Degrees of Ambiguity of Büchi Tree Automata

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

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

Abstract

An automaton is unambiguous if for every input it has at most one accepting computation. An automaton is finitely (respectively, countably) ambiguous if for every input it has at most finitely (respectively, countably) many accepting computations. An automaton is boundedly ambiguous if there is $k \in \mathbb{N}$, such that for every input it has at most k accepting computations. We consider nondeterministic Büchi automata (NBA) over infinite trees and prove that it is decidable in polynomial time, whether an automaton is unambiguous, boundedly ambiguous, finitely ambiguous, or countably ambiguous.

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

Keywords and phrases automata on infinite trees, ambiguous automata

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.50

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

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 [4, 5] for recent surveys).

Probably, the oldest restricted form of nondeterminism is unambiguity. An automaton is unambiguous if for every input there is at most one accepting run. For automata over finite words there is a rich and well-developed theory on the relationship between deterministic, unambiguous and nondeterministic automata [5]. All three models have the same expressive power. Unambiguous automata are exponentially more succinct than deterministic ones, and nondeterministic automata are exponentially more succinct than unambiguous ones [6, 7].

Many other notions of ambiguity were suggested and investigated. A recent paper [5] 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

[1] corresponding author

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. Our main result is:

▶ **Theorem 1.** *There are polynomial time algorithms that decide whether a Büchi automaton over trees is unambiguous, boundedly ambiguous, finitely ambiguous, or countably ambiguous.*

Over infinite trees, Büchi tree automata are less expressive than Monadic Second-Order Logic or parity automata. In Sect. 8 we will show that the problem whether a parity tree automaton is ambiguous is co-NP complete.

Related Works

Weber and Seidl [14] investigated several classes of ambiguous automata on words and obtained polynomial time algorithms for deciding the membership in each of these classes. Their algorithms were derived from structural characterizations of the classes.

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

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

Weber and Seidl [14] proved that an NFA is not boundedly ambiguous iff it contains the forbidden pattern for bounded ambiguity. This pattern is testable in polynomial time; hence, it can be decided in polynomial time whether the degree of ambiguity of a NFA is bounded.

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

Löding and Pirogov [8] and Rabinovich [10] provided structural characterizations and polynomial algorithms for bounded, finite and countable ambiguity of Büchi automata on ω-words. These characterizations and algorithms can be adopted for other acceptance conditions: parity, Rabin, Muller, etc.

Our proof of Theorem 1 will first provide structural characterizations of bounded, finite and countable ambiguity of automata on infinite trees, and then derive polynomial algorithms.

As far as we know, the degrees of ambiguity for automata over infinite trees have not been investigated. The decidability whether an automaton on infinite trees is finitely ambiguous or countably ambiguous can be obtained from the results of Bárány et al. in [2], where an extension of monadic second-order logic of order with the cardinality quantifiers "there exist uncountably many sets," "there are countably many sets," "there are finitely many sets" (MSO($\exists^{<\aleph_0}, \exists^{>\aleph_0}$)) was investigated. It was proved that, over the class of finitely branching trees, MSO($\exists^{<\aleph_0}, \exists^{>\aleph_0}$) is (effectively) equally expressive to plain monadic second-order logic of order (MSO). It is a routine exercise for a given automaton on infinite trees to write sentences in MSO($\exists^{<\aleph_0}, \exists^{>\aleph_0}$) that express "the automaton has finitely many accepting runs," "the automaton has countably many accepting runs," and "the automaton has uncountably many accepting runs." By combining these with Rabin's theorem on decidability of MSO over infinite trees we obtain that it is decidable whether an automaton is finitely or countably ambiguous. Unfortunately, the complexity of the algorithm extracted from this proof is (at least) triple exponential. Our proofs are inspired by techniques used in [2].

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

Organization of the paper. The next section contains standard definitions and notations about tree automata. The main results are stated in Sect. 3 and are proved in Sects. 4–7. The last section presents the conclusion and further results. Missing proofs will appear in the full paper.

2 Preliminaries

We recall here standard terminology and notations about trees and automata [13, 9].

