordingly) can increase the closure size, a translation to tree automata that does not rely on cleanness has recently been given in [17]. For brevity of presentation we assume throughout that target formulas are clean but remark that this does not affect the stated complexity results since the more involved translation from [17] can be used to obtain tree automata (of suitable size and rank) from arbitrary formulas.

Remark 2.1. Another common constraint on the syntactic structure of formulas is *guardedness*, which requires that there is always at least one modal operator

$$
\begin{aligned}
[\![p]\!]_\eta &= \{w \mid p \in L(w)\} \\
[\![\neg p]\!]_\eta &= \{w \mid p \notin L(w)\} \\
[\![\varphi \vee \psi]\!]_\eta &= [\![\varphi]\!]_\eta \cup [\![\psi]\!]_\eta \\
[\![\varphi \wedge \psi]\!]_\eta &= [\![\varphi]\!]_\eta \cap [\![\psi]\!]_\eta \\
[\![\Diamond\varphi]\!]_\eta &= \{w \mid R(w) \cap [\![\varphi]\!]_\eta \neq \emptyset\} \\
[\![\Box\varphi]\!]_\eta &= \{w \mid R(w) \subseteq [\![\varphi]\!]_\eta\} \\
[\![X]\!]_\eta &= \eta(X) \\
[\![\mu X.\varphi]\!]_\eta &= \bigcap \{T \subseteq W \mid [\![\varphi]\!]_{\eta[X \leftarrow T]} \subseteq T\} \\
[\![\nu X.\varphi]\!]_\eta &= \bigcup \{T \subseteq W \mid T \subseteq [\![\varphi]\!]_{\eta[X \leftarrow T]}\}
\end{aligned}
$$

Fig. 1. Semantics of the μ-calculus

between a fixpoint operator and occurrences of the fixpoint variable that it binds. It is currently an open question whether there is a guardedness-transformation with polynomial blow-up of the closure size, and it has been shown that such a polynomial transformation would yield a polynomial algorithm for parity game solving [17]. Throughout this work, we assume that formulas are guarded.

Alternation-depth. Given a clean formula φ, and two subformulas $\xi X.\,\psi$ and $\xi' Y.\,\chi$ of φ, $\xi' Y.\,\chi$ *depends* on $\xi X.\,\psi$ if X has a free occurrence in χ. We define the *dependent nesting order* $\succeq_\varphi$ to be the partial order on fixpoint subformulas of φ obtained by taking the reflexive-transitive closure of the dependency ordering. The *alternation-depth* $\mathsf{ad}(\varphi)$ of φ then is defined to be the maximal length of an alternating $\succ_\varphi$-path, where a $\succ_\varphi$-path is alternating if all its transitions switch the fixpoint type. Given a fixpoint subformula $\chi = \eta X.\,\psi$ of φ, let d be the maximal length of an alternating $\succeq_\varphi$-path that starts at χ. We define the *alternation-level* $\mathsf{al}(\chi)$ of χ to be $2\lceil d/2 \rceil - 1$ if $\eta = \mu$ and $2\lfloor d/2 \rfloor$ if $\eta = \nu$. Hence least fixpoint formulas have odd alternation-level while greatest fixpoint formulas have even alternation-level; furthermore, we always have $\mathsf{al}(\chi) \leq \mathsf{ad}(\varphi)$.

Semantics. Formulas of the μ-calculus are evaluated over pointed Kripke structures. A pointed Kripke structure is $K = (W, w_0, R, L)$, where W is a set of worlds, $w_0 \in W$ is an initial world, $R \subseteq W \times W$ is a transition relation, and $L : W \to \mathcal{P}(\mathsf{At})$ is a labeling function. We denote by $R(w) = \{w' \in W \mid (w, w') \in R\}$ the set of worlds connected by R to w. We restrict attention to structures where for every $w \in W$ we have $R(w) \neq \emptyset$.

Given a μ-calculus formula φ and a pointed Kripke structure K, the semantics of the formula is defined based on a valuation function η assigning each variable appearing in φ to a set of worlds of K. Given such a function η we denote by $\eta[X \leftarrow S]$ the function η' where $\eta'(X) = S$ and $\eta'(Y) = \eta(Y)$ for every $Y \neq X$. The semantics of the μ-calculus is included in Figure 1. It is simple to see that in the case that a formula φ is closed its semantics does not depend on the initial valuation η. Thus, for a closed formula we write $[\![\varphi]\!]$. A formula φ is satisfiable if there exists a structure $K = (W, w_0, R, L)$ such that $w_0 \in [\![\varphi]\!]$.

Theorem 2.2 ([5]). *Given a μ-calculus formula φ, deciding whether φ is satisfiable is* ExpTime*-complete.*

Fragments of the μ-calculus. We consider the following fragments of the μ-calculus:

- The *limit-linear* fragment of the μ-calculus consists of all formulas φ such that for all subformulas $\mu X.\,\psi$ of φ, X has exactly one occurrence in ψ, and this occurrence is not in the scope of a fixpoint subformula of ψ. Computation tree logic (CTL) is a fragment of the limit-linear μ-calculus.
- The *alternation-free* fragment of the μ-calculus consists of all formulas φ such that $\mathsf{ad}(\varphi) \leq 1$. It has been shown that satisfiability checking for a guarded alternation-free formula of size n can be done by solving a Büchi game of size 3^n [9].
- The *aconjunctive* fragment of the μ-calculus consists of all formulas φ such that for all conjunctions $\psi \wedge \chi$ that occur as a subformula in φ, at most one of the conjuncts ψ or χ contains a free least fixpoint variable. Satisfiability checking for a (weakly) aconjunctive formula of size n and with k priorities can be done by solving a parity game of size $e \cdot (nk)!$ and with $2nk$ priorities [12].
- The *alternation-free aconjunctive* fragment of the μ-calculus is the intersection of the alternation-free fragment and the aconjunctive fragment. In particular, every limit-linear formula is alternation-free and aconjunctive.

3 Two-Player Games and Alternating Parity Tree Automata

We give background on two-player games and tree automata. Our notations are based on those developed by Wilke [27].

Definition 3.1. A *game* is $G = (V, V_\Diamond, V_\Box, E, \alpha)$, where V is a set of nodes, $V_\Diamond$ and $V_\Box$ form a partition of V to player $\Diamond$ and player $\Box$ nodes, $E \subseteq V \times V$ is a set of edges, and $\alpha \subseteq V^\omega$ is a winning condition. A *play* is a sequence $\pi = v_0, v_1, \ldots$ such that for every i we have $(v_i, v_{i+1}) \in E$. A play π is winning for player $\Diamond$ if $\pi \in \alpha$. A strategy for player $\Diamond$ is $\sigma : V^* \cdot V_\Diamond \to V$ such that $(v, \sigma(wv)) \in E$ for $wv \in V^* \cdot V_\Diamond$. A play π is *compatible* with σ if whenever $v_i \in V_\Diamond$ we have $v_{i+1} = \sigma(v_0, \ldots, v_i)$. A strategy for player $\Diamond$ is winning from node v if all the plays starting in v that are compatible with σ are winning for her. Strategies for player $\Box$ are defined similarly.

In a *parity* game, there exists a priority function $\Omega : V \to \mathbb{N}$ and a play π is winning for player $\Diamond$ if the maximal priority occurring infinitely often in π is even. A *Büchi* game is a parity game with just the priorities 1 and 2. Given an infinite sequence $\pi \in V^\omega$ let $\mathsf{inf}(\pi)$ denote the set of nodes occuring in π infinitely often and put $\mathsf{inf}_\Omega(\pi) = \{\Omega(v) \mid v \in \mathsf{inf}(\pi)\}$. Then the parity winning condition induced by Ω is $\alpha = \{\pi \in V^\omega \mid \max(\mathsf{inf}_\Omega(\pi)) \text{ is even}\}$. The

complexity of analyzing parity games is a hot area of research [1,3,4]. Here we denote by PARITY(n,k)-TIME the time complexity of solving parity games with n nodes and k priorities. We do not refer to space complexity, however, a similar general dependency on space can be stated. Our results produce parity games of different parameters depending on the exact shape of a μ-calculus formula. We hence use this parametric form to give complexity results.

Theorem 3.2 (Parity and Büchi games [1,2]). *Parity games with n nodes and k priorities can be solved in time quasipolynomial in n and k, more specifically*[1] *in time $n^{2\log(k/\log n)+\mathcal{O}(1)}$, and in polynomial time if $k < \log n$. Büchi games with n nodes can be solved in time $\mathcal{O}(n^2)$.*

We sometimes consider games where edges are labeled. In such a case, there exists a set of labels D and we have $E \subseteq V \times D \times V$. Then, α can be a subset of $V \cdot (D \cdot V)^\omega$.

When the winning condition is not important, we call $(V, V_\Diamond, V_\Box, E)$ an *arena*.

