Degrees of Ambiguity for Parity Tree Automata

Alexander Rabinovich
Tel Aviv University, Israel
https://www.cs.tau.ac.il/~rabinoa/
rabinoa@tauex.tau.ac.il

Doron Tiferet[1]
Tel Aviv University, Israel
sdoron5.t2@gmail.com

Abstract

An automaton is unambiguous if for every input it has at most one accepting computation. An automaton is finitely (respectively, countably) ambiguous if for every input it has at most finitely (respectively, countably) many accepting computations. An automaton is boundedly ambiguous if there is $k \in \mathbb{N}$, such that for every input it has at most k accepting computations. We consider Parity Tree Automata (PTA) and prove that the problem whether a PTA is not unambiguous (respectively, is not boundedly ambiguous, not finitely ambiguous) is co-NP complete, and the problem whether a PTA is not countably ambiguous is co-NP hard.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects

Keywords and phrases automata on infinite trees, degree of ambiguity, omega word automata, parity automata

Digital Object Identifier 10.4230/LIPIcs.CSL.2021.36

Funding Supported in part by Len Blavatnik and the Blavatnik Family foundation.

Acknowledgements We would like to thank the reviewers for their useful suggestions. The first author is grateful to Michał Skrzypczak for very fruitful discussions.

1 Introduction

Degrees of Ambiguity

The relationship between deterministic and nondeterministic machines plays a central role in computer science. An important topic is the comparison of expressiveness, succinctness and complexity of deterministic and nondeterministic models. Various restricted forms of nondeterminism were suggested and investigated (see [3, 4] for recent surveys).

Probably, the oldest restricted form of nondeterminism is unambiguity. An automaton is unambiguous if for every input there is at most one accepting run. For automata over finite words there is a rich and well-developed theory on the relationship between deterministic, unambiguous and nondeterministic automata [4]. All three models have the same expressive power. Unambiguous automata are exponentially more succinct than deterministic ones, and nondeterministic automata are exponentially more succinct than unambiguous ones [6, 7].

Some problems are easier for unambiguous than for nondeterministic automata. As shown by Stearns and Hunt [13], the equivalence and inclusion problems for unambiguous automata are in polynomial time, while these problems are PSPACE-complete for nondeterministic automata.

[1] corresponding author

Figure 1 Finitely ambiguous and 2-ambiguous Büchi automata.

The complexity of basic regular operations on languages represented by unambiguous finite automata was investigated in [5], and tight upper bounds on state complexity of intersection, concatenation and many other operations on languages represented by unambiguous automata were established. It is well-known that the tight bound on the state complexity of the complementation of nondeterministic automata is 2^n. In [5], it was shown that the complement of the language accepted by an n-state unambiguous automaton is accepted by an unambiguous automaton with $2^{0.79n+\log n}$ states.

Many other notions of ambiguity were suggested and investigated. A recent paper [4] surveys works on the degree of ambiguity and on various nondeterminism measures for finite automata on words.

An automaton is *k-ambiguous* if on every input it has at most k accepting runs; it is *boundedly ambiguous* if it is k-ambiguous for some k; it is *finitely ambiguous* if on every input it has finitely many accepting runs.

It is clear that an unambiguous automaton is k-ambiguous for every $k > 0$, and a k-ambiguous automaton is finitely ambiguous. The reverse implications fail. For ϵ-free automata over words (and over finite trees), on every input there are at most finitely many accepting runs. Hence, every ϵ-free automaton on finite words and on finite trees is finitely ambiguous. However, over ω-words there are nondeterministic automata with uncountably many accepting runs. Over ω-words and over infinite trees, finitely ambiguous automata are a proper subclass of the class of countably ambiguous automata, which is a proper subclass of nondeterministic automata.

Weber and Seidl [15] investigated several classes of ambiguous automata on words, and obtained polynomial time algorithms for deciding the membership in each of these classes. Their algorithms were derived from structural characterizations of the classes.

In particular, they proved that the following Bounded Ambiguity Criterion (BA) characterizes whether there is a bound k such that a nondeterministic automaton on words has at most k accepting runs on each word.

Forbidden Pattern for Bounded Ambiguity: There are distinct useful[2] states $p, q \in Q$ such that for some word u, there are runs on u from p to p, from p to q and from q to q.

Weber and Seidl [15] proved that an NFA is not boundedly ambiguous iff it contains the forbidden pattern for bounded ambiguity. This pattern is testable in polynomial time; hence, it can be decided in polynomial time whether the degree of ambiguity of an NFA is bounded.

Seidl [12] provided a structural characterization of bounded ambiguity for automata on finite trees, and derived a polynomial algorithm to decide whether such an automaton is boundedly ambiguous.

[2] A state is useful if it is on an accepting run.

Löding and Pirogov [8] and Rabinovich [10] provided structural characterizations and polynomial algorithms for bounded, finite and countable ambiguity of Büchi automata on ω-words.

Rabinovich and Tiferet [11] provided a structural characterizations and polynomial time algorithms for bounded, finite and countable ambiguity of Büchi automata on infinite trees.

Over infinite trees, Büchi tree automata are less expressive than Monadic Second-Order Logic which is equivalent to parity tree automata. Our main result is:

▶ **Theorem 1** (Main).

1. *The problem whether a parity tree automaton is not unambiguous (respectively, not boundedly ambiguous, or not finitely ambiguous) is co-NP-complete.*
2. *The problem whether a parity tree automaton is not countably ambiguous is co-NP hard.*

The paper's organization. The next section contains standard definitions and notations about tree automata. The proof of Theorem 1 relies on the complexity of many-dimensional parity games. In Sect. 3, we first recall many-dimensional parity games, introduced by Chatterjee, Henzinger, and Piterman in [2]. We provide reductions between the emptiness problem for multidimensional parity tree automata and multidimensional parity games. Then, we show that the non-emptiness problem for intersection of PTA is polynomial time reducible to the non-emptiness problem of multi-dimensional PTA, and therefore is in co-NP.

In Sect. 4, we prove co-NP-hardness for the degrees of ambiguity of a PTA. This establishes the lower bounds stated in Theorem 1. The proof of the upper bounds of Theorem 1 is based on structural characterizations of degrees of ambiguity for PTA. In Sect. 5 we lift the structural characterizations of Büchi ω-word automata [8, 10] to structural characterizations of parity ω-word automata and provide a polynomial algorithm for the degrees of ambiguity of parity ω-word automata. In Sect. 6 we present characterizations for the finite and bounded ambiguity of PTA. Finally, in Sect. 7 we prove the co-NP upper bounds of the Main Theorem. The last section presents the conclusion.

2 Preliminaries

We recall here standard terminology and notations about trees and automata [14, 9].

Trees. We view the set $\{l, r\}^*$ of finite words over alphabet $\{l, r\}$ as the domain of a full-binary tree, where the empty word ϵ is the root of the tree, and for each node $v \in \{l, r\}^*$, we call $v \cdot l$ the left child of v, and $v \cdot r$ the right child of v.

We define a tree order "$\leq$" as a partial order such that $\forall u, v \in \{l, r\}^* : u \leq v$ iff u is a prefix of v. Nodes u and v are incomparable - denoted by $u \perp v$ - if neither $u \leq v$ nor $v \leq u$.

We say that an infinite sequence $\pi = v_0, v_1, \ldots$ is a **tree branch** if $v_0 = \epsilon$ and $\forall i \in \mathbb{N}$: $v_{i+1} = v_i \cdot l$ or $v_{i+1} = v_i \cdot r$.

If Σ is a finite alphabet, then a Σ-labeled full-binary tree t is a labeling function $t : \{l, r\}^* \to \Sigma$. We denote by T_Σ^ω the set of all Σ-labeled full-binary trees. We often use "tree" for "labeled full-binary tree."

Given a Σ-labeled tree t and a node $v \in \{l, r\}^*$, the tree $t_{\geq v}$ (called the subtree of t, rooted at v) is defined by $t_{\geq v}(u) := t(v \cdot u)$ for each $u \in \{l, r\}^*$.

A tree language L over an alphabet Σ is a set of Σ-labeled trees.

Automata on ω-words and on infinite trees. An automaton on ω-words is a tuple $\mathcal{A} := (Q_{\mathcal{A}}, \Sigma, Q_I, \delta, Acc)$ where Σ is a finite alphabet, $Q_{\mathcal{A}}$ is a finite set of states, $Q_I \subseteq Q_{\mathcal{A}}$ is a set of initial states, $\delta \subseteq Q_{\mathcal{A}} \times \Sigma \times Q_{\mathcal{A}}$ is a transition relation, and Acc is an acceptance condition. A run of $\mathcal{A}$ on an ω-word $y = a_0 a_1 \dots$ is an infinite sequence $\rho = q_0, q_1, \dots$ such that $q_0 \in Q_I$, and for all $i \in \mathbb{N} : (q_i, a_i, q_{i+1}) \in \delta$.

The Büchi acceptance conditions are given by a set $F \subseteq Q_{\mathcal{A}}$. We say that a run ρ is accepting if there is a state $f \in F$ which appears infinitely often in ρ.

The parity acceptance conditions are given by a function $\mathbb{C} : Q_{\mathcal{A}} \to \mathbb{N}$. We say that a run ρ is accepting if the maximal number which appears infinitely often in $\mathbb{C}(q_0), \mathbb{C}(q_1), \dots$ is even.

We denote the set of all accepting runs of $\mathcal{A}$ on an ω-word y by $ACC(\mathcal{A}, y)$. The language of $\mathcal{A}$ is defined as $L(\mathcal{A}) := \{y \in \Sigma^\omega \mid ACC(\mathcal{A}, y) \neq \emptyset\}$.

An automaton on infinite trees is a tuple $\mathcal{A} := (Q_{\mathcal{A}}, \Sigma, Q_I, \delta, Acc)$ where $Q_{\mathcal{A}}$, Σ, Q_I are as in an automaton on ω-words, $\delta \subseteq Q_{\mathcal{A}} \times \Sigma \times Q_{\mathcal{A}} \times Q_{\mathcal{A}}$ is a transition relation, and Acc is an acceptance condition. A computation of $\mathcal{A}$ on a Σ-labeled tree t is a function $\phi : \{l, r\}^* \to Q_{\mathcal{A}}$ such that $\phi(\epsilon) \in Q_I$, and $\forall v \in \{l, r\}^* : (\phi(v), t(v), \phi(v \cdot l), \phi(v \cdot r)) \in \delta$.

The Büchi acceptance conditions are given by a set $F \subseteq Q_{\mathcal{A}}$. We say that ϕ is accepting if for each branch $\pi = v_0, v_1, \dots$ there is a state $f \in F$ such that the sequence $\phi(v_0), \phi(v_1), \dots$ contains infinitely many occurrences of f. The parity acceptance conditions are given by a function $\mathbb{C} : Q_{\mathcal{A}} \to \mathbb{N}$. We say that ϕ is accepting if for each branch $\pi = v_0, v_1, \dots$ the maximal number which appears infinitely often in $\mathbb{C}(\phi(v_0)), \mathbb{C}(\phi(v_1)), \dots$ is even.

We denote the set of all accepting computations of $\mathcal{A}$ on t by $ACC(\mathcal{A}, t)$. The language of $\mathcal{A}$ is defined as $L(\mathcal{A}) := \{t \mid ACC(\mathcal{A}, t) \neq \emptyset\}$.

