AMBIGUITY HIERARCHY OF REGULAR INFINITE TREE LANGUAGES

ALEXANDER RABINOVICH AND DORON TIFERET

Tel Aviv University, Israel
e-mail address: rabinoa@tauex.tau.ac.il
URL: `https://www.cs.tau.ac.il/~rabinoa`

Tel Aviv University, Israel
e-mail address: sdoron5.t2@gmail.com

Abstract. An automaton is unambiguous if for every input it has at most one accepting computation. An automaton is k-ambiguous (for $k > 0$) if for every input it has at most k 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. An automaton is finitely (respectively, countably) ambiguous if for every input it has at most finitely (respectively, countably) many accepting computations.

The degree of ambiguity of a regular language is defined in a natural way. A language is k-ambiguous (respectively, boundedly, finitely, countably ambiguous) if it is accepted by a k-ambiguous (respectively, boundedly, finitely, countably ambiguous) automaton. Over finite words every regular language is accepted by a deterministic automaton. Over finite trees every regular language is accepted by an unambiguous automaton. Over ω-words every regular language is accepted by an unambiguous Büchi automaton and by a deterministic parity automaton. Over infinite trees Carayol et al. showed that there are ambiguous languages.

We show that over infinite trees there is a hierarchy of degrees of ambiguity: For every $k > 1$ there are k-ambiguous languages which are not $k - 1$ ambiguous; there are finitely (respectively countably, uncountably) ambiguous languages which are not boundedly (respectively finitely, countably) ambiguous.

1. Introduction

Degrees of Ambiguity. The relationship between deterministic and nondeterministic machines plays a central role in computer science. An important topic is a comparison of expressiveness, succinctness and complexity of deterministic and nondeterministic models. Various restricted forms of nondeterminism were suggested and investigated (see [6, 10] 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

Key words and phrases: regular tree languages, parity tree automata, automata ambiguity.

finite words there is a rich and well-developed theory on the relationship between deterministic, unambiguous and nondeterministic automata [10]. 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 [12, 13].

Some problems are easier for unambiguous than for nondeterministic automata. As shown by Stearns and Hunt [19], the equivalence and inclusion problems for unambiguous automata are in polynomial time, while these problems are PSPACE-complete for nondeterministic automata.

The complexity of basic regular operations on languages represented by unambiguous finite automata was investigated in [11], 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 [11], 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 [10] 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.

The cardinality of the set of accepting computations of an automaton over an infinite tree t is bounded by the cardinality of the set of functions from the nodes of t to the state of the automaton, and therefore, it is at most continuum $2^{\aleph_0}$. The set of accepting computations on t is definable in Monadic Second-Order Logic (MSO). In Bárány et al. in [2] it was shown that the continuum hypothesis holds for MSO-definable families of sets. Therefore, if the set of accepting computations of an automaton on a tree t is uncountable, then its cardinality is $2^{\aleph_0}$. Hence, there are exactly two infinite degrees of ambiguity.

The degree of ambiguity of a regular language is defined in a natural way. A language is k-ambiguous if it is accepted by a k-ambiguous automaton. A language is boundedly ambiguous if it is k-ambiguous for some k; it is finitely (respectively, countably) ambiguous if it is accepted by a finitely (respectively, countably) ambiguous automaton.

Over finite words, every regular language is accepted by a deterministic automaton. Over finite trees, every regular language is accepted by a deterministic bottom-up tree automaton and by an unambiguous top-down tree automaton. Over ω-words every regular language is accepted by an unambiguous Büchi automaton [1] and by a deterministic parity automaton.

Hence, the regular languages over finite words, over finite trees and over ω-words are unambiguous.

In [5] it was shown that the aforementioned situation is different for infinite trees. Carayol et al. [5] proved that the language $L_{\exists a}$ of infinite full-binary trees over the alphabet $\{a, c\}$, defined as $L_{\exists a} := \{t \mid t$ has at least one node labeled by $a\}$ is ambiguous. The proof is based on the undefinability of a choice function in Monadic Second-Order logic (MSO) [9, 4].

Our results imply that the complement of every countable regular language is not finitely ambiguous. Since $L_{\exists a}$ is the complement (with respect to the alphabet $\{a, c\}$) of the language which consists of a single tree (i.e. the tree with all nodes labeled by c), we conclude that $L_{\exists a}$ is not finitely ambiguous (this strengthens the above mentioned result of [5]). Our main result states that over infinite trees there is a hierarchy of degrees of ambiguity:

Theorem 1.1 (Hierarchy)**.** (1) *For every $k > 1$ there are k-ambiguous languages which are not $(k-1)$-ambiguous.*
(2) *There are finitely ambiguous languages which are not boundedly ambiguous.*
(3) *There are countably ambiguous languages which are not finitely ambiguous.*
(4) *There are uncountably ambiguous languages which are not countably ambiguous.*

Some natural tree languages which witness items (1), (3) and (4) of Theorem 1.1 are described in the examples below. We have not found a “natural” finitely ambiguous language which is not boundedly ambiguous (Theorem 1.1(2)).

Examples 1.2. Let T^{ω}_{Σ} be the set of all infinite full-binary trees over an alphabet Σ. Let $\Sigma_k = \{c, a_1, a_2, ..., a_k\}$, and let $L_{\neg a_i} := \{t \in T^{\omega}_{\Sigma_k} \mid$ no node in t is labeled by $a_i\}$ for $1 \leq i \leq n$. Define:

(1) $L_{\neg a_1 \vee \cdots \vee \neg a_k} := L_{\neg a_1} \cup \cdots \cup L_{\neg a_k}$. We show that this language is k-ambiguous, but is not $(k-1)$-ambiguous (see Sect. 5). In [3] it was shown that $L_{\neg a_1 \vee \neg a_2}$ is two ambiguous.
(2) $L_{\exists a_1} := \{t \in T^{\omega}_{\Sigma_1} \mid$ there exists an a_1-labeled node in $t\}$. This is a countably ambiguous language which is not finitely ambiguous (see Sect. 4).
(3) $L_{no-\max-a_1} := \{t \in T^{\omega}_{\Sigma_1} \mid$ above every a_1-labeled node in t there is an a_1-labeled node$\}$. This is an uncountably ambiguous language which is not countably ambiguous (see Sect. 7).

Organization of the paper: In Sect. 2 we recall notations and basic results about automata and monadic second-order logic. In Sect. 3 simple properties of languages are proved. Sect. 4 gives a sufficient condition for a language to be not finitely ambiguous. The proof techniques used in Sect. 4 refine the proof techniques of [5], and rely on the fact that a choice function is not MSO-definable. Sect. 5 deals with k-ambiguous languages - for every $k \in \mathbb{N}$, we describe a k-ambiguous language which is not $(k-1)$-ambiguous. Sect. 6 provides an example of a finitely ambiguous language which is not boundedly ambiguous. Sect. 7 introduces a scheme for obtaining uncountably ambiguous languages from languages which are not boundedly ambiguous, and presents some natural examples of uncountably ambiguous languages. In Sect. 8, relying on the characterization of countable regular languages given by Niwiński [15], we prove that every countable tree language is unambiguous. Conclusion is given in Sect. 9

An extended abstract of this paper was published in [18]. In this paper we added missing proofs, presented natural examples of uncountably ambiguous languages (in Sect. 7) and added Sect. 8 in which we prove that countable languages are unambiguous.

2. Preliminary

We recall here standard terminology and notations about trees, automata and logic [16, 17].

2.1. **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 $v \leq u$; a set U of nodes is an **antichain**, if its elements are incomparable with each other.

We say that an infinite sequence $\pi = v_0, v_1, \dots$ 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^ω_Σ 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 $u \in \{l,r\}^*$.

Grafting. Given two labeled trees t_1 and t_2 and a node $v \in \{l,r\}^*$, the grafting of t_2 on v in t_1, denoted by $t_1 \circ_v t_2$, is the tree t which is obtained from t_1 by replacing the subtree of t_1 rooted at v by t_2. Formally, $t(u) := \begin{cases} t_2(w) & \exists w \in \{l,r\}^* : u = v \cdot w \\ t_1(u) & \text{otherwise} \end{cases}$

More generally, given a tree t_1, an antichain $Y \subseteq \{l,r\}^*$ and a tree t_2, the grafting of t_2 on Y in t_1, denoted by $t_1 \circ_Y t_2$, is obtained by replacing each subtree of t_1 rooted at a node $y \in Y$ by the tree t_2.

Tree Language. A language L over an alphabet Σ is a set of Σ-labeled trees. We denote by $\overline{L} := T^\omega_\Sigma \setminus L$ the complement of L.

2.2. **Automata.**

2.2.1. *ω-word Automata.*

Parity ω-word Automata (PWA). A PWA is a tuple $(Q_\mathcal{A}, \Sigma, Q_I, \delta, \mathbb{C})$ where Σ is a finite alphabet, Q is a finite set of states, $Q_I \subseteq Q$ is a set of initial states, $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation, and $\mathbb{C} : Q \to \mathbb{N}$ is a coloring function. 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 $(q_i, a_i, q_{i+1}) \in \delta$ for all $i \in \mathbb{N}$. We say that ρ is accepting if the maximal number which occurs infinitely often in $\mathbb{C}(q_0)\mathbb{C}(q_1)\dots$ is even.

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

2.2.2. *Infinite Tree Automata.*

Parity Tree Automata (PTA). A PTA is a tuple $(Q_{\mathcal{A}}, \Sigma, Q_I, \delta, \mathbb{C})$ where $\delta \subseteq Q \times \Sigma \times Q \times Q$, and Σ, Q, Q_I, F are defined as in PWA. A computation of $\mathcal{A}$ on a tree t is a function $\phi : \{l, r\}^* \to Q$ such that $\phi(\epsilon) \in Q_I$, and $\forall v \in \{l, r\}^* : (\phi(v), t(v), \phi(v \cdot l), \phi(v \cdot r)) \in \delta$. We say that ϕ is accepting if for each tree branch $\pi = v_0 v_1 \dots$, the maximal number which occurs infinitely often in $\mathbb{C}(\phi(v_0))\mathbb{C}(\phi(v_1)) \dots$ is even.

Given a PTA $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, Q_I, \delta_{\mathcal{A}}, \mathbb{C}_{\mathcal{A}})$ and a set $Q' \subseteq Q_{\mathcal{A}}$, we define $\mathcal{A}_{Q'} := (Q_{\mathcal{A}}, \Sigma, Q', \delta_{\mathcal{A}}, \mathbb{C}_{\mathcal{A}})$ as the automaton obtained from $\mathcal{A}$ by replacing the set of initial states Q_I with Q'. For a singleton $Q' = \{q\}$, we simplify this notation by $\mathcal{A}_q := \mathcal{A}_{Q'}$.

Language. 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 $L(\mathcal{A}) := \{t \in T^{\omega}_{\Sigma} \mid ACC(\mathcal{A}, t) \neq \emptyset\}$. A tree language is said to be *regular* if it is accepted by a PTA.

A state $q \in Q$ of a PTA $\mathcal{A}$ is called useful if there is a tree $t \in L(\mathcal{A})$, a computation $\phi \in ACC(\mathcal{A}, t)$ and a node $v \in \{l, r\}^*$ such that $\phi(v) = q$. Throughout the paper we will assume all states of PTA are useful.

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 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.

We order the degrees of ambiguity in a natural way: $i < j < finite < \aleph_0 < 2^{\aleph_0}$, for $i < j \in \mathbb{N}$.

Degree of Ambiguity of a Language. We say that a regular tree language L is unambiguous (respectively, k-ambiguous, finitely ambiguous, countably ambiguous) if it is accepted by an unambiguous (respectively, k-ambiguous, finitely ambiguous, countably ambiguous) automaton. We define $da(L) := min_{\mathcal{A}}\{da(\mathcal{A}) \mid L(\mathcal{A}) = L\}$.

2.3. **Monadic Second-Order Logic.** We use standard notations and terminology about monadic second-order logic (MSO) [17, 21, 20].

Let τ be a relational signature. A structure (for τ) is a tuple $M = (D, \{R^M \mid R \in \tau\})$ where D is a domain, and each symbol $R \in \tau$ is interpreted as a relation R^M on D.

MSO-formulas use first-order variables, which are interpreted by elements of the structure, and monadic second-order variables, which are interpreted as sets of elements. Atomic MSO-formulas are of the following form:

- $R(x_1, \dots, x_n)$ for an n-ary relational symbol R and first order variables $x_1, \dots, x_n$
- $x = y$ for two first-order variables x and y
- $x \in X$ for a first-order variable x and a second-order variable X

MSO-formulas are constructed from the atomic formulas, using boolean connectives, the first-order quantifiers, and the second-order quantifiers.

We write $\psi(X_1, \dots, X_n, x_1, \dots, x_m)$ to indicate that the free variables of the formula ψ are $X_1, \dots, X_n$ (second order variables) and $x_1, \dots, x_m$ (first order variables). We write $M \models \psi(A_1, \dots, A_n, a_1, \dots a_m)$ if ψ holds in M when subsets A_i are assigned to X_i for $i = 1, \dots, n$ and elements a_i are assigned to variables $x_1, \dots, x_m$ for $i = 1, \dots, m$.

Coding. Let Δ be a finite set. We can code a function from a set D to Δ by a tuple of unary predicates on D. This type of coding is standard, and we shall use explicit variables which range over such mappings and expressions of the form "$F(u) = d$" (for $d \in \Delta$) in MSO-formulas, rather than their codings.

Formally, for each finite set Δ we have second-order variables $X_1^\Delta, X_2^\Delta, \dots$ which range over the functions from D to Δ, and atomic formulas $X_i^\Delta(u) = d$ for $d \in \Delta$ and u a first order variables [21]. Often the type of the second order variables will be clear from the context and we drop the superscript Δ.

Definable Relations. The powerset of D is denoted by $\mathcal{P}(D)$. We say that a relation $R \subseteq \mathcal{P}(D)^n \times D^m$ is MSO-definable in a structure S with universe D if there is an MSO-formula $\psi(X_1, \dots, X_n, x_1, \dots, x_m)$ such that $R = \{(D_1, \dots, D_n, u_1, \dots, u_m) \in \mathcal{P}(D)^n \times D^m \mid S \models \psi(D_1 \dots, D_n, u_1 \dots, u_n)\}$.

An element $d \in D$ is MSO-definable in a structure S if there is a formula $\psi(x)$ such that $S \models \phi(u)$ iff $u = d$. A set $U \subseteq D$ is MSO-definable if there is a formula $\phi(X)$ such that $S \models \phi(V)$ iff $V = U$. A function is MSO-definable if its graph is.

The unlabeled binary tree is the structure $(\{l, r\}^*, \{E_l, E_r\})$ where E_l and E_r are binary symbols, respectively interpreted as $\{(v, v \cdot l) \mid v \in \{l, r\}^*)\}$ and $\{(v, v \cdot r) \mid v \in \{l, r\}^*)\}$.