Trees. A **tree order** $\leq$ on a set t is a partial order with a unique minimal element (the root of t) such that for every $u \in t$, the set $\{v \mid v \leq u\}$ is finite and linearly ordered by $\leq$. We use standard terminology and notations: u is an *ancestor* of v if $u \leq v$, u is a *child* of v, u is a *leaf*, u and v are incomparable - denoted by $u \perp v$ - if neither $u \leq v$ nor $v \leq u$; a subset A of t is an antichain, if its elements are incomparable with each other.

If $\leq$ is a tree order over t and $u \in t$, we denote by $t_{\geq u}$ the restriction of $\leq$ to the set $\{v \mid v \geq u\}$; $t_{\geq u}$ is called the subtree of t rooted at u. A binary tree is a tree order with a partition of children into two sets - left/right child such that every non-leaf node has exactly one left child and one right child. The full binary tree is a binary tree without leaves.

There is a natural order isomorphism between the full binary trees and the set of strings $\{l, r\}^*$ with prefix order; it maps ϵ to the root, l to the left child of root, etc. We will often refer to a node in the full binary tree by the corresponding string over $\{l, r\}$.

If Σ is an alphabet, then a Σ-labeled tree is a tree t and a function σ_t from elements of t to Σ. We often use "Σ-tree" for "Σ-labeled tree"; Σ-labeled full binary trees are defined similarly. We often use "tree" or variables t, t', T for "full binary tree" for "labeled tree" or for "full binary labeled tree." It will be clear from the context or unimportant what kind of tree is used. In particular, $t_{\geq u}$ is naturally defined over all such kinds of trees.

Grafting. If t, t_1 are trees and $u \in t$, then the grafting of t_1 at u in t, denoted by $t \circ_u t_1$, is a tree which is obtained from t when $t_{\geq u}$ is replaced by t_1. Formally, this is defined by taking an isomorphic copy t'_1 of t_1 with the domain disjoint from the domain of t and defining a tree order $\leq'$ on $t \setminus t_{\geq u} \cup t'_1$, by $v_1 \leq' v_2$ if v_1 is an ancestor of v_2 in t or in t'_1, or if $v_1 \in t$ and $v_2 \in t'_1$. More generally, if t is a tree, A is an antichain in t and t_1 is a tree, then the grafting of t_1 at A in t is a tree which is obtained from t by replacing every subtree $t_{\geq a}$ by t_1 for $a \in A$. Even more generally, if t is a tree, A is an antichain in t and f assigns a tree t_a to every $a \in A$, then grafting f at A in t is a tree obtained from t, by replacing every subtree $t_{\geq a}$ by t_a for $a \in A$.

Automata. We use standard notations and terminology about Büchi automata on (infinite) full binary Σ-labeled trees and on ω-strings. A Büchi automaton $\mathcal{A}$ has an alphabet Σ, a finite set of states $Q_{\mathcal{A}}$, initial states $Q_I \subseteq Q_{\mathcal{A}}$, a transition relation $\delta_{\mathcal{A}}$, and a set of final states $F \subseteq Q_{\mathcal{A}}$. For a Büchi automaton on ω-string, $\delta_{\mathcal{A}} \subseteq Q_{\mathcal{A}} \times \Sigma \times Q_{\mathcal{A}}$; for a Büchi tree automaton, $\delta_{\mathcal{A}} \subseteq Q_{\mathcal{A}} \times \Sigma \times Q_{\mathcal{A}} \times Q_{\mathcal{A}}$.

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

The notion of a computation/run, accepting computation of $\mathcal{A}$ on an ω-string/tree is defined as usual. We use letter f for a final state of automata, and we use the letters ϕ, ϕ' for computations. 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 of $\mathcal{A}$.

A state q of $\mathcal{A}$ is useful if there is a tree, an accepting computation ϕ on t and a node u such that $\phi(u) = q$. There is a polynomial algorithm that for $\mathcal{A}$ and q checks whether q is useful. There is a polynomial algorithm which for every $\mathcal{A}$ computes an automaton $\mathcal{B}$ with all states useful and $L(\mathcal{A}) = L(\mathcal{B})$. We will always assume that all states are useful.