Definition 3.3. An *alternating parity tree automaton* $A = (\Sigma, Q, q_0, \delta, \Omega)$ consists of a finite alphabet Σ, a finite set of states Q, an initial state $q_0 \in Q$, a transition function $\delta : Q \times \Sigma \to \mathcal{P}(Q)$, and a priority function $\Omega : Q \to \mathbb{N}$; furthermore, each state $q \in Q$ is marked as either local-existential, local-universal, modal-existential or modal-universal (denoted by $q \in Q_\vee$, $q \in Q_\wedge$, $q \in Q_\Diamond$, and $q \in Q_\Box$, respectively). We also denote $Q_l = Q_\vee \cup Q_\wedge$, where l stands for local, and $Q_m = Q_\Diamond \cup Q_\Box$, where m stands for modal. Without loss of generality, we assume that modal-existential and modal-universal states have exactly one successor, formally: $|\delta(q)| = 1$ if $q \in Q_\Diamond$ or $q \in Q_\Box$. We overload $\delta(q)$ to denote q' if $\delta(q) = \{q'\}$. The *rank* of A is the number $|\Omega(Q)|$ of priorities appearing in its priority function. We assume that A does not have local loops. That is, for every letter σ and for every sequence of states $q_1, \ldots, q_l \in Q_l^+$ such that for every $i \geq 1$ we have $q_{i+1} \in \delta(q_i, \sigma)$ we have $q_l \neq q_1$.

A tree automaton is *weak* if for all its strongly connected components C, either all states in C have priority 0 or all states in C have priority 1. A weak tree automaton is *limit-linear* if for each $q \in Q$ such that $\Omega(q) = 1$, there is exactly one path from q to q. That is, within rejecting strongly connected components the looping behavior is deterministic. A tree automaton is *limit-deterministic* if for each odd priority p and each state q such that $\Omega(q) = p$, we have that for all $a \in \Sigma$ and all states $q' \in Q_\wedge$ that are reachable from q by visiting nodes with priority at most p, $|\delta(q', a) \cap Q_{\leq p}| \leq 1$, where $Q_{\leq p} = \{q \in Q \mid \Omega(q) \leq p\}$. In particular, every limit-linear automaton is limit-deterministic.

Alternating tree automata read *pointed Kripke structures*. An alternating tree automaton A accepts a Kripke structure $K = (W, w_0, R, L)$ if player $\Diamond$ wins the node (w_0, q_0) in the acceptance game $G_{A,K}$. Formally, $G_{A,K} = (V, V_\Diamond, V_\Box, E, \alpha)$, where $V = W \times Q$, $V_\Diamond = W \times (Q_\vee \cup Q_\Diamond)$, $V_\Box = W \times (Q_\wedge \cup Q_\Box)$, α is induced by the priority function $\Omega'(w, q) = \Omega(q)$, and E is defined as follows.

$$E = \{((w,q),(w,q')) \mid q \in Q_l \text{ and } q' \in \delta(q, L(w))\} \quad \cup \\ \{((w,q),(w',q')) \mid q \in Q_m, q' \in \delta(q, L(w)), \text{ and } w' \in R(w)\}$$

[1] This improved bound has been shown in [14]

An automaton A is *non-empty* if there exists a Kripke structure that it accepts.

We now state (the well known result) that given a μ-calculus formula, we can construct an alternating tree automaton accepting exactly the models of the formula.

Definition 3.4 (Formula automaton). Given a closed and clean μ-calculus formula φ that mentions atoms $\mathsf{A} \subseteq \mathsf{At}$, we define an alternating parity tree automaton $A(\varphi) = (\Sigma, Q, q_0, \delta, \Omega)$ by putting $\Sigma = \mathcal{P}(\mathsf{A})$, $Q = \mathsf{FL}(\varphi) \cup \{\top, \bot\}$, and $q_0 = \varphi$. We define a partial priority function $\Omega' : Q \rightharpoonup \{0, \ldots, \mathsf{ad}(\varphi)\}$ by putting $\Omega'(\eta X.\, \psi) = \mathsf{al}(\theta(X))$ (recalling that $\theta(X)$ is *the* subformula of φ that binds X), $\Omega'(\bot) = 1$ and $\Omega'(\top) = 0$; then Ω' assigns a priority to at least one state on each cycle in $A(\varphi)$. The total priority function $\Omega : V \to \{0, \ldots, \mathsf{ad}(\varphi)\}$ is obtained by putting, for each state $q \in Q$ such that $\Omega'(q)$ is undefined, $\Omega(q) = p$ where p is the minimum priority such that all paths from q to q visit priority at most p; states that do not belong to a strongly connected component obtain priority 0. Furthermore, we put

$$\delta(q, P) = \begin{cases} \{\psi_0, \psi_1\} & \text{if } q = \psi_0 \wedge \psi_1 \text{ or } q = \psi_0 \vee \psi_1 \\ \{\psi\} & \text{if } q = \Diamond\psi \text{ or } q = \Box\psi \\ \{\psi[X \mapsto \eta X.\, \psi]\} & \text{if } q = \eta X.\, \psi \\ \{\top\} & \text{if } q = p \text{ and } p \in P \text{ or } q = \neg p \text{ and } p \notin P \\ \{\bot\} & \text{if } q = p \text{ and } p \notin P \text{ or } q = \neg p \text{ and } p \in P \\ \{q\} & \text{if } q = \top \text{ or } q = \bot \end{cases}$$

for $q \in Q$, $P \in \Sigma$. Finaly, we put

$$Q_\exists = \{\psi_0 \vee \psi_1, \eta X.\, \psi \in \mathsf{FL}(\varphi)\} \cup \{\bot\} \qquad Q_\Diamond = \{\Diamond\psi \in \mathsf{FL}(\varphi)\}$$
$$Q_\forall = \{\psi_0 \wedge \psi_1, p, \neg p \in \mathsf{FL}(\varphi)\} \cup \{\top\} \qquad Q_\Box = \{\Box\psi \in \mathsf{FL}(\varphi)\}.$$

Theorem 3.5 ([27,17]). *We have $L(A(\varphi)) = \{(W, w_0, R, L) \mid w_0 \in [\![\varphi]\!]\}$. Furthermore, $|Q| \leq |\mathsf{FL}(\varphi)| + 2$ and $A(\varphi)$ has rank $\mathsf{ad}(\varphi) + 1$.*

Corollary 3.6. *Deciding if a Kripke structure with set of worlds W satisfies a μ-calculus formula φ is in* PARITY$(|W| \cdot (\mathsf{FL}(\varphi) + 2), \mathsf{ad}(\varphi) + 1)$-TIME.

It follows from Theorem 3.5 that by checking whether the language of $A(\varphi)$ is empty we can decide whether φ is satisfiable. In the next section, we proceed to show how to determine whether the language of an automaton is empty.

Before proceeding, we show that in case the μ-calculus formula has a special structure, as defined in Section 2, the automaton resulting from the translation above has also a special structure.

Lemma 3.7.
- *If φ is alternation-free, then $A(\varphi)$ is a weak tree automaton.*
- *If φ is limit linear, then $A(\varphi)$ is limit linear.*
- *If φ is aconjunctive, then $A(\varphi)$ is limit deterministic.*

Proof. – Let φ be alternation-free so that $\mathsf{ad}(\varphi) \leq 1$ and $\mathsf{al}(\psi) \leq 1$ for all $\psi \in \mathsf{FL}(\varphi)$. Hence $A(\varphi)$ uses just the priorities $\{0, 1\}$. Furthermore, every strongly connected component in $A(\varphi)$ belongs to either a greatest or a least fixpoint and hence consists only of states with priority 0 or only of states with priority 1, as claimed.
– Let φ be limit linear so that for all subformulas $\mu X.\, \psi$ of φ, X has exactly one occurrence in ψ. Then φ is alternation-free so that $A(\phi)$ is weak by the previous item. Furthermore, all states in the strongly connected component of $\theta(X)$ belong to $\mu X.\, \psi$ and hence have priority 1. Since φ is limit-linear, there is exactly one circular path in the strongly connected component of $\theta(X)$. Hence $A(\varphi)$ is limit linear.
– Let φ be aconjunctive, let p be an odd number, let $q \in Q$ such that $\Omega(q) = p$ and let $q' \in Q_\forall$ be a state that is reachable from q by visiting states with priority at most p. It remains to show that for all $a \in \Sigma$, we have $|\delta(q', \Sigma) \cap Q_{\leq p}| \leq 1$, where $Q_{\leq p} = \{q \in Q \mid \Omega(q) \leq p\}$. Since $q \in Q_\wedge$, we have $q = \psi_1 \wedge \psi_2$ for some $\psi_1, \psi_2 \in \mathsf{FL}(\varphi)$. As φ is aconjunctive, there is at most one i such that ψ_i contains a free least fixpoint variable, and such that an odd priority is reachable from ψ_i without first passing a priority greater than p. Hence $\Omega(\psi_1) > p$ or $\Omega(\psi_2) > p$, showing that $|\delta(q', \Sigma) \cap Q_{\leq p}| \leq 1$. □

4 Emptiness of Alternating Tree Automata

We now show how the decision whether the language of an alternating automaton is non-empty can be reduced to deciding the winner in a two-player game. We start by constructing a game with labeled edges and an acceptance condition that is defined by a nondeterministic word automaton. We show that player $\Diamond$ wins in this game if the language of the alternating automaton is not empty. Then, by manipulating the word automaton, we construct a parity game with the same quality: player $\Diamond$ wins if the language of the alternating automaton is not empty. This is interesting because it unifies many results about fragments of the μ-calculus to results about word automata.