It is easy to verify the correctness of the following lemma:

Lemma 2.1. *The following relations are MSO-definable in the unlabeled full-binary tree.*

- *The ancestor relation $\leq$.*
- *"A set of nodes is a branch," "A set of nodes is an antichain."*
- *Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ be a PTA. We use ϕ for a function $\{l, r\}^* \to Q$ and σ for a function $\{l, r\}^* \to \Sigma$.*
  - *"ϕ is a computation of $\mathcal{A}$ on the tree σ."*
  - *"ϕ is an **accepting** computation of $\mathcal{A}$ on the tree σ."*

Theorem 2.2 (Rabin [17])**.** *A tree language is regular iff it is MSO-definable in the unlabeled binary tree structure.*

A labeled tree is regular iff it has finitely many different subtrees. An equivalent definition is: a tree is regular iff its labeling is MSO-definable [17]. Hence, for every regular Σ-labeled tree t_0, there is an MSO-formula $\psi_{t_0}(\sigma^\Sigma)$ which is satisfied by t iff $t = t_0$.

Theorem 2.3 (Rabin's basis theorem [17])**.** *Any non-empty regular tree language contains a regular tree.*

Choice Function. A choice function is a mapping which assigns to each non-empty set of nodes one element from the set.

Theorem 2.4 (Gurevich and Shelah [9])**.** *There is no MSO-definable choice function on the full-binary tree.*

The following lemma follows from Theorem 2.4.

Lemma 2.5. *There is no MSO-definable function which assigns to every non-empty antichain Y a finite non-empty subset $X \subseteq Y$.*

Proof. Assume, for the sake of contradiction, that a function which returns a non-empty subset for each non-empty antichain is MSO-definable in the unlabeled full-binary tree, by an MSO-formula $FiniteAntichainSubset(X, Y)$.

Claim 2.5.1 (Choice function over finite sets)**.** *There is an MSO-definable function which assigns to each non-empty finite set $X \subseteq \{l, r\}^*$ an element $x \in X$.*

Proof. We first define a lexicographic order "$\leq_{lex}$" on $\{l, r\}^*$, by $u \leq_{lex} v$ iff u is a prefix of v or $u = w \cdot l \cdot u'$ and $v = w \cdot r \cdot v'$ for some $w, u', v' \in \{l, r\}^*$.

It is easy to verify that $\leq_{lex}$ is MSO-definable in the unlabeled full-binary tree. $\leq_{lex}$ is a linear order, and therefore each non-empty finite set has a exactly one $\leq_{lex}$-minimal element. We conclude that a finite set choice function is definable by $FiniteChoice(X, x) :=$"x is the $\leq_{lex}$-minimal element in X". ■

Let $FiniteChoice(X, x)$ be an MSO-formula which defines a function as in Claim 2.5.1. We will use formulas $FiniteAntichainSubset(X, Y)$ and $FiniteChoice(X, x)$ to define a choice function by an MSO-formula $Choice(X, x)$ which is the conjunction of the following conditions:

(1) $\exists Z$: "Z is the set of $\leq$-minimal elements in X"
(2) $\exists Y : FiniteAntichainSubset(Z, Y)$
(3) $FiniteChoice(Y, x)$

For each non-empty set X there is a unique subset $Z \subseteq X$ of the $\leq$-minimal elements in X. This set is a non-empty antichain, and therefore $FiniteAntichainSubset(Z, Y)$ returns a finite subset $Y \subseteq Z$. Therefore, $FiniteChoice(Y, x)$ returns an element in Y. We conclude that $Choice(X, x)$ returns an element $x \in X$ and therefore defines a choice function in the unlabeled full-binary tree, in contradiction to Theorem 2.4. □

3. Simple Properties of Automata and Languages

In this section some simple lemmas are collected.

Lemma 3.1. *Let $\mathcal{A}_1 = (Q_1, \Sigma_1, Q^1_{I_1}, \delta_1, \mathbb{C}_1)$ and $\mathcal{A}_2 = (Q_2, \Sigma_2, Q^2_{I_1}, \delta_2, \mathbb{C}_2)$ be two PTA. Then:*

(1) *There exists an automaton $\mathcal{B}$ such that $L(\mathcal{B}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ and for each $t \in L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$, $|ACC(\mathcal{B}, t)| \leq |ACC(\mathcal{A}_1, t)| + |ACC(\mathcal{A}_2, t)|$*
(2) *There exists an automaton $\mathcal{B}$ such that $L(\mathcal{B}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ and for each $t \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$, $|ACC(\mathcal{B}, t)| \leq |ACC(\mathcal{A}_1, t)| \cdot |ACC(\mathcal{A}_2, t)|$*

Proof. (1) Assume that Q_1 and Q_2 are disjoint, and let $\mathcal{B} := (Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, Q_I^1 \cup Q_I^2, \delta_1 \cup \delta_2, \mathbb{C}_1 \cup \mathbb{C}_2)$. It is clear that $L(\mathcal{B}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.

Let $t \in L(\mathcal{B})$. By definition of $\mathcal{B}$, for each $\phi \in ACC(\mathcal{B}, t)$ we either have $\phi \in ACC(\mathcal{A}_1, t)$ or $\phi \in ACC(\mathcal{A}_2, t)$. Therefore, we obtain $|ACC(\mathcal{B}, t)| = |ACC(\mathcal{A}_1, t)| + |ACC(\mathcal{A}_2, t)|$.

(2) It is easy to verify that there is an MSO-formula over ω-words which holds for $w = (c_1, c'_1), \dots, (c_i, c'_i), \dots \in (Image(\mathbb{C}_1) \times Image(\mathbb{C}_2))^\omega$ iff the maximal color which appears infinitely often in the first coordinate of w and the maximal color which appears infinitely often in the second coordinate of w are both even. Therefore (by McNaughton's Theorem [14]) there is a deterministic PWA $\mathcal{D} = (Q_\mathcal{D}, \Sigma_\mathcal{D}, q_I^\mathcal{D}, \delta_\mathcal{D}, \mathbb{C}_\mathcal{D})$ over alphabet $\Sigma_\mathcal{D} = Image(\mathbb{C}_1) \times Image(\mathbb{C}_2)$ such that $w \in L(\mathcal{D})$ iff the maximal color which appears infinitely often in the first coordinate of w and the maximal color which appears infinitely often in the second coordinate of w are both even.

We will use the automata $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{D}$ to define a PTA $\mathcal{B} = (Q_\mathcal{B}, \Sigma_\mathcal{B}, Q_I^\mathcal{B}, \delta_\mathcal{B}, \mathbb{C}_\mathcal{B})$ which accepts $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

- $Q_\mathcal{B} = Q_1 \times Q_2 \times Q_\mathcal{D}$
- $\Sigma_\mathcal{B} := \Sigma_1 \cap \Sigma_2$
- $Q_I^\mathcal{B} := Q_I^1 \times Q_I^2 \times \{q_I^\mathcal{D}\}$
- $((q, p, s), a, (q_1, p_1, s_1), (q_2, p_2, s_2)) \in \delta_\mathcal{B}$ iff $(q, a, q_1, q_2) \in \delta_1$, $(p, a, p_1, p_2) \in \delta_2$, and $s_1 = s_2 = \delta_\mathcal{D}(s, (\mathbb{C}_1(q), \mathbb{C}_2(p)))$.
- $\mathbb{C}_\mathcal{B}(q_1, q_2, p) := \mathbb{C}_\mathcal{D}(p)$

It is easy to verify that $L(\mathcal{B}) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

Assume, for the sake of contradiction, that there exists t such that $|ACC(\mathcal{B}, t)| > |ACC(\mathcal{A}_1, t)| \cdot |ACC(\mathcal{A}_2, t)|$. Since $\mathcal{D}$ is deterministic, it follows that there is a computation in $ACC(\mathcal{B}, t)$ such that either the projection of the first coordinate of ϕ on Q_1, denoted ϕ_1, is not in $ACC(\mathcal{A}_1, t)$ or the projection of the second coordinate of ϕ on Q_2, denoted ϕ_2, is not in $ACC(\mathcal{A}_2, t)$. Assume w.l.o.g. that $\phi \notin ACC(\mathcal{A}_1, t)$. Therefore, there is a tree branch $\pi = v_0, v_1, \dots$ such that the maximal color which $\mathbb{C}_1$ assigns to the states which occurs infinitely often in $\phi_1(\pi)$ is odd. By definition of $\mathcal{D}$ we conclude that $w := (c_0, c'_0), (c_1, c'_1), \dots \notin L(\mathcal{D})$, where $c_i := \mathbb{C}_1(\phi_1(v_i))$ and $c'_i := \mathbb{C}_2(\phi_2(v_i))$. Hence, by definition of $\mathcal{B}$ we conclude that the sequence of colors which $\mathbb{C}_\mathcal{B}$ assigns to the states $\phi(\pi)$ is exactly w, and therefore $\phi \notin ACC(\mathcal{B}, t)$ - a contradiction. □

From Lemma 3.1, we obtain:

Corollary 3.2. *Boundedly, finitely and countably ambiguous tree languages are closed under finite union and intersection.*

We often use implicitly the following simple Lemma.

Lemma 3.3 (Grafting)**.** *Let $\mathcal{A}$ be an automaton, t, t_1 trees, $v \in \{l, r\}^*$ and $\phi \in ACC(\mathcal{A}, t)$, and $\phi_1 \in ACC(\mathcal{A}_q, t_1)$. If $\phi(v) = q$, then $\phi \circ_v \phi_1$ is an accepting computation of $\mathcal{A}$ on $t \circ_v t_1$.*

A similar lemma holds for general grafting. As an immediate consequence, we obtain the following lemma:

Lemma 3.4. *$da(\mathcal{A}) \geq da(\mathcal{A}_q)$ for every useful state q of $\mathcal{A}$.*

Corollary 3.5. *Let $\mathcal{A}$ be a boundedly (respectively, finitely, countably) ambiguous PTA with a set Q of useful states, and let $Q' \subseteq Q$. Then $\mathcal{A}_{Q'}$ is boundedly (respectively, finitely, countably) ambiguous.*

Lemma 3.6. *Let L_1 and L_2 be two tree languages such that $da(L_1) \neq da(L_2)$ and $L_1 \subseteq L_2$. Then, there exists a tree $t \in L_2 \setminus L_1$.*

Proof. The lemma follows immediately, since otherwise we have $L_1 = L_2$ and therefore $da(L_1) = da(L_2)$, in contradiction to $da(L_1) \neq da(L_2)$. □

Lemma 3.7. *Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ be a PTA. Then, there is a PTA $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, \{q_I^{\mathcal{B}}\}, \delta_{\mathcal{B}}, \mathbb{C})$ with single initial state such that $L(\mathcal{B}) = L(\mathcal{A})$, and $da(\mathcal{B}) \leq da(\mathcal{A})$.*

Proof. Let $Q_{\mathcal{B}} := Q \cup \{q_I^{\mathcal{B}}\}$ and $\delta_{\mathcal{B}} := \delta_{\mathcal{A}} \cup \{(q_I^{\mathcal{B}}, a, q_l, q_r) \mid q_I \in Q_I \text{ and } (q_I, a, q_l, q_r) \in \delta\}$. It is easy to see that $L(\mathcal{B}) = L(\mathcal{A})$.

Let $t \in L(\mathcal{A})$, and let g_t be a function from $ACC(\mathcal{A}, t)$ to $ACC(\mathcal{B}, t)$ which maps each computation $\phi \in ACC(\mathcal{A}, t)$ to a computation ϕ' which assigns $q_I^{\mathcal{B}}$ to node ϵ, and $\phi(v)$ to other nodes. It is easy to see that $\phi' \in ACC(\mathcal{B}, t)$, and that g_t is surjective, and therefore $\forall t : |ACC(\mathcal{A}, t)| \geq |ACC(\mathcal{B}, t)|$, as requested. □

Definition 3.8 (Moore machine)**.** A Moore machine is a tuple $M = (\Sigma, \Gamma, Q, q_I, \delta, out)$, where Σ is a finite input alphabet, Q is a finite set of states, $q_I \in Q$ is an initial state, $\delta : Q \times \Sigma \to Q$ is a transition function, Γ is an output alphabet, and $out : Q \to \Gamma$ is an output function.

Define $\widehat{\delta} : \Sigma^* \to Q$ by $\widehat{\delta}(\epsilon) := q_I$ and $\widehat{\delta}(w) := \delta(\widehat{\delta}(w'), a)$ for $w = w' \cdot a$ where $w' \in \Sigma^*$ and $a \in \Sigma$. We say that a function $F : \Sigma^* \to \Gamma$ is definable by a Moore machine if there is a Moore machine M such that $F(w) = out(\widehat{\delta}(w))$ for all $w \in \Sigma^*$.

Definition 3.9. Let $F : \Sigma_1^* \to \Sigma_2$ be a function definable by a Moore machine, and let $t_1 \in T_{\Sigma_1}^{\omega}$. We define $t_2 := \widehat{F}(t_1)$ as a tree in $T_{\Sigma_2}^{\omega}$ such that $t_2(v) := F(t_1(v_1) \cdot \dots \cdot t_1(v_k))$ where $v_1, v_2, \dots, v_k$ is the path from the root to v.

For a tree language $L \subseteq T_{\Sigma_1}^{\omega}$, we define $\widehat{F}(L) := \{\widehat{F}(t) \mid t \in L\} \subseteq T_{\Sigma_2}^{\omega}$.

Lemma 3.10 (Reduction)**.** *Let L_1 and L_2 be regular tree languages over alphabets Σ_1 and Σ_2, respectively. Let $F : \Sigma_1^* \to \Sigma_2$ be a function definable by a Moore machine. Assume that for each $t \in T_{\Sigma_1}^{\omega}$, $t \in L_1$ iff $\widehat{F}(t) \in L_2$. Then $da(L_1) \leq da(L_2)$.*

Proof. Let $\mathcal{A}_2 = (Q_2, \Sigma_2, Q_I^2, \delta_2, \mathbb{C}_2)$ such that $\mathcal{A}_2$ accepts L_2 and $da(\mathcal{A}_2) = da(L_2)$.

Let $M = (\Sigma_1, \Sigma_2, Q_M, q_I^M, \delta_M, out_M)$ be a Moore machine defining F. We will use $\mathcal{A}_2$ and M to define an automaton $\mathcal{A}_1 = (Q_1, \Sigma_1, Q_I^1, \delta_1, \mathbb{C}_1)$ such that $t \in L(\mathcal{A}_1)$ iff $\widehat{F}(t) \in L(\mathcal{A}_2)$, by:

- $Q_1 := Q_2 \times Q_M$
- $Q_I^1 := Q_I^2 \times \{q_I^M\}$
- $((q, p), a, (q_1, p_1), (q_2, p_2)) \in \delta_1$ iff $p_1 = p_2 = \delta_M(p, a)$ and $(q, out_M(p), q_1, q_2) \in \delta_2$
- $\mathbb{C}_1(q, p) := \mathbb{C}_2(q)$

First notice that $\forall t \in T_{\Sigma}^{\omega} : t \in L(\mathcal{A}_1) \Leftrightarrow \widehat{F}(t) \in L(\mathcal{A}_2) \Leftrightarrow \widehat{F}(t) \in L_2 \Leftrightarrow t \in L_1$, and therefore $L(\mathcal{A}_1) = L_1$ as needed.

Let $\phi \in ACC(\mathcal{A}_1, t)$, and define a computation ϕ' by $\phi'(v) = q_1$ for $\phi(v) = (q_1, q_2) \in Q_2 \times Q_M$. It is easy to see that $\phi' \in ACC(\mathcal{A}_2, \widehat{F}(t))$ and since M is deterministic, we conclude that $|ACC(\mathcal{A}_1, t)| \leq |ACC(\mathcal{A}_2, \widehat{F}(t)|)|$, and therefore $da(\mathcal{A}_1) \leq da(\mathcal{A}_2)$.