A state $q \in Q_{\mathcal{A}}$ of $\mathcal{A}$ is *useful* if it appears on an accepting computation.

Given an automaton $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, Q_I, \delta, Acc)$ and a state $q \in Q_{\mathcal{A}}$, $\mathcal{A}_q$ is defined as $\mathcal{A}_q := (Q_{\mathcal{A}}, \Sigma, \{q\}, \delta, Acc)$, by replacing the set of initial states of $\mathcal{A}$ by $\{q\}$.

We will use PTA for Parity Tree Automata, BTA for Büchi Tree Automata, PWA for Parity Word Automata, and BWA for Büchi Word Automata.

Degree of Ambiguity of an Automaton. We denote by $|X|$ the cardinality of a set X. An automaton $\mathcal{A}$ is k-ambiguous if $|ACC(\mathcal{A}, t)| \leq k$ for all $t \in L(\mathcal{A})$. $\mathcal{A}$ is unambiguous if it is 1-ambiguous. $\mathcal{A}$ is boundedly ambiguous if there is $k \in \mathbb{N}$ such that $\mathcal{A}$ is k-ambiguous, $\mathcal{A}$ is finitely ambiguous if $ACC(\mathcal{A}, t)$ is finite for all t, $\mathcal{A}$ is countably ambiguous if $ACC(\mathcal{A}, t)$ is countable for all t. The set $ACC(\mathcal{A}, t)$ is definable in the Monadic Second-Order logic (MSO). The results of Bárány et al. in [1] imply that every uncountable MSO-definable (in the full-binary tree) set has cardinality $2^{\aleph_0}$.

The degree of ambiguity of $\mathcal{A}$ (notation $da(\mathcal{A})$) is defined by $da(\mathcal{A}) := k$ if $\mathcal{A}$ is k-ambiguous and either $k = 1$ or $\mathcal{A}$ is not $k-1$ ambiguous, $da(\mathcal{A}) := finite$ if $\mathcal{A}$ is finitely ambiguous and not boundedly ambiguous, $da(\mathcal{A}) := \aleph_0$ if $\mathcal{A}$ is countably ambiguous and not finitely ambiguous, and $da(\mathcal{A}) := 2^{\aleph_0}$ if $\mathcal{A}$ is not countably ambiguous.

3 Non-Emptiness Problem for Intersection of PTA

In this section we will prove that deciding the non-emptiness of the intersection of k PTA is in co-NP. Our proof relies on a reduction from k-dimensional parity tree automata to k-dimensional parity games. We first recall k-dimensional parity games [2] and introduce k-dimensional parity automata.

▶ **Definition 2.** *A k-dimensional parity game is a tuple $G = (S_1, S_2, E, \mathbb{P})$, where (S_1, S_2, E) is a directed bipartite graph: S_i the set of states of Player i, E a set of edges such that each state $s \in S_1 \cup S_2$ has at least one outgoing edge $(s, s') \in E$; and $\mathbb{P} : S_1 \cup S_2 \to \mathbb{N}^k$ a (priority) function. The game starts at some state $s \in S$. The players construct an infinite sequence of states (called a play) as follows: Let s be the last state in the sequence. If $s \in S_i$, then Player i chooses an edge $(s, s') \in E$ and the state s' is added to the sequence. Since each state has at least one successor, the sequence can always be continued.*

Let $s_1, s_2, \ldots$ be the play which is constructed by the selections of the two players, and let $\mathbb{P}(s_1), \mathbb{P}(s_1), \ldots$ be a sequence of priorities in $(\mathbb{N}^k)^\omega$. We say Player 1 wins the play if for every $i \leq k$ the maximal value which is seen infinitely often in the i-th coordinates is even. Otherwise, we say that Player 2 wins the play.

A strategy for Player i specifies for each sequence $s_1, \ldots s_m$ where $s_m \in S_i$, the next state s' such that $(s_m, s') \in E$. A play is consistent with a strategy of Player i if each move of Player i in the play is according to the strategy.

The winning region of Player 1 is a subset $S' \subseteq S_1 \cup S_2$, for which there exists a strategy of Player 1 such that each play from $s' \in S'$ which is consistent with it is winning for Player 1. The winning region of Player 2 is defined similarly.

Notice that each play $s_1, s_2, \ldots$ could equivalently be represented by a sequence of edges $e_1, e_2, \ldots$ such that $e_i = (s_i, s_{i+1})$.

▶ **Definition 3** (k-dimensional PTA)**.** *A k-dimensional PTA is a tuple $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ where Q, Σ, Q_I, and δ are as in PTA, and $\mathbb{C} : Q \to \mathbb{N}^k$ is a function which assigns a k-dimensional color vector to each state in Q. A computation ϕ of $\mathcal{A}$ on t is accepting if for each branch $\pi = v_0, v_1, \ldots$ and each coordinate $i \leq k$, the maximal color which occurs infinitely often in the i-th coordinate of $\mathbb{C}(\phi(v_0)), \mathbb{C}(\phi(v_1)), \ldots$ is even.*

▶ **Theorem 4** (Chatterjee, Henzinger and Piterman [2])**.** *Let $G = (S_1, S_2, E, \mathbb{P})$ be a k-dimensional parity game, for $k > 1$. The problem of deciding whether a node $s \in S_1$ is in the winning region of Player 1 is co-NP-complete.*

We use Theorem 4 to obtain the following result regarding the non-emptiness problem of k-dimensional parity tree automata:

▶ **Proposition 5.** *(1) The non-emptiness problem for k-dimensional PTA is in co-NP. (2) The non-emptiness problem for deterministic 2-dimensional PTA is co-NP-hard.*

Proof. (1) We use the standard reduction from the emptiness problem for automata to games; it works also for k-dimensional parity conditions. Given a k-dimensional PTA $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$, we define a k-dimensional parity game $G(\mathcal{A}) = (S_1, S_2, E, \mathbb{P})$ as follows:

- $S_1 = Q$
- $S_2 = \delta$
- $(q, (p, a, p_1, p_2)) \in E$ iff $(p, a, p_1, p_2) \in \delta$ and $p = q$
- $((p, a, p_1, p_2), q) \in E$ iff $q = p_1$ or $q = p_2$
- $\forall s \in S_1 : \mathbb{P}(s) := \mathbb{C}(s)$, and $\forall s \in S_2 : \mathbb{P}(s) := (\underbrace{0, \ldots, 0}_{k \text{ times}})$

Recall that for $s \in Q$ an automaton $\mathcal{A}_s$ is defined by replacing the set of initial states of $\mathcal{A}$ by $\{s\}$. For every state $s \in Q$, $L(\mathcal{A}_s)$ is non-empty iff Player 1 has a winning strategy from s. Hence, by Theorem 4 we conclude that deciding the non-emptiness of $\mathcal{A}$ is in co-NP.

(2) We prove this using a reduction from the problem of deciding whether Player 1 has a winning strategy from state $s \in S_1$ in a 2-dimensional parity game. Since this problem is co-NP-hard (by Theorem 4), the result will follow.

Let $G = (S_1, S_2, E, \mathbb{P})$ be a 2-dimensional parity game with priority function $\mathbb{P} : S_1 \cup S_2 \to \mathbb{N}^2$. We will assume without restriction that:

- The out degree of each node in G is 2.
- $\forall s \in S_2 : \mathbb{P}(s) = (0, 0)$.

For each node $s \in S_1$, we denote one of its successors by $E_a(s)$ and the other by $E_b(s)$, and for each node $s \in S_2$ we denote one of its successors by $E_l(s)$ and the other by $E_r(s)$. We refer to the selection of edge $(s', E_h(s')) \in E$ for $h \in \{a, b, l, r\}$ by Player i as the h move of Player i. Notice that a sequence of states in the game could equivalently be represented using a sequence of h moves.

We construct a deterministic 2-dimensional parity automaton $\mathcal{A} := \mathcal{A}(G) = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ as follows:

- $Q := S_1$
- $\Sigma := \{a, b\}$
- $Q_I := S_1$
- $\forall c \in \Sigma, \forall q \in Q : \delta(q, c) = (E_l(E_c(q)), E_r(E_c(q)))$
- $\mathbb{C} := \mathbb{P}|_{S_1}$ (the restriction of $\mathbb{P}$ onto S_1)

We will prove that Player 1 has a winning strategy from a state $s \in S_1$ iff $\mathcal{A}_s$ is non-empty. This implies the co-NP-hardness of deciding whether $\mathcal{A}_s$ is non-empty.

$\Rightarrow$: A strategy of Player 1 from a state s can be represented as a function from a sequence of moves of Player 2 to a move of Player 1. Assume Player 1 has a winning strategy $strategy_1 : \{l, r\}^* \to \{a, b\}$ from a state s. We will use $strategy_1$ to construct a tree $t \in L(\mathcal{A})$ and an accepting computation $\phi \in ACC(\mathcal{A}_s, t)$.

Define $t(d_1 \dots d_n) := stategy_1(d_1, \dots, d_n)$. Define a computation ϕ on t as $\phi(\epsilon) := s$ and $\phi(v \cdot d_{n+1}) := E_{d_{n+1}}(E_{t(v)}(\phi(v)))$ for $v \in \{l, r\}^n$ and $d_{n+1} \in \{l, r\}$. It is easy to verify that ϕ is a computation of $\mathcal{A}_s$ on t (this holds even for the non-winning strategies of Player 1).

A proof that ϕ is an accepting computation of $\mathcal{A}_s$ on t is also simple. Let $\pi = v_1, v_2, \dots$ be a tree branch such that $v_1 = \epsilon$ and $\forall i \in \mathbb{N} : v_{i+1} = v_i \cdot d_i$ where $d_i \in \{l, r\}$. By definition of ϕ, $\phi(\pi)$ corresponds to the states of Player 1 in a play which is consistent with winning strategy $strategy_1$, and with Player 2 choosing move d_i on his i-th turn. This play is winning for Player 1; hence, it satisfies the 2-dimensional parity condition; therefore, $\phi(\pi)$ satisfies the acceptance conditions. Since π was an arbitrary branch, this implies that $\phi \in ACC(\mathcal{A}_s, t)$, and $\mathcal{A}_s$ is non-empty.

$\Leftarrow$: Assume that $\mathcal{A}_s$ is non-empty. Therefore, there exists $\phi \in ACC(\mathcal{A}_s, t)$. We use t to define a strategy $strategy_1 : \{l, r\}^* \to \{a, b\}$ for Player 1, by $strategy_1(d_1, \dots, d_n) := t(d_1 \dots d_n)$. We leave to the reader the verification that $strategy_1$ is winning for Player 1 from s. ◀

We will now proceed to show the connection between k-dimensional automata and the intersection of parity automata.

▶ **Definition 6** (Product of PTA). *Given two PTA $\mathcal{A} = (Q_\mathcal{A}, \Sigma, Q_I^\mathcal{A}, \delta_\mathcal{A}, \mathbb{C}_\mathcal{A})$ and $\mathcal{B} = (Q_\mathcal{B}, \Sigma, Q_I^\mathcal{B}, \delta_\mathcal{B}, \mathbb{C}_\mathcal{B})$ we define the 2-dimensional PTA $\mathcal{A} \times \mathcal{B} := (Q_\times, \Sigma, Q_I^\times, \delta_\times, \mathbb{C}_\times)$ where:*

- $Q_\times := Q_\mathcal{A} \times Q_\mathcal{B}$
- $Q_I^\times := Q_I^\mathcal{A} \times Q_I^\mathcal{B}$
- $((p, q), a, (p_l, q_l), (p_r, q_r)) \in \delta_\times$ *iff* $(q, a, q_l, q_r) \in \delta_\mathcal{A}$ *and* $(p, a, p_l, p_r) \in \delta_\mathcal{B}$
- $\forall (q, p) \in Q_\mathcal{A} \times Q_\mathcal{B} : \mathbb{C}_\times(q, p) = (\mathbb{C}_\mathcal{A}(q), \mathbb{C}_\mathcal{B}(p))$

The product automata of k PTA is defined similarly, as a k-dimensional PTA.

▶ **Lemma 7.** *Let $\mathcal{A}_1, \dots, \mathcal{A}_k$ be parity tree automata, and define $\mathcal{A}_\times$ as the k-dimensional product automaton $\mathcal{A}_1 \times \dots \times \mathcal{A}_k$. Then $L(\mathcal{A}_1) \cap \dots \cap L(\mathcal{A}_k) \neq \emptyset$ iff $L(\mathcal{A}_\times) \neq \emptyset$.*

Proof. $\Rightarrow$: Assume that $L(\mathcal{A}_1) \cap \cdots \cap L(\mathcal{A}_k) \neq \emptyset$. Therefore, there is an infinite tree $t \in L(\mathcal{A}_1) \cap \cdots \cap L(\mathcal{A}_k)$ and accepting computations $\phi_i \in ACC(\mathcal{A}_i, t)$ for all $1 \leq i \leq k$. Define a computation $\phi_\times$ such that $\phi_\times(u) := (\phi_1(u), \ldots, \phi_k(u))$ for all $u \in \{l, r\}^*$. By definition of $\mathcal{A}_\times$ we conclude that $\phi_\times \in ACC(\mathcal{A}_\times, t)$, and therefore $t \in L(\mathcal{A}_\times)$ and $L(\mathcal{A}_\times) \neq \emptyset$

$\Leftarrow$: Assume that $L(\mathcal{A}_\times) \neq \emptyset$. Therefore, there is a tree $t \in L(\mathcal{A}_\times)$ and an accepting computation $\phi_\times \in ACC(\mathcal{A}_\times, t)$. For each $u \in \{l, r\}^*$ there are k states $q_1^u, \ldots, q_k^u$ such that $\phi_\times(u) = (q_1^u, \ldots, q_k^u)$. Define a computation ϕ_i by $\phi_i(u) := q_i^u$. By definition of $\mathcal{A}_\times$ it follows that $\phi_i \in ACC(\mathcal{A}_i, t)$. We conclude that for each automaton $\mathcal{A}_i$ there is an accepting computation ϕ_i of $\mathcal{A}_i$ on t, and therefore $t \in L(\mathcal{A}_1) \cap \cdots \cap L(\mathcal{A}_k)$ and we obtain $L(\mathcal{A}_1) \cap \cdots \cap L(\mathcal{A}_k) \neq \emptyset$. ◀

▶ **Lemma 8.** *For every $k \in \mathbb{N}$, the problem of deciding whether the intersection of k PTA is non-empty is in co-NP.*

Proof. Given k PTA $\mathcal{A}_1, \ldots, \mathcal{A}_k$, the product automaton $\mathcal{A}_1 \times \cdots \times \mathcal{A}_k$ can be computed in polynomial time in the size of $\mathcal{A}_1, \ldots, \mathcal{A}_k$. By Lemma 7 we conclude that deciding the non-emptiness of $L(\mathcal{A}_1) \cap \cdots \cap L(\mathcal{A}_k)$ is equivalent to deciding the non-emptiness of $L(\mathcal{A}_1 \times \cdots \times \mathcal{A}_k)$. $\mathcal{A}_1 \times \cdots \times \mathcal{A}_k$ is a k-dimensional PTA and therefore, by Proposition 5(1), this problem is in co-NP. ◀

4 co-NP-hardness of Deciding the Degree of Ambiguity of a PTA

In this section we will use the results of Sect. 3 and prove the co-NP hardness lower bounds stated in Theorem 1. We will first prove the co-NP-hardness of deciding whether a PTA is ambiguous, and then use a polynomial time reduction to show that co-NP-hardness holds for other ambiguity degree problems (Proposition 10).

▶ **Lemma 9.** *The problem of deciding whether a PTA is ambiguous is co-NP-hard.*

Proof. We will use a reduction from the problem of deciding the non-emptiness of deterministic 2-dimensional PTA (which is co-NP-hard, by Proposition 5(2)), to the problem of deciding whether a PTA is ambiguous.

Let $\mathcal{D} = (Q, \Sigma, q_{init}, \delta, \mathbb{C})$ be a deterministic 2-dimensional PTA. Define $\mathcal{D}_i := (Q, \Sigma, q_{init}, \delta, \mathbb{C}_i)$ for $i = 1, 2$ as the deterministic PTA obtained from $\mathcal{D}$ by defining a coloring function $\mathbb{C}_i$ such that $\mathbb{C}_i(q)$ returns the i-th coordinate of $\mathbb{C}(q)$ for each $q \in Q$.

Let $\mathcal{D}_2'$ be an automaton which is isomorphic to $\mathcal{D}_2$, and does not share common states with $\mathcal{D}_1$ (this could be achieved by renaming the states of $\mathcal{D}_2$). Since $\mathcal{D}_1$ and $\mathcal{D}_2'$ are deterministic, it easily follows that $L(\mathcal{D}) \neq \emptyset$ iff $L(\mathcal{D}_1) \cap L(\mathcal{D}_2') \neq \emptyset$.

Given two PTA $\mathcal{A} = (Q_\mathcal{A}, \Sigma, Q_I^\mathcal{A}, \delta_\mathcal{A}, \mathbb{C}_\mathcal{A})$ and $\mathcal{B} = (Q_\mathcal{B}, \Sigma, Q_I^\mathcal{B}, \delta_\mathcal{B}, \mathbb{C}_\mathcal{B})$ such that $Q_\mathcal{A} \cap Q_\mathcal{B} = \emptyset$, we define the parity automaton $\mathcal{A} \cup \mathcal{B} := (Q_\mathcal{A} \cup Q_\mathcal{B}, \Sigma, Q_I^\mathcal{A} \cup Q_I^\mathcal{B}, \delta_\mathcal{A} \cup \delta_\mathcal{B}, \mathbb{C}_\mathcal{A} \cup \mathbb{C}_\mathcal{B})$. Notice that $L(\mathcal{A} \cup \mathcal{B}) = L(\mathcal{A}) \cup L(\mathcal{B})$.

We will prove that $da(\mathcal{D}_1 \cup \mathcal{D}_2') > 1$ iff $L(\mathcal{D}_1) \cap L(\mathcal{D}_2')$ is non-empty:

$\Rightarrow$: Assume that $da(\mathcal{D}_1 \cup \mathcal{D}_2') > 1$. Then, there exist two accepting computations $\phi_1, \phi_2 \in ACC(\mathcal{D}_1 \cup \mathcal{D}_2', t)$. Since $\mathcal{D}_1$ and $\mathcal{D}_2'$ are deterministic, each of them has at most one accepting computation, and therefore, if $\phi_1 \in ACC(\mathcal{D}_1, t)$ then $\phi_2 \in ACC(\mathcal{D}_2', t)$. Hence, $t \in L(\mathcal{D}_1) \cap L(\mathcal{D}_2')$.

$\Leftarrow$: Assume $t \in L(\mathcal{D}_1) \cap L(\mathcal{D}_2')$. Therefore, there are computations $\phi_1 \in ACC(\mathcal{D}_1, t)$, $\phi_2 \in ACC(\mathcal{D}_2', t)$. By definition of $\mathcal{D}_1 \cup \mathcal{D}_2'$ we have $\phi_1, \phi_2 \in ACC(\mathcal{D}_1 \cup \mathcal{D}_2', t)$. Since $\mathcal{D}_1$ and $\mathcal{D}_2'$ have no common states, we have $\phi_1 \neq \phi_2$ and therefore $da(\mathcal{D}_1 \cup \mathcal{D}_2') > 1$.

We conclude that $L(\mathcal{D}) \neq \emptyset$ iff $da(\mathcal{D}_1 \cup \mathcal{D}_2') > 1$, and therefore deciding whether a PTA is ambiguous is co-NP-hard. ◀

▶ **Proposition 10.** *The problem of deciding whether a PTA is not countably ambiguous (respectively, is not finitely ambiguous, or is not boundedly ambiguous) is co-NP-hard.*

Proof. We will prove it using a reduction from the problem of deciding whether a parity tree automaton is ambiguous, which is co-NP-hard by Lemma 9.

Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ be a PTA, and let $a \in \Sigma$. Construct an automaton $\mathcal{B} := (Q_\mathcal{B}, \Sigma, Q_I^\mathcal{B}, \delta_\mathcal{B}, \mathbb{C}_\mathcal{B})$, where:

- $Q_\mathcal{B} := Q \cup \{f\}$ and $f \notin Q$ is a new state,
- $Q_I^\mathcal{B} := \{f\}$,
- $\delta_\mathcal{B} = \delta \cup \{(f, a, f, q) \mid q \in Q_I\}$,
- $\mathbb{C}_\mathcal{B}(f) := 0$ and for all $q \in Q : \mathbb{C}_\mathcal{B}(q) := \mathbb{C}(q)$.

▷ Claim 11. **1.** If $\mathcal{A}$ is ambiguous, then $\mathcal{B}$ is not countably ambiguous.
2. If $\mathcal{B}$ is ambiguous, then $\mathcal{A}$ is ambiguous.

Proof. (1) Assume that $\mathcal{A}$ is ambiguous. Therefore, there is a tree $t \in L(\mathcal{A})$ and two computations $\phi_1, \phi_2 \in ACC(\mathcal{A}, t)$ such that $\phi_1 \neq \phi_2$. Let t' be a tree such that $\forall v \in l^* : t'(v) = a$ and $t'_{\geq v \cdot r} = t$. For each $S \subseteq \mathbb{N}$, define a computation ϕ_S by

$$\phi_S(u) := \begin{cases} f & u \in l^* \\ \phi_1(v) & u = l^i \cdot r \cdot v \text{ for } i \in S \\ \phi_2(v) & u = l^i \cdot r \cdot v \text{ for } i \notin S \end{cases}$$

It is easy to see that $\forall S \subseteq \mathbb{N} : \phi_S \in ACC(\mathcal{B}, t')$, and that $\forall S_1, S_2 \subseteq \mathbb{N} : S_1 \neq S_2 \to \phi_{S_1} \neq \phi_{S_2}$. Therefore, $ACC(\mathcal{B}, t')$ is not countable, and $\mathcal{B}$ is not countably ambiguous.

(2) Assume that $\mathcal{B}$ is ambiguous. Therefore, there is a tree $t \in L(\mathcal{B})$ and computations $\phi', \phi'' \in ACC(\mathcal{B}, t)$ such that $\phi' \neq \phi''$. Let $v \in \{l, r\}^*$ such that $\phi'(v) \neq \phi''(v)$. By definition of $\mathcal{B}$ we have $\forall u \in l^* : \phi'(u) = \phi''(u) = f$, and therefore, $v \notin l^*$. Hence, there is $w \in l^* \cdot r$ such that $v \geq w$. Let $\phi'_{\geq w}$ and $\phi''_{\geq w}$ be the restrictions of ϕ' and ϕ'', respectively, on $t_{\geq w}$. It is clear that $\phi'_{\geq w} \neq \phi''_{\geq w}$. By definition of $\mathcal{B}$ we obtain $\phi'_{\geq w}, \phi''_{\geq w} \in ACC(\mathcal{A}, t_{\geq w})$, and therefore $\mathcal{A}$ is ambiguous. ◁

The parity tree automaton $\mathcal{B}$ can be constructed in a polynomial time in the size of $\mathcal{A}$. By Claim 11 we obtain that the following conditions are equivalent:

- $\mathcal{A}$ is not unambiguous
- $\mathcal{B}$ is not countably ambiguous
- $\mathcal{B}$ is not finitely ambiguous
- $\mathcal{B}$ is not boundedly ambiguous

Therefore, there are polynomial time reductions from the problem of deciding whether a PTA is not unambiguous (which is co-NP-hard, by Lemma 9), to the problems of deciding whether a PTA is not countably ambiguous/not finitely ambiguous/not boundedly ambiguous. Hence, those problems are co-NP-hard. ◀

5 Degree of Ambiguity for Parity Automata on ω-words

In this section we will present structural characterizations and polynomial algorithms for deciding the degree of ambiguity of parity automata on ω-word (PWA). These characterizations and algorithms are derived from similar characterizations for Büchi automata on ω-words (BWA) given in [8, 10]. Throughout this section we will assume all states of the automata are useful (it is computable in polynomial time whether a state of a PWA/BWA is useful).

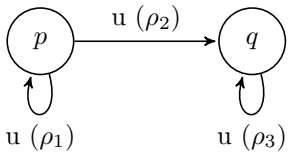

Figure 2 Forbidden Pattern for Finite Ambiguity of PWA. The runs ρ_1 and ρ_2 are distinct, $\mathbb{C}(q)$ is even, and $\mathbb{C}(q)$ is maximal among the colors which $\mathbb{C}$ assigns to the states in ρ_3.

The next definition and theorem are taken from [8, 10]. They provide a forbidden pattern characterization of the degrees of ambiguity of BWA.

▶ **Definition 12** (Forbidden pattern for BWA). *Let $\mathcal{B}$ be a BWA such that all its states are useful.*

- *$\mathcal{B}$ contains a* forbidden pattern for countable ambiguity*, if there is a final state f and there are two distinct runs of $\mathcal{B}_f$ on the same word u from f to f.*
- *$\mathcal{B}$ contains a* forbidden pattern for finite ambiguity*, if it contains the forbidden pattern for countable ambiguity or there is a final state f, a state $q \neq f$, and a word u such that there are runs of $\mathcal{B}_q$ on u from q to q and from q to f, and a run of $\mathcal{B}_f$ on u from f to f.*
- *$\mathcal{B}$ contains a* forbidden pattern for bounded ambiguity*, if there are distinct states p, q such that for a (finite) word u, there are runs of $\mathcal{B}_p$ on u from p to p and from p to q, and there is a run of $\mathcal{B}_q$ on u from q to q.*

▶ **Theorem 13** (Structural characterization). *Let $\mathcal{B}$ be a BWA.*

1. *$\mathcal{B}$ has uncountably many accepting runs on some ω-word iff $\mathcal{B}$ contains the forbidden pattern for countable ambiguity.*
2. *$\mathcal{B}$ has infinitely many accepting runs on some ω-word iff $\mathcal{B}$ contains the forbidden pattern for finite ambiguity.*
3. *$\mathcal{B}$ is not boundedly ambiguous iff it contains the forbidden pattern for bounded ambiguity.*

Now, we define forbidden patterns for PWA. Then, Proposition 16 proves their correctness.

▶ **Definition 14** (Forbidden pattern for PWA). *Let $\mathcal{A}$ be a PWA such that all its states are useful.*

- *$\mathcal{A}$ contains a* forbidden pattern for countable ambiguity*, if there is a state q such that $\mathbb{C}(q)$ is even, and two distinct runs ρ_1 and ρ_2 of $\mathcal{A}_q$ from q to q on the same finite word y such that $\forall p \in \rho_1 : \mathbb{C}(q) \geq \mathbb{C}(p)$ and $\forall p \in \rho_2 : \mathbb{C}(q) \geq \mathbb{C}(p)$.*
- *$\mathcal{A}$ contains a* forbidden pattern for finite ambiguity*, if there is a state q such that $\mathbb{C}(q)$ is even, a state p (which can be equal to q), and a word y such that there is a run ρ_1 of $\mathcal{A}_p$ on y from p to p, a run ρ_2 of $\mathcal{A}_p$ on y from p to q, and a run ρ_3 of $\mathcal{A}_q$ on y from q to q such that $\max\{\mathbb{C}(p') \mid p' \in \rho_3\}$ is even, and $\rho_1 \neq \rho_2$.*
- *$\mathcal{A}$ contains a* forbidden pattern for bounded ambiguity*, if there are two states $p \neq q$ and a finite word y such that there are runs of $\mathcal{A}_p$ on y from p to p and from p to q, and a run of $\mathcal{A}_q$ on y from q to q.*

First, we prove that the forbidden patterns provide sufficient conditions for the corresponding ambiguity.

▶ **Lemma 15** (Sufficient conditions for degrees of ambiguity). *Let $\mathcal{A}$ be a PWA.*

1. *If $\mathcal{A}$ contains the forbidden pattern for countable ambiguity, then it is not countably ambiguous.*
2. *If $\mathcal{A}$ contains the forbidden pattern for finite ambiguity, then it is not finitely ambiguous.*
3. *If $\mathcal{A}$ contains the forbidden pattern for bounded ambiguity, then it is not boundedly ambiguous.*

Proof. (1) Let y be a finite word, and let q be a state of $\mathcal{A}$ such that $\mathbb{C}(q)$ is even and there are two distinct runs ρ_1 and ρ_2 of $\mathcal{A}_q$ from q to q on y where $\forall p \in \rho_1 : \mathbb{C}(q) \geq \mathbb{C}(p)$ and $\forall p \in \rho_2 : \mathbb{C}(q) \geq \mathbb{C}(p)$.

We will show that there are infinitely many accepting runs of $\mathcal{A}_q$ on y^ω. Denote by ρ'_1 and ρ'_2 the runs ρ_1 and ρ_2, respectively, with the last state removed. For each $B \subseteq \mathbb{N}$, let $\rho^B := \rho_0^B \cdot \rho_1^B \cdots \in Q^\omega$, where $\rho_i^B := \rho'_1$ if $i \in B$, and $\rho_i^B := \rho'_2$ otherwise. It is clear that ρ^B is a computation of $\mathcal{A}_q$ on y^ω. Notice that q occurs infinitely often in ρ^B, and by the definition of ρ_1 and ρ_2 we conclude that $\mathbb{C}(q)$ is the maximal color which is assigned to a state in ρ^B.

Let $B_1, B_2 \subseteq \mathbb{N}$ such that $\exists b \in B_1 \setminus B_2$. We obtain $\rho_b^{B_1} = \rho'_1 \neq \rho'_2 = \rho_b^{B_2}$, and therefore $\rho^{B_1} \neq \rho^{B_2}$. There are uncountably many subsets of $\mathbb{N}$, and therefore there are uncountably many accepting computation of $\mathcal{A}_q$ on y^ω. Since q is a useful state we conclude that $\mathcal{A}$ is uncountably ambiguous.

(2) Let y be a finite word, q a state such that $\mathbb{C}(q)$ is even, and p a state such that there are three runs on y: a run ρ_1 of $\mathcal{A}_p$ from p to p, a run ρ_2 of $\mathcal{A}_p$ from p to q, and a run ρ_3 of $\mathcal{A}_q$ from q to q such that $\max\{\mathbb{C}(p') \mid p' \in \rho\}$ is even and $\rho_1 \neq \rho_2$.

We will show that there are infinitely many accepting runs of $\mathcal{A}_p$ on y^ω. Denote by ρ'_1, ρ'_2 and ρ'_3 the runs ρ_1, ρ_2 and ρ_3, respectively, with the last state removed. For each $k \in \mathbb{N}$, define a computation $\rho^k := \rho_0^k \cdot \rho_1^k \ldots$, where $\rho_i^k := \begin{cases} \rho'_1 & i < k \\ \rho'_2 & i = k \\ \rho'_3 & i > k \end{cases}$

It is easy to verify that ρ^k is a run of $\mathcal{A}_p$ on y^ω. Notice that q occurs infinitely often in ρ^k. Since each state which occurs infinitely often in ρ^k is in ρ'_3, we conclude that $\mathbb{C}(q)$ is the maximal color among the colors which are assigned to states which occurs infinitely often in ρ^k; hence, ρ^k is accepting. Let $k_1 < k_2 \in \mathbb{N}$. We obtain $\rho_{k_1}^{k_1} = \rho'_2 \neq \rho'_1 = \rho_{k_1}^{k_2}$, and therefore, $\rho^{k_1} \neq \rho^{k_2}$ and we conclude that $\mathcal{A}$ is not finitely ambiguous.

(3) Let y be a finite word such that there are two states $p \neq q$, and three runs: a run ρ_1 of $\mathcal{A}_p$ on y from p to p, a run ρ_2 of $\mathcal{A}_p$ from p to q, and a run ρ_3 of $\mathcal{A}_q$ from q to q.

Since q is a useful state, there is an ω-word $\widehat{y} \in L(\mathcal{A}_q)$, and an accepting run $\widehat{\rho}$ of $\mathcal{A}_q$ on $\widehat{y}$. We will prove that for each $k \in \mathbb{N}$, there are at least k accepting runs of $\mathcal{A}_p$ on $y^k \cdot \widehat{y}$.

Denote by ρ'_1, ρ'_2 and ρ'_3 the runs ρ_1, ρ_2 and ρ_3, respectively, with the last state removed. For each $0 \leq j < k$ we define a run $\rho^j := \rho_0^j \cdot \rho_1^j \ldots \rho_k^j$, where

$$\rho_i^j := \begin{cases} \rho'_1 & i < j \\ \rho'_2 & i = j \\ \rho'_3 & i > j \end{cases}$$

It is easy to verify that ρ^j is a run of $\mathcal{A}_p$ on y^k from p to q, and therefore, $\rho^j \cdot \widehat{\rho}$ is an accepting run of $\mathcal{A}_p$ on $y^k \cdot \widehat{y}$. Moreover, for each $j_1 < j_2 < k$ we have $\rho_{j_1}^{j_1} = \rho'_2 \neq \rho'_1 = \rho_{j_1}^{j_2}$, and therefore, $\rho^{j_1} \neq \rho^{j_2}$. We conclude that for each $k \in \mathbb{N}$ there are at least k different accepting computation of $\mathcal{A}_p$ on $y^k \cdot \widehat{y}$, and therefore $\mathcal{A}_p$ is not boundedly ambiguous. Since p is useful, we conclude that $\mathcal{A}$ is not boundedly ambiguous. ◀

▶ **Proposition 16** (Structural characterization). *Let $\mathcal{A}$ be a PWA.*

1. *$\mathcal{A}$ has uncountably many accepting runs on some ω-word iff $\mathcal{A}$ contains the forbidden pattern for countable ambiguity.*
2. *$\mathcal{A}$ has infinitely many accepting runs on some ω-word iff $\mathcal{A}$ contains the forbidden pattern for finite ambiguity.*

3. *$\mathcal{A}$ is not boundedly ambiguous iff it contains the forbidden pattern for bounded ambiguity.*

The $\Leftarrow$ direction of the proposition was proved in Lemma 15. To prove the $\Rightarrow$ direction of Proposition 16 we will use the standard reduction of PWA to BWA.

Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ be a PWA, and let $\mathbb{C}_{even} := \{\mathbb{C}(q) \mid q \in Q \text{ and } \mathbb{C}(q) \text{ is even}\}$.

Define a BWA $\mathcal{B} := (Q', \Sigma, Q'_I, \delta', F)$, where:

- $Q' := Q \cup \{(c, p) \in \mathbb{C}_{even} \times Q \mid c \geq \mathbb{C}(p)\}$
- $Q'_I := Q_I \cup \{(c, p) \in Q' \mid p \in Q_I\}$
- δ' is the union of the following sets:
  - δ
  - $\{(p, a, (c, p')) \mid (p, a, p') \in \delta, \mathbb{C}(p) > c \text{ and } (c, p') \in Q'\}$
  - $\{((c, p), a, (c, p')) \mid (p, a, p') \in \delta \text{ and } (c, p') \in Q'\}$
- $F := \{(c, q) \mid \mathbb{C}(q) = c\}$

Proposition 16 immediately follows from Theorem 13, Lemma 17 and Lemma 18.

▶ **Lemma 17.** $|ACC(\mathcal{A}, y)| \leq |ACC(\mathcal{B}, y)|$

Proof. By the definition of PWA, for each accepting run $\rho := p_0, \dots p_i, \dots$ of $\mathcal{A}$ on y there is $i \in \mathbb{N}$ such that p_i occurs infinitely often in ρ, $c := \mathbb{C}(p_i)$ is even and $\forall j > i : \mathbb{C}(p_i) \geq \mathbb{C}(p_j)$.

If $\mathbb{C}(p_i) \geq \mathbb{C}(p)$ for all $p \in \rho$ then define $g(\rho) := (c, p_0), (c, p_1), \dots$. Otherwise, let k be such that $\mathbb{C}(p_k) > c$ and $\forall j > k : c \geq \mathbb{C}(p_j)$, and define $g(\rho) := p_0, \dots p_k, (c, p_{k+1}), (c, p_{k+2}), \dots$. By the definition of $\mathcal{B}$ we conclude that $g(\rho)$ is an accepting computation of $\mathcal{B}$ on y, since (c, p_i) occurs infinitely often in $g(\rho)$. It is easy to verify that g is injective, and therefore, we obtain $|ACC(\mathcal{A}, y)| \leq |ACC(\mathcal{B}, y)|$. ◀

▶ **Lemma 18.** **1.** *If $\mathcal{B}$ contains the forbidden pattern for countable ambiguity (of BWA) then $\mathcal{A}$ contains the forbidden pattern for countable ambiguity (of PWA).*

2. *If $\mathcal{B}$ contains the forbidden pattern for finite ambiguity (of BWA) then $\mathcal{A}$ contains the forbidden pattern for finite ambiguity (of PWA).*

3. *If $\mathcal{B}$ contains the forbidden pattern for bounded ambiguity (of BWA) then $\mathcal{A}$ contains the forbidden pattern for bounded ambiguity (of PWA).*

Proof. The lemma is proved by inspecting the transition relations of $\mathcal{A}$ and the corresponding Büchi automaton $\mathcal{B}$.

1. By definition of the forbidden pattern of countable ambiguity we conclude that there is a final state f and two distinct runs ρ_1 and ρ_2 of $\mathcal{B}_f$ on the same finite word y from f to f. By the definition of $\mathcal{B}$ there is a state $q \in Q$ with even color $c := \mathbb{C}(q)$ such that $f = (c, q)$, the run ρ_1 is of the form $(c, p_0), \dots, (c, p_n)$ and the run ρ_2 is of the form $(c, q_0), \dots, (c, q_n)$, where $p_0 = p_n = q_0 = q_n = q$.

The runs ρ_1 and ρ_2 are distinct, and therefore there is $0 < i < n$ such that $p_i \neq q_i$. We conclude that $q_0, \dots, q_n$ and $p_0, \dots, p_n$ are two distinct runs of $\mathcal{A}$ on y from q to q such that $\mathbb{C}(q)$ is even, $\forall p \in \rho_1 : \mathbb{C}(q) \geq \mathbb{C}(p)$ and $\forall p \in \rho_2 : \mathbb{C}(q) \geq \mathbb{C}(p)$. Therefore, $\mathcal{A}$ has a forbidden pattern for countable ambiguity.

2. If $\mathcal{B}$ contains the forbidden pattern for countable ambiguity, then by (1) we conclude that $\mathcal{A}$ contains the forbidden pattern for countable ambiguity. Hence, there is a state q such that $\mathbb{C}(q)$ is even, and there are two distinct runs ρ_1 and ρ_2 of $\mathcal{A}_q$ from q to q on the same finite word y, and for the run $\rho_3 := \rho_1$ we have $\forall p \in \rho_3 : \mathbb{C}(q) \geq \mathbb{C}(p)$. That is, $\mathcal{A}$ contains the forbidden pattern for finite ambiguity.

Otherwise, $\mathcal{B}$ has a final state f, a state $q \neq f$, and a finite word y such that there are runs ρ_1 of $\mathcal{B}_q$ on y from q to q, ρ_2 of $\mathcal{B}_q$ on y from q to f, and ρ_3 of $\mathcal{B}_f$ on y from f to f.

Since $f \in F$, we conclude that there is a state $q' \in Q$ with even color $c := \mathbb{C}(q')$ such that $f = (c, q')$. The run ρ_3 is therefore of the form $(c, p_0), \ldots, (c, p_n)$ where $p_0 = p_n = q'$. Hence, $p_0, \ldots, p_n$ is a run of $\mathcal{A}$ on y from q' to q' such that $\mathbb{C}(q')$ is even and $\forall i \leq n : c \geq \mathbb{C}(p_i)$.

Case 1: $q \in Q' \setminus Q$. By the definition of Q' we have $q = (c', p')$ for $c' \in \mathbb{C}_{even}$ and $p' \in Q$. Notice that the run ρ_2 is from (c', p') to (c, q'). By the definition of δ' we conclude that $c = c'$, and since $q \neq f$ we obtain $p' \neq q'$. By the definition of ρ_1 and ρ_2 we conclude that there are runs of $\mathcal{A}_{p'}$ on y from p' to p' and from p' to q', where $p' \neq q'$. Along with the run $p_0, \ldots, p_n$ from q' to q', we conclude that $\mathcal{A}$ contains the forbidden pattern for finite ambiguity.

Case 2: $q \in Q$. Notice that the run ρ_2 of $\mathcal{B}$ is from q to (c, q'). Therefore, by the definition of δ', there is a run $\rho_{\mathcal{A}}$ of $\mathcal{A}$ on y from q to q' which passes through a state with priority greater than c. If $q = q'$, then we conclude that there are two distinct runs of $\mathcal{A}_q$ on y from q to q: The run $\rho_1' := \rho_{\mathcal{A}}$, which visits a state with a color greater than c, and the run $\rho_2' := p_0, \ldots, p_n$ which only visits states of color at most c. Taking $\rho_3' := \rho_2$, we conclude that $\mathcal{A}$ has the forbidden pattern for finite ambiguity. Otherwise, $q \neq q'$ and by the definition of ρ_1 and ρ_2 we conclude that there are runs of $\mathcal{A}_q$ from q to q and from q to q', and together with the run $p_0, \ldots, p_n$ from q' to q', we conclude that $\mathcal{A}$ contains the forbidden pattern for finite ambiguity.

3. $\mathcal{B}$ contains the forbidden pattern for bounded ambiguity. Therefore, there are distinct states $p \neq q$ and a finite word y such that there are runs of $\mathcal{B}_p$ on y from p to p and from p to q, and there is a run of $\mathcal{B}_q$ on y from q to q.

   Case 1: $q \in Q$. Notice that there is a run of $\mathcal{B}$ from p to q, and by the definition of δ' we obtain $p \in Q$. Therefore, there are runs of $\mathcal{A}_p$ on y from p to p and from p to q, and a run of $\mathcal{A}_q$ on y from q to q. Hence, $\mathcal{A}$ contains a forbidden pattern for bounded ambiguity.

   Case 2: $q \in Q' \setminus Q$. By the definition of Q', there are $c \in \mathbb{C}_{even}$ and $q' \in Q$ such that $q = (c, q')$. If $p \in Q' \setminus Q$ then there are $c' \in \mathbb{C}_{even}$ and $p' \in Q$ such that $p = (c', p')$. Notice that there is a run from (c', p') to (c, q') and by the definition of δ' we have $c = c'$. Since $p \neq q$ we conclude that $p' \neq q'$. Therefore, there are runs of $\mathcal{A}_{p'}$ on y from p' to p' and from p' to q', and a run of $\mathcal{A}_{q'}$ on y from q' to q'. We conclude that $\mathcal{A}$ contains the forbidden pattern for bounded ambiguity.

   If $p \in Q$ and $p \neq q'$, then there are runs of $\mathcal{A}_p$ on y from p to p and from p to q', and a run of $\mathcal{A}_{q'}$ on y from q' to q'. We conclude that $\mathcal{A}$ contains the forbidden pattern for bounded ambiguity. Otherwise, we have $p = q'$. The run ρ_2 is from a state in Q to a state in $Q' \setminus Q$. By the definition of δ' we conclude that there is a run ρ' of $\mathcal{A}_p$ on y from p to p which passes though a state with color greater than $\mathbb{C}(q)$. The run ρ_3 is from a state in $Q' \setminus Q$ to a state in $Q' \setminus Q$, and by the definition of δ' we conclude that there is a run ρ'' of $\mathcal{A}_p$ on y from p to p which only visits states with color not greater than $\mathbb{C}(q)$. Let $\rho' = p_0, \ldots, p_n$ and $\rho'' = q_0, \ldots, q_n$. We conclude that $p_0 = p_n = q_0 = q_n = p$, and $\rho' \neq \rho''$. Therefore, there is $0 < i < n$ such that $p_i \neq q_i$. Let y_1, y_2 be two finite words such that y_1 is the prefix of y of length i, and $y_1 \cdot y_2 = y$. We conclude that there are runs of $\mathcal{A}_{p_i}$ on $y_2 \cdot y_1$ from p_i to p_i and from q_i to q_i, and there is a run of $\mathcal{A}_{q_i}$ on $y_2 \cdot y_1$ from q_i to q_i. Therefore, $\mathcal{A}$ contains the forbidden pattern for bounded ambiguity, as requested. ◀

We will now show that the problem of deciding the degree of ambiguity of PWA is in PTIME (in fact, this problem is even in NL).

▶ **Definition 19.** *Given a PWA $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ and $k > 0$, we define a graph $G^k_{\mathcal{A}}$ where the set of nodes is Q^k, and there is an edge from $(q_1, \ldots, q_k)$ to $(p_1, \ldots, p_k)$ iff there is $a \in \Sigma$ such that $(q_i, a, p_i) \in \delta$ for all $1 \leq i \leq k$.*

The following lemma supplies equivalent conditions to the forbidden patterns presented in Definition 14:

▶ **Lemma 20.**

- *$\mathcal{A}$ contains a forbidden pattern for countable ambiguity iff there is a state q such that $\mathbb{C}(q)$ is even, and $G^2_{\mathcal{A}}$ contains a path from (q, q) to itself which passes through a node (p, p') where $p \neq p'$, and for each node (p_1, p_2) in the path, $\mathbb{C}(q) \geq \mathbb{C}(p_1)$ and $\mathbb{C}(q) \geq \mathbb{C}(p_2)$.*
- *$\mathcal{A}$ contains a forbidden pattern for finite ambiguity iff there are states q and p such that $\mathbb{C}(q)$ is even, and $G^3_{\mathcal{A}}$ contains a path from (p, p, q) to (p, q, q) such that $\mathbb{C}(q)$ is the maximal color which is assigned to a state in the third coordinate of each node in the path, and the path contains a node (p_1, p_2, p_3) where $p_1 \neq p_2$.*
- *$\mathcal{A}$ contains a forbidden pattern for bounded ambiguity, if there are two states $p \neq q$ such that $G^3_{\mathcal{A}}$ contains a path from (p, p, q) to (p, q, q).*

▶ **Proposition 21.** *There is a polynomial time algorithm for deciding the degree of ambiguity of PWA.*

Proof. Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ be a PWA. By Lemma 20, it is sufficient to show that the equivalent conditions on $G^2_{\mathcal{A}}$ and $G^3_{\mathcal{A}}$ are decidable in polynomial time.

Indeed, each of the conditions in Lemma 20 can be reduced to polynomially many reachability problems in $G^2_{\mathcal{A}}$ and $G^3_{\mathcal{A}}$. Constructing $G^2_{\mathcal{A}}$ and $G^3_{\mathcal{A}}$ can be done in polynomial time in the size of $\mathcal{A}$, and the proposition follows. ◀

6 Finite Ambiguity and Bounded Ambiguity of PTA

In this section we will provide characterizations for finite ambiguity and bounded ambiguity of PTA. These characterizations are similar to the characterizations for finite ambiguity and bounded ambiguity of BTA (see Propositions 17 and 18 in [11]).

▶ **Definition 22** (Projection of a computation on a branch)**.** *Let $\phi \in ACC(\mathcal{A}, t)$ and let $\pi := v_0, v_1, \ldots$ be a tree branch. We say that $\phi(\pi) := \phi(v_0), \phi(v_1), \cdots \in Q^\omega_{\mathcal{A}}$ is the projection of ϕ on π, and define $ACC(\mathcal{A}, t, \pi) := \{\phi(\pi) \mid \phi \in ACC(\mathcal{A}, t)\}$.*

▶ **Definition 23** (Branch ambiguity)**.** *$\mathcal{A}$ is at most n branch-ambiguous if $|ACC(\mathcal{A}, t, \pi)| \leq n$ for every t and branch π. $\mathcal{A}$ is bounded branch ambiguous if it is at most n branch ambiguous for some n. $\mathcal{A}$ is finitely (respectively, countably) branch ambiguous if $|ACC(\mathcal{A}, t, \pi)|$ is finite (respectively, countable) for every t and π.*

Let $\mathcal{A}$ be a PTA. We define a PWA $\mathcal{A}_B$ which has the same ambiguity as the branch ambiguity of $\mathcal{A}$:

▶ **Definition 24** (Branch automaton)**.** *For a PTA $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$, the corresponding branch automaton is a PWA $\mathcal{A}_B := (Q, \Sigma_B, Q_I, \delta_B, \mathbb{C})$, where*

1. *$\Sigma_B := \Sigma \times \Sigma_d \times \Sigma_{cons}$ with*
   a. *$\Sigma_d := \{l, r\}$ - directions alphabet (left/right).*
   b. *$\Sigma_{cons} := \{S \subseteq Q \mid \bigcap_{q \in S} L(\mathcal{A}_q) \neq \emptyset\}$ - sets of states, which we consider "consistent."*
2. *$(q, a, q') \in \delta_B$ iff $a = (\sigma, l, S)$ and $\exists p \in S : (q, \sigma, (q', p)) \in \delta$ or $a = (\sigma, r, S)$ and $\exists p \in S : (q, \sigma, (p, q')) \in \delta$.*

The following lemma reduces the branch ambiguity to the ambiguity of branch automaton.

▶ **Lemma 25.** *The branch ambiguity of a PTA $\mathcal{A}$ is bounded (respectively, finite, countable) iff the ambiguity of the corresponding branch ω-automaton $\mathcal{A}_B$ is bounded (respectively, finite, countable).*

The proof of the lemma, which appears in the appendix, is a minor modification of the proof of Lemma 11 in [11] which reduces the branch ambiguity of BTA to the ambiguity of the corresponding branch automaton.

▶ **Definition 26** (Ambiguous Transition Pattern). *Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ be a PTA with corresponding branch automaton $\mathcal{A}_B = (Q, \Sigma_B, Q_I, \delta_B, \mathbb{C})$. $\mathcal{A}$ has a **q-ambiguous transition pattern** if $q \in Q$ and there are $p_1, p_2 \in Q$ and $y_1 \in \Sigma_B^*$, $y_2 \in \Sigma_B^+$ with runs of $\mathcal{A}_B$ from q to p_1 on y_1 and from p_2 to q on y_2 such that at least one of the following holds:*

1. *There are two transitions $(p_1, (a, d, \{q_1\}), p_2), (p_1, (a, d, \{q_2\}), p_2) \in \delta_B$ with $q_1 \neq q_2$ and $L(\mathcal{A}_{q_1}) \cap L(\mathcal{A}_{q_2}) \neq \emptyset$, or*
2. *There is a transition $(p_1, (a, d, \{q_1\}), p_2) \in \delta_B$ with $da(\mathcal{A}_{q_1}) > 1$.*

*A q-ambiguous transition pattern is said to be **fine** if $\mathbb{C}(q)$ is even, and $\mathbb{C}(q) \geq \mathbb{C}(p)$ for each state p in the runs of $\mathcal{A}_B$ from q to p_1 on y_1 and from p_2 to q on y_2.*

*$\mathcal{A}$ is said to have an **ambiguous transition pattern** if there is $q \in Q$ such that $\mathcal{A}$ has a q-ambiguous transition pattern. $\mathcal{A}$ is said to have a **fine ambiguous transition pattern** if there is $q \in Q$ such that $\mathcal{A}$ has a fine q-ambiguous transition pattern.*

We are now ready to provide the characterization of finite and bounded ambiguity of parity tree automata.

▶ **Proposition 27** (Bounded ambiguity of parity automata). *The following are equivalent:*

1. *A PTA $\mathcal{A}$ is not boundedly ambiguous.*
2. *At least one of the following conditions holds:*
   a. *$\mathcal{A}$ is not bounded branch ambiguous.*
   b. *$\mathcal{A}$ has an ambiguous transition pattern.*

▶ **Proposition 28** (Finite ambiguity of parity automata). *The following are equivalent:*

1. *A PTA $\mathcal{A}$ is not finitely ambiguous.*
2. *At least one of the following conditions holds:*
   a. *$\mathcal{A}$ is not finitely branch ambiguous.*
   b. *$\mathcal{A}$ has a fine ambiguous transition pattern.*

The proofs of Propositions 28 and 27 are simple variations of the proofs of Propositions 17 and 18 of [11], which deal with the characterization of bounded and finite ambiguity of Büchi tree automata. See the appendix for the proof of Prop. 28.

In Section 7 we use Propositions 27 and 28 to show that the problems of deciding whether a PTA is not boundedly ambiguous/not finitely ambiguous are in co-NP.

7 co-NP Upper Bound of Theorem 1

Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ be a PTA. Deciding whether a state $q \in Q$ of a PTA is useful is reducible to the emptiness problem of PTA, and can be tested in NP$\cap$ co-NP as follows: Let $Q_{non-empty} := \{p \mid L(\mathcal{A}_p) \neq \emptyset\}$. If $q \notin Q_{non-empty}$, then q is not useful. Otherwise, let $\mathcal{B}$ be the restriction of the branch automaton $\mathcal{A}_B$ to the transitions over one state letters in $\Sigma \times \Sigma_d \times \{\{p\} \mid p \in Q_{non-empty}\}$. Now, q is reachable from an initial state of $\mathcal{B}$ iff it is useful.

Therefore, we will assume in the rest of the proof that all states in $\mathcal{A}$ are useful.

It is easy to verify that $\mathcal{A}$ is ambiguous iff there exist two states $p, q \in Q_I$ such that $L(\mathcal{A}_p) \cap L(\mathcal{A}_q) \neq \emptyset$ or there exist two transitions $(q, a, q_1, q_2), (q, a, q_1', q_2') \in \delta$ from a state q such that $L(\mathcal{A}_{q_1}) \cap L(\mathcal{A}_{q_1'}) \neq \emptyset$ and $L(\mathcal{A}_{q_2}) \cap L(\mathcal{A}_{q_2'}) \neq \emptyset$.

Since deciding whether $L(\mathcal{A}_p) \cap L(\mathcal{A}_q) \neq \emptyset$ is in co-NP (by Lemma 8), and the number of pairs $p, q \in Q$ is polynomial in $|\mathcal{A}|$, we conclude:

▶ **Lemma 29.** *Deciding whether a PTA is ambiguous is in co-NP.*

The following Lemma easily follows from Definition 24 of the branch automaton.

▶ **Lemma 30.** *Let $\mathcal{A}_B$ be the branch automaton of $\mathcal{A}$. Assume that $r_i \in Q^{l+1}$ for $i = 1, \ldots, k$ are runs of $\mathcal{A}_B$ on $u = (\sigma_1, d_1, S_1) \ldots (\sigma_l, d_l, S_l) \in \Sigma_B^*$. Then, for $i = 1, \ldots, l$ there are $S_i' \subseteq S_i$ such that $|S_i'| \leq k$ and r_i for $i = 1, \ldots, k$ are runs of $\mathcal{A}_B$ on $u = (\sigma_1, d_1, S_1') \ldots (\sigma_l, d_l, S_l')$.*

A letter $(\sigma, d, S) \in \Sigma_B$ is called a k-state letter if S has at most k states. If $\mathcal{A}$ has n states, then the alphabet Σ_B of the branch automaton $\mathcal{A}_B$ might be of size $2|\Sigma| \times 2^n$, yet the number of k-state letters is bounded by $2|\Sigma| \times \sum_{i=1}^{k} \binom{n}{i} \leq 2|\Sigma|n^k$. To test whether a k-state letter (σ, d, S) is in Σ_B, we can check whether the intersection of the tree languages $L(\mathcal{A}_q)$ for $q \in S$ is non-empty. By Lemma 8, this is in co-NP for every fixed $k \in \mathbb{N}$. We denote by $\mathcal{A}_B^{(k)}$ the restriction of the branch automaton $\mathcal{A}_B$ to k-state letters.