4.1 Nondeterministic Parity Word Automata

Before proceeding we introduce nondeterministic and history deterministic word automata.

Definition 4.1 (Nondeterministic Parity Word Automata). A nondeterministic parity word automaton is $N = (\Sigma, Q, q_0, \delta, \Omega)$, where Σ is a finite alphabet, Q a finite set of states, $q_0 \in Q$ an initial state, and $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ a transition function. The *priority function* $\Omega : Q \to \mathbb{N}$ assigns priorities to states. Given an automaton N, the rank of N is its maximal priority, that is $\max\{\Omega(q) \mid q \in Q\}$. Given an infinite word $w = a_0 a_1 \ldots \in \Sigma^\omega$, a *run of N on w* is an infinite sequence $\tau = q_0, q_1, \ldots$ of states such that $q_{i+1} \in \delta(q_i, a_i)$ for

all $i \geq 0$. A run $\tau = q_0, q_1, \ldots$ is *accepting* if the highest priority that occurs infinitely often in τ is even. Formally, reusing the notation inf_Ω introduced for parity games, run τ is accepting if and only if $\max\{\mathsf{inf}_\Omega(\tau)\}$ is an even number. The language accepted by N is

$$L(N) = \{w \in \Sigma^\omega \mid \text{there is an accepting run of } N \text{ on } w\}.$$

Definition 4.2 (History-deterministic Word Automata [13]). Given a nondeterministic word automaton N, a *resolver* for N is a function $\sigma : \Sigma^* \to Q$ such that $\sigma(\epsilon) = q_0$ and for all sequences $wa \in \Sigma^+$, we have $\sigma(wa) \in \delta(\sigma(w), a)$. Given a word $w = a_0 a_1 \cdots \in \Sigma^\omega$ the outcome of σ on w, denoted $\sigma(w)$, is the run $r = q_0, q_1, \ldots$ such that for all $i \geq 0$ we have $q_i = \sigma(a_0 \ldots a_{i-1})$. We say that N is *history-deterministic* if there is a resolver σ such that for every word w we have that

$$w \in L(N) \text{ if and only if } \sigma(w) \text{ is an accepting run of } N.$$

A word automaton is *deterministic* if for every state $q \in Q$ and every letter $a \in \Sigma$ we have $|\delta(q, a)| \leq 1$. In particular, every deterministic automaton is history deterministic.

Theorem 4.3 ([23,22,13]). *Given a nondeterministic parity word automaton N, there exist a history deterministic parity automaton H and a deterministic parity automaton D such that $L(N) = L(H) = L(D)$.*

In Section 5 we mention several determinization and history determinization constructions that take nondeterministic word automata and construct equivalent (history) deterministic automata.

4.2 The Emptiness Games

Using these definitions we are ready to proceed with the construction of the games capturing emptiness of an alternating parity tree automaton.

Definition 4.4 (Strategy Arena). Given an alternating parity tree automaton $A = (\Sigma, Q, q_0, \delta, \Omega)$ we define the *strategy arena* $G_A = (V, V_\Diamond, V_\Box, E)$, where the components of G_A are as follows. We label the edges in E as we explain below.

- $V = (\mathcal{P}(Q) \times \Sigma) \cup \mathcal{P}(Q)$
- $V_\Diamond = \mathcal{P}(Q) \cup \{(s, \sigma) \mid s \cap Q_l \neq \emptyset\}$
- $V_\Box = \{(s, \sigma) \mid s \cap Q_l = \emptyset\}$

  That is, nodes correspond to either sets of states of A with a letter from Σ or just a set of states of A. A node is in $V_\Diamond$ if either it is a plain subset of states of A or if it contains local states of A. A node is in $V_\Box$ if it does not contain local states of A.

– A *choice function* for $a \in \Sigma$ is $d : Q_\vee \to Q$ such that for every $q \in Q_\vee$ we have $d(q) \in \delta(q, a)$. We denote by D_a all the choice functions for a and by D all the choice functions for all letters $a \in \Sigma$.
Let $D_A = D \cup Q_\Diamond \cup \Sigma$ be the set of labels.
Intuitively, an edge e from a node (set of states) s to set of states s' corresponds to one of three cases.
  - Either e corresponds to a set of transitions taken by local states of A, in which case e is labeled by the choice function associating each existential state in s to the successor chosen for it.
  - Edge e corresponds to a set of transitions taken by modal states of A. In this case the edge corresponds to the transitions of exactly *one* existential modal state and potentially many universal modal states. In this case e is labeled by the existential modal state whose transition was taken.
  - Or e corresponds to a choice of a letter in Σ.
– Given a set of states $s \subseteq Q$, $q \in s \cap Q_\Diamond$, a letter σ, and a choice $d \in D_\sigma$ we define the update of s as follows:

$$update_l(s, \sigma, d) = (s \setminus Q_l) \cup \left\{ q' \;\middle|\; \begin{array}{l} \exists q \in s \cap Q_\wedge \text{ and } q' \in \delta(q, \sigma) \text{ or} \\ \exists q \in s \cap Q_\vee \text{ and } q' = d(q) \end{array} \right\}$$
$$update_m(s, \sigma, q) = \{q' \mid q' \in \delta(q, \sigma) \text{ or } \exists q'' \in s \cap Q_\Box \text{ and } q' \in \delta(q'', \sigma)\}$$

That is, a local update consists of the set of all the successors of all the local universal states in s and all the chosen successors of all the local existential states in s. A modal update consists of the successors of the (modal existential) state $q \in s$ and all the successors of all the modal universal states in s.
The set of edges is:

$$\begin{aligned} E = &\left\{ ((s,\sigma),d,(s',\sigma)) \;\middle|\; \begin{array}{l} (s,\sigma) \in V_\Diamond, d \in D_\sigma, \text{ and} \\ s' = update_l(s,\sigma,d) \end{array} \right\} \cup \\ &\left\{ ((s,\sigma),q,s') \;\middle|\; \begin{array}{l} (s,\sigma) \in V_\Box, q \in s \cap Q_\Diamond, \\ s' = update_m(s,\sigma,q) \end{array} \right\} \cup \\ &\{ \quad (s,\sigma,(s,\sigma)) \mid \sigma \in \Sigma\} \end{aligned}$$

That is, a node (s, σ), where s contains local states of A, has successors that correspond to taking a transition from all the local states. For existential local states only one successor is taken (according to the choice labeling the edge) and for universal local states all successors are taken. A node (s, σ), where s contains no local states, has successors that correspond to taking a transition from one existential modal state in s (according to the state labeling the edge) and taking the transitions of all the universal modal states in s. A node $s \subseteq Q$, has successors that correspond to choosing a letter $\sigma \in \Sigma$ (labeling the edge) and moving to (s, σ).

We now define the winning condition associated with the strategy arena. For a labeled arena, the winning condition is a subset of $V \cdot (D_A \cdot V)^\omega$. We construct a word automaton to define the winning condition.

Definition 4.5 (Tracking Automaton). Given an alternating parity tree automaton $A = (\Sigma, Q, q_0, \delta, \Omega)$, and the arena G_A, we define the *tracking automaton* $T_A = (\Sigma_A, Q, q_0, \Gamma, \overline{\Omega})$ to be a nondeterministic parity word automaton. The alphabet of T_A is $\Sigma_A = (\Sigma \times (D \cup Q_\Diamond)) \cup \Sigma$. That is T_A reads either a letter in Σ and either a choice function or a modal existential state or simply a letter in Σ. The transition function of T_A is defined by putting

$$\Gamma(q,(\sigma,d)) = \begin{cases} d(q) & \text{if } q \in Q_\vee \text{ and } d \in D \\ \emptyset & \text{if } q \in Q_\vee \text{ and } d \in Q_\Diamond \\ \delta(q,\sigma) & \text{if } q \in Q_\wedge \\ \delta(q,\sigma) & \text{if } q \in Q_\Diamond \text{ and } d = q \\ \emptyset & \text{if } q \in Q_\Diamond \text{ and } d \in Q \setminus \{q\} \\ q & \text{if } q \in Q_\Diamond \text{ and } d \notin Q \\ \delta(q,\sigma) & \text{if } q \in Q_\Box \text{ and } d \in Q \\ q & \text{if } q \in Q_\Box \text{ and } d \notin Q \end{cases}$$

for $q \in Q$, $\sigma \in \Sigma$, and $d \in D \cup Q_\Diamond$.
For $q \in Q$ and $\sigma \in \Sigma$ we put $\Gamma(q,\sigma) = \{q\}$.