We conclude that $da(L_1) \leq da(\mathcal{A}_1) \leq da(\mathcal{A}_2) = da(L_2)$, as requested. □

Let us state another well-known characterization of regular trees.

Fact 3.11. A tree t is regular iff its labelling $t : \{l,r\}^* \to \Sigma$ is definable by a Moore machine.

4. Not-Finitely Ambiguous Languages

We provide here sufficient conditions for a language to be not finitely ambiguous. First, we state our main technical result - Proposition 4.1. Then, we derive some consequences. Finally, a proof of Proposition 4.1 is given. Our proof relies on the fact that there is no MSO-definable function which assigns to every non-empty antichain Y a finite non-empty subset $X \subseteq Y$ (Lemma 2.5), and our proof techniques refine the proof techniques of [5].

Notations. For trees t and t' and an antichain Y, we denote by $t[t'/Y]$ the tree obtained from t by grafting t' at every node in Y.

Proposition 4.1. *Let t_0 and t_1 be regular trees and L be a regular language such that $t_0 \notin L$ and $t_0[t_1/Y] \in L$ for every non-empty antichain Y. Then L is not finitely ambiguous.*

Definition 4.2. For a tree language L over alphabet Σ, we denote by *Subtree*(L) the tree language $\{t \in T^\omega_\Sigma \mid \exists t' \in L\, \exists v : t'_{\geq v} = t\}$.

Corollary 4.3. *Let L be a non-empty regular language over an alphabet Σ such that Subtree$(L) \neq T^\omega_\Sigma$. Then, the complement of L is not finitely ambiguous.*

Proof. Let L be as in Corollary 4.3. We claim that there are regular Σ-labeled trees $t_0 \in L$ and $t_1 \notin$ *Subtree*(L). Indeed, by Rabin's basis theorem there is a regular $t_0 \in L$. Since L is regular, there is an automaton $\mathcal{B} = (Q, \Sigma, \{q_I\}, \delta, \mathbb{C})$ (with only useful states) which accepts L. It is clear that $\mathcal{B}_Q$ accepts *Subtree*(L), and therefore *Subtree*(L) is regular. The complement of *Subtree*(L) is regular (as the complement of a regular language) and non-empty (since *Subtree*$(L) \neq T^\omega_\Sigma$), and therefore contains a regular tree t_1 (by Rabin's basis theorem). Note that $t_0[t_1/Y] \notin L$ for every non-empty antichain Y.

The complement of L satisfies the assumption of Proposition 4.1. Therefore, it is not finitely ambiguous. □

Corollary 4.4 (not finitely ambiguous languages)**.** *The following languages are not finitely ambiguous:*

(1) *The complement of a non-empty regular countable tree language.*
(2) *The complement of a regular language which contains a single tree.*
(3) *The language $L_{\exists a_1} := \{t \in T^\omega_\Sigma \mid t$ has at least one node labeled by $a_1\}$ over alphabet $\Sigma = \{a_1, \dots, a_m, c\}$.*

Proof. (1) Every tree has countably many subtrees. Since L is countable, we conclude that *Subtree*(L) is countable. Therefore, *Subtree*(L) does not contain all trees. By Proposition 4.3, we conclude that $\overline{L}$ is not finitely ambiguous.

(2) Follows immediately from (1).

(3) By the definition of $L_{\exists a_1}$ we have $L_{\exists a_1} \cap T^\omega_{\{c,a_1\}} = T^\omega_{\{c,a_1\}} \setminus \{t_c\}$, and therefore by (2), $L_{\exists a_1} \cap T^\omega_{\{c,a_1\}}$ is not finitely ambiguous. It is easy to see that $T^\omega_{\{c,a_1\}}$ is unambiguous (since there is a deterministic automaton which accepts it). Therefore, by Corollary 3.2 we conclude that $L_{\exists a_1}$ is not finitely ambiguous. □

It is easy to prove that the complement of every finite language is countably ambiguous. Therefore, we obtain:

Corollary 4.5. *If L is regular and its complement is finite and non-empty, then $da(L) = \aleph_0$.*

Proof of Corollary 4.5. We first prove the following claim:

Claim 4.5.1. *Let L be a regular tree language containing a single tree. Then $\overline{L}$ is countably ambiguous.*

Proof. Assume that $L = \{t\}$. L is a regular language, and therefore t is regular. We conclude that there is a Moore machine $M = (\{l,r\}, \Sigma, Q_M, q_I^M, \delta_M, out_M)$ such that for each $v \in \{l,r\}^*$, $out(\widehat{\delta}(v)) = \sigma$ iff $t(v) = \sigma$ (that is, M defines the function $t : \{l,r\}^* \to \Sigma$).

We will use M to construct a countably ambiguous automaton $\mathcal{A}$ which accepts $\overline{L}$ by guessing a node $v \in \{l,r\}^*$ such that $t(v) \neq t'(v)$ for each tree $t' \in \overline{L}$.

Let $\mathcal{A} := (Q_{\mathcal{A}}, \Sigma, Q_I, \delta, \mathbb{C})$ such that:

- $Q_{\mathcal{A}} := \{q, q'\} \times Q_M$
- $Q_I := \{(q', q_I^M)\}$
- δ is defined by:
  - $((q,p), a, (q,p'), (q,p'')) \in \delta$ iff $\delta_M(p,l) = p'$, $\delta_M(p,r) = p''$
  - $((q',p), a, (q,p'), (q,p'')) \in \delta$ iff $\delta_M(p,l) = p'$, $\delta_M(p,r) = p''$ and $out(p) \neq a$
  - $((q',p), a, (q',p'), (q,p'')), ((q',p), a, (q,p'), (q',p'')) \in \delta$ iff $\delta_M(p,l) = p'$, $\delta_M(p,r) = p''$ and $out(p) = a$.
- $\forall p \in Q_M : \mathbb{C}(q,p) := 0$ and $\mathbb{C}(q',p) := 1$

By definition of $\mathcal{A}$, it is clear that $t' \in L(\mathcal{A})$ iff there is a node v such that $t'(v) \neq t(v)$, and therefore $t' \in L(\mathcal{A})$ iff $t' \neq t$.

For each computation ϕ of $\mathcal{A}$ on t', the Q_M component is determined deterministically by M and t. If ϕ is accepting, there are finitely many nodes v such that the first component of $\phi(v)$ is q' - otherwise, there would be a branch where the maximal color assigned infinitely often by $\mathbb{C}$ is odd, in contradiction to ϕ being an accepting computation. Therefore, there are countably many accepting computations on each tree $t' \in L(\mathcal{A})$, and $\mathcal{A}$ is countably ambiguous. ■

L is finite and therefore there are $t_1, \dots, t_k \in T_\Sigma^\omega$ such that $L = \{t_1, \dots, t_k\}$. A finite tree language does not contain a non-regular tree, and therefore $t_1, \dots, t_k$ are regular. By Claim 4.5.1, for each tree $t_i \in L$, there is a countably ambiguous automaton $\mathcal{A}_i$ such that $t \in L(\mathcal{A}_i)$ iff $t \neq t_i$. Notice that $\overline{L} = L(\mathcal{A}_1) \cap \dots, \cap L(\mathcal{A}_k)$, and therefore by Lemma 3.1 we conclude that $\overline{L}$ is countably ambiguous. □

On the proof of Proposition 4.1. In the rest of this section, Proposition 4.1 is proved. Let us sketch some ideas of the proof. For a language L, as in Proposition 4.1, and any non-empty antichain Y we show that if $\mathcal{A}$ does not accept t_0 and accepts $t := t_0[t_1/Y]$, then every $\phi \in ACC(\mathcal{A}, t)$ chooses (in an MSO-definable way) an element from Y. Hence, the computations in $ACC(\mathcal{A}, t)$ choose together a subset X of Y of cardinality $\leq |ACC(\mathcal{A}, t)|$ (each computation chooses a single element). Therefore, if $\mathcal{A}$ accepts L and is finitely ambiguous, then X is finite - a contradiction to Lemma 2.5. To implement this plan, in Subsect. 4.1 we recall a game theoretical interpretation of "a tree is accepted by an automaton." Then, in Subsect. 4.2 we analyze which concepts related to these games are MSO-definable. Finally, in Subsect. 4.3, the proof is completed.

4.1. **Membership Game.** Let $\mathcal{A} = (Q, \Sigma, \{q_I\}, \delta, \mathbb{C})$ be a PTA, and let t be a Σ-labeled tree. A two-player game $G_{t,\mathcal{A}}$ (called a "membership game") between Automaton and Pathfinder is defined as follows. The positions of Automaton are $\{l, r\}^* \times Q$, and the positions of Pathfinder are $\{l, r\}^* \times Q \times Q$. The initial position is (ϵ, q_I).

From a position $(v, q) \in \{l, r\}^* \times Q$ Automaton chooses a tuple $(q_l, q_r) \in Q \times Q$ such that $\exists a \in \Sigma : (q, a, q_l, q_r) \in \delta$, and moves to the position (v, q_l, q_r). From a position $(v, q_l, q_r) \in \{l, r\}^* \times Q \times Q$ Pathfinder chooses a direction $d \in \{l, r\}$, and moves to the position $(v \cdot d, q_d)$.

We define a **play** $\overline{s} := e_0, d_0, e_1, d_1, \ldots, e_i, d_i, \cdots \in (Q \times Q \times \{l, r\})^\omega$ as an infinite sequence of moves, corresponding to the choices of Automaton and Pathfinder from the initial position. We say that the move $e_i = (q_l, q_r)$ from position (q, v) is **invalid** for Automaton if $(q, t(v), q_l, q_r) \notin \delta$.

A **strategy** for a player in $G_{t,\mathcal{A}}$ is a function which determines the next move of the player based on previous moves of both players.

A **positional strategy** for a player in $G_{t,\mathcal{A}}$ is a strategy which determines the next move of the player based only on the current position. A positional strategy for Automaton is a function $str : \{l, r\}^* \times Q \to Q \times Q$, and a positional strategy for Pathfinder is a function $STR : \{l, r\}^* \times Q \times Q \to \{l, r\}$.

Let $\mathbb{C}_G$ be a coloring function which maps each position in $G_{t,\mathcal{A}}$ to a color in $\mathbb{N}$. We define $\mathbb{C}_G(v, q) := \mathbb{C}(q)$ for Automaton's positions, and $\mathbb{C}_G(v, q_l, q_r) := 0$ for Pathfinder's positions.

For each play $\overline{s}$ define $\pi_{\overline{s}}$ as the infinite sequence of positions corresponding to the moves in $\overline{s}$. A play $\overline{s}$ is winning for Automaton iff $\overline{s}$ does not contain an invalid move for Automaton, and the maximal color which $\mathbb{C}_G$ assigns infinitely often to the positions in $\pi_{\overline{s}}$ is even. Since all Pathfinder's positions are colored by 0, it is sufficient to consider the coloring of Automaton's positions in $\pi_{\overline{s}}$.

We say that a play is consistent with a strategy of a player if all moves of the player are according to the strategy. A **winning strategy** for a player is a strategy such that each play which is consistent with the strategy is winning for the player.

Parity games are positionally determined [7], i.e., for each parity game, one of the players has a positional winning strategy. Therefore, if a player has a winning strategy, then he has a positional winning strategy. Additionally, if a positional strategy of a player wins against all positional strategies of the other player, then it is a winning strategy.

We recall standard definitions and facts about the connections between games and tree automata [8, 16].

Let $\phi : \{l, r\}^* \to Q$ be a function such that $\phi(\epsilon) = q_I$ and $\forall v \in \{l, r\} : \exists a \in \Sigma : (\phi(v), a, \phi(v \cdot l), \phi(v \cdot r)) \in \delta$. We define a positional strategy $str_\phi : \{l, r\}^* \times Q \to Q \times Q$ for Automaton, by $str_\phi(v, q) := (\phi(v \cdot l), \phi(v \cdot r))$. Conversely, for each positional strategy $str : \{l, r\}^* \times Q \to Q \times Q$ of Automaton we construct a function $\phi_{str} : \{l, r\}^* \to Q$ by $\phi(\epsilon) := q_I$ and for all $v \in \{l, r\}^*$ we set $\phi(v \cdot l) := q_l$, and $\phi(v \cdot r) := q_r$ where $str(v, \phi(v)) = (q_l, q_r)$.

Claim 4.1.1. (1) *Let $\overline{s}$ be a play which is consistent with str_ϕ, and let (v_i, q_i) be the i-th position of Automaton in $\pi_{\overline{s}}$. Then, $\phi(v_i) = q_i$.*

(2) *If $\phi \in ACC(\mathcal{A}, t)$, then str_ϕ is a positional winning strategy for Automaton.*

(3) *If str is a positional winning strategy for Automaton, then $\phi_{str} \in ACC(\mathcal{A}, t)$.*

Proof. (1) We will prove by induction on i. For $i = 0$ we have $(v_0, q_0) = (\epsilon, q_I)$ (by definition of $G_{t,\mathcal{A}}$), and indeed $\phi(v_0) = \phi(\epsilon) = q_I$. Assume the claim holds for $i = k$ and we prove for $i = k + 1$.

Let $d \in \{l, r\}$ be the i-th move of Pathfinder in $\overline{s}$. By definition of $G_{t,\mathcal{A}}$ we have $v_{i+1} = v_i \cdot d$, and $q_{i+1} = q_d$, where $str_\phi(v_i, q_i) = (q_l, q_r)$.

By definition str_ϕ we have $(q_l, q_r) = (\phi(v_i \cdot l), \phi(v_i \cdot l))$, and therefore $q_{i+1} = \phi(v_i \cdot d) = \phi(v_{i+1})$, as requested.

(2) and (3) are well known results about membership games [16]. ■

The next claim describes what happens when Pathfinder plays his winning strategy in $G_{t,\mathcal{A}}$ against an Automaton's winning strategy in $G_{t',\mathcal{A}}$ (for $t' \neq t$).

Claim 4.1.2. *Assume $t \notin L(\mathcal{A})$ and let ϕ be an accepting computation of $\mathcal{A}$ on a tree t', and STR be a winning strategy of Pathfinder in $G_{t,\mathcal{A}}$. Let $\overline{s} := e_0, d_0, e_1, d_1, \ldots, e_i, d_i, \ldots$ be the play which is consistent with str_ϕ and STR. Then, there is $i \in \mathbb{N}$ such that e_i is an invalid move for Automaton in $G_{t,\mathcal{A}}$. Moreover, if e_i is the first invalid move for Automaton in $\overline{s}$, then $t(v) \neq t'(v)$ for $v := d_0 \ldots d_{i-1}$.*

Proof. Assume, for the sake of contradiction, that $\overline{s}$ does not contain an invalid move for Automaton, and let (v_i, q_i) be the i-th position of Automaton in $\pi_{\overline{s}}$. By definition of $G_{t,\mathcal{A}}$ it is easy to see that $\pi = v_0, \ldots, v_i, \ldots$ is a branch in the full-binary tree. Since ϕ is an accepting computation of $\mathcal{A}$ on t', we conclude that the maximal color which $\mathbb{C}$ assigns infinitely often to states in $\phi(\pi)$ is even. By Claim 4.1.1(1) we have $\phi(v_i) = q_i$, and therefore $\phi(\pi) = q_0 \ldots q_i \ldots$. By the definition of $\mathbb{C}_G$ we have $\mathbb{C}_G(v_i, q_i) = \mathbb{C}(q_i)$ and we conclude that the maximal color which $\mathbb{C}$ assigns infinitely often in $\pi_{\overline{s}}$ is even, and therefore the play is winning for Automaton - a contradiction to STR being a winning strategy of Pathfinder.

Therefore, Automaton makes an invalid move in $\overline{s}$. Let $e_i = (q_l, q_r)$ be the first invalid move of Automaton in $\overline{s}$. Since e_i is invalid we have $(q_i, t(v_i), q_l, q_r) \notin \delta$, and by definition of str_ϕ we obtain $(q_l, q_r) = (\phi(v_i \cdot l), \phi(v_i \cdot r))$. Since $\phi(v_i) = q_i$ we have $(\phi(v_i), t(v_i), \phi(v_i \cdot l), \phi(v_i \cdot r)) \notin \delta$. ϕ is a computation of $\mathcal{A}$ on t' and therefore $(\phi(v_i), t'(v_i), \phi(v_i \cdot l), \phi(v_i \cdot r)) \in \delta$, and we conclude that $t(v_i) \neq t'(v_i)$. Notice that by the definition of $G_{t,\mathcal{A}}$ we have $v_i = d_0 \ldots d_{i-1}$, and the claim follows. ■

4.2. **MSO-definability.** Throughout this section we will use the following conventions and terminology.

Positional Pathfinder strategies as labeled trees: A positional strategy STR for Pathfinder is a function in $\{l, r\}^* \times Q \times Q \to \{l, r\}$. Hence, it can be considered as a $Q \times Q \to \{l, r\}$ labeled tree. Below we will not distinguish between a positional Pathfinder's strategy and the corresponding $Q \times Q \to \{l, r\}$ labeled full-binary tree. In particular, we call such a strategy regular, if the corresponding tree is regular.

MSO-definability: We will use "MSO-definable" for "MSO-definable in the unlabeled full-binary tree."

The rest of the proof deals with MSO-definability. By Claim 4.1.2, there is a function $Invalid_{\mathcal{A}}(\phi, STR, t, v)$ which, for every accepting computation ϕ of $\mathcal{A}$ on t', returns a node v such that $t'(v) \neq t(v)$. This function depends on the strategy STR of Pathfinder. The restriction of $Invalid_{\mathcal{A}}$ to the Pathfinder positional winning strategies in $G_{t,\mathcal{A}}$ is MSO-definable (with parameters t and STR) by the following formula $Leads_{\mathcal{A}}(\phi, STR, t, v)$, which describes in MSO the play of ϕ against STR up to the first invalid move of Automaton (at the position $(v,\phi(v))$.

Define $Leads_{\mathcal{A}}(\phi, STR, t, v)$ as the conjunction of:

(1) $\phi(\epsilon)=q_I$ -the play starts from the initial position.
(2) $\forall u<v : ((\phi(u),t(u),\phi(u\cdot l),\phi(u\cdot r))\in\delta$ - all Automaton's moves at the positions (u,q), where u is an ancestor of v respect δ. (By Claim 4.1.1(1), in any play consistent with ϕ, Automaton can reach only the positions of the form $(u,\phi(u))$).
(3) $(\phi(v),t(v),\phi(v\cdot l),\phi(v\cdot r))\notin\delta$ - the Automaton move at $(v,\phi(v))$ is invalid.
(4) $\forall u<v : (STR(u,\phi(u\cdot l),\phi(u\cdot r))=l)\leftrightarrow u\cdot l\leq v))$ - the Pathfinder moves $d_0\dots d_j\dots$ are consistent with STR and are along the path from the root to v, i.e., $d_0d_1\dots d_j\leq v$.

To sum up, we have the following claim:

Claim 4.1.3. *$Leads_{\mathcal{A}}(\phi, STR, t, v)$ defines a function which, for every tree $t\notin L(\mathcal{A})$, every Pathfinder's positional (in $G_{t,\mathcal{A}}$) winning strategy STR, and every $\phi\in ACC(\mathcal{A},t')$, returns a node v such that $t(v)\neq t'(v)$.*

Claim 4.1.3 plays a crucial role in our proof. It is instructive to compare it with Theorem 2.4 which implies that there is no MSO-definable function $F(t,D,v)$ which for a tree $t\neq t'$ and $D:=\{u\mid t(u)\neq t'(u)\}$ returns a node v such that $t(v)\neq t'(v)$.

The following claim is folklore. Due to the lack of references, it is proved in the Appendix.

Claim 4.1.4. *Let t_0 be a regular tree such that $t_0\notin L(\mathcal{A})$. Then, Pathfinder has a regular positional winning strategy in $G_{t_0,\mathcal{A}}$.*

Let t_0 be a regular tree such that $t_0\notin L(\mathcal{A})$. By Claim 4.1.4 there is a regular positional winning strategy $\widehat{STR}$ of Pathfinder in $G_{t_0,\mathcal{A}}$. Now, we can substitute $\widehat{STR}$ and t_0 for arguments STR and t of $Leads_{\mathcal{A}}$ and obtain the following Proposition:

Proposition 4.6. *For every regular tree $t_0\notin L(\mathcal{A})$ and a regular positional winning strategy $\widehat{STR}$ for Pathfinder in $G_{t_0,\mathcal{A}}$, there is an MSO-definable function which, for each accepting computation ϕ of $\mathcal{A}$ on t', returns a node v such that $t_0(v)\neq t'(v)$.*

Proof. Let $\psi_{t_0}(\sigma)$ and $\psi_{\widehat{STR}}(STR)$ be MSO-formulas that define t_0 and $\widehat{STR}$. Then, by Claim 4.1.3, $\exists\sigma\exists STR : \psi_{t_0}(\sigma)\wedge\psi_{\widehat{STR}}(STR)\wedge Leads_{\mathcal{A}}(\phi,STR,\sigma,v)$ defines such a function. ■

Let us continue with the proof of Proposition 4.1. Recall that for trees t and t' and an antichain Y, we denote by $t[t'/Y]$ the tree obtained from t by grafting t' at every node in Y.

Claim 4.1.5. *Let t_0 and t_1 be regular trees. Then, there is an MSO-formula $graft_{t_0,t_1}(Y,\sigma)$ defining a function which for every antichain Y returns the tree $t_0[t_1/Y]$.*

Proof of Claim 4.1.5. t_0 and t_1 are regular, and therefore there are MSO-formulas $\psi_{t_0}(\sigma)$ and $\psi_{t_1}(\sigma)$ which defines t_0 and t_1.

Let $\psi_{t_1}^{\geq y}(y,\sigma)$ be a formula which is obtained from $\psi_{t_1}(\sigma)$ by relativizing the first-order quantifiers to $\geq y$, i.e., by replacing subformulas of the form $\exists x(\dots)$ and $\forall x(\dots)$ by $\exists x(x\geq y)\wedge(\dots)$ and $\forall x(x\geq y)\rightarrow(\dots)$. Then, $v,t\models\psi_{t_1}^{\geq y}(y,\sigma)$ iff $t_{\geq v}=t_1$. Hence, $graft_{t_0,t_1}(Y,\sigma)$ can be defined as the conjunction of:

(1) $\exists \sigma_0 \psi_{t_0}(\sigma_0) \wedge \forall v$ - "if no Y node is an ancestor of v then $\sigma(v) = \sigma_0(v)$," and
(2) $\forall y (y \in Y) \rightarrow \psi_{t_1}^{\geq y}(y, \sigma)$ - "at every node in Y a tree t_1 is grafted."

■

4.3. **Finishing Proof of Proposition 4.1.** Now, we have all the ingredients ready for the proof of Proposition 4.1.

Let $\mathcal{A}$ be such that $L(\mathcal{A}) = L$, and let $\alpha_{t_0, \mathcal{A}, \widehat{STR}}(\phi, v)$ be a formula which defines the function from Proposition 4.6 ($t_0[t_1/Y]$ now takes the role of t').

Define a formula: $Choice_{\mathcal{A}, t_0, t_1, \widehat{STR}}(Y, \phi, y) := y \in Y \wedge \exists v (\alpha_{t_0, \mathcal{A}, \widehat{STR}}(\phi, v) \wedge v \geq y)$.

Claim 4.1.6. *$Choice_{\mathcal{A}, t_0, t_1, \widehat{STR}}(Y, \phi, y)$ defines a function which for every non-empty antichain Y and an accepting computation ϕ of $\mathcal{A}$ on $t_0[t_1/Y]$, returns a node $y \in Y$.*

Proof. By Proposition 4.6, $\alpha_{t_0, \mathcal{A}, \widehat{STR}}(\phi, v)$ returns a node v such that $t_0(v) \neq (t_0[t_1/Y])(v)$. By definition of $t_0[t_1/Y]$, there is a unique node $y \in Y$ such that $v \geq y$. ■

Define $ChooseSubset_{\mathcal{A}, t_0, t_1, \widehat{STR}}(Y, X) := \forall x : x \in X$ iff the following conditions hold:
(1) $x \in Y$ and
(2) $\exists \sigma$ such that
  (a) $graft_{t_0, t_1}(Y, \sigma)$ - "$\sigma = t_0[t_1/Y]$" and
  (b) $\exists \phi AcceptingRun_{\mathcal{A}}(\sigma, \phi) \wedge Choice_{\mathcal{A}, t_0, t_1, \widehat{STR}}(Y, \phi, x)$, where $AcceptingRun_{\mathcal{A}}(\sigma, \phi)$ defines "ϕ is an accepting computation of $\mathcal{A}$ on the tree σ."

Claim 4.1.7. *$ChooseSubset_{\mathcal{A}, t_0, t_1, \widehat{STR}}(Y, X)$ defines a function which maps every non-empty antichain Y to a non-empty subset $X \subseteq Y$. Moreover, $|X| \leq |ACC(A, t_0[t_1/Y])|$.*

Proof. If Y is non-empty, then $t_0[t_1/Y] \in L$. Hence, $\mathcal{A}$ has at least one accepting computation on $t_0[t_1/Y]$. Therefore, X is non-empty, by Claim 4.1.6. The "Moreover" part immediately follows from Claim 4.1.6. ■

Let $\mathcal{A}$ be such that $L(\mathcal{A}) = L$ and assume, for the sake of contradiction, that $\mathcal{A}$ is finitely ambiguous. In particular, there are finitely many accepting computations of $\mathcal{A}$ on $t_0[t_1/Y]$, and therefore by Claim 4.1.7, we conclude that $ChooseSubset_{\mathcal{A}, t_0, t_1, \widehat{STR}}(Y, X)$ assigns to every non-empty antichain Y a finite non-empty $X \subseteq Y$ - a contradiction to Lemma 2.5.

5. k-Ambiguous Languages

In this section we prove that for every $0 < k \in \mathbb{N}$, there is a tree language with the degree of ambiguity equal to k. First, we introduce some notations. For a letter σ, we denote by t_σ, the full-binary tree with all nodes labeled by σ. Let $L_{\neg a_1 \vee \cdots \vee \neg a_k} := L_{\neg a_1} \cup \cdots \cup L_{\neg a_k}$ be a tree language over alphabet $\Sigma_n = \{c, a_1, a_2, ..., a_n\}$, where $L_{\neg a_i} := \{t \in T_{\Sigma_n}^{\omega} \mid \text{no node in } t \text{ is labeled by } a_i\}$.

Proposition 5.1. *The degree of ambiguity of $L_{\neg a_1 \vee \cdots \vee \neg a_k}$ for $k \leq n$ is k.*

It is easy to see that $L_{\neg a_i}$ are accepted by deterministic PTA. Therefore, by Lemma 3.1, we obtain that $L_{\neg a_1 \vee \cdots \vee \neg a_k}$ is k-ambiguous. In the rest of this section we will show that $L_{\neg a_1 \vee \cdots \vee \neg a_k}$ is not $(k-1)$-ambiguous. It was shown in [3] that $L_{\neg a_1 \vee \neg a_2}$ is ambiguous.

Lemma 5.2. *Let $L_{\exists a_1 \wedge \dots \wedge \exists a_m} := \{t \in T^\omega_{\Sigma_n} \mid$ for every $i \leq m$ there is a node in t labeled by $a_i\}$, and let L be a tree language such that $t_c \notin L$ and $L_{\exists a_1 \wedge \dots \wedge \exists a_m} \cap T^\omega_{\{c,a_1,\dots,a_m\}} \subseteq L$. Then, L is not finitely ambiguous.*

Proof. Define a function $F : \Sigma^* \to \Sigma$ such that $F(\sigma_1 \dots \sigma_k) := a_{k-i+1}$ if there is i such that $\sigma_i = a_1$, for all $j < i : \sigma_j \neq a_1$ and $k - i + 1 \leq m$. Otherwise, $F(\sigma_1 \dots \sigma_k) := c$.

It is easy to see that F is definable by a Moore machine, and $\forall t \in T^\omega_\Sigma : t \in L_{\exists a_1}$ iff $\widehat{F}(t) \in L$. Therefore, by Lemma 3.10 we conclude that $da(L) \geq da(L_{\exists a_1})$. Since $L_{\exists a_1}$ is not finitely ambiguous (by Corollary 4.4 (3)), we conclude that L is not finitely ambiguous. □

Notations. Let $a \in \Sigma$, $t_1 \in T^\omega_\Sigma$ and $t_2 \in T^\omega_\Sigma$. We define $Tree(a, t_1, t_2) \in T^\omega_\Sigma$ as a tree t where $t(\epsilon) = a$, $t_{\geq l} = t_1$ and $t_{\geq r} = t_2$.

Lemma 5.3. *Let $\mathcal{A}$ be a finitely ambiguous automaton over alphabet Σ_n such that $L(\mathcal{A}) = L_{\neg a_1 \vee \dots \vee \neg a_k}$ for $k \leq n$. Then $|ACC(\mathcal{A}, t_c)| \geq k$.*

Proof. We will prove by induction on k. For $k = 1$ the claim holds trivially, since $t_c \in L(\mathcal{A})$ implies that $|ACC(\mathcal{A}, t_c)| \geq 1$.

Assume the claim holds for all $k < m \leq n$ and prove for $k = m$.

Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ be a finitely ambiguous automaton which accepts $L_{\neg a_1 \vee \dots \vee \neg a_m}$. Define $R := \{(q_1, q_2) \in Q \times Q \mid \exists q_i \in Q_I : (q_i, c, q_1, q_2) \in \delta)\}$, and let $R[1]$ and $R[2]$ be the projections of the first and second coordinate of R on Q, respectively.

Define $Q_{\exists a_m} := \{q \in R[1] \mid L(A_q) \cap L_{\exists a_m} \neq \emptyset\}$, and let $Q_{\exists a_m \wedge t_c} := \{q \in Q_{\exists a_m} \mid t_c \in L(\mathcal{A}_q)\}$ and $Q_{\exists a_m \wedge \neg t_c} := Q_{\exists a_m} \setminus Q_{\exists a_m \wedge t_c}$.