▶ **Lemma 31** (Computability of branch ambiguity)**.** *The problem whether the branch ambiguity of $\mathcal{A}$ is not bounded (respectively, not finite, not countable) is in co-NP.*

Proof. By Lemma 25 and Proposition 16, deciding whether $\mathcal{A}_B$ is not finitely/not boundedly ambiguous is equivalent to deciding whether $\mathcal{A}_B$ has a forbidden pattern for finite/bounded ambiguity. The forbidden patterns involve conditions on at most three runs on the same word. By Lemma 30, we conclude that $\mathcal{A}_B$ has a forbidden pattern for finite/bounded ambiguity iff $\mathcal{A}_B^{(3)}$ has a forbidden pattern for finite/bounded ambiguity. Finding $\mathcal{A}_B^{(3)}$ requires deciding the non-emptiness of $L(\mathcal{A}_{q_1}) \cap L(\mathcal{A}_{q_2}) \cap L(\mathcal{A}_{q_3})$ for all triplets $q_1, q_2, q_3 \in Q$. This problem is in co-NP by Lemma 8; hence, $\mathcal{A}_B^{(3)}$ can be constructed in co-NP and its size is polynomial in the size of $\mathcal{A}$. The problem of deciding if $\mathcal{A}_B^{(3)}$ has a forbidden pattern for bounded/finite/countable ambiguity is in PTIME in the size of $\mathcal{A}_B^{(3)}$ (by Lemma 21). All these imply the co-NP bound of Lemma 31. ◀

▶ **Lemma 32** (Computability of q-ambiguous pattern)**.** *Deciding whether $\mathcal{A}$ has a q-ambiguous (respectively, fine q-ambiguous) transition for a state $q \in Q$ is in co-NP.*

Proof. Deciding if item (1) or (2) of Definition 26 holds for a fixed pair of states requires testing the non-emptiness of PTA intersection, which is in co-NP by Lemma 8. $\mathcal{A}$ has a q-ambiguous transition pattern iff there is a path in $\mathcal{A}_B$ from q to p_1 and from p_1 to q, such that items (1) or (2) hold for (p_1, p_2). If, additionally, $\mathbb{C}(q)$ is even and all states in the paths have a color which is not greater than $\mathbb{C}(q)$, then $\mathcal{A}$ has a fine q-ambiguous transition. Both these cases are reducible to the reachability problem in $\mathcal{A}_B^{(1)}$. Assuming all states are useful, finding $\mathcal{A}_B^{(1)}$ can be done in polynomial time. ◀

Lemmas 25, 31 and 32 imply that deciding whether condition 2(a) and 2(b) of Propositions 28 and 27 are in co-NP. Therefore, the problem whether a PTA is not boundedly (respectively, finitely) ambiguous is in co-NP. This together with Lemma 29 prove the upper bounds of Theorem 1.

8 Conclusion

We investigated the complexity of deciding the degree of ambiguity for PTA. The co-NP hardness lower bound was obtained by reductions from multi-dimensional parity games [2]. The co-NP upper bound was obtained by structural characterizations of degrees of ambiguity for PTA which is similar to the corresponding characterizations for BTA [11]. Unfortunately, we have not succeeded to find a characterization and an upper bound for countable ambiguity. It is also interesting to find natural problems for PTA/BTA which are easier for PTA/BTA with small degrees of ambiguity than for arbitrary PTA/BTA.

References

1 Vince Bárány, Łukasz Kaiser, and Alex Rabinovich. Expressing cardinality quantifiers in monadic second-order logic over trees. *Fundamenta Informaticae*, 100(1-4):1–17, 2010.

2 Krishnendu Chatterjee, Thomas A Henzinger, and Nir Piterman. Generalized parity games. In *International Conference on Foundations of Software Science and Computational Structures*, pages 153–167. Springer, 2007.

3 Thomas Colcombet. Unambiguity in automata theory. In *International Workshop on Descriptional Complexity of Formal Systems*, pages 3–18. Springer, 2015.

4 Yo-Sub Han, Arto Salomaa, and Kai Salomaa. Ambiguity, nondeterminism and state complexity of finite automata. *Acta Cybernetica*, 23(1):141–157, 2017.

5 Jozef Jirásek, Galina Jirásková, and Juraj Šebej. Operations on unambiguous finite automata. In *International Conference on Developments in Language Theory*, pages 243–255. Springer, 2016.

6 Ernst Leiss. Succinct representation of regular languages by boolean automata. *Theoretical computer science*, 13(3):323–330, 1981.

7 Hing Leung. Descriptional complexity of nfa of different ambiguity. *International Journal of Foundations of Computer Science*, 16(05):975–984, 2005.

8 Christof Löding and Anton Pirogov. On finitely ambiguous büchi automata. In Mizuho Hoshi and Shinnosuke Seki, editors, *Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings*, volume 11088 of *Lecture Notes in Computer Science*, pages 503–515. Springer, 2018. doi:10.1007/978-3-319-98654-8_41.

9 Dominique Perrin and Jean-Éric Pin. *Infinite words: automata, semigroups, logic and games*, volume 141. Academic Press, 2004.

10 Alexander Rabinovich. Complementation of finitely ambiguous Büchi automata. In *International Conference on Developments in Language Theory*, pages 541–552. Springer, 2018.

11 Alexander Rabinovich and Doron Tiferet. Degrees of ambiguity of büchi tree automata. In Arkadev Chattopadhyay and Paul Gastin, editors, *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019, December 11-13, 2019, Bombay, India*, volume 150 of *LIPIcs*, pages 50:1–50:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.FSTTCS.2019.50.

12 Helmut Seidl. On the finite degree of ambiguity of finite tree automata. *Acta Informatica*, 26(6):527–542, 1989.

13 Richard Edwin Stearns and Harry B Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*, 14(3):598–611, 1985.

14 Wolfgang Thomas. Automata on infinite objects. In *Formal Models and Semantics*, pages 133–191. Elsevier, 1990.

15 Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theoretical Computer Science*, 88(2):325–349, 1991.

A Selected Proofs

A.1 Proof of Lemma 25

▶ **Lemma 25.** *The branch ambiguity of a PTA $\mathcal{A}$ is bounded (respectively, finite, countable) iff the ambiguity of the corresponding branch ω-automaton $\mathcal{A}_B$ is bounded (respectively, finite, countable).*

The proof will make use the following two lemmas, which deals with the connection between computations of $\mathcal{A}$ and runs of $\mathcal{A}_B$:

▶ **Lemma 33.** *Let $t \in L(\mathcal{A})$, and let $\pi = v_0, v_1, \ldots$ be a tree branch. Then there exists $y \in L(\mathcal{A}_B)$ such that $ACC(\mathcal{A}, t, \pi) \subseteq ACC(\mathcal{A}_B, y)$.*

Proof. Let $y := (a_1, d_1, S_1) \ldots (a_i, d_i, S_i) \ldots$ be an ω-word over alphabet Σ_B, such that:

- $d_i \in \{l, r\}$ and $d_i := l$ iff v_i is the left child of v_{i-1}
- $a_i := t(v_{i-1})$
- $S_i := \{\phi(v'_i) \mid \phi \in ACC(\mathcal{A}, t)\}$ where v'_i is the child of v_{i-1} which is not v_i

Let $\phi \in ACC(\mathcal{A}, t)$. We will prove that $\rho := \phi(\pi)$ is a run of $\mathcal{A}_B$ on y. Assume that $\rho = p_0, p_1, \ldots$. For each $1 \leq i \leq n$ we have $p_{i-1} = \phi(v_{i-1})$ and $p_i = \phi(v_i)$. If v_i is the left child of v_{i-1} then we obtain $(\phi(v_{i-1}), t(v_{i-1}), \phi(v_i), \phi(v'_{i-1})) \in \delta$, and otherwise $(\phi(v_{i-1}), t(v_{i-1}), \phi(v'_{i-1}), \phi(v_i)) \in \delta$. By definition of S_i we obtain $\phi(v'_{i-1}) \in S_i$. Notice that $a_i = t(v_{i-1})$, and $d_i = l$ iff v_i is the left child of v_{i-1}. Therefore, by definition of $\mathcal{A}_B$, we conclude that $(\phi(v_{i-1}), (a_i, d_i, S_i), \phi(v_i)) = (p_{i-1}, (a_i, d_i, S_i), p_i) \in \delta_B$, and the lemma follows. ◀

▶ **Lemma 34.** *Let $\mathcal{A}$ be a PTA with corresponding branch automaton $\mathcal{A}_B$. If $y = (a_1, d_1, S_1) \ldots (a_i, d_i, S_i) \ldots$ is an ω-word over alphabet Σ_B such that $y \in L(\mathcal{A}_B)$, then there exist a tree $t \in L(\mathcal{A})$ and a tree branch $\pi = v_0, v_1, \ldots$ such that:*

- *$t(v_i) = a_{i+1}$*
- *v_{i+1} is the left child of v_i iff $d_i = l$*
- *For each run $\rho \in ACC(\mathcal{A}_B, y)$ there is a computation $\phi \in ACC(\mathcal{A}, t)$ such that $\phi(\pi) = \rho$.*

Proof. For each S_i, let $t_i \in \bigcap_{q \in S_i} L(\mathcal{A}_q)$ (there is such t_i, since $S_i \in \Sigma_{cons}$).

Let $\pi := v_0, v_1, \ldots$ where $v_0 := \epsilon$ and $\forall i \in \mathbb{N} : v_{i+1} := v_i \cdot d_i$; and denote by v'_i be the child of v_i which is not v_{i+1}.

We define a Σ-labeled full-binary tree t by $t(u) := \begin{cases} a_{i+1} & \exists i : u = v_i \\ t_{i+1}(w) & \exists i : u = v'_i \cdot w \end{cases}$

Let $\rho := p_0, p_1, \ldots$ be an accepting run of $\mathcal{A}_B$ on y. By definition of $\mathcal{A}_B$, for each $i \in \mathbb{N}$ there is a state $q_i \in Q$ such that $(p_i, a_i, p_{i+1}, q_i) \in \delta$ if $d_i = l$ or $(p_i, a_i, q_i, p_{i+1}) \in \delta$ if $d_i = r$. Recall that $t_i \in L(\mathcal{A}_{q_i})$, and therefore there is a computation $\phi_i \in ACC(\mathcal{A}_{q_i}, t_i)$. We use ρ and ϕ_i to define a computation ϕ of $\mathcal{A}$ on t, as follows:

$$\phi(u) := \begin{cases} p_i & \exists i : u = v_i \\ \phi_{i+1}(w) & \exists i : u = v'_i \cdot w \end{cases}$$

It is easy to see that ϕ is a computation of $\mathcal{A}$ on t. We will show that ϕ is accepting. For each tree branch π', if $\pi' = \pi$ then $\phi(\pi') = \phi(\pi) = \rho$ and since $\rho \in ACC(\mathcal{A}_B, y)$ we conclude that the maximal color which is assigned infinitely often to a state in $\phi(\pi')$ is even. Otherwise, by definition of t, there is $i \in \mathbb{N}$ such that $v'_i \in \pi'$. By the definition of ϕ we obtain $\phi(u) = \phi_i(w)$ for all nodes $u = v'_i \cdot w$, and since ϕ_i is accepting we conclude that the maximal color which is assigned infinitely often to a state in $\phi(\pi')$ is also even. Hence, $\phi \in ACC(\mathcal{A}, t)$ as requested. ◀

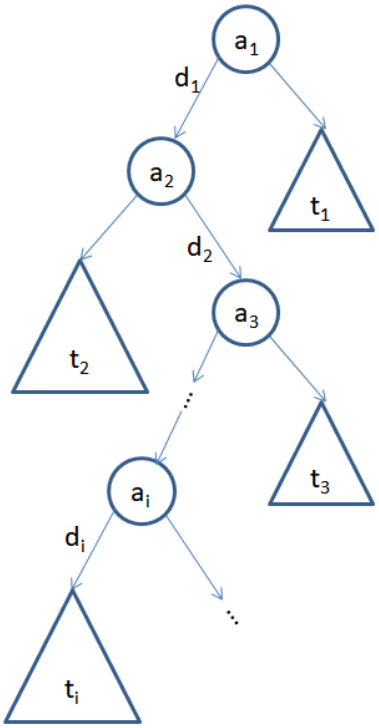

Figure 3 The tree t.

We are now ready to prove Lemma 25.

$\Rightarrow$: By Lemma 33, for each tree $t \in L(\mathcal{A})$ and a tree branch π there is an ω-word $y \in L(\mathcal{A}_B)$ such that $ACC(\mathcal{A}, t, \pi) \subseteq ACC(\mathcal{A}_B, y)$. Therefore, if $\mathcal{A}$ is not boundedly (respectively, finitely, countably) branch ambiguous then $\mathcal{A}_B$ is not boundedly (respectively, finitely, countably) ambiguous.

$\Leftarrow$: By Lemma 34, for each $y \in L(\mathcal{A}_B)$ there is a tree $t \in L(\mathcal{A})$ and a tree branch π such that $ACC(\mathcal{A}_B, y) \subseteq ACC(\mathcal{A}, t, \pi)$. Therefore, if $\mathcal{A}_B$ is not boundedly (respectively, finitely, countably) ambiguous then $\mathcal{A}$ is not boundedly (respectively, finitely, countably) branch ambiguous. ◀

A.2 Proof of Proposition 28

▶ **Proposition 28** (Finite ambiguity of parity automata)**.** *The following are equivalent:*

1. *A PTA $\mathcal{A}$ is not finitely ambiguous.*
2. *At least one of the following conditions holds:*
   a. *$\mathcal{A}$ is not finitely branch ambiguous.*
   b. *$\mathcal{A}$ has a fine ambiguous transition pattern.*

We will first prove the following auxiliary lemma:

▶ **Lemma 35.** *If a PTA $\mathcal{A}$ has a fine ambiguous transition pattern then its ambiguity degree is not countable.*

Proof. Let $\mathcal{A}_B = (Q, \Sigma_B, Q_I, \delta_B, F)$ be the corresponding branch automaton of $\mathcal{A}$, and let q be a state such that $\mathcal{A}$ has a fine q-ambiguous transition pattern. Therefore, there exist $p'_1, p'_2 \in Q$ and $z_1 \in \Sigma_B^*$, $z_2 \in \Sigma_B^+$ such that there is a run ρ_1 of $(\mathcal{A}_B)_q$ on z_1 from q to p'_1,

and a run ρ_2 of $(\mathcal{A}_B)_{p'_2}$ on z_2 from p'_2 to q; $\mathbb{C}(q)$ is even, and $\mathbb{C}(q) > \mathbb{C}(p)$ for each state p in the runs ρ_1 and ρ_2.

We choose $z' \in \Sigma_B$ as follows:

- If there are transitions $(p'_1, (a', d', \{q_1\}), p'_2), (p'_1, (a', d', \{q_2\}), p'_2) \in \delta_B$ with $L(\mathcal{A}_{q_1}) \cap L(\mathcal{A}_{q_2}) \neq \emptyset$, then by definition of $\mathcal{A}_B$ there exists a transition $(a', d', \{q_1, q_2\}) \in \delta_B$. Let $z' := (a', d', \{q_1, q_2\})$.
- Otherwise, by definition of fine q-ambiguous transition, there exists a transition: $(p'_1, (a', d', \{q_1\}), p'_2) \in \delta_B$ with $da(\mathcal{A}_{q_1}) > 1$. In this case, let $z' := (a', d', \{q_1\})$.

Define a word $y := z_1 \cdot z' \cdot z_2$ over alphabet Σ_B, and let $\rho := \rho_1 \cdot \rho_2$. Notice that ρ is a run of $\mathcal{A}_B$ on y from q to q.

y^ω is an ω-word in Σ_B^ω. Denote by ρ' the run ρ without the last state. By definition of ρ we conclude that $(\rho')^\omega$ is a run of $(\mathcal{A}_B)_f$ on y^ω. Notice that ρ' contains a final state, and therefore $(\rho')^\omega$ is an accepting run, and $y^\omega \in L((\mathcal{A}_B)_f)$.

y^ω is of the form $(a_1, d_1, S_1)\dots(a_i, d_i, S_i)\dots$ where $a_i \in \Sigma$, $d_i \in \{l, r\}$ and $S_i \subseteq Q$. Assume $z' = (a_z, d_z, S_z)$, and let $t_z \in \bigcap_{q' \in S_z} L(\mathcal{A}_{q'})$ such that there are two accepting computations ϕ_1 and ϕ_2 on t_z, where $\phi_1(\epsilon), \phi_2(\epsilon) \in S_z$ (there is such t_z by definition of z').

By Lemma 34, there is a tree $t \in L(\mathcal{A}_q)$, a computation $\phi \in ACC(\mathcal{A}_q, t)$ and a tree branch $\pi = v_0 v_1 \dots$ such that $\phi(\pi) = (\rho')^\omega$; and for each $i \in \mathbb{N}$ we have $t(v_i) = a_i$, and v_{i+1} is the left child of v_i iff $d_i = l$.

Let $J := \{i \mid \text{the } i\text{-th transition of } (\rho')^\omega \text{ is from } p'_1 \text{ to } p'_2 \text{ over } z'\}$. By definition of ρ we conclude that J is an infinite subset of $\mathbb{N}$.

Define a tree t' by $t'(u) = \begin{cases} t_z(w) & \exists i : u = v'_i \cdot w \\ t(u) & \text{otherwise} \end{cases}$

For each $B \subseteq A$, we define a computation ϕ_B by:

$$\phi_B(u) = \begin{cases} \phi_1(w) & \exists i : u = v'_i \cdot w \text{ and } v'_i \in A \setminus B \\ \phi_2(w) & \exists i : u = v'_i \cdot w \text{ and } v'_i \in B \\ \phi(u) & \text{otherwise} \end{cases}$$

It is routine to verify that ϕ_B is an accepting computation of $\mathcal{A}_q$ on t', and that $B_1 \neq B_2 \to \phi_{B_1} \neq \phi_{B_2}$. Since the number of subsets of A is uncountable, and each subset B yields a unique accepting computations ϕ_B of $\mathcal{A}_q$ on t', it follows that $ACC(\mathcal{A}_q, t')$ is not countable, and since q is useful, we conclude that $\mathcal{A}$ is not countably ambiguous. ◀

We are now ready to proceed with the proof of Prop. 28. The (2) ⇒ (1) direction follows from Lemma 25 and Lemma 35. Below we prove the (1) ⇒ (2) direction.

Let t be a tree such that $ACC(\mathcal{A}, t)$ is not finite. We define a branch $\pi := v_0, \dots, v_i, \dots$ in t and an ω-sequence of states $q_0 \dots q_i \dots$ such that for every i:

1. From q_i there are infinitely many accepting computations of $\mathcal{A}_{q_i}$ on the subtree $t_{\geq v_i}$.
2. There is an accepting computation ϕ_i on t such that $\phi_i(v_j) = q_j$ for every $j \leq i$.

Define v_0 as the root of t and q_0 as an initial state from which there are infinitely many accepting computations.

Assume that v_i and q_i were defined. Since there are infinitely many accepting computations from the state q_i on the subtree $t_{\geq v_i}$, infinitely many of them take the same first transition from q_i to $\langle q_l, q_r \rangle$ and either there are infinitely many accepting computations from state q_l on the subtree rooted at the left child of v_i, or from state q_r on the subtree rooted at the right child of v_i. Define v_{i+1} and q_{i+1} according to these cases.

If $|ACC(\mathcal{A}, t, \pi)|$ is infinite, then by the definition of branch ambiguity we have that $\mathcal{A}$ is not finitely branch ambiguous, and 2(a) holds. Otherwise, there exist $\phi_1, \dots, \phi_k \in ACC(\mathcal{A}, t)$ such that $ACC(\mathcal{A}, t, \pi) = \{\phi_i(\pi) \mid 1 \leq i \leq k\}$. Choose n such that for all $1 \leq i < j \leq k$: $\phi_i(v_0) \dots \phi_i(v_n) \neq \phi_j(v_0) \dots \phi_j(v_n)$.

There is $1 \leq j \leq k$ such that $\phi_j(v_0) \dots \phi_j(v_n) = q_0 \dots q_n$. Notice that by definition of n, each computation $\phi \in ACC(\mathcal{A}, t)$ which assigns $q_0, \dots, q_n$ to the nodes $v_0, \dots, v_n$ must also agree with ϕ_j on each node v_i for $i \in \mathbb{N}$. Therefore, again by the definition of π and $q_0, \dots q_i, \dots$, we conclude that $\phi_j(\pi) = q_0, q_1, \dots$.

Let q be a state which occurs infinitely often in $\phi_j(\pi)$ such that $\mathbb{C}(q)$ is even, and $\mathbb{C}(q)$ is maximal among the colors which $\mathbb{C}$ assigns to states which occurs infinitely often in $\phi_j(\pi)$.

Choose $N > n$ such that $\phi_j(v_N) = q_N = q$, and each state q_i where $i \geq N$ occurs infinitely often in $\phi_j(\pi)$. By selection of q_N, there are infinitely many accepting computations of $\mathcal{A}_q$ on $t_{\geq v_N}$. Take two different accepting computations $\phi', \phi'' \in ACC(\mathcal{A}_q, t_{\geq v_N})$. Note that $\forall i \geq N : \phi_j(v_i) = \phi'(v_i) = \phi''(v_i) = q_i$. Therefore, ϕ' and ϕ'' differ at some node $w \notin \pi$, and there exists $M > N$ such that $\phi_j(v_M) = q = \phi'(v_M) = \phi''(v_M)$ and $v_M \perp w$.

Let u be the node of maximal depth on the path from v_N to v_M such that $w > u$. Let u', u'' be the children of u such that $w \geq u'$. Assume w.l.o.g. that u' is the left child of u.

Look at the transitions $(\phi'(u), t(u), \phi'(u'), \phi'(u'')), (\phi''(u), t(u), \phi''(u'), \phi''(u'')) \in \delta$. Since $u'' \in \pi$, we have $\phi'(u'') = \phi''(u'')$. If $\phi'(u') = \phi''(u')$ then the restriction of ϕ' and ϕ'' on $t_{\geq u'}$ are two different computations in $ACC(\mathcal{A}_{\phi(u')}, t_{\geq u'})$ and therefore $da(\mathcal{A}_{\phi(u')}) > 1$ and condition 2 of q-ambiguous transition pattern definition applies. Otherwise, we have $\phi'(u') \neq \phi''(u')$ and $t_{\geq u'} \in L(\mathcal{A}_{\phi(u')}) \cap L(\mathcal{A}_{\phi'(u')})$ and therefore condition 1 of q-ambiguous transition pattern definition applies. By selection of q we conclude that $\mathcal{A}$ has a fine q-ambiguous transition pattern, and condition 2(b) of Prop. 28 holds. ◀