Notice that the only transitions of T_A that lead to sets with more than one element are when $q \in Q_\wedge$. Indeed, when $q \in Q_m$, by assumption, we have $|\delta(q,\sigma)| = 1$. Finally, $\overline{\Omega}$ is obtained from Ω by setting $\overline{\Omega}(q) = \Omega(q) + 1$.

We are now ready to define the acceptance condition α. Recall, that a labeled play π in G_A is $\pi \in V \cdot (D_A \cdot V)^\omega$. Given a pair (v,d), their projection onto Σ_A, denoted $\lfloor v,d \rfloor$, is $\lfloor (s,\sigma),d \rfloor = (\sigma,d)$ and $\lfloor s,\sigma \rfloor = \sigma$. Given an infinite sequence $\pi = v_0 d_0 v_1 d_1 \cdots \in V \cdot (D_A \cdot V)^\omega$, we denote by $\lfloor \pi \rfloor$ the sequence $\lfloor v_0, d_0 \rfloor \lfloor v_1, d_1 \rfloor \cdots$. We define α_A as follows.

$$\alpha_A = \{\pi \mid \lfloor \pi \rfloor \notin L(T_A)\}$$

Let G_A^+ be the combination of the arena G_A with α_A. We sum up the relation between A and G_A^+ as follows.

Theorem 4.6 (Simulation). *Let A be an alternating tree automaton A. Then A is non-empty if and only if player $\Diamond$ wins G_A^+ from $\{q_0\}$.*

A proof is included in [11].

Consider the automaton T_A. By Theorem 4.3, there exists an equivalent history deterministic automaton H_A. Let $H_A = (\Sigma_A, T, t_0, \rho, \Omega')$. By using H_A we can turn the game G_A^+ to a parity game G_A^* capturing the non-emptiness of A.

Definition 4.7 (G_A^*). Consider the strategy arena $G_A = (V, V_\Diamond, V_\Box, E)$ and the automaton H_A. We construct the parity game $G_A^* = (V', V_\Diamond', V_\Box', E', \Omega'')$, where the components of G_A^* are as follows.

- $V' = (V \times T) \cup (\Sigma_A \times V \times T)$

- $V'_{\Diamond} = V_{\Diamond} \times T$
- $V'_{\Box} = (V_{\Box} \times T) \cup (\Sigma_A \times V \times T)$
- The set of edges is:

$$\begin{aligned} E' = \{ & \quad (((s,\sigma),t)\ ,\ ((\sigma,d),v',t) \mid ((s,\sigma),d,v') \in E\} \quad \cup \\ \{ & \quad ((s,t)\ ,\ (\sigma,v',t)) \quad \mid (s,\sigma,v') \in E \quad \} \quad \cup \\ \{ & (((\sigma,d),v',t)\ ,\ (v',t')) \quad \mid t' \in \rho(t,(\sigma,d)) \} \quad \cup \\ \{ & \quad ((\sigma,v',t)\ ,\ (v',t')) \quad \mid t' \in \rho(t,\sigma) \quad \} \end{aligned}$$

- The priority function Ω'' is obtained from Ω' by setting $\Omega''(v,t) = \Omega'(t)+1$ and $\Omega''(a,v,t) = \Omega'(t)+1$.

Theorem 4.8 (Game Translation). *Player $\Diamond$ wins in G_A^+ from some state (s,σ) if and only if player $\Diamond$ wins in G_A^* from $((s,\sigma),t_0)$.*

Proof. We can show that player $\Box$ wins in G_A^* if and only if she wins in G_A^+. The proof follows the proof that history deterministic automata can be used in combination with games as in [13].

Remark 4.9. An alternative way to view the construction of T_A and H_A is to think about the dual of T_A as a universal automaton recognizing plays that are winning for player $\Diamond$. Then, the dual of H_A would be a history-deterministic universal parity automaton recognizing the same language. The resolution of the transition function of a history-deterministic universal automaton is delegated to player $\Box$ just like it is in the construction of G_A^*. In particular, every history-determinization for nondeterministic automata is, in fact, also a history-determinization for universal automata. This implies that the history-determinization construction of Henzinger and Piterman [13] can be also used for under-approximating the losing region in an LTL game, which was left as an open question in their paper.

The following is a direct implication of Theorems 4.6 and 4.8.

Corollary 4.10 (Emptiness). *Let A be an alternating tree automaton. Then A is non-empty if and only if player $\Diamond$ wins G_A^* from the node $(\{q_0\},t_0)$.*

We now consider the complexity of the decision problem. Let A be an alternating tree automaton reading an alphabet of size m with n states and rank k. Then G_A has $(m+1)\cdot 2^n$ vertices and T_A has n states and rank k as well. Let $S_{HD}(n,k)$ denote the number of states and $R_{HD}(n,k)$ denote the rank of a history deterministic automaton obtained from a nondeterministic word automation with n states and rank k.

Corollary 4.11 (Complexity). *Let A be an alternating tree automaton reading an alphabet of size m, with n states and rank k. The complexity of emptiness of A is* PARITY$((m+1)\cdot 2^n \cdot S_{HD}(n,k), R_{HD}(n,k))$-TIME.

Remark 4.12. In case that H_A is a deterministic automaton, a winning strategy for player $\Diamond$ in G_A^* directly induces a Kripke structure K with set of worlds $W = V_{\Box} \times T$ such that A accepts K. Hence non-empty alternating parity tree automata accept some structure of size at most $(m+1)\cdot 2^n \cdot S_{HD}(n,k)$ which is in $\mathcal{O}((m+1)\cdot 2^n \cdot ((nk)!)^2)$ by Lemma 5.1 and Lemma 5.11 below.

5 Transformations of Word Automata

We specialize parity acceptance conditions to the special cases of Büchi and Co-Büchi conditions. In a Büchi condition the priority function uses only the priorities $1, 2$. In a Co-Büchi condition the priority function uses only the priorities $0, 1$. For Büchi automata, we put $F = \{q \in Q \mid \Omega(q) = 2\}$; for Co-Büchi automata, we put $F = \{q \in Q \mid \Omega(q) = 0\}$; in both cases, we put $\overline{F} = Q \setminus F$. The Büchi acceptance requires accepting runs to contain infinitely many accepting states, while Co-Büchi acceptance requires accepting runs to contain only finitely many non-accepting states. A Co-Büchi automaton is weak if for all its strongly connected components C, we have $C \subseteq F$ or $C \subseteq \overline{F}$. For deterministic automata, we extend δ from letters to finite words in the obvious way.

Lemma 5.1 ([15]). *Let $A = (\Sigma, Q, q_0, \delta, \Omega)$ be a nondeterministic parity word automaton of rank k. Then there is a nondeterministic Büchi word automaton $A' = (\Sigma, Q', q_0, \delta', F)$ such that $L(A) = L(A')$ and $|Q'| \leq \left(\left\lceil \frac{k+1}{2} \right\rceil + 1\right) \cdot |Q|$.*

Proof. We just recall the construction of A' and refer to [15,12] for the proof of $L(A) = L(A')$. Intuitively, the automaton A' nondeterministically guesses a position and an even priority p such that there is a run of A on the input word such that from the guessed position on, no state with priority greater than p is visited and some state with priority p is visited infinitely often. Formally, we put $Q'' = Q \times \{i \in \mathbb{N} \mid 0 \leq i \leq k \text{ and } i \text{ is even}\}$ and

$$Q' = Q \cup Q'' \qquad\qquad F = \{(q, i) \in Q'' \mid \Omega(q) = i\}$$

so that the claimed bound on the size of A' follows immediately. The transition function δ' is defined, for $q \in Q$, even i such that $0 \leq i \leq k$ and $a \in \Sigma$, by putting

$$\delta'(q, a) = \delta(q, a) \cup \{(q', i) \in Q'' \mid q' \in \delta(q, a) \text{ and } \Omega(q') = i\}$$

and

$$\delta'((q, i), a) = \{(q', i) \in \delta(q, a) \times \{i\} \mid \Omega(q') \leq i\}.$$

□

Definition 5.2 (Limit-linear Co-Büchi Automata). A Co-Büchi automaton $A = (\Sigma, Q, q_0, \delta, F)$ is *limit-linear* if for all $q \in F$, there is exactly one δ-path that stays in F and leads from q to q.

Definition 5.3 (Limit-deterministic Word Automata). Fix a parity word automaton $A = (\Sigma, Q, q_0, \delta, \Omega)$. Given a state $q \in Q$, the *compartment* C_q of q consists of all states that are reachable from q by a path that visits states with priority at most $\Omega(q)$. We say that A is *limit-deterministic (LD)* if for all states q such that $\Omega(q)$ is even, C_q is *internally deterministic*, that is, $|\delta(q', a) \cap C_q| \leq 1$ for all $q' \in C_q$ and $a \in \Sigma$.

Thus a Büchi automaton is limit-deterministic if all its states that are reachable from an accepting state are deterministic. A Co-Büchi automaton is limit-deterministic if all its accepting states are deterministic. In particular, every limit-linear Co-Büchi automaton is limit-deterministic.

Lemma 5.4. *The construction in Lemma 5.1 preserves limit determinism.*

Proof. Let A be a limit-deterministic parity automaton. Then we claim that A' as constructed in Lemma 5.1 is limit-deterministic. Since A' is a Büchi automaton and since all states that are reachable from some state in F are contained in $Q \times \{p\}$ for some even p, it suffices to show that for all even p, all states $(q, p) \in Q \times \{p\}$ and all $a \in \Sigma$, we have $|\delta'((q, p), a)| \leq 1$. So let $(q, p) \in Q \times \{p\}$. Then, by construction of A', (q, p) is contained in the compartment C of some state with priority p in A. By definition of δ', we have $\delta'((q, p), a) \subseteq C$ since C is a compartment. Since A is limit-deterministic, C is internally deterministic which shows $|\delta'((q, p), a)| \leq 1$, as required. □

5.1 Determinizing Word Automata

We give specialized determinization constructions for limit-linear Co-Büchi automata, nondeterministic Co-Büchi automata, limit-deterministic Büchi automata, and finally for general Büchi automata and parity automata.

Lemma 5.5 (Circle method). *Let A be a limit-linear Co-Büchi automaton with n states. Then there is a deterministic Co-Büchi automaton A' with n' states such that $L(A) = L(A')$ and $n' \leq n^2 \cdot 2^n$.*

Proof. Let $A = (\Sigma, Q, q_0, \delta, F)$. If $F = Q$, then A is deterministic and we put $A' = A$. Otherwise, we have $|F| < n$ and proceed with the following construction, which is similar to the powerset construction, but additionally annotates macro-states with a single state and a counter. The states of accepting components are arranged in a cycle since A is limit-linear. Intuitively, the single state component of macro-states identifies exactly one state in exactly one accepting cycle that has a token. The determinized automaton then checks whether it is possible to stay within this cycle forever, moving the token according to the letters that are read. If this is not possible, the automaton reduces the counter by one and moves the token to *the* next state in the current cycle and again checks whether is possible to stay in the cycle forever when moving the token according to the read word. When this fails so often that the counter reaches 0, the automaton picks a state from another accepting cycle, moves the token to this state and resets the counter. It is crucial that the moving of tokens between accepting cycles is done in a fair way, so that if the token changes cycles infinitely often, the token visits every accepting cycle infinitely often. Then the token eventually stays forever within one accepting cycle if and only if there is an accepting run.

Formally, we proceed as follows. For moving the token between accepting cycles, we assume a function $\mathsf{next} : Q \to Q$ such that for $q \in Q$, $\mathsf{next}(q)$ is some abitrary but fixed state from an accepting cycle of A such that iterative

application of next cycles through all accepting cycles of A in a fair manner. We also assume a function $\mathsf{step} : F \to F$ that cycles through the states of a single accepting cycle of A in a fair manner; formally, we put $\mathsf{step}(q) = q'$ where q' is *the* state such that there is some $a \in \Sigma$ such that $\delta(q, a) \cap F = \{q'\}$. We define the deterministic Co-Büchi automaton $A' = (\Sigma, Q', u_0, \delta', F')$ by putting

$$Q' = 2^Q \times Q \times \{0, \ldots, |F|\} \qquad F' = \{(U, q, c) \in Q' \mid c \neq 0\}$$

and $u_0 = (\{q_0\}, q_0, 0)$. The claimed bound on the size of A' follows immediately since $|F| < n$ so that $|\{0, \ldots, |F|\}| \leq n$. Finally, the transition relation δ' is defined by putting, for $(U, q, c) \in Q'$ and $a \in \Sigma$,

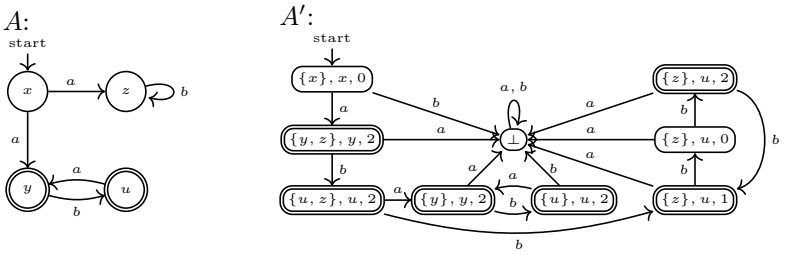
$$\delta'((U, q, c), a) = (\delta(U, a), q', c)$$

if $c \neq 0$, $\delta(q, a) \cap F = \{q'\}$ and $q \in U$; this moves the token within the current accepting cycle according to the input letter, if possible. Otherwise, the run represented by the token does not stay in the current accepting cycle and we move the token to another state. This is achieved by putting

$$\delta'((U, q, c), a) = \begin{cases} (\delta(U, a), \mathsf{next}(q), |F|) & \text{if } c = 0 \\ (\delta(U, a), \mathsf{step}(\mathsf{step}(q)), c - 1) & \text{if } c > 0 \end{cases}$$

If $c = 0$, then the token is moved to the next accepting cycle and the counter is reset to $|F|$; if $c > 0$, then the token is moved to the next state in the current accepting cycle (to also incorporate the a-transition that takes place, we apply step twice) and the counter is reduced by 1. We show in [11] that $L(A) = L(A')$. □

Example 5.6. Consider the limit-linear Co-Büchi automaton A depicted below, and the equivalent deterministic Co-Büchi automaton A' obtained by using the construction from Lemma 5.5; to be able show a complete example, A is picked to be a very simple automaton (accepting just the word $(ab)^\omega$). For brevity, we depict only the reachable part of A' and collapse all macro-states of the shape $(\emptyset, q, c)$ to a single non-accepting sink state $\bot$. Any macro-state in A' that has a nonzero counter value is accepting. We have $\mathsf{step}(y) = u$ and $\mathsf{step}(u) = y$. Since there is just one accepting strongly connected component in A, we assume that $\mathsf{next}(x) = \mathsf{next}(y) = \mathsf{next}(z) = y$, and $\mathsf{next}(u) = u$.



The automaton A' starts with the token at x and with counter value 0. When

reading a, the token is moved to $\mathsf{next}(x) = y$ and the counter is reset to 2. Afterwards, there are two cases: If the automaton reads bb, it is not possible in A to move the token accordingly from y and stay in the accepting cycle between y and u. Thus A' transitions to $(\{z\}, u, 1)$, intuitively moving the token to the next accepting cycle, which in this example moves the token to u. This state however is not contained in the powerset component $\{z\}$ so that the automaton rejects the word, which is reflected by the fact that $(\{z\}, u, 1)$ accepts the empty language. The other option to proceed from $(\{y, z\}, y, 2)$ is by reading sequences $(ba)^*$, which results in repeatedly moving the token from y to u and back to y; if this continues forever, the word is accepted by A'. Otherwise, a sequence aa or bb is read eventually and the automaton transitions to the sink state and rejects the word.

Lemma 5.7 (Miyano-Hayashi [19]). *Given a nondeterministic Co-Büchi automaton $A = (\Sigma, Q, q_0, \delta, F)$, there is a deterministic Co-Büchi automaton $A' = (\Sigma, Q', u_0, \delta', F')$ such that $L(A) = L(A')$ and $|Q'| \leq 3^{|Q|}$.*

Proof. We just show the construction of A'; for the proof of $L(A) = L(A')$ we refer to [19]. The construction is similar to the powerset construction but additionaly tracks subsets V of the accepting states $U \cap F$ of macro-states $U \subseteq Q$. Intuitively, there is, for each state in V, a run of A that has not left F recently. Whenever this set is the empty set, it is reset to all accepting states of the current macro-state. A run of A' then is accepting if such resetting steps happen only finitely often, ensuring the existence of a run of A that from some point on stays within F forever. Formally, we put

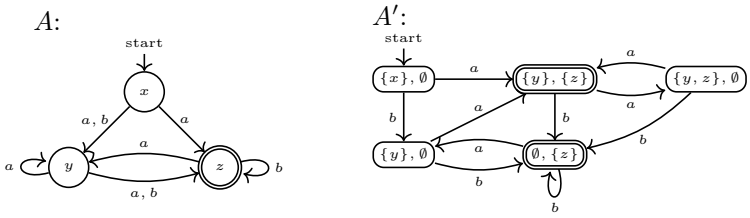
$$Q' = \{(U, V) \mid U \subseteq Q, V \subseteq U \cap F\} \qquad F' = \{(U, V) \in Q' \mid V \neq \emptyset\}$$

and $u_0 = (\{q_0\}, \emptyset)$. The claimed bound on the size of A' follows since macro-states $(U, V) \in Q'$ can be coded by functions $f : Q \to \{0, 1, 2\}$ where $f(q) = 0$ if $q \notin U$, $f(q) = 1$ if $q \in U$ but $q \notin V$ and $f(q) = 2$ if $q \in V$; the number of such functions is bounded by $3^{|Q|}$. We define δ' by putting

$$\delta'((U, V), a) = \begin{cases} (\delta(U, a), \delta(V, a) \cap F) & \text{if } V \neq \emptyset \\ (\delta(U, a), \delta(U, a) \cap F) & \text{if } V = \emptyset \end{cases}$$

for $(U, V) \in Q'$ and $a \in \Sigma$. □

Example 5.8. Consider the nondeterministic Co-Büchi automaton A depicted below, and the equivalent deterministic automaton A' obtained by using the construction from Lemma 5.7; both automata accept exactly the infinite words over $\Sigma = \{a, b\}$ that contain a finitely often. For brevity, we depict only the reachable part of A' and label macro-states (U, V) with $U \setminus V, V$.



The a-transition from the accepting macro-state $(\{y\},\{z\})$ in A' leads to the non-accepting macro-state $(\{y,z\},\emptyset)$ and not to $(\{y\},\{z\})$; the tracked set of accepting states is then reset to $\{z\}$ after a further a- or b-transition. This reflects the fact that no run of A can stay in the accepting state z by reading the letter a so that all words that contain a infinitely often are rejected by both A and A'.

Lemma 5.9 (Permutation method [7,12]). *Let A be a limit-deterministic Büchi automaton with n states. Then there is a deterministic parity automaton A' with n' states and $2n$ priorities such that $L(A) = L(A')$ and $n' \leq e(n+1)!$.*

Proof. We sketch just the construction of A' and refer to [7,12] for the proof of equivalence of A and A'. Intuively, A' is similar to the powerset automaton of A, but additionally keeps a permutation on the deterministic states in macro-states, indicating the order in which runs leading to the respective states have last seen an accepting state. Additionaly, states in A' contain a third component which indicates the leftmost position in the permutation that is *active* or *ending* by the transitions leading to the current state in A'. Here, a position is active in an a-transition, if the state at this position in the current permutation is accepting; a position is said to be ending if all runs of A that are represented by the state at this position end when reading the letter a or lead to a state at an older position. A parity condition then uses this information to detect a position in the permutation components that is active infinitely often but, from some point on, never ends. This ensures the existence of a continuous run of A that visits some accepting state infinitely often.

Formally, we proceed as follows. Given a limit-deterministic Büchi automaton $A = (\Sigma, Q, q_0, \delta, F)$ with sets $Q_D, Q_N \subseteq Q$ of deterministic and nondeterministic states, respectively, we have that every state reachable from F is contained in Q_D. We assume without loss of generality that $q_0 \in Q_N$. We let $\mathsf{perm}(Q_D)$ denote the set of partial permutations over Q_D, that is, $\mathsf{perm}(Q_D)$ consists of all partial functions $f : Q_D \rightharpoonup |Q_D|$ such that $f(q) \neq f(q')$ for all $q, q' \in \mathsf{dom}(f)$ such that $q \neq q'$. We denote the empty permutation by $[]$ ($\mathsf{dom}([]) = \emptyset$). Then we define the deterministic parity automaton $A' = (\Sigma, Q', u_0, \delta', \Omega)$ by putting

$$Q' = 2^{Q_N} \times \mathsf{perm}(Q_D) \times \{1, \ldots, 2|Q_D| + 1\} \qquad \Omega(U, f, p) = p$$

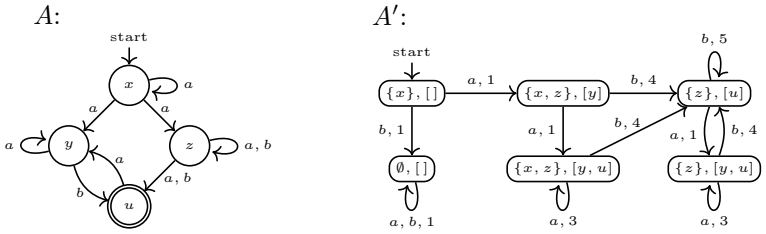
and $u_0 = (\{q_0\}, [], 1)$. The claimed bounds on the size and number of priorities of A' follows. The transition function δ' is defined by putting, for $(U, f, p) \in Q'$

and $a \in \Sigma$,

$$\delta'((U, f, p), a) = (\delta(U, a) \cap Q_N, f', p'),$$

where f' denotes the partial permutation that is obtained by applying a-transitions from δ to the partial permutation f, keeping the ordering intact but removing elements that do not have an outgoing a-transition; here it is crucial that all states in f are deterministic so that it is never the case that additional elements are inserted between any two elements of the permutation. Furthermore, we add all states from $\delta(U, a) \cap Q_D$ that do not already occur in this new permutation to the end of it (the order of these elements is irrelevant). Let $i \geq 1$ be the leftmost position in f' such that $f(i)$ is defined and $\delta(f(i), a) \neq f'(i)$ (including the case that $f'(i)$ is undefined), or we have $f'(i) \in F$. Thus i identifies the leftmost position in the partial permutation that is active or ending (possibly both). If no such i exists, put $p' = 1$. Otherwise, if $\delta(f(i), a) \neq f'(i)$, then put $p' = 2(|Q_D| - i) + 3$; if $\delta(f(i), a) = f(i) \in F$, then put $p' = 2(|Q_D| - i) + 2$. □

Example 5.10. For the limit-deterministic Büchi automaton A with $Q_N = \{x, z\}$ and $Q_D = \{y, u\}$ depicted below, we obtain the equivalent deterministic parity automaton A' using the construction from Lemma 5.9. For brevity, we depict A' with *edge priorities*, thus moving the priority component p of macro-states (U, f, p) to the edges.



Let δ be the transition relation of A. In A' there is an a-transition with priority 1 from the initial state $(\{x\}, [])$ to $(\{x, z\}, [y])$. This is the case since $\delta(\{x\}, a) = \{x, y, z\}$ so that $\{x, y, z\} \cap Q_N = \{x, z\}$. Since $y \in \delta(\{x\}, a) \cap Q_D$, we add it to the end of the permutation component which thereby changes from $[]$ to $[y]$. We have $y \notin F$ so that there is no position in the permutation that ends or is active. Thus the priority of this transition is 1. There is an a-loop with priority 3 at $(\{x, z\}, [y, u])$. This is the case since $\delta(\{x, z\}, a) = \{x, y, z, u\}$ so that $\delta(\{x, z\}, a) \cap Q_N = \{x, z\}$. Also we update the permutation component $[y, u]$ according to reading the letter a: We have $\delta(y, a) = y$ and $\delta(u, a) = y$ and hence obtain a temporary permutation $[y]$. Now $\delta(\{x, z\}, a) \cap Q_D$ contains the state u that is appended to the permutation, resulting in $[y, u]$ as new permutation component. As $y \notin F$ and $\delta(y, a) = y$, the leftmost position in the permutation component is neither active nor ending. We also have $u \in F$ and $\delta(u, a) \neq u$ so that position 2 is both ending and active. Hence the priority of this transition is $2(|Q_D| - i) + 3 = 2(2 - 2) + 3 = 3$. This reflects the fact that even though an accepting state can be reached from $\{x, y, z, u\}$ by an a-transition in A (as

$u \in \delta(z, a)$), all runs that have visited an accepting state at least once before are residing in the state y after reading a. Thus it is not possible to construct a continuous run that only reads the letter a and still visits u more than once. Intuitively, reading the letter a merges all runs leading to y or u, so that both positions in the permutation component $[y, u]$ are merged into the new first position containing just y; the second position thus is ending. The deterministic state u to which there is no a-transition from y or u then is appended as new (accepting) position to the permutation.