By definition of $Q_{\exists a_m \wedge \neg t_c}$ we have $t_c \notin L(\mathcal{A}_{Q_{\exists a_m \wedge \neg t_c}})$ and therefore $L(\mathcal{A}_{Q_{\exists a_m \wedge \neg t_c}}) \cap T^\omega_{\{c,a_m\}} \subseteq T^\omega_{\{c,a_m\}} \setminus \{t_c\}$. The language $T^\omega_{\{c,a_m\}} \setminus \{t_c\}$ is not finitely ambiguous by Corollary 4.4 (2). $L(\mathcal{A}_{Q_{\exists a_m \wedge \neg t_c}})$ is finitely ambiguous (by Corollary 3.5) and since $T^\omega_{\{c,a_m\}}$ is unambiguous we conclude that $L(\mathcal{A}_{Q_{\exists a_m \wedge \neg t_c}}) \cap T^\omega_{\{c,a_m\}}$ is finitely ambiguous, by Corollary 3.2. Therefore, by Lemma 3.6, there is a tree $t' \in T^\omega_{\{c,a_m\}} \setminus \{t_c\} = L_{\exists a_m} \cap T^\omega_{\{c,a_m\}}$ such that $t' \notin L(\mathcal{A}_{Q_{\exists a_m \wedge \neg t_c}})$, and since $L_{\exists a_m} \cap T^\omega_{\{c,a_m\}} \subseteq L(\mathcal{A}_{Q_{\exists a_m}}) = L(\mathcal{A}_{Q_{\exists a_m \wedge t_c}}) \cup L(\mathcal{A}_{Q_{\exists a_m \wedge \neg t_c}})$ we conclude that $t' \in L(\mathcal{A}_{Q_{\exists a_m \wedge t_c}})$.

Define $Q' := \{q \in R[1] \mid t' \in L(\mathcal{A}_q)\}$ and $R' := \{(q_1, q_2) \in R \mid q_1 \in Q'\}$. Since $t' \in L_{\exists a_m} \cap T^\omega_{\{c,a_m\}}$, we conclude that $\{t \in T^\omega_\Sigma \mid Tree(c, t', t) \in L_{\neg a_1 \vee \dots \vee \neg a_m}\} = L_{\neg a_1 \vee \dots \vee \neg a_{m-1}}$. Therefore, $L(\mathcal{A}_{R'[2]}) = L_{\neg a_1 \vee \dots \vee \neg a_{m-1}}$, and by induction assumption we obtain $|ACC(\mathcal{A}_{R'[2]}, t_c)| \geq m - 1$.

For each computation $\phi \in ACC(\mathcal{A}_{R'[2]}, t_c)$ we will construct a computation $g(\phi) \in ACC(\mathcal{A}, t_c)$, as following. Let $q_2 := \phi(\epsilon)$. By the definition of R', there is $(q_1, q_2) \in R'$ such that $t' \in L(\mathcal{A}_{q_1})$. Since $t' \in L(\mathcal{A}_{Q_{\exists a_m \wedge t_c}})$ we have $t_c \in L(\mathcal{A}_{q_1})$, and therefore there is a computation $\phi_c \in ACC(\mathcal{A}_{q_1}, t_c)$. Let $q_i \in Q_I$ such that $(q_i, c, q_1, q_2) \in \delta$. By defining $g(\phi) := Tree(q_i, \phi_c, \phi)$ we obtain that $g(\phi) \in ACC(\mathcal{A}, t_c)$, as requested.

Let $\Phi := \{g(\phi) \mid \phi \in ACC(\mathcal{A}_{R'[2]}, t_c)\}$. $g(\phi)_{\geq r} = \phi$ and therefore g is injective, and we conclude that $|\Phi| = |ACC(\mathcal{A}_{R'[2]}, t_c)| \geq m - 1$.

We now need to find an additional computation $\phi \in ACC(\mathcal{A}, t_c)$ such that $\phi \notin \Phi$, resulting $|ACC(\mathcal{A}, t_c)| \geq m$.

Let $Q_{\exists a_1 \wedge \dots \wedge \exists a_{m-1}} := \{q \in R[2] \mid L(\mathcal{A}_q) \cap L_{\exists a_1 \wedge \dots \wedge \exists a_{m-1}} \neq \emptyset\}$ and let $Q_{t_c \wedge \exists a_1 \wedge \dots \wedge \exists a_{m-1}} := \{q \in Q_{\exists a_1 \wedge \dots \wedge \exists a_{m-1}} \mid t_c \in L(\mathcal{A}_q)\}$ and $Q_{\neg t_c \wedge \exists a_1 \wedge \dots \wedge \exists a_{m-1}} := Q_{\exists a_1 \wedge \dots \wedge \exists a_{m-1}} \setminus Q_{t_c \wedge \exists a_1 \wedge \dots \wedge \exists a_{m-1}}$.

Claim 5.3.1. *There is a tree $t'' \in L_{\exists a_1 \wedge \cdots \wedge \exists a_{m-1}} \cap T^{\omega}_{\{c,a_1,\ldots,a_{m-1}\}}$ such that $t'' \in L(\mathcal{A}_{Q_{t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}}})$ and $t'' \notin L(\mathcal{A}_{Q_{\neg t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}}})$.*

Proof. By the definition of $R[2]$ we have $L_{\exists a_1 \wedge \cdots \wedge \exists a_{m-1}} \cap T^{\omega}_{\{c,a_1,\ldots,a_{m-1}\}} \subseteq L(\mathcal{A}_{R[2]})$ and therefore by the definition of $Q_{t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}}$ and $Q_{\neg t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}}$, we have $L_{\exists a_1 \wedge \cdots \wedge \exists a_{m-1}} \cap T^{\omega}_{\{c,a_1,\ldots,a_{m-1}\}} \subseteq L(\mathcal{A}_{Q_{t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}}}) \cup L(\mathcal{A}_{Q_{\neg t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}}})$.

Assume, for the sake of contradiction, that the claim does not hold. Then, we obtain $L_{\exists a_1 \wedge \cdots \wedge \exists a_{m-1}} \cap T^{\omega}_{\{c,a_1,\ldots,a_{m-1}\}} \subseteq L(\mathcal{A}_{Q_{\neg t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}}})$. We have $t_c \notin L(\mathcal{A}_{Q_{\neg t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}}})$, and therefore by Lemma 5.2 we conclude that $L(\mathcal{A}_{Q_{\neg t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}}})$ is not finitely ambiguous - a contradiction to $\mathcal{A}$ being finitely ambiguous. ■

Let t'' be a tree as in Claim 5.3.1. We have $t'' \in L_{\exists a_1 \wedge \cdots \wedge \exists a_{m-1}} \cap T^{\omega}_{\{c,a_1,\ldots,a_{m-1}\}}$, and therefore $Tree(c, t_c, t'') \in L_{\neg a_1 \vee \cdots \vee \neg a_m} = L(\mathcal{A})$, and there is a computation $\phi \in ACC(\mathcal{A}, Tree(c, t_c, t''))$. Let $q := \phi(r)$. By definition of t'', we have $q \in Q_{t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}}$ and therefore $t_c \in L(\mathcal{A}_q)$. Let $\phi_c \in ACC(\mathcal{A}_q, t_c)$, and let ϕ' be the computation obtained from ϕ by grafting ϕ_c on r. We conclude that $\phi' \in ACC(\mathcal{A}, t_c)$.

Assume, for the sake of contradiction, that $\phi' \in \Phi$, and let $q_1 := \phi'(l)$ and $q_2 := \phi'(r)$. We have $t' \in L(\mathcal{A}_{q_1})$ (by definition of $|\Phi|$) and $t'' \in L(\mathcal{A}_{q_2})$ (by definition of ϕ'). Therefore, by grafting computations $\phi_{t'} \in ACC(\mathcal{A}_{q_1}, t')$ and $\phi_{t''} \in ACC(\mathcal{A}_{q_2}, t'')$ to the left and right children of the root of t_c, respectively, we obtain $Tree(c, t', t'') \in L(\mathcal{A})$. That is a contradiction, since t' contains an a_m labeled node, and t'' contains $a_1, \ldots, a_{m-1}$ labeled nodes, and therefore $Tree(c, t', t'') \notin L_{\neg a_1 \vee \cdots \vee \neg a_m}$.

We conclude that $\phi' \notin \Phi$, and therefore $|ACC(\mathcal{A}, t_c)| \geq 1 + |\Phi| = 1 + (m-1) = m$. □

6. Finitely Ambiguous Languages

Definition 6.1. Let $\Sigma = \{a_1, a_2, c\}$. We define the following languages over Σ:

- For $k, m \in \mathbb{N}$ such that $k < m$, we define $L_{k,m}$ as the set of trees t which are obtained from t_c by grafting a tree $t' \in L_{\neg a_1 \vee \neg a_2}$ on node $l^k r$, and grafting t_{a_1} on node l^m.
- For $m \in \mathbb{N}$ we define $L_m := \cup_{k<m} L_{k,m}$.
- $L^{fa} := \cup_{m \in \mathbb{N}} L_m$.

Proposition 6.2. *The degree of ambiguity of L^{fa} is finite.*

The proposition follows from Lemma 6.3 and Lemma 6.6 proved below.

Lemma 6.3. *There is a finitely ambiguous automaton which accepts L^{fa}*

Proof. On a tree $t \in L_m$ the automaton "guesses" a position $i < m$, checks that $t_{\geq l^i r} \in L_{\neg a_1 \vee \neg a_2}$ (using a 2-ambiguous automaton), checks that $t_{\geq l^j r} = t_c$ for all $j \neq i \wedge j < m$, and checks that $t_{\geq l^m} = t_{a_1}$ (using deterministic automata). Below, a more detailed proof is given.

First, notice that there are deterministic PTA $\mathcal{A}_c$, $\mathcal{A}_{a_1}$, $\mathcal{A}_{\neg a_1}$ and $\mathcal{A}_{\neg a_2}$ which accepts languages $\{t_c\}$, $\{t_{a_1}\}$, $L_{\neg a_1}$ and $L_{\neg a_2}$, respectively.

By Lemma 3.1, there is a 2-ambiguous automaton $\mathcal{A}_{\neg a_1 \vee \neg a_2}$ which accepts the language $L_{\neg a_1 \vee \neg a_2} := L_{\neg a_1} \cup L_{\neg a_2}$.

We will construct an automaton $\mathcal{B} := (Q_{\mathcal{B}}, \Sigma_{\mathcal{B}}, Q_{I_{\mathcal{B}}}, \delta_{\mathcal{B}}, \mathbb{C}_{\mathcal{B}})$ which accepts L^{fa}.

- $Q_{\mathcal{B}}$ is defined as the union of states of $\mathcal{A}_{a_1}$, $\mathcal{A}_c$ and $\mathcal{A}_{\neg a_1 \vee \neg a_2}$, along with additional states q_1, q_2.
- $\Sigma_{\mathcal{B}} := \{a_1, a_2, c\}$
- $Q_{I_{\mathcal{B}}} := \{q_1\}$
- $\delta_{\mathcal{B}}$ will consists of the transitions of $\mathcal{A}_{a_1}$, $\mathcal{A}_c$ and $\mathcal{A}_{\neg a_1 \vee \neg a_2}$, along with additional transitions:
  - $(q_1, c, q_1, p) \in \delta_{\mathcal{B}}$ for p an initial state in $\mathcal{A}_c$
  - $(q_1, c, q_2, p) \in \delta_{\mathcal{B}}$ for p an initial state in $\mathcal{A}_{\neg a_1 \vee \neg a_2}$
  - $(q_2, c, q_2, p) \in \delta_{\mathcal{B}}$ for p an initial state in $\mathcal{A}_c$
  - $(q_2, a_1, p, p) \in \delta_{\mathcal{B}}$ for p an initial state in $\mathcal{A}_{a_1}$
- $\mathbb{C}_{\mathcal{B}}(q_1) := 1$, $\mathbb{C}_{\mathcal{B}}(q_2) := 1$, and for other states, the assigned color would be the same as in the automaton the state has originated from ($\mathcal{A}_{a_1}$, $\mathcal{A}_c$ or $\mathcal{A}_{\neg a_1 \vee \neg a_2}$)

It is easy to see that $L(\mathcal{B}) = L^{fa}$.

Let $t \in L(\mathcal{B})$. By definition of L^{fa}, there is $m \in \mathbb{N}$ such that $t \in L_m$. If ϕ is an accepting computation on t, then ϕ assigns to the first $m+2$ nodes on the leftmost branch the sequence $\underbrace{q_1, \dots, q_1}_{i \text{ times}} \cdot \underbrace{q_2, \dots, q_2}_{m-i+1 \text{ times}} \cdot q_{a_1}$ for some $i \in \{1, \dots, m\}$, where q_{a_1} is the initial state of $\mathcal{A}_{a_1}$ (total m possibilities). ϕ assigns to $l^j \cdot r$ the initial state of $\mathcal{A}_c$ if $j < i-1$ or $i-1 < j < m$; the initial state of $\mathcal{A}_{\neg a_1 \vee \neg a_2}$ if $j = i-1$; and the initial state of $\mathcal{A}_{a_1}$ if $j \geq m$. Since $\mathcal{A}_c$ and $\mathcal{A}_{a_1}$ are deterministic and $\mathcal{A}_{\neg a_1 \vee \neg a_2}$ is 2-ambiguous, the number of accepting computations on t is at most $2m$, hence, finite. □

Lemma 6.4. *Let L be a tree language such that $L_m \subseteq L \subseteq L^{fa}$. Then, L is not $m-1$ ambiguous.*

Proof. Let $\mathcal{A}$ be an automaton with states Q which accepts L, and assume $\mathcal{A}$ is finitely ambiguous. Define a set $Q' \subseteq Q$ by $Q' := \{\phi(l^i r) \mid i < m \wedge \exists t \in L : \phi \in ACC(\mathcal{A}, t)\}$ and $Q_{\exists a_1} := \{q \in Q' \mid L_{\exists a_1} \cap L(\mathcal{A}_q) \neq \emptyset\}$, and let $Q_{t_c \wedge \exists a_1} := \{q \in Q_{\exists a_1} \mid t_c \in L(\mathcal{A}_q)\}$ and $Q_{\neg t_c \wedge \exists a_1} := Q_{\exists a_1} \setminus Q_{t_c \wedge \exists a_1}$.

Relying on the fact that $T^{\omega}_{\{c,a_1\}} \setminus \{t_c\}$ is not finitely ambiguous (by Corollary 4.4 (2)), we derive the following claim:

Claim 6.4.1. *There is a tree $t_{\exists a_1} \in \left(T^{\omega}_{\{c,a_1\}} \setminus \{t_c\}\right) \cap \left(L(\mathcal{A}_{Q_{t_c \wedge \exists a_1}}) \setminus L(\mathcal{A}_{Q_{\neg t_c \wedge \exists a_1}})\right)$.* ■

Recall that t^m is the tree which is obtained from t_c by grafting t_{a_1} on node l^m. For each $i < m$, define t^m_i as the tree which is obtained from t^m by grafting $t_{\exists a_1}$ on node $l^i r$. It is clear that $t^m_i \in L(\mathcal{A})$, and therefore there is an accepting computation ϕ_i of $\mathcal{A}$ on t^m_i.