Degree of Ambiguity. We denote by $|X|$, the cardinality of a set X. $da(\mathcal{A})$ is defined as $\sup\{|ACC(\mathcal{A}, t)| \mid t \in L(\mathcal{A})\}$. We say that $\mathcal{A}$ is unambiguous if $da(\mathcal{A}) = 1$, boundedly ambiguous if there is $k \in \mathbb{N}$ such that $da(\mathcal{A}) \leq k$, finitely ambiguous if $|ACC(\mathcal{A}, t)|$ is finite for every t, countably ambiguous if $ACC(\mathcal{A}, t)$ is countable for every t.

▶ **Example 2.** Consider Büchi tree automata $\{\mathcal{A}_i\}_{i=1}^3$ over the unary alphabet.

1. $\mathcal{A}_1$ has a single state q; it is initial and final. $\Delta_1 = (q, q, q)$. $\mathcal{A}_1$ is deterministic.
2. The set of states of $\mathcal{A}_2$ is $Q_2 := \{q, q_1\}$, both are initial and final, and $\Delta_2 := Q \times Q \times Q$. $\mathcal{A}_2$ is uncountably ambiguous.
3. The set of states of $\mathcal{A}_3$ is $Q_3 := \{q, f\}$, f is the final state, q is initial, and $\Delta_3 = \{q\} \times Q \times Q \cup \{(f, f, f)\})$. $\mathcal{A}_3$ is countably ambiguous.
4. Over the unary alphabet there is only one full binary tree; therefore, every finitely ambiguous automata is boundedly ambiguous.

A computation of $\mathcal{A}$ on a Σ-tree t can be considered as a $Q_{\mathcal{A}}$-labeling of t. The grafting of computations $\phi \circ_v \phi'$ is defined as for the corresponding $Q_{\mathcal{A}}$-trees. We often use implicitly the following simple Lemma.

▶ **Lemma 3** (Grafting). *Let $\mathcal{A}$ be an automaton, t, t_1 trees, $v \in t$ 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 4.** *$da(\mathcal{A}) \geq da(\mathcal{A}_q)$ for every useful state q of $\mathcal{A}$.*

We suspect that the following lemma is folklore. For lack of reference, we provide a proof in the full version of the paper.

▶ **Lemma 5.** *It is computable in polynomial time whether a Büchi tree automaton is unambiguous.*

The next definition and theorem are taken from [8, 10]. They provide a forbidden pattern characterization of degrees of ambiguity of automata on ω-words.

▶ **Definition 6** (Forbidden pattern for ω-word automata). *Let $\mathcal{B}$ be a Büchi automaton on ω-words such that all its states are useful.*

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

▶ **Theorem 7.** *Let $\mathcal{B}$ be a Büchi automaton on ω-words.*
1. *$\mathcal{B}$ has uncountably many accepting runs on some ω-word if and only if $\mathcal{B}$ contains the forbidden pattern for countable ambiguity.*
2. *$\mathcal{B}$ has infinitely many accepting runs on some ω-word if and only if $\mathcal{B}$ contains the forbidden pattern for finite ambiguity.*
3. *$\mathcal{B}$ is not boundedly ambiguous iff it contains the forbidden pattern for bounded ambiguity.*

3 Main Result

In this section we first introduce branch ambiguity and ambiguous transition patterns and then state our main results.

3.1 Branch Ambiguity

▶ **Definition 8** (Projection of a computation on a branch). *Let $\phi \in ACC(\mathcal{A}, t)$ and $\pi := v_0 v_1 \dots$ be a branch of t. $\phi(\pi) := \phi(v_0)\phi(v_1)\cdots \in Q_{\mathcal{A}}^{\omega}$ is the projection of ϕ on π. We define $ACC(\mathcal{A}, t, \pi) := \{\phi(\pi) \mid \phi \in ACC(\mathcal{A}, t)\}$.*

▶ **Definition 9** (Branch ambiguity). *$\mathcal{A}$ is at most n branch-ambiguous if $|ACC(\mathcal{A}, t, \pi)| \leq n$ for every t and branch π. $\mathcal{A}$ is bounded branch ambiguous if it is at most n branch ambiguous for some n. $\mathcal{A}$ is finitely (countably) branch ambiguous if $|ACC(\mathcal{A}, t, \pi)|$ is finite (respectively, countable) for every t and π.*

Let $\mathcal{A}$ be a Büchi tree automaton. We define a Büchi ω-automaton $\mathcal{A}_B$ which has the same ambiguity as branch ambiguity of $\mathcal{A}$:

▶ **Definition 10** (Branch automaton). *For a Büchi tree automaton $\mathcal{A} = (Q, \Sigma, Q_I, \delta, F)$, the corresponding branch automaton $\mathcal{A}_B$ is an ω-word automaton $(Q, \Sigma_B, Q_I, \delta_B, F)$, where*
1. *$\Sigma_B := \Sigma \times \Sigma_d \times \Sigma_{cons}$ with*
   a. *$\Sigma_d := \{l, r\}$ directions alphabet (left/right).*
   b. *$\Sigma_{cons} := \{S \subseteq Q \mid \bigcap_{q \in S} L(\mathcal{A}_q) \neq \emptyset\}$ sets of states, which we consider "consistent."*
2. *$(q, a, q') \in \delta_B$ iff $a = (\sigma, l, S)$ and $\exists p \in S : (q, \sigma, (q', p)) \in \delta$ or $a = (\sigma, r, S)$ and $\exists p \in S : (q, \sigma, (p, q')) \in \delta$.*

▶ **Lemma 11.** *The branch ambiguity of a tree automaton $\mathcal{A}$ is bounded (respectively, finite, countable) iff the ambiguity of the corresponding branch ω-automaton $\mathcal{A}_B$ is bounded (respectively, finite, countable).*

▶ **Proposition 12** (Computability of branch ambiguity). *It is computable in polynomial time whether the branch ambiguity of $\mathcal{A}$ is bounded, finite, or countable.*

3.2 Ambiguous Transition Pattern

▶ **Definition 13** (Ambiguous transition pattern). *Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, F)$ be a Büchi automaton with corresponding branch automaton $\mathcal{A}_B = (Q, \Sigma_B, Q_I, \delta_B, F)$. $\mathcal{A}$ has a **q-ambiguous transition pattern** if $q \in Q$ and there are $p_1, p_2 \in Q$ and $y_1 \in \Sigma_B^*$, $y_2 \in \Sigma_B^+$ with runs of $\mathcal{A}_B$ from q to p_1 on y_1 and from p_2 to q on y_2 such that at least one of the following holds:*
1. *There are two transitions $(p_1, (a, d, \{q_1\}), p_2), (p_1, (a, d, \{q_2\}), p_2) \in \delta_B$ with $q_1 \neq q_2$ and $L(\mathcal{A}_{q_1}) \cap L(\mathcal{A}_{q_2}) \neq \emptyset$, or*
2. *There is a transition $(p_1, (a, d, \{q_1\}), p_2) \in \delta_B$ with $da(\mathcal{A}_{q_1}) > 1$.*

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

Lemma 14 provides sufficient conditions for an ambiguous transition pattern.

▶ **Lemma 14** (q-ambiguous transition pattern). *Let $\mathcal{A}$ be a Büchi tree automaton and $v \perp w$. If one of the following conditions holds, then $\mathcal{A}$ has a q-ambiguous transition pattern.*

1. *There is $\phi \in ACC(\mathcal{A}_q, t)$ such that $q = \phi(v)$ and $\phi(w) = p$, where $\mathcal{A}_p$ is ambiguous.*
2. *There are $\phi, \phi' \in ACC(\mathcal{A}_q, t)$ such that $q = \phi(v)$ and $\forall v'(v' \leq v) \rightarrow (\phi(v') = \phi'(v'))$, and $\phi(w) \neq \phi'(w)$.*

▶ **Lemma 15.** *If $\mathcal{A}$ has an ambiguous transition pattern then its ambiguity degree is not bounded. If $\mathcal{A}$ has an f-ambiguous transition pattern (for a final state f), then its ambiguity degree is not countable.*

▶ **Lemma 16.** *It is computable in polynomial time whether $\mathcal{A}$ has an ambiguous transