# Ambiguity Hierarchy of Regular Infinite Tree Languages

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

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

## Abstract

An automaton is unambiguous if for every input it has at most one accepting computation. An automaton is $k$-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 $\omega$-words every regular language is accepted by an unambiguous Büchi automaton [1] and by a deterministic parity automaton. Over infinite trees there are ambiguous languages [5].

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.

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

**Keywords and phrases** automata on infinite trees, ambiguous automata, monadic second-order logic

**Digital Object Identifier** 10.4230/LIPIcs.MFCS.2020.80

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

**Acknowledgements** We would like to thank an anonymous referee for pointing out to [3].

## 1 Introduction

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 it is $k$-ambiguous for some $k \in \mathbb{N}$. An automaton is finitely (respectively, countably) ambiguous if for every input it has at most finitely (respectively, countably) many accepting computations.

For automata over finite words (and over finite trees), on every input there are at most finitely many accepting computations. Hence, every automaton on finite words and on finite trees is finitely ambiguous. However, over $\omega$-words and over infinite trees there are nondeterministic automata with uncountably many accepting computations. Over $\omega$-words

[1] corresponding author

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 $\omega$-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 $\omega$-words are unambiguous.

In [5] it was shown that the aforementioned situation is different for infinite trees. Carayol et al. [5] proved that a language $L_{\exists a}$ of infinite full-binary trees over the alphabet $\{a, c\}$, defined as $L_{\exists a} := \{t \mid t \text{ has at least one node labeled by } a\}$ is ambiguous. The proof is based on the undefinability of a choice function in Monadic Second-Order logic (MSO) [8, 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** (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.*

The paper is organized as follows. 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 here use the fact that a choice function is not MSO-definable and refine the proof techniques of [5]. Sect. 5 deals with $k$-ambiguous languages. For every $k \in \mathbb{N}$, we describe here 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 provides a scheme that generates languages which are not countably ambiguous from non-boundedly ambiguous languages. Conclusion is given in Sect. 8.

## 2 Preliminary

We recall here standard terminology and notations about trees, automata and logic [10, 11].

**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 $\epsilon$ 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.

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

Given a $\Sigma$-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 $t_1$ 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 $t_1$ is obtained by replacing each subtree of $t_1$ rooted at a node $y \in Y$ by the tree $t_2$.

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

**Automata on infinite trees.** We use standard notations and terminology about parity automata on $\Sigma$-labeled full-binary trees, and on $\omega$-strings. A parity automaton $\mathcal{A}$ is a tuple $(Q_\mathcal{A}, \Sigma, Q_I, \delta_\mathcal{A}, \mathbb{C}_\mathcal{A})$ with an alphabet $\Sigma$, a finite set of states $Q_\mathcal{A}$, initial states $Q_I \subseteq Q_\mathcal{A}$, a transition relation $\delta_\mathcal{A}$, and a coloring function $\mathbb{C}_\mathcal{A}$. For a parity automaton on $\omega$-string, $\delta_\mathcal{A} \subseteq Q_\mathcal{A} \times \Sigma \times Q_\mathcal{A}$; for a parity tree automaton, $\delta_\mathcal{A} \subseteq Q_\mathcal{A} \times \Sigma \times Q_\mathcal{A} \times Q_\mathcal{A}$.

Given a parity automaton $\mathcal{A} = (Q_\mathcal{A}, \Sigma, Q_I, \delta_\mathcal{A}, \mathbb{C}_\mathcal{A})$ 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 the notation by $\mathcal{A}_q := \mathcal{A}_{Q'}$.

The notion of a computation/run of a parity automaton $\mathcal{A}$ on a tree/$\omega$-string is defined as usual. We use the letters $\phi, \phi'$ for computations. A computation $\phi$ is accepting if the maximal number which $\mathbb{C}$ assigns infinitely often to the states along every branch of $\phi$ is even. We denote by $ACC(\mathcal{A}, t)$ the set of accepting computations of $\mathcal{A}$ on $t$. We denote by $L(\mathcal{A}) := \{t \mid ACC(\mathcal{A}, t) \text{ is not empty}\}$ the language accepted by $\mathcal{A}$.