$t_{\exists a_1} \in L(\mathcal{A}_{Q_{t_c \wedge \exists a_1}}) \setminus L(\mathcal{A}_{Q_{\neg t_c \wedge \exists a_1}})$ and since $t_{\exists a_1} \in \mathcal{A}_{\phi_i(l^i r)}$ we conclude that $\phi_i(l^i r) \in Q_{t_c \wedge \exists a_1}$ and therefore $t_c \in L(\mathcal{A}_{\phi_i(l^i r)})$. Let $\phi^c_i \in ACC(\mathcal{A}_{\phi_i(l^i r)}, t_c)$, and construct a computation ϕ'_i from ϕ_i by grafting ϕ^c_i on $l^i r$. This tree which is obtained from t^m_i by grafting t_c on $l^i r$ is the tree t^m and therefore $\phi'_i \in ACC(\mathcal{A}, t^m)$.

We are going to show that for all $i < j < m$, the computations $\phi'_i, \phi'_j \in ACC(\mathcal{A}, t^m)$ are different. Assume towards a contradiction $\phi'_i = \phi'_j$ and let $\widehat{\phi} := \phi'_i$. Define $p_i := \widehat{\phi}(l^i r)$, $p_j := \widehat{\phi}(l^j r)$, and let $\phi_{p_i} \in ACC(\mathcal{A}_{p_i}, t_{\exists a_1})$ and $\phi_{p_j} \in ACC(\mathcal{A}_{p_2}, t_{\exists a_1})$. Construct t' from t^m by grafting $t_{\exists a_1}$ on nodes $l^i r$ and $l^j r$, and construct ϕ' from $\widehat{\phi}$ by grafting ϕ_{p_i} on $l^i r$ and ϕ_{p_2} on $l^j r$. It follows that ϕ' is an accepting computation of $\mathcal{A}$ on t', which is a contradiction, since $t' \notin L^{fa}$ (since $t'_{\geq l^j r} = t'_{\geq l^i r} = t_{\exists a_1} \neq t_c$) and therefore $t' \notin L$ (since $L \subseteq L^{fa}$). We conclude that there are at least m different accepting computations of $\mathcal{A}$ on t^m. □

Remark 6.5. The language L_m is $2m$ ambiguous but not $m-1$ ambiguous. This implies that the hierarchy of ambiguous languages is infinite. The point of the more complex construction in Sect. 5 is to show that this hierarchy is populated at every level.

Lemma 6.6. *L^{fa} is not boundedly ambiguous*

Proof. $\forall m \in \mathbb{N} : L_m \subseteq L^{fa}$, and therefore from Lemma 6.4 it follows that L^{fa} is not $(m-1)$-ambiguous. That is, L^{fa} is not boundedly ambiguous. □

7. Uncountably Ambiguous Languages

In this section we introduce a scheme for obtaining uncountably ambiguous languages from languages which are not boundedly ambiguous. We then use this scheme to obtain natural examples of tree languages which are uncountably ambiguous.

Definition 7.1. Let $L^{\neg ba}$ be an arbitrary regular tree language over alphabet Σ which is not boundedly ambiguous, and let L_0 be an arbitrary regular tree language over alphabet Σ such that $L_0 \cap L^{\neg ba} = \emptyset$. Let $c \in \Sigma$ and define a language $\mathfrak{L}[L_0, L^{\neg ba}]$ over alphabet Σ: $t \in \mathfrak{L}[L_0, L^{\neg ba}]$ iff the following conditions hold:

- $\forall v \in l^* : t(v) = c$
- There is an infinite set $I \subseteq \mathbb{N}$ such that $\forall i \in I : t_{\geq l^i \cdot r} \in L^{\neg ba}$ and $\forall i \notin I : t_{\geq l^i \cdot r} \in L_0$.

Proposition 7.2. *The degree of ambiguity of $\mathfrak{L}[L_0, L^{\neg ba}]$ is $2^{\aleph_0}$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ be a PTA which accepts $\mathfrak{L}[L_0, L^{\neg ba}]$. We will show that $da(\mathcal{A}) = 2^{\aleph_0}$.

Let $Q' := \{\phi(u) \mid u \in l^* \cdot r \text{ and } \exists t : \phi \in ACC(\mathcal{A}, t)\}$, and define $Q_{unamb \wedge \neg L_0} := \{q \in Q' \mid \mathcal{A}_q \text{ is unambiguous and } L(\mathcal{A}_q) \cap L_0 = \emptyset\}$.

Claim 7.2.1. $L(\mathcal{A}_{Q_{unamb\wedge\neg L_0}}) \subseteq L^{\neg ba}$.

Proof. Assume, for the sake of contradiction, that there is a tree $t \in L(\mathcal{A}_{Q_{unamb\wedge\neg L_0}})$ such that $t \notin L^{\neg ba}$. By definition of $Q_{unamb\wedge\neg L_0}$ we conclude that $t \notin L_0$.

Let $q \in Q_{unamb\wedge\neg L_0}$ such that $t \in L(\mathcal{A}_q)$ and let $\phi \in ACC(\mathcal{A}_q, t)$. Recall that $q \in Q'$ (since $Q_{unamb\wedge\neg L_0} \subseteq Q'$) and therefore there is a tree $t' \in L(\mathcal{A})$, a computation $\phi' \in ACC(\mathcal{A}, t)$ and a node $u \in l^* \cdot r$ such that $\phi'(u) = q$. By the grafting lemma we conclude that $\phi' \circ_u \phi$ is an accepting computation of $\mathcal{A}$ on $t' \circ_u t$. Therefore, $t' \circ_u t \in L(\mathcal{A})$ for $t \notin L^{\neg ba} \cup L_0$ - a contradiction to definition of $\mathcal{A}$. ■

Notice that $L(\mathcal{A}_{Q_{unamb\wedge\neg L_0}})$ is boundedly ambiguous by Corollary 3.2 (as a finite union of unambiguous languages), and since $L^{\neg ba}$ is not boundedly ambiguous we conclude that $da(L(\mathcal{A}_{Q_{unamb\wedge\neg L_0}})) \neq da(L^{\neg ba})$. By Claim 7.2.1 we obtain $L(\mathcal{A}_{Q_{unamb\wedge\neg L_0}}) \subseteq L^{\neg ba}$, and applying Lemma 3.6 we conclude that there is a tree $t_{\neg ba} \in L^{\neg ba}$ such that $t_{\neg ba} \notin L(\mathcal{A}_{Q_{unamb\wedge\neg L_0}})$.

Let $c \in \Sigma$ be as in the definition of $\mathfrak{L}[L_0, L^{\neg ba}]$, and let t_c be a tree where all nodes are labeled by c. Let $A := l^* \cdot r$ be an antichain, and define $t'' := t_c \circ_A t_{\neg ba}$. By the definition of $\mathcal{A}$ it is clear that $t'' \in L(\mathcal{A})$. Let $\phi'' \in ACC(\mathcal{A}, t'')$, and let $B := \{u \in A \mid L(\mathcal{A}_{\phi''(u)}) \cap L_0 \neq \emptyset\}$.

For each $u \in B$ there is a tree $t_u \in L_0$ and a computation $\phi_u \in ACC(\mathcal{A}_{\phi''(u)}, t_u)$. Therefore, by the grafting lemma, we conclude that the tree t''' which is obtained from t'' by grafting t_u on each node $u \in B$ is in $L(\mathcal{A})$.

Assume, for the sake of contradiction, that $A \setminus B$ is finite. By definition of t''', for each $i \in \mathbb{N}$ such that $u := l^i \cdot r \in B$ we have $t'''_{\geq l^i \cdot r} = t_u \in L_0$. Therefore, $|\{i \in \mathbb{N} \mid t'''_{\geq l^i \cdot r} \in L^{\neg ba}\}| = |\{u \in A \mid t'''_{\geq u} \in L^{\neg ba}\}| = |\{u \in A \setminus B \mid t'''_{\geq u} \in L^{\neg ba}\}| = |A \setminus B| < \aleph_0$, and by definition of $\mathfrak{L}[L_0, L^{\neg ba}]$ we conclude that $t''' \notin \mathfrak{L}[L_0, L^{\neg ba}]$ - a contradiction to the definition of $\mathcal{A}$.

$A \setminus B$ is infinite, and therefore there is a state q and an infinite set $\widehat{A} \subseteq A \setminus B$ such that $\phi''(u) = q$ for all $u \in \widehat{A}$. Recall that $\forall u \in \widehat{A} : t''_{\geq u} = t_{\neg ba}$. Notice that for each $u \in \widehat{A}$ we have $u \notin B$, and by definition of B we obtain $L(\mathcal{A}_{\phi''(u)}) \cap L_0 = L(\mathcal{A}_q) \cap L_0 = \emptyset$. Since $t_{\neg ba} \notin L(\mathcal{A}_{Q_{unamb \wedge \neg L_0}})$ we conclude that $q \notin Q_{unamb \wedge \neg L_0}$ - hence, $\mathcal{A}_q$ is ambiguous.

Let $t_{amb} \in L(\mathcal{A}_q)$ be a tree with at least two accepting computations $\phi_1, \phi_2 \in ACC(\mathcal{A}_q, t_{amb})$. Let $\widehat{t} := t'' \circ_{\widehat{A}} t_{amb}$, and $\widehat{\phi} := \phi \circ_{\widehat{A}} \phi_1$. By the grafting lemma we obtain $\widehat{\phi} \in ACC(\mathcal{A}, \widehat{t})$. For each $A' \subseteq \widehat{A}$, define a computation $\phi_{A'} := \widehat{\phi} \circ_{A'} \phi_2$. Notice that $\phi_{A'} \in ACC(\mathcal{A}, \widehat{t})$ (by the grafting lemma) and that $\forall A_1, A_2 \subseteq \widehat{A} : A_1 \neq A_2 \to \phi_{A_1} \neq \phi_{A_2}$ (since $\phi_1 \neq \phi_2$). Therefore, $|ACC(\mathcal{A}, \widehat{t})| \geq |\{A' \mid A' \subseteq \widehat{A}\}| = 2^{\aleph_0}$, and $da(\mathcal{A}) = 2^{\aleph_0}$, as requested. □

We will now introduce a couple of definitions, and present three natural examples of infinite tree languages which are not countable ambiguous.

Definition 7.3 (characteristic tree)**.** The characteristic tree of $U_1, \ldots, U_n \subseteq \{l, r\}^*$ is a $\{0,1\}^n$-labeled tree $t[U_1, \ldots, U_n]$ such that $t[U_1, \ldots, U_n](u) := (b_1, \ldots, b_n)$ where $b_i = 1$ iff $u \in U_i$ for each $1 \leq i \leq n$.

Definition 7.4. For a set $U \subseteq \{l, r\}^*$ we define $U\downarrow$ as the downward closure of U.

Definition 7.5. A set $X \subseteq \{l, r\}^*$ is called **perfect** if $X \neq \emptyset$ and $\forall u \in X : \exists v_1, v_2 \in X$ such that $v_1, v_2 > u$ and $v_1 \perp v_2$.

Proposition 7.6. *The following regular languages are not countably ambiguous:*

(1) $L_{X \subseteq Y\downarrow} := \{t[X, Y] \mid X \subseteq Y\downarrow\}$ - *"for each node in X there is a greater or equal node in Y."*
(2) $L_{no-max} := \{t[X] \mid X$ *has no maximal element*$\}$ - *"for each node in X there is a greater node in X."*
(3) $L_{perf} := \{t[X] \mid X$ *is perfect* $\}$ - *"for each node in X there are at least two greater incomparable nodes in X."*

In the rest of this section we will prove Proposition 7.6.

Proof of Proposition 7.6(1). Let $L_{left} := \{t[X, Y] \mid X = l^*$ and $Y \cap l^* = \emptyset\}$. It is easy to see that L_{left} can be accepted by a deterministic PTA, and therefore $da(L_{left}) = 1$.

By Lemma 3.1 we conclude that $da(L_{X \subseteq Y\downarrow} \cap L_{left}) \leq da(L_{X \subseteq Y\downarrow}) \cdot da(L_{left}) = da(L_{X \subseteq Y\downarrow})$. We will show that $L_{X \subseteq Y\downarrow} \cap L_{left}$ is not countably ambiguous. By the above inequality, this implies that $L_{X \subseteq Y\downarrow}$ is not countably ambiguous.

Claim 7.6.1. *Let $L_{X=\emptyset, Y \neq \emptyset} := \{t[X, Y] \mid X = \emptyset$ and $Y \neq \emptyset\}$. Then $t' \in L_{X \subseteq Y\downarrow} \cap L_{left}$ iff the following conditions hold:*

(1) $\forall u \in l^* : t'(u) = (1, 0)$
(2) *There is an infinite set $I \subseteq \mathbb{N}$ such that:*
    (a) *If $i \in I$ then $t'_{\geq l^i \cdot r} \in L_{X=\emptyset, Y \neq \emptyset}$*
    (b) *If $i \notin I$ then $t'_{\geq l^i \cdot r} \in \{t[\emptyset, \emptyset]\}$*

Proof. $\Rightarrow$: Let $t' \in L_{X\subseteq Y\downarrow} \cap L_{left}$. By definition of L_{left} it is clear that the condition (1) holds, and that for each $i \in \mathbb{N} : t'_{\geq v^i \cdot l} \in L_{X=\emptyset,Y\neq\emptyset}$ or $t'_{\geq v^i \cdot l} = t[\emptyset, \emptyset]$. Assume, for the sake of contradiction, that the set $\{i \in \mathbb{N} \mid t'_{\geq v^i \cdot l} \in L_{X=\emptyset,Y\neq\emptyset}\}$ is finite. Therefore, by the second condition, there is an index $k \in \mathbb{N}$ such that $\forall i \geq k : t'_{\geq v^i \cdot l} = t[\emptyset, \emptyset]$. Let $u := l^k$. By the definition of L_{left} we have $u \in X$, and for each $v \geq u$ we have either $t'(v) = (1, 0)$ if $v \in l^*$, or $t'(v) = (0, 0)$ otherwise. Hence, $\forall v \geq u : v \notin Y$, in contradiction to $t' \in L_{X\subseteq Y\downarrow}$.

$\Leftarrow$: Assume that the conditions hold for t'. It is easy to see that $t' \in L_{left}$. We will show that $t' \in L_{X\subseteq Y\downarrow}$. Assume, for the sake of contradiction, that there is a node $u \in X$ such that $v \notin Y$ for each node $v \geq u$. Since all nodes in X are in l^* we conclude that there is $i \in \mathbb{N}$ such that $u = l^i$. Notice that the set $I \subseteq \mathbb{N}$ is infinite, and therefore there is $j > i$ such that $t'_{\geq l^j \cdot r} \in L_{X=\emptyset,Y\neq\emptyset}$. Therefore, there is a node $v \geq l^j \cdot r > l^i = u$ such that $v \in Y$ - a contradiction. ■

Observe that the language $L_{X=\emptyset,Y\neq\emptyset} := \{t[X, Y] \mid X = \emptyset \text{ and } Y \neq \emptyset\}$ can be considered as a tree language over alphabet $\{0\} \times \{0, 1\}$, and that $L_{X=\emptyset,Y\neq\emptyset} = T^\omega_{\{0\}\times\{0,1\}} \setminus \{t[\emptyset, \emptyset]\}$. Therefore, by Corollary 4.4(2) we conclude that $L_{X=\emptyset,Y\neq\emptyset}$ is not finitely ambiguous.