Lemma 5.11 (Safra-Piterman [23,22]). *Let $A = (\Sigma, Q, q_0, \delta, F)$ be a Büchi automaton. Then there is a deterministic parity automaton $A' = (\Sigma, Q', u_0, \delta', \Omega)$ such that $L(A) = L(A')$, $|Q'| \in \mathcal{O}(|Q|!^2)$ and A' has at most $2|Q|$ priorities.*[2]

Lemma 5.12 (Parity Determinization [25]). *Let $A = (\Sigma, Q, q_0, \delta, \Omega)$ be a parity automaton of rank k. Then there is a deterministic parity automaton $A' = (\Sigma, Q', u_0, \delta', \Omega')$ such that $L(A) = L(A')$, $|Q'| \in \mathcal{O}(|Q|!^{2k})$ and A' has at most $2|Q|k$ priorities.*

5.2 History-determinizing Word Automata

Next we give specialized history determinization constructions for limit-deterministic Co-Büchi automata and for general Büchi automata.

Lemma 5.13 (History-determinizing by focusing). *Let $A = (\Sigma, Q, q_0, \delta, F)$ be a limit-deterministic Co-Büchi word automaton. Then there is a history-deterministic Co-Büchi word automaton $A' = (\Sigma, Q', u_0, \delta', F')$ such that $L(A) = L(A')$ and $|Q'| \leq (|F| + 1) \cdot 2^{|Q|}$.*

Proof. Inituively, the determinization procedure is similar to the powerset construction but uses the limited nondeterminism that is allowed in history-deterministic automata to guess a run that eventually stays in F forever. Information about the guessed runs is kept by annotating macro-states with a *focus*, that is, the state in which the run currently resides. If the guess turns out to be wrong and the run leaves F, a new guess is taken (a refocusing step takes place). The resulting automaton then can be shown to be history-deterministic by using a resolver function that refocuses in a fair manner, guaranteeing that no run is overlooked.

Formally, we put $Q'' = \{(U, q) \in 2^Q \times F \mid q \in U\}$ and

$$Q' = 2^Q \cup Q'' \qquad u_0 = \{q_0\} \qquad F' = Q'',$$

from which the claimed bound on the size of A' follows since

$$|Q'| = |2^Q \cup Q''| = 2^{|Q|} + (2^{|Q|} \cdot |F|) = (|F| + 1) \cdot 2^{|Q|}.$$

The transition relation δ' is defined by putting, for $a \in \Sigma$ and $U \subseteq Q$,

$$\delta'(U, a) = \{\delta(U, a)\} \cup \{(\delta(U, a), q') \mid q' \in \delta(U, a) \cap F\}$$

[2] The tight complexity analysis of the construction in [22] is in [24].

and, for $a \in \Sigma$ and $(U, q) \in Q''$,

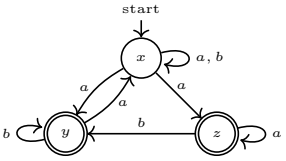
$$\delta'((U,q),a) = \begin{cases} \{(\delta(U,a),q')\} & \text{if } \delta(q,a) \cap F = \{q'\} \\ \{\delta(U,a)\} & \text{if } \delta(q,a) \cap F = \emptyset, \end{cases}$$

noting that since A is limit-deterministic, we have $|\delta(q,a) \cap F| \leq 1$ if $q \in F$, so that the case distinction above is exhaustive. Given $(U, q) \in Q''$, we refer to q as the *focus* and for $U \in Q'$ we say that the focus is *finished* at U. Outgoing transitions from $U \subseteq Q$ to $(U', q) \in Q''$ are *refocusing transitions*. We show in [11] that $L(A) = L(A')$. □

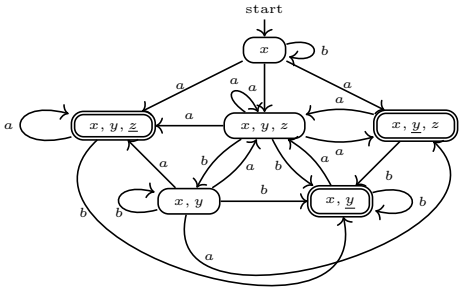
We note that the automaton A' in the above construction is not a weak automaton, even if A is a weak automaton.

Example 5.14. Consider the limit-deterministic Co-Büchi automaton A depicted below, and the equivalent history-deterministic Co-Büchi automaton A' obtained by using the construction from Lemma 5.13; both automata accept exactly the infinite words over $\Sigma = \{a, b\}$ that contain either a or b finitely often. For brevity, we depict only the reachable part of A' and label macro-states (U, q) with the elements of U with focus q underlined. Every macro-state in A' that has a focus is accepting.

A:



A':



Let δ be the transition relation of A. Then we have a-transitions from x to $x, y, \underline{z}$, to x, y, z and to $x, \underline{y}, z$. This is the case since $\delta(x, a) = \{x, y, z\}$. Since both y and z are accepting states in A, A' has, when reading a at the state x, the history-deterministic choice to focus either y or z (or none of the two). On the other hand, we have e.g. an a-transition from $x, \underline{y}$ to the non-accepting macro-state x, y, z since $\delta(y, a) = x$ is not an accepting state. Hence the a-transition from $x, \underline{y}$ finishes the focus and another refocusing step is necessary in order for a run to be accepting. Thus A' accepts for instance the word $(aba)b^\omega$ (also accepted by A) by staying unfocused when reading $abab$, leading to the partial run $x \xrightarrow{a} x, y, z \xrightarrow{b} x, y \xrightarrow{a} x, y, z \xrightarrow{b} x, y$; then the automaton can focus on y, continuing the run with $x, y \xrightarrow{b} x, \underline{y} \xrightarrow{b} x, \underline{y} \ldots$, resulting in an overall accepting run. For the word a^ω however, there is a non-accepting run $x \xrightarrow{a} x, y, z \xrightarrow{a} x, \underline{y}, z \xrightarrow{a} x, y, z \xrightarrow{a} x, \underline{y}, z$ in which y is focused infinitely often, but

also finished infinitely often. This shows that fair order of focusing is crucial in resolving the history-determinism: Every run in which the automaton eventually focuses the state z is accepting.

Lemma 5.15 (Henzinger-Piterman [13]). *Let A be a nondeterministic Büchi word automaton with n states. Then there is a history-deterministic parity word automaton A' with n' states such that $L(A) = L(A')$ and $n' \in \mathcal{O}(3^{n^2})$.*

Notice that the size of a history-deterministic automaton, in the general case, is larger than the size of the deterministic automaton. The potential advantage of using history determinization would be to have a simpler structure of the resulting automaton.

5.3 Application to Emptiness Checking and μ-Calculus Satisfiability

To conclude this section, we state the connection between the structure of the alternating parity tree automaton A and the structure of the tracking automaton T_A.

Lemma 5.16.
- *If A is an alternating weak tree automaton, then the tracking automaton T_A is a weak word automaton.*
- *If A is limit deterministic, then the tracking automaton T_A is limit deterministic.*

Proof.
- Let A be a weak automaton. Then all strongly connected components in T_A either contain only states with priority 2 or only states with priority 1. Regarding states with priority 1 as non-accepting and states with priority 2 as accepting, T_A can be seen as a weak automaton (and as a Co-Büchi automaton).
- Recall that the transition relation of T_A is Γ and the priority function of T_A is $\overline{\Omega}$. Let A be limit deterministic, let $q \in Q$ such that $\overline{\Omega}(q) = p$ is even, let $q' \in C_q$ and let $a \in \Sigma$. We have to show that $|\Gamma(q', a) \cap C_q| \leq 1$. Since $q' \in C_q$, q' is reachable from q by a path that visits states with priority at most p. The only case where we have $|\Gamma(q', a)| > 1$ is when $q' \in Q_\wedge$. Also $\Omega(q) = \overline{\Omega}(q) - 1 = p - 1$ is odd and q' is reachable in A by a path that visits nodes with priority at most $p - 1$. Since A is limit-deterministic, we have $|\delta(q', a) \cap Q_{\leq p-1}| \leq 1$ which implies $|\Gamma(q', a) \cap C_q| \leq 1$ since $C_q \subseteq Q_{\leq p-1}$ by definition of compartments and since $\Gamma(q', a) \subseteq \delta(q', a)$ by definition of Γ.

□

Remark 5.17. If A is limit linear, then A is weak. By the above lemma, T_A is a weak automaton with F being the states with priority 2. Given $q \in F$ so that $\Omega(q) = \overline{\Omega}(q) - 1 = 1$, there is exactly one path from q to q in A, since A is limit linear. Except for the self-loops introduced by non-manipulating transitions in T_A, there is exactly one path from q to q in T_A that stays in F. We note that the concept of limit-linear word automata can be slightly extended to accommodate self-loops, using a notion of *synchronizing* transitions in such a way that the method from Lemma 5.5 can be employed, obtaining the same complexity result. For brevity, we omit the technical details here and refer to [10] instead.

By using the bespoke determinization and history-determinization constructions stated above we achieve below better complexity bounds.

Corollary 5.18. *Let A be an alternating parity tree automaton reading an alphabet of size m, with n states and rank k. Depending on the structure of A, the complexity of emptiness checking for A is as follows (where $s = (nm+1) \cdot 2^n$).*

- *If A is limit-linear:* $\textsc{parity}(s \cdot n^2 \cdot 2^n, 2)$-$\textsc{Time}$
- *If A is limit-deterministic and weak:* $\textsc{parity}(s \cdot n \cdot 2^n, 2)$-$\textsc{Time}$
- *If A is weak:* $\textsc{parity}(s \cdot 3^n, 2)$-$\textsc{Time}$
- *If A is limit-deterministic:* $\textsc{parity}(s \cdot e \cdot (nk)!, 2nk)$-$\textsc{Time}$
- *In any case:* $\textsc{parity}(s \cdot \mathcal{O}((n)!^{2k}), 2nk)$-$\textsc{Time}$

Remark 5.19. Given a formula φ, the automaton $A(\varphi)$ makes very limited use of the alphabet Σ (in fact, it is only used to check for satisfaction of propositional atoms). For satisfiability checking, the guessing and memorizing of letters in the emptiness game G_A^* can hence be avoided by letting player $\Box$ immediately win all nodes whose state component $s \subseteq Q$ in the strategy arena contains $\{p, \neg p\}$ for some atom p. Furthermore, the state component $s \subseteq Q$ of nodes in the strategy arena is always contained in the label of states of the (history) deterministic variant of the tracking automaton H_A. Hence G_A^* can be slightly adapted to obtain the following complexity bounds, matching previously known results for guarded formulas.

Corollary 5.20. *Let φ be a μ-calculus formula and let $n = |\mathsf{FL}(\varphi)|$, $k = \mathsf{ad}(\varphi)$. Then the time complexity of deciding satisfiability of φ is as follows.*

- *If φ is limit-linear:* $\textsc{parity}(n^2 \cdot 2^n, 2)$-$\textsc{Time}$
- *If φ is aconjunctive and alternation-free:* $\textsc{parity}(n \cdot 2^n, 2)$-$\textsc{Time}$
- *If φ is alternation-free:* $\textsc{parity}(3^n, 2)$-$\textsc{Time}$
- *If φ is aconjunctive:* $\textsc{parity}(e \cdot (nk)!, 2nk)$-$\textsc{Time}$
- *In any case:* $\textsc{parity}(\mathcal{O}((n)!^{2k}), 2nk)$-$\textsc{Time}$

Let φ be satisfiable. Then φ has a model of size $2^{O(nk)\log n}$, and of size 3^n if φ is alternation-free.

Remark 5.21. In [8], the authors present a tableaux-based satisfiability algorithm for unguarded formulas; unguardedness is handled by an auxilliary tableau rule and by extending the tracking automaton with an additional priority to detect *inactive* traces. Using this approach however, the tracking automaton for unguarded alternation-free formulas is (in contrast to our framework) *not* a Co-Büchi automaton and Co-Büchi methods for (history)-determinization can not be used to obtain Büchi games that characterize satisfiability.

Our treatment of aconjunctive and alternation-free formulas employs a focusing method (Lemma 5.13) to history-determinize limit-deterministic Co-Büchi automata. This generalizes the focus games for CTL [18] to the aconjunctive alternation-free μ-calculus and sheds light on the automata theoretic background of focus games.

6 Conclusions

We surveyed the approach to deciding the satisfiability of the modal μ-calculus through a reduction to alternating parity tree automata emptiness. We present the solution to the emptiness of alternating parity tree automata as a combination of a structural game construction with word automata for defining the winning condition. Interestingly the structural game construction remains fixed regardless of the exact structure of the automaton. The exact structure, however, greatly affects the properties of the word automata for the winning condition. This, in turn, can be exploited to give improved complexity results for various fragments of the μ-calculus by concentrating on bespoke word automata conversion constructions.

References

1. Calude, C., Jain, S., Khoussainov, B., Li, W., Stephan, F.: Deciding parity games in quasipolynomial time. In: Theory of Computing, STOC 2017. pp. 252–263. ACM (2017)
2. Chatterjee, K., Henzinger, M.: An $O(n^2)$ time algorithm for alternating büchi games. In: SODA. pp. 1386–1399. SIAM (2012)
3. Colcombet, T., Fijalkow, N.: Universal graphs and good for games automata: New tools for infinite duration games. In: FoSSaCS. Lecture Notes in Computer Science, vol. 11425, pp. 1–26. Springer (2019)
4. Czerwinski, W., Daviaud, L., Fijalkow, N., Jurdzinski, M., Lazic, R., Parys, P.: Universal trees grow inside separating automata: Quasi-polynomial lower bounds for parity games. In: SODA. pp. 2333–2349. SIAM (2019)
5. Emerson, E.A., Jutla, C.: The complexity of tree automata and logics of programs. SIAM J. Comput. **29**(1), 132–158 (Sep 1999)
6. Emerson, E.A., Lei, C.: Efficient model checking in fragments of the propositional mu-calculus (extended abstract). In: LICS. pp. 267–278. IEEE Computer Society (1986)
7. Esparza, J., Kretínský, J., Raskin, J., Sickert, S.: From LTL and limit-deterministic büchi automata to deterministic parity automata. In: Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2017. LNCS, vol. 10205, pp. 426–442. Springer (2017)
8. Friedmann, O., Lange, M.: Deciding the unguarded modal μ-calculus. J. Appl. Non-Classical Log. **23**, 353–371 (2013)
9. Friedmann, O., Latte, M., Lange, M.: Satisfiability games for branching-time logics. Log. Methods Comput. Sci. **9** (2013)
10. Hausmann, D.: Satisfiability Checking for the Coalgebraic μ-Calculus. Ph.D. thesis, University of Erlangen-Nuremberg, Germany (2018), `https://opus4.kobv.de/opus4-fau/frontdoor/index/index/docId/9932`
11. Hausmann, D., Piterman, N.: A survey on satisfiability checking for the μ-calculus through tree automata. CoRR **abs/2207.00517** (2022), `https://arxiv.org/abs/2207.00517`
12. Hausmann, D., Schröder, L., Deifel, H.: Permutation games for the weakly aconjunctive μ-calculus. In: Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2018. LNCS, vol. 10806, pp. 361–378. Springer (2018)

13. Henzinger, T.A., Piterman, N.: Solving games without determinization. In: CSL. Lecture Notes in Computer Science, vol. 4207, pp. 395–410. Springer (2006)
14. Jurdzinski, M., Morvan, R.: A universal attractor decomposition algorithm for parity games. CoRR **abs/2001.04333** (2020), `https://arxiv.org/abs/2001.04333`
15. King, V., Kupferman, O., Vardi, M.: On the complexity of parity word automata. In: Foundations of Software Science and Computation Structures, FoSSaCS 2001. LNCS, vol. 2030, pp. 276–286. Springer (2001)
16. Kozen, D.: Results on the propositional μ-calculus. Theor. Comput. Sci. **27**, 333–354 (1983)
17. Kupke, C., Marti, J., Venema, Y.: Succinct graph representations of μ-calculus formulas. In: Computer Science Logic, CSL 2022. LIPIcs, vol. 216, pp. 29:1–29:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
18. Lange, M., Stirling, C.: Focus games for satisfiability and completeness of temporal logic. In: Logic in Computer Science, LICS 2001. pp. 357–365. IEEE Computer Society (2001)
19. Miyano, S., Hayashi, T.: Alternating finite automata on ω-words. Theor. Comput. Sci. **32**, 321–330 (1984)
20. Muller, D.E., Saoudi, A., Schupp, P.E.: Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In: LICS. pp. 422–427. IEEE Computer Society (1988)
21. Muller, D.E., Schupp, P.E.: Alternating automata on infinite trees. Theor. Comput. Sci. **54**, 267–276 (1987)
22. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. Log. Meth. Comput. Sci. **3** (2007)
23. Safra, S.: On the complexity of omega-automata. In: Foundations of Computer Science, FOCS 1988. pp. 319–327. IEEE Computer Society (1988)
24. Schewe, S.: Tighter bounds for the determinisation of büchi automata. In: Foundations of Software Science and Computational Structures, FOSSACS 2009. LNCS, vol. 5504, pp. 167–181. Springer (2009)
25. Schewe, S., Varghese, T.: Determinising parity automata. In: Mathematical Foundations of Computer Science, MFCS 2014. LNCS, vol. 8634, pp. 486–498. Springer (2014)
26. Streett, R.S., Emerson, E.A.: An automata theoretic decision procedure for the propositional mu-calculus. Inf. Comput. **81**(3), 249–264 (1989)
27. Wilke, T.: Alternating tree automata, parity games, and modal μ-calculus. Bulletin of The Belgian Mathematical Society-simon Stevin **8**, 359–391 (2001)