A tree language $L$ is called **regular** if it is accepted by a parity tree automaton.

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

**Degree of Ambiguity**. 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 $\mathcal{A}$, denoted by $da(\mathcal{A})$, 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 $i < j \in \mathbb{N}$.

**Language Ambiguity** 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 $da(L) := min_{\mathcal{A}}\{da(\mathcal{A}) \mid L(\mathcal{A}) = L\}$.

**Monadic Second-Order Logic.** We use standard notations and terminology about monadic second-order logic (MSO) [11, 13, 12].

Let $\tau$ be a relational signature. A structure (for $\tau$) 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:

- $R(x_1, \ldots, x_n)$ for an $n$-ary relational symbol $R$ and first order variables $x_1, \ldots, x_n$
- $x = y$ 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 $\psi(X_1, \ldots, X_n, x_1, \ldots, x_m)$ to indicate that the free variables of the formula $\psi$ are $X_1, \ldots, X_n$ (second order variables) and $x_1, \ldots, x_m$ (first order variables).

**Coding** Let $\Delta$ be a finite set. We can code a function from a set $D$ to $\Delta$ 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 $\Delta$ we have second-order variables $X_1^\Delta, X_2^\Delta, \ldots$ which range over the functions from $D$ to $\Delta$, and atomic formulas $X_i^\Delta(u) = d$ for $d \in \Delta$ and $u$ a first order variables [13]. Often the type of the second order variables will be clear from the context and we drop the superscript $\Delta$.

**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 $\psi(X_1, \ldots, X_n, x_1, \ldots, x_m)$ such that $R = \{(D_1, \ldots, D_n, u_1, \ldots, u_m) \in \mathcal{P}(D)^n \times D^m \mid S \models \psi(D_1 \ldots, D_n, u_1 \ldots, u_n)\}$.

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 $\{(v, v \cdot r) \mid v \in \{l, r\}^*)\}$.

▶ **Lemma 2.** *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 parity automaton. We use $\phi$ for a function $\{l, r\}^* \to Q$ and $\sigma$ for a function $\{l, r\}^* \to \Sigma$.*
  - *"$\phi$ is a computation of $\mathcal{A}$ on the tree $\sigma$."*
  - *"$\phi$ is an **accepting** computation of $\mathcal{A}$ on the tree $\sigma$."*

▶ **Theorem 3** (Rabin [11])**.** *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 [11]. Hence, for every $\Sigma$-labeled regular tree $t_0$, there is an MSO-formula $\psi_{t_0}(\sigma^\Sigma)$ which is satisfied by $t$ iff $t = t_0$.

▶ **Theorem 4** (Rabin's basis theorem [11]). *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 5** (Gurevich and Shelah [8]). *There is no MSO-definable choice function on the full-binary tree.*

The following lemma follows from Theorem 5.

▶ **Lemma 6.** *There is no MSO-definable function which assigns to every non-empty antichain $Y$ a finite non-empty subset $X \subseteq Y$.*

## 3 Simple Properties of Languages

In this section some simple lemmas are collected. Their proofs are easy.

▶ **Lemma 7.** *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 parity tree automata. 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 $t \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$, $|ACC(\mathcal{B}, t)| \leq |ACC(\mathcal{A}_1, t)| \cdot |ACC(\mathcal{A}_2, t)|$*

From Lemma 7, we obtain:

▶ **Corollary 8.** *Boundedly, finitely and countably ambiguous tree languages are closed under finite union and intersection.*

▶ **Lemma 9.** *Let $\mathcal{A}$ be a parity automaton with a set $Q$ of useful states. Then, for each state $q \in Q$, $da(\mathcal{A}_q) \leq da(\mathcal{A})$.*

▶ **Corollary 10.** *Let $\mathcal{A}$ be a boundedly (respectively, finitely, countably) ambiguous parity tree automaton with a set $Q$ of useful states. Let $Q' \subseteq Q$. Then $\mathcal{A}_{Q'}$ is boundedly (respectively, finitely, countably) ambiguous.*

▶ **Lemma 11.** *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 $t \in L_2 \setminus L_1$.*

▶ **Lemma 12.** *Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ be a parity tree automaton. Then, there is a parity tree automaton $\mathcal{B} = (Q_{\mathcal{B}}, \Sigma, \{q^{\mathcal{B}}_I\}, \delta_{\mathcal{B}}, \mathbb{C})$ with single initial state such that $L(\mathcal{B}) = L(\mathcal{A})$, and $da(\mathcal{B}) \leq da(\mathcal{A})$.*

▶ **Definition 13** (Moore machine). *A Moore machine is a tuple $M = (\Sigma, \Gamma, Q, q_I, \delta, out)$, where $\Sigma$ 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, $\Gamma$ 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 14.** *Let $F : \Sigma_1^* \to \Sigma_2$ be a function definable by a Moore machine, and let $t_1 \in T^\omega_{\Sigma_1}$. We define $t_2 := \widehat{F}(t_1)$ as a tree in $T^\omega_{\Sigma_2}$ such that $t_2(v) := F(t_1(v_1) \cdot \cdots \cdot t_1(v_k))$ where $v_1, v_2, \ldots, v_k$ is the path from the root to $v$.*

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

▶ **Lemma 15** (Reduction). *Let $L_1$ and $L_2$ be regular tree languages over alphabets $\Sigma_1$ and $\Sigma_2$, respectively. Let $F : \Sigma_1^* \to \Sigma_2$ be a function definable by a Moore machine. Assume that for each $t \in T^\omega_{\Sigma_1}$, $t \in L_1$ iff $\widehat{F}(t) \in L_2$. Then $da(L_1) \leq da(L_2)$.*

## 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 17. Then, we derive some consequences. Finally, a proof of Proposition 17 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 6), and our proof techniques refine the proof techniques of [5].

▶ **Definition 16.** *For a tree language $L$ over alphabet $\Sigma$, we denote by $Subtree(L)$ the tree language $\{t \in T^\omega_\Sigma \mid \exists t' \in L\ \exists v : t'_{\geq v} = t\}$.*

▶ **Proposition 17.** *Let $L$ be a non-empty regular language over an alphabet $\Sigma$ such that $Subtree(L) \neq T^\omega_\Sigma$. Then, the complement of $L$ is not finitely ambiguous.*

▶ **Corollary 18** (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 $\Sigma = \{a_1, \ldots, a_m, c\}$.*

**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 17, 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 8 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 19.** *If $L$ is regular and its complement is finite and non-empty, then $da(L) = \aleph_0$.*

In the rest of this section, Proposition 17 is proved. Let us sketch some ideas of the proof. For a language $L$, as in Proposition 17, and any antichain $Y$ we construct trees $t_0 \in L$ and $t \in \overline{L}$ with the following property: if $\mathcal{A}$ does not accept $t_0$ and accepts $t$, 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 $\overline{L}$ and is finitely ambiguous, then $X$ is finite - a contradiction to Lemma 6. 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 parity tree automaton, and let $t$ be a $\Sigma$-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 $(\epsilon, q_I)$.

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 [6], 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 [7, 10].

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 17.1.

1. Let $\overline{s}$ be a play which is consistent with $str_\phi$, 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 $str_\phi$ is a positional winning strategy for Automaton.
3. If $str$ is a positional winning strategy for Automaton, then $\phi_{str} \in ACC(\mathcal{A}, t)$.

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 17.2. Assume $t \notin L(\mathcal{A})$ and let $\phi$ 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 $str_\phi$ 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 towards a 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 $\phi$ 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 17.1(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 $str_\phi$ we obtain $(q_l, q_r) = (\phi(v_i \cdot l), \phi(v_i \cdot r))$. Since $\phi(v_i) = q_i$ we have $(\phi(v_i), t(v_i), \phi(v_i \cdot l), \phi(v_i \cdot r)) \notin \delta$. $\phi$ 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 17.2, there is a function $Invalid_{\mathcal{A}}(\phi, STR, t, v)$ which, for every accepting computation $\phi$ 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 $\phi$ 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 $\delta$. (By Claim 17.1(1), in any play consistent with $\phi$, 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 $d_0 \ldots d_j \ldots$ are consistent with $STR$ and are along the path from the root to $v$, i.e., $d_0 d_1 \ldots d_j \leq v$.

To sum up, we have the following claim:

▷ Claim 17.3. $Leads_{\mathcal{A}}(\phi, STR, t, v)$ 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 17.3 plays a crucial role in our proof. It is instructive to compare it with Theorem 5 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.

▷ Claim 17.4. 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 17.4 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 claim:

▷ Claim 17.5. 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 $\phi$ 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 17.3, $\exists\sigma\exists STR : \psi_{t_0}(\sigma) \wedge \psi_{\widehat{STR}}(STR) \wedge Leads_{\mathcal{A}}(\phi, STR, \sigma, v)$ defines such a function. ◁

**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 $Y$.

▷ Claim 17.6. 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 $t_0[t_1/Y]$.

## 4.3 Finishing Proof of Proposition 17

Now, we have all the ingredients ready for the proof of Proposition 17. Let $L$ be as in Proposition 17. We claim that there are regular $\Sigma$-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 $Subtree(L) \neq T_\Sigma^\omega$), 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 $Y$.

Let $\mathcal{A}$ be such that $L(\mathcal{A}) = \overline{L}$, and let $\alpha_{t_0,\mathcal{A},\widehat{STR}}(\phi, v)$ be a formula which defines the function from Claim 17.5 ($t_0[t_1/Y]$ 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 17.7. $Choice_{\mathcal{A},t_0,t_1,\widehat{STR}}(Y, \phi, y)$ defines a function which for every non-empty antichain $Y$ and an accepting computation $\phi$ of $\mathcal{A}$ on $t_0[t_1/Y]$, returns a node $y \in Y$.

Proof. By Claim 17.5, $\alpha_{t_0,\mathcal{A},\widehat{STR}}(\phi, v)$ 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 "$\phi$ is an accepting computation of $\mathcal{A}$ on the tree $\sigma$."

▷ Claim 17.8. $ChooseSubset_{\mathcal{A},t_0,t_1,\widehat{STR}}(Y,X)$ 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 $t_0[t_1/Y] \in \overline{L}$, because $t_1 \notin Subtree(L)$. Hence, $\mathcal{A}$ has at least one accepting computation on $t_0[t_1/Y]$. Therefore, $X$ is non-empty, by Claim 17.7. The "Moreover" part immediately follows from Claim 17.7. ◁

Let $\mathcal{A}$ be such that $L(\mathcal{A}) = \overline{L}$ and assume towards a 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 17.8, 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 6.

## 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 $\sigma$, we denote by $t_\sigma$, the full-binary tree with all nodes labeled by $\sigma$. 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 $L_{\neg a_i} := \{t \in T^\omega_{\Sigma_n} \mid \text{no node in } t \text{ is labeled by } a_i\}$.

▶ **Proposition 20.** *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 automata. Therefore, by Lemma 7, 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 21.** *Let $L_{\exists a_1 \wedge \cdots \wedge \exists a_m} := \{t \in T^\omega_{\Sigma_n} \mid$ 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 $L_{\exists a_1 \wedge \cdots \wedge \exists a_m} \cap T^\omega_{\{c,a_1,\ldots,a_m\}} \subseteq L$. Then, $L$ is not finitely ambiguous.*

**Proof.** Define a function $F : \Sigma^* \to \Sigma$ such that $F(\sigma_1 \ldots \sigma_k) := a_{k-i+1}$ 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, $F(\sigma_1 \ldots \sigma_k) := c$.

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 15 we conclude that $da(L) \geq da(L_{\exists a_1})$. Since $L_{\exists a_1}$ is not finitely ambiguous (by Corollary 18 (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 22.** *Let $\mathcal{A}$ be a finitely ambiguous automaton over alphabet $\Sigma_n$ such that $L(\mathcal{A}) = L_{\neg a_1 \vee \cdots \vee \neg a_k}$ 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 $L_{\neg a_1 \vee \cdots \vee \neg a_m}$. 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 18 (2). $L(\mathcal{A}_{Q_{\exists a_m \wedge \neg t_c}})$ is finitely ambiguous (by Corollary 10) 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 8. Therefore, by Lemma 11, 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 $\{t \in T^{\omega}_{\Sigma} \mid Tree(c, t', t) \in L_{\neg a_1 \vee \cdots \vee \neg a_m}\} = L_{\neg a_1 \vee \cdots \vee \neg a_{m-1}}$. Therefore, $L(\mathcal{A}_{R'[2]}) = L_{\neg a_1 \vee \cdots \vee \neg a_{m-1}}$, 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 $Q_{\exists a_1 \wedge \cdots \wedge \exists a_{m-1}} := \{q \in R[2] \mid L(\mathcal{A}_q) \cap L_{\exists a_1 \wedge \cdots \wedge \exists a_{m-1}} \neq \emptyset\}$ and let
$Q_{t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}} := \{q \in Q_{\exists a_1 \wedge \cdots \wedge \exists a_{m-1}} \mid t_c \in L(\mathcal{A}_q)\}$ and
$Q_{\neg t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}} := Q_{\exists a_1 \wedge \cdots \wedge \exists a_{m-1}} \setminus Q_{t_c \wedge \exists a_1 \wedge \cdots \wedge \exists a_{m-1}}$.

▷ Claim 22.1. 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 towards 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 21 we obtain 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 22.1. 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 $\phi'$ be the computation obtained from $\phi$ by grafting $\phi_c$ on $r$. We conclude that $\phi' \in ACC(\mathcal{A}, t_c)$.

Assume towards 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 $\phi'$). 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 23.** *Let $\Sigma = \{a_1, a_2, c\}$. We define the following languages over $\Sigma$:*

- *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 24.** *The degree of ambiguity of $L^{fa}$ is finite.*
The proposition follows from Lemma 25 and Lemma 27 proved below.

▶ **Lemma 25.** *There is a finitely ambiguous automaton which accepts $L^{fa}$*

**Sketch.** On a tree $t \in L_m$ the automaton "guesses" a position $i < m$ and checks that $t_{\geq l^i r} \in L_{\neg a_1 \vee \neg a_2}$ (using a 2-ambiguous automaton), and 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). ◀

▶ **Lemma 26.** *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 18 (2)), we derive the following claim:

▷ Claim 26.1. 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 $\phi_i$ 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 $\phi'_i$ from $\phi_i$ by grafting $\phi^c_i$ 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 $\phi'$ from $\widehat{\phi}$ by grafting $\phi_{p_i}$ on $l^i r$ and $\phi_{p_2}$ on $l^j r$. It follows that $\phi'$ 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. 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 27.** *$L^{fa}$ is not boundedly ambiguous*

**Proof.** $\forall m \in \mathbb{N} : L_m \subseteq L^{fa}$, and therefore from Lemma 26 it follows that $L^{fa}$ is not $(m-1)$-ambiguous. That is, $L^{fa}$ is not boundedly ambiguous. ◀

## 7 Uncountably Ambiguous Languages

▶ **Definition 28.** *Let $L^{\neg ba}$ be an arbitrary regular tree language over alphabet $\Sigma$ which is not boundedly ambiguous. Let $c \in \Sigma$ and define a language $L^{uc}$ over alphabet $\Sigma$. $t \in L^{uc}$ iff the following conditions hold:*

- $\forall v \in l^* : t(v) = c$
- $\forall v \in l^*r : t_{\geq v} \in L^{\neg ba}$.

It is immediate to formalize $L^{uc}$ using an MSO-formula, and therefore it is regular.