Notice that by Claim 7.6.1 we obtain $L_{X\subseteq Y\downarrow} \cap L_{left} = \mathfrak{L}[L_0, L^{\neg ba}]$, for $L_0 = \{t[\emptyset, \emptyset]\}$ and $L^{\neg ba} = L_{X=\emptyset,Y\neq\emptyset}$. Therefore, applying Proposition 7.2 we conclude that $L_{X\subseteq Y\downarrow} \cap L_{left}$ is not countably ambiguous. □

To prove Proposition 7.6(2), we will first prove the following lemma:

Lemma 7.7. *L_{no-max} is not finitely ambiguous.*

Proof. Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ be a PTA which accepts L_{no-max}. Let $Q' := \{q \in Q \mid \exists q_i \in Q_I \exists q' \in Q : (q_i, 1, q, q') \in \delta \text{ and } t[\emptyset] \in L(\mathcal{A}_{q'})\}$.

Claim 7.7.1. *Define $L_{\neg\emptyset} := T^\omega_\Sigma \setminus \{t[\emptyset]\}$. Then:*
(1) $L_{no-max} \setminus \{t[\emptyset]\} \subseteq L(\mathcal{A}_{Q'})$
(2) $L(\mathcal{A}_{Q'}) \subseteq L_{\neg\emptyset}$

Proof. (1) Let $t' \in L_{no-max} \setminus \{t[\emptyset]\}$, and let $t_\epsilon := t[\{\epsilon\}]$ (that is, $t_\epsilon(\epsilon) := 1$, and $\forall u \neq \epsilon : t_\epsilon(u) := 0$). Let $t'' := t_\epsilon \circ_l t' \circ_r t[\emptyset]$. By the definition of L_{no-max} we obtain $t'' \in L_{no-max}$. Therefore, there is a computation $\phi \in ACC(\mathcal{A}, t'')$ such that $\phi(l) \in Q'$ and $t' \in L(\mathcal{A}_{\phi(l)})$, as requested.

(2) Assume, for the sake of contradiction, that $t[\emptyset] \in L(\mathcal{A}_{Q'})$. Then there is a transition $(q_i, 1, q_1, q_2) \in \delta$ from an initial state q_i such that $t[\emptyset] \in L(\mathcal{A}_{q_1})$ and $t[\emptyset] \in L(\mathcal{A}_{q_2})$. Therefore, we conclude that $t_\epsilon := t[\{\epsilon\}]$ is accepted by $\mathcal{A}$ - a contradiction to the definition of L_{no-max}. ■

Let $\Sigma := \{0, 1\}$. Define a function $F : \Sigma^* \to \Sigma$ such that

$$F(\sigma_1, \ldots, \sigma_m) := \begin{cases} 1 & \exists 1 \leq i \leq m : \sigma_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

It is easy to see that F is definable by a Moore machine. We show that F reduces $L_{\neg\emptyset}$ to $L(\mathcal{A}_{Q'})$.

Notice that $\forall t' \in T^\omega_\Sigma : t' \in L_{\neg\emptyset} \to \widehat{F}(t') \in L_{no-max} \setminus \{t[\emptyset]\}$. Since $L_{no-max} \setminus \{t[\emptyset]\} \subseteq L(\mathcal{A}_{Q'})$ (by Claim 7.7.1(1)) we conclude that $\forall t' \in T^\omega_\Sigma : t' \in L_{\neg\emptyset} \to \widehat{F}(t') \in L(\mathcal{A}_{Q'})$.

Conversely, $\forall t' \in T^{\omega}_{\Sigma} : \widehat{F}(t') \in L_{\neg\emptyset} \to t' \in L_{\neg\emptyset}$, and since $L(\mathcal{A}_{Q'}) \subseteq L_{\neg\emptyset}$ (by Claim 7.7.1(2)) we obtain $\forall t' \in T^{\omega}_{\Sigma} : \widehat{F}(t') \in L(\mathcal{A}_{Q'}) \to t' \in L_{\neg\emptyset}$.

Therefore, by Lemma 3.10, we conclude that $da(L(\mathcal{A}_{Q'})) \geq da(L_{\neg\emptyset})$. Notice that $L_{\neg\emptyset} = T^{\omega}_{\Sigma} \setminus \{t[\emptyset]\}$ and by Corollary 4.4(2) we obtain $da(L_{\neg\emptyset}) \geq \aleph_0$. Hence, $\mathcal{A}_{Q'}$ is not finitely ambiguous, and by Corollary 3.5 we conclude that $da(\mathcal{A}) \geq \aleph_0$. □

Proof of Proposition 7.6(2). Let $L_{l^* \cap X = \emptyset} := \{t[X] \mid X \cap l^* = \emptyset\}$. It is easy to construct a deterministic PTA which accepts $L_{l^* \cap X = \emptyset}$, and therefore $da(L_{l^* \cap X = \emptyset}) = 1$.

By Lemma 3.1 we conclude that $da(L_{no-max} \cap L_{l^* \cap X = \emptyset}) \leq da(L_{no-max}) \cdot da(L_{l^* \cap X = \emptyset}) = da(L_{no-max})$. We will show that $da(L_{no-max} \cap L_{l^* \cap X = \emptyset}) = 2^{\aleph_0}$, and the lemma will follow.

Notice that $t' \in L_{no-max} \cap L_{l^* \cap X = \emptyset}$ iff the following hold:

- $\forall u \in l^* : t(u) = 0$
- $\forall u \in l^* \cdot r : t'_{\geq u} \in L_{no-max}$

It is easy to see that $L_{no-max} \cap L_{l^* \cap X = \emptyset} = \mathfrak{L}[L_0, L^{\neg ba}]$ for $L^{\neg ba} := L_{no-max}$ (which is not boundedly ambiguous, by Lemma 7.7) and $L_0 := \emptyset$. Therefore, by Proposition 7.2 we conclude that $da(L_{no-max} \cap L_{l^* \cap X = \emptyset}) = 2^{\aleph_0}$, as requested. □

Proof of Proposition 7.6(3). Let $L_{contains-l^*} := \{t[X] \mid l^* \subseteq X\}$. It is easy to see that $L_{contains-l^*}$ can be accepted by a deterministic PTA, and therefore $da(L_{contains-l^*}) = 1$.

Look at the language $L_{perf} \cap L_{contains-l^*}$. By Lemma 3.1 we obtain $da(L_{perf} \cap L_{contains-l^*}) \leq da(L_{perf}) \cdot da(L_{contains-l^*}) = da(L_{perf})$. We will show that $L_{perf} \cap L_{contains-l^*}$ is not countably ambiguous. By the above inequality, this implies that $da(L_{perf}) = 2^{\aleph_0}$.

Claim 7.7.2. *L_{perf} is not finitely ambiguous.*

Proof. Define a function $F : \Sigma^* \to \Sigma$ such that $F(\sigma_1, \ldots, \sigma_m) := \begin{cases} 1 & \exists 1 \leq i \leq m : \sigma_i = 1 \\ 0 & \text{otherwise.} \end{cases}$.
It is easy to see that F is definable by a Moore machine, and that $\forall t' \in T^{\omega}_{\Sigma} : t' \in T^{\omega}_{\Sigma} \setminus \{t[\emptyset]\} \leftrightarrow \widehat{F}(t) \in L_{perf}$. Notice that $T^{\omega}_{\Sigma} \setminus \{t[\emptyset]\}$ is not finitely ambiguous (by Corollary 4.4(2)), and therefore by Lemma 3.10 we conclude that L_{perf} is not finitely ambiguous. ■

Claim 7.7.3. *$t' \in L_{perf} \cap L_{contains-l^*}$ iff the following conditions hold:*
(1) *$\forall u \in l^* : t'(u) = 1$*
(2) *There is an infinite set $I \subseteq \mathbb{N}$ such that $\forall i \in I : t'_{\geq l^i \cdot r} \in L_{perf}$ and $\forall i \notin I : t'_{\geq l^i \cdot r} \in \{t[\emptyset]\}$.*

Proof. $\Rightarrow$: Let $t' \in L_{perf} \cap L_{contains-l^*}$. By definition of $L_{contains-l^*}$ it is clear that condition (1) holds for t'. Notice that $\forall i \in \mathbb{N} : t'_{\geq l^i \cdot r} \in L_{perf}$ or $t'_{\geq l^i \cdot r} = t[\emptyset]$. Assume, for the sake of contradiction, that $\{i \in \mathbb{N} \mid t'_{\geq l^i \cdot r} \in L_{perf}\}$ is finite. Therefore, there is $k \in \mathbb{N}$ such that $\forall i \geq k : t'_{\geq l^i \cdot r} = t[\emptyset]$. Let $u := l^k$, and notice that $t'(u) = 1$, and $\forall v > u : t'(v) = 1 \leftrightarrow v \in l^*$. Hence, each pair of 1-labeled nodes which are greater than u are comparable - a contradiction to the definition of L_{perf}.

$\Leftarrow$: Let t' such that the conditions hold. By the first condition it is clear that $t' \in L_{contains-l^*}$. We will prove that $t' \in L_{perf}$, and the claim will follow. First, notice that $t'(\epsilon) = 1$, and therefore $t' \neq t[\emptyset]$. Let u be a node such that $t'(u) = 1$. If $u \in l^*$ then by the second condition, there is a node $v \in l^* \cdot r$ such that $v > u$ and $t_{\geq v} \in L_{perf}$. Therefore, there are two nodes $w_1, w_2 > v > u$ such that $w_1 \perp w_2$ and $t'(w_1) = t'(w_2) = 1$. Otherwise ($u \notin l^*$), there is a node $v \in l^* \cdot r$, such that $u > v$ and $t_{\geq v} \in L_{perf}$, and by definition of L_{perf}

we conclude that there are two nodes $w_1, w_2 > u$ such that $w_1 \perp w_2$ and $t'(w_1) = t'(w_2) = 1$ - hence, $t' \in L_{perf}$. ■

It is easy to see that $L_{perf} \cap L_{contains-l^*} = \mathfrak{L}[L_0, L^{\neg ba}]$ for $L^{\neg ba} := L_{perf}$ (which is not boundedly ambiguous, by Claim 7.7.2) and $L_0 := \{t[\emptyset]\}$. Therefore, by Proposition 7.2 we conclude that $L_{perf} \cap L_{contains-l^*} = 2^{\aleph_0}$, as requested. □

Observe that our proof shows that $L_{perf \wedge \min} := \{t[X] \mid X$ is perfect and has the $\leq$-minimal element$\}$ is also uncountable ambiguous. We conclude with an instructive example of an unambiguous language which is similar to $L_{perf \wedge \min}$. Let $X \subseteq \{l, r\}^*$ be a set of nodes. We say that $u \in X$ is a X-successor of v if $u > v$ and there is no node $w \in X$ such that $v < w < u$. We call X a full-binary subset-tree if X has a minimal node, and each node in X has two X-successors.

Note that if X is a full-binary subset tree then X is perfect and has the $\leq$-minimal element. However the language $L_{binary} := \{t[X] \mid X$ is a full-binary subset tree$\}$ is unambiguous.

8. Countable Languages are Unambiguous

In this section we prove the following Proposition:

Proposition 8.1. *Each regular countable tree language is unambiguous*

To prove Proposition 8.1 we first recall finite tree automata (Subsec. 8.1). Then, we present Niwiński's Representation for Countable Languages (Subsec. 8.2). Finally, the proof of Proposition 8.1 is given (Subsec. 8.3).

8.1. **Finite Trees and Finite Tree Automata.**

Finite Trees. A finite tree is a finite set $U \subseteq \{l, r\}^*$ which is closed under prefix relation. U is called a finite **binary** tree if $\forall u \in U : u \cdot l \in U \leftrightarrow u \cdot r \in U$.

Finite Σ-labeled Binary Trees. Let Σ be partitioned into two sets: Σ_2 - labels of internal nodes, and Σ_0 - labels of leaves. A finite Σ-labeled binary tree is a function $t_U : U \to \Sigma$, where $U \subseteq \{l, r\}^*$ is a finite binary tree, $t_U(v) \in \Sigma_0$ if v is a leaf, and $t_U(v) \in \Sigma_2$ if v has children.

When it is clear from the context, we will use "finite tree" or "labeled finite tree" for "Σ-labeled finite binary tree".

Finite Tree Automata (FTA). An automaton over Σ-labeled finite trees is a tuple $\mathcal{B} = (Q, \Sigma, Q_I, \delta)$, where Q is a finite set of states, $\Sigma = \Sigma_0 \cup \Sigma_2$ is an alphabet, Q_I is a set of initial states, and $\delta \subseteq (Q \times \Sigma_0) \cup (Q \times \Sigma_2 \times Q \times Q)$ is a set of transitions.

An accepting computation of $\mathcal{B}$ on a finite tree t_U is a function $\phi : U \to Q$, such that $\phi(\epsilon) \in Q_I$, and for each node $u \in U$, if u is not a leaf then $(\phi(u), t_U(u), \phi(u \cdot l), \phi(u \cdot r)) \in \delta$, and otherwise $(\phi(u), t_U(u)) \in \delta$.

The language of a FTA $\mathcal{B}$ is the set of finite trees t such that $\mathcal{B}$ has an accepting computation on t. A finite tree language is regular iff it is accepted by a FTA. It is well-known that every regular finite tree language is unambiguous (i.e., for every finite tree language, there is an unambiguous automaton which accepts it).

8.2. **Niwiński's Representation for Countable Languages.**

Definition 8.2. Define $T^{fin}_{\Sigma(\{x_1,\dots,x_n\})}$ as the set of finite trees over alphabet $\Sigma \cup \{x_1, \dots, x_n\}$ where the internal nodes are Σ-labeled, and the leaves are $\{x_1, \dots, x_n\}$-labeled.

Let $\tau \in T^{fin}_{\Sigma(\{x_1,\dots,x_n\})}$ be a finite tree, and let $t_1, \dots, t_n \in T^{\omega}_{\Sigma}$ be infinite binary trees over alphabet Σ. We define $\tau[t_1/x_1, \dots, t_n/x_n]$ as the infinite tree which is obtained from τ by grafting t_i on leaves labeled by x_i.

For a set $M \subseteq T^{fin}_{\Sigma(\{x_1,\dots,x_n\})}$, we define $M[t_1/x_1, \dots, t_n/x_n] := \bigcup_{\tau \in M} \tau[t_1/x_1, \dots, t_n/x_n]$.

Theorem 8.3 (D. Niwiński [15])**.** *Let L be a countable regular tree language over alphabet Σ. Then there is a finite set of trees $\{t_1, \dots, t_n\}$ such that the following hold:*

(1) *For each tree $t \in L$ and a tree branch π, there is a node $v \in \pi$ and a number $1 \le i \le n$ such that $t_{\ge v} = t_i$.*
(2) *There is a regular finite tree language $M \subseteq T^{fin}_{\Sigma(\{x_1,\dots,x_n\})}$ such that $L = M[t_1/x_1, \dots, t_n/x_n]$.*

The following lemma strengthen item (2) of Theorem 8.3 by adding another condition on M, implying a unique representation of each tree in L:

Lemma 8.4. *Let L be a countable regular tree language over alphabet Σ, and let $\{t_1, \dots, t_n\}$ be a finite set of trees as in Theorem 8.3. Then there is a regular finite trees language $M \subseteq T^{fin}_{\Sigma(\{x_1,\dots,x_n\})}$ such that $L = M[t_1/x_1, \dots, t_n/x_n]$, and for each $t \in L$ there is a* ***unique*** *finite tree $\tau \in M$ such that $t = \tau[t_1/x_1, \dots, t_n/x_n]$.*

Proof. For each tree $t \in L$, let $g(t)$ be the tree which is obtained from t by changing the label of each node $v \in \{l, r\}^*$ where $t_{\ge v} = t_i$ to x_i, and removing all descendants of $\{x_1, \dots, x_n\}$-labeled node.

Claim 8.4.1. *$g(t)$ is finite for all $t \in L$.*

Proof. Assume, for the sake of contradiction, that there is $t \in L$ such that the set of nodes $U \subseteq \{l, r\}^*$ of $g(t)$ is infinite. The number of children of each node in U is bounded by 2, and therefore, by König's Lemma, there is a tree branch π such that $\forall v \in \pi : v \in U$. Therefore, by definition of $g(t)$, we conclude that $t_{\ge v} \ne t_i$ for each $v \in \pi$ and $1 \le i \le n$ - a contradiction to item (1) of Theorem 8.3. ■

Notice that for each $t \in L$ we obtain $g(t)[t_1/x_1, \dots, t_n/x_n] = t$, and therefore g is injective. Hence, $L = M[t_1/x_1, \dots, t_n/x_n]$ where $M := \{g(t) \mid t \in L\}$. We will show that M is a regular language of finite trees.

It is easy to see that for each $t \in L$ and finite tree $\tau \in T^{fin}_{\Sigma(\{x_1,\dots,x_n\})}$, $\tau = g(t)$ iff the following conditions hold:

- $t = \tau[t_1/x_1, \dots, t_n/x_n]$
- $t_{\ge v} \ne t_i$ for each node v in τ which is not a leaf, and for each $1 \le i \le n$.

Since both conditions could be formulated in MSO, we conclude that M is MSO-definable, and therefore regular. □

8.3. **Proof of Proposition 8.1.** Let L be a countable regular tree language over alphabet Σ. We will show that L can be accepted by an unambiguous PTA.

By Lemma 8.4, there is a regular finite tree language $M \subseteq T^{fin}_{\Sigma(\{x_1,\dots,x_n\})}$ and regular infinite trees $t_1,\dots,t_n$ such that $L = M[t_1/x_1,\dots,t_n/x_n]$. Additionally, for each $t \in L$ there is a unique $\tau \in M$ such that $t = \tau[t_1/x_1,\dots,t_n/x_n]$.

Each infinite tree $t_i : \{l,r\}^* \to \Sigma$ is regular, and therefore definable by a Moore machine $M_i = (\{l,r\},\Sigma,Q_i,q^i_I,\delta^M_i,out^M_i)$. Let $\mathcal{A}_i := (Q_i,\Sigma,q^i_I,\delta_i,F_i)$ where $F_i := Q_i$, and $(q,a,q_1,q_2) \in \delta_i$ iff $q_1 = \delta(q,l)$, $q_2 = \delta(q,r)$ and $a = out^M_i(q)$. It is easy to verify that $\mathcal{A}_i$ is unambiguous, and $L(\mathcal{A}_i) = \{t_i\}$. M is regular and therefore can be accepted by an unambiguous FTA $\mathcal{B} = (Q_\mathcal{B},\Sigma \cup \{x_1,\dots,x_n\},q^\mathcal{B}_I,\delta_\mathcal{B})$.

We use these automata to construct a PTA $\mathcal{A} := (Q,\Sigma,Q_I,\delta,\mathbb{C})$, by:

- $Q := \cup_{1\le i\le n} Q_i \cup Q_\mathcal{B}$
- $q^i_I := \{q^\mathcal{B}_I\} \cup \{q^i_I \mid (q^\mathcal{B}_I,x_i) \in \delta_\mathcal{B}\}$
- δ is the union of the following:
  - $\{(q,a,q_1,q_2) \in \delta_\mathcal{B} \mid a \in \Sigma\}$ (all transitions of $\mathcal{B}$ on inner nodes)
  - $\cup_{1\le i\le n}\delta_i$
  - $\{(q,a,q^i_I,q^j_I) \mid \exists (q,a,q_1,q_2) \in \delta_\mathcal{B} : (q_1,x_i) \in \delta_\mathcal{B} \text{ and } (q_2,x_j) \in \delta_\mathcal{B}\}$
  - $\{(q,a,q_1,q^j_I) \mid \exists (q,a,q_1,q_2) \in \delta_\mathcal{B} : (q_2,x_j) \in \delta_\mathcal{B}\}$
  - $\{(q,a,q^i_I,q_2) \mid \exists (q,a,q_1,q_2) \in \delta_\mathcal{B} : (q_1,x_i) \in \delta_\mathcal{B}\}$
- $\mathbb{C}(q) := \begin{cases} \mathbb{C}_i(q) & \exists i : q \in Q_i \\ 1 & \text{otherwise} \end{cases}$

It is easy to see that $L(\mathcal{A}) = M[t_1/x_1,\dots,t_n/x_n] = L$.

We will show that $\mathcal{A}$ is unambiguous. For each accepting computation $\phi \in ACC(\mathcal{A},t)$, define a set of nodes $U_\phi := \{u \in \{l,r\}^* \mid \forall v < u : \phi(v) \in Q_\mathcal{B}\}$. It is easy to see that U_ϕ is downward closed. Assume towards contradiction that U_ϕ is infinite - by König Lemma, U_ϕ contains an infinite tree branch π. By definition of U_ϕ all states in $\phi(\pi)$ are in $Q_\mathcal{B}$, and therefore colored by 1. That is a contradiction to ϕ being an accepting computation.

Define a labeled finite tree $t_\phi : U_\phi \to \Sigma \cup \{x_1,\dots,x_n\}$ by:

$$t_\phi := \begin{cases} x_i & \exists i : \phi(u) = q^i_I \\ t(u) & \text{otherwise} \end{cases}$$

By definition of t_ϕ we obtain $t = t_\phi[t_1/x_1,\dots,t_n/x_n]$, and by definition of $\mathcal{B}$ we conclude that $t_\phi \in M$.

Assume, for the sake of contradiction, that $\mathcal{A}$ is ambiguous. Therefore, there is a tree $t \in L$ and two distinct accepting computations $\phi_1,\phi_2 \in ACC(\mathcal{A},t)$. $\mathcal{A}_i$ is deterministic for each $1 \le i \le n$, and therefore $\phi_1 \ne \phi_2$ iff $t_{\phi_1} \ne t_{\phi_2}$. We conclude that $t_{\phi_1}[t_1/x_1,\dots,t_n/x_n] = t_{\phi_2}[t_1/x_1,\dots,t_n/x_n]$ for $t_{\phi_1},t_{\phi_2} \in M$ - a contradiction to the uniqueness property of M.

9. Conclusion and Open Questions

We proved that the ambiguity hierarchy is strict for regular languages over infinite trees. We proved that countable regular languages are unambiguous.

A natural question is whether the ambiguity degree is decidable. However, this is not a trivial matter. In [3] some partial solutions for variants of the problem whether a given language is unambiguous are provided.

A less ambitious task is to develop techniques for computing degrees of ambiguity and compute the degree of ambiguity of some natural languages. Let $\Sigma_1 := \{c, a_1\}$ and $L_{\exists^\infty a_1} := \{t \in T^\omega_{\Sigma_1} \mid$ there are infinitely many a_1-labeled nodes in $t\}$. $L_{\exists^\omega a_1} := \{t \in T^\omega_{\Sigma_1} \mid$ there is a branch with infinitely many a_1-labeled nodes in $t\}$. $L_{a_1-\infty\text{antichain}} := \{t \in T^\omega_{\Sigma_1} \mid$ the set of a_1-labeled nodes in t contain an infinite antichain$\}$. All these languages are regular. There are (Moore) reductions from $L_{\exists a_1}$ to these languages, hence they are not finitely ambiguous. We believe that their ambiguity degree is uncountable, but we were unable to prove this.

We provided sufficient conditions for a language to be not finitely ambiguous and for a language to have uncountable degree of ambiguity.

In particular, we proved that the degree of ambiguity of the complement of a countable regular language is $\aleph_0$ or $2^{\aleph_0}$, and provided natural examples of such languages with countable degree of ambiguity. We proved that the degree of ambiguity of the complement of a finite regular language is $\aleph_0$ Yet, it is open whether the degree of ambiguity of the complement of countable regular languages is $\aleph_0$.

References

[1] Andr Arnold. Rational omega-languages are non-ambiguous. *Theor. Comput. Sci.*, 26:221–223, 09 1983.

[2] 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.

[3] Marcin Bilkowski and Michal Skrzypczak. Unambiguity and uniformization problems on infinite trees. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*, pages 81–100. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013.

[4] Arnaud Carayol and Christof Löding. MSO on the infinite binary tree: Choice and order. In *International Workshop on Computer Science Logic*, pages 161–176. Springer, 2007.

[5] Arnaud Carayol, Christof Löding, Damian Niwinski, and Igor Walukiewicz. Choice functions and well-orderings over the infinite binary tree. *Open Mathematics*, 8(4):662–682, 2010.

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

[7] E Allen Emerson and Charanjit S Jutla. Tree automata, mu-calculus and determinacy. In *FoCS*, volume 91, pages 368–377. Citeseer, 1991.

[8] Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 60–65, 1982.

[9] Yuri Gurevich and Saharon Shelah. Rabin's uniformization problem 1. *The Journal of Symbolic Logic*, 48(4):1105–1119, 1983.

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

[11] 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.

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

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

[14] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and control*, 9(5):521–530, 1966.

[15] Damian Niwiński. On the cardinality of sets of infinite trees recognizable by finite automata. In Andrzej Tarlecki, editor, *Mathematical Foundations of Computer Science 1991*, pages 367–376, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[16] D. Perrin and J.É. Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. ISSN. Elsevier Science, 2004.

[17] Michael O Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the american Mathematical Society*, 141:1–35, 1969.

[18] Alexander Rabinovich and Doron Tiferet. Ambiguity hierarchy of regular infinite tree languages. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 80:1–80:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[19] 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.

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

[21] Boris A. Trakhtenbrot and Ya. M. Barzdin. Finite automata, behavior and synthesis. 1973.

Appendix A. Proof of Claim 4.1.4

Claim 4.1.4. *Let t_0 be a regular tree such that $t_0 \notin L(\mathcal{A})$. Then, Pathfinder has a regular positional winning strategy in $G_{t_0,\mathcal{A}}$.*

Proof. t_0 is regular, and therefore there is a formula $\psi_{t_0}(\sigma)$ which defines t_0 in the unlabeled full-binary tree.

We will use $\psi_{t_0}(\sigma)$ to define the following formula $PathfinderWins_{\mathcal{A},t_0}(\phi, STR)$, as the conjunction of the following conditions:

(1) $\exists \pi$ such that:
   (a) π is a branch
   (b) $\forall u \in \pi : (STR(u, \phi(u \cdot l), \phi(u \cdot r)) = l) \leftrightarrow u \cdot l \in \pi)$ - the Pathfinder moves $d_0 \dots d_j \dots$ are consistent with STR and are along the branch π.
(2) $\exists \sigma : \psi_{t_0}(\sigma)$ and at least one of the following holds:
   (a) $\exists v \in \pi$ such that $(\phi(v), \sigma(v), \phi(v \cdot l), \phi(v \cdot r)) \notin \delta$ - the Automaton move at $(v, \phi(v))$ is invalid.
   (b) The maximal color which $\mathbb{C}$ assigns infinitely often to states in $\phi(\pi)$ is odd.

Claim A.1. $PathfinderWins_{\mathcal{A},t_0}(\phi, STR)$ holds for a positional strategy STR of Pathfinder and a computation ϕ of $\mathcal{A}$ on a tree t' iff the play $\overline{s}$ of STR against str_ϕ in $G_{t_0,\mathcal{A}}$ is winning for Pathfinder.

Proof. By definition of $G_{t_0,\mathcal{A}}$, Pathfinder wins if either Automaton makes an invalid move (condition 2a) or the maximal color which is assigned infinitely often to the positions in $\pi_{\overline{s}}$ is odd. Since all Pathfinder positions have color 0, this is equivalent to the maximal color assigned infinitely often to Automaton positions being odd.

Let $\overline{s} = e_0, d_0, e_1, d_1, \dots, e_i, d_i, \dots$. Notice that by condition 1, there is a unique branch π such that $\pi = v_0, \dots v_i, \dots$ where $v_i = d_0 \dots d_{i-1}$. By Claim 4.1.1, we have $\phi(v_i) = q_i$, where the i-th position of Automaton in $\pi_{\overline{s}}$ is (v_i, q_i). Since $\mathbb{C}_G(v_i, q_i) = \mathbb{C}(q_i)$, we conclude that the maximal color which $\mathbb{C}$ assigns infinitely often to states in $\phi(\pi)$ is odd iff the maximal color which $\mathbb{C}_G$ assigns infinitely often to positions in $\pi_{\overline{s}}$ is odd. This is assured by condition 2b. ■

Let $WinningStrategy_{t_0,\mathcal{A}}(STR) := \forall \phi$ such that the following holds:

- If there is t such that ϕ is an accepting computation of $\mathcal{A}$ on t, then:
  - $PathfinderWins_{\mathcal{A},t_0}(\phi, STR)$ holds

Recalling that the set of all computation of $\mathcal{A}$ is MSO-definable, we conclude that $WinningStrategy_{t_0,\mathcal{A}}(STR)$ is MSO-definable in the unlabeled full-binary tree.

Claim A.2. $WinningStrategy_{t_0,\mathcal{A}}(STR)$ holds for a positional strategy STR of Pathfinder iff STR is a positional winning strategy of Pathfinder.

Proof. $\Rightarrow$: By Claim A.1, STR wins in $G_{t_0,\mathcal{A}}$ against each positional strategy of Automaton. Assume, for the sake of contradiction, that is a non-positional strategy str' of automaton which wins against STR. Then by positional determinacy of parity games, we conclude that there is a positional strategy str'' which wins against STR - a contradiction.

$\Leftarrow$: Follow immediately from Claim A.1. ■

$t_0 \notin L(\mathcal{A})$ and therefore by Claim 4.1.1(3), Automaton does not have a positional winning strategy. From positional determinacy of parity games we conclude that Pathfinder has a positional winning strategy. Therefore, there is a strategy STR' which satisfies $WinningStrategy_{t_0,\mathcal{A}}(STR)$ in the unlabeled full-binary tree.

Therefore, $WinningStrategy_{t_0,\mathcal{A}}(STR)$ defines a non-empty tree language over alphabet $Q \times Q \to \{l, r\}$. By Rabin's basis Theorem, we conclude that there is a regular tree $\widehat{STR}$ in this language, and by Claim A.2 we conclude that $\widehat{STR}$ is a positional winning strategy for Pathfinder in $G_{t_0,\mathcal{A}}$. ■

Remark (Logic Free Proof of Claim 4.1.4). One can reduce a membership game for a regular tree t_0 to a game on a finite graph. By positional determinacy Theorem, Pathfinder will have a positional winning strategy in the reduced game. From this strategy a regular winning strategy in $G_{t_0,\mathcal{A}}$ for Pathfinder is easily constructed.