▶ **Proposition 29.** *The degree of ambiguity of $L^{uc}$ is $2^{\aleph_0}$.*

**Proof.** Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, \mathbb{C})$ be a parity automaton such that $L(\mathcal{A}) = L^{uc}$. Let $Q' := \{\phi(v) \mid v \in l^*r \text{ and } \exists t : \phi \in ACC(\mathcal{A}, t)\}$. First, observe the following Claim:

▷ Claim 29.1. $L(\mathcal{A}_{Q'}) = L^{\neg ba}$

Next, define $Q_{unam} := \{q \in Q' \mid \mathcal{A}_q \text{ is unambiguous}\}$, and $Q_{amb} := Q' \setminus Q_{unam}$. Let $k := |Q_{unam}|$. By Lemma 7(1), $L(\mathcal{A}_{Q_{unam}})$ is $k$-ambiguous, since it is accepted by an automaton $\mathcal{A}_{Q_{unam}}$ and $L(\mathcal{A}_{Q_{unam}})$ is a union of $k$ unambiguous languages. Clearly, $L(\mathcal{A}_{Q_{unam}}) \subseteq L(\mathcal{A}_{Q'})$, and therefore by Claim 29.1 we obtain $L(\mathcal{A}_{Q_{unam}}) \subseteq L^{\neg ba}$. Applying Lemma 11 we obtain a tree $t' \in L^{\neg ba}$ such that $t' \notin L(\mathcal{A}_{Q_{unam}})$. Since $L(\mathcal{A}_{Q_{unam}}) \cup L(\mathcal{A}_{Q_{amb}}) = L(\mathcal{A}_{Q'}) = L^{\neg ba}$, we conclude that $t' \in L(\mathcal{A}_{Q_{amb}})$.

Construct a tree $\widehat{t}$ from $t_c$ by grafting $t'$ on all nodes $l^*r$. It is clear that $\widehat{t} \in L(\mathcal{A}) = L^{uc}$, and therefore there is a computation $\widehat{\phi} \in ACC(\mathcal{A}, \widehat{t})$. By definition of $t'$, $\forall v \in l^*r : \widehat{\phi}(v) \in Q_{amb}$.

For every $q \in Q_{amb}$, there is a tree $t^q \in L(\mathcal{A}_q)$ such that there are two different computations $\phi_1^q, \phi_2^q$ of $\mathcal{A}_q$ on $t^q$.

Let $q_i := \widehat{\phi}(l^ir)$; let $\bar{t}$ be the tree which is obtained from $\widehat{t}$ as follows: graft $t^{q_i}$ on $l^ir$ in $\widehat{t}$ for $i \in \mathbb{N}$. We will show that there are uncountable many accepting computations on $\bar{t}$.

For $S \subseteq \mathbb{N}$, let $\phi_S$ be a computation on $\bar{t}$ which coincides with $\widehat{\phi}$ on the nodes in $l^* \cup l^*r$, and for each $i$: if $i \in S$, then $\phi_S$ coincides with $\phi_1^{q_i}$ on $\bar{t}_{\geq l^ir} = t^{q_i}$; if $i \notin S$, then $\phi_S$ coincides with $\phi_2^{q_i}$ on $\bar{t}_{\geq l^ir} = t^{q_i}$. It is easy to see that $\phi_S$ is an acceptiong computation on $\bar{t}$, and $\phi_{S_1} \neq \phi_{S_2}$ for $S_1 \neq S_2$. Hence, $|ACC(\mathcal{A}, \bar{t})| \geq |\{S \mid S \subseteq \mathbb{N}\}| = 2^{\aleph_0}$, and $da(L^{uc}) = 2^{\aleph_0}$. ◀

## 8 Conclusion

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

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. Yet, it is open whether the degree of ambiguity of the complement of countable regular languages is $\aleph_0$. Relying on Niwiński's characterization of countable regular languages [9], we can prove that every countable language is unambiguous.

## References


1 André Arnold. Rational omega-languages are non-ambiguous. *Theor. Comput. Sci.*, 26:221–223, September 1983. doi:10.1016/0304-3975(83)90086-5.

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. doi:10.4230/LIPIcs.CSL.2013.81.

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

7 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.

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

9 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.

10 D. Perrin and J.É. Pin. *Infinite Words: Automata, Semigroups, Logic and Games*. ISSN. Elsevier Science, 2004. URL: https://books.google.fr/books?id=S7hHhJc4iNgC.

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

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

13 Boris A. Trakhtenbrot and Ya. M. Barzdin. Finite automata, behavior and synthesis. 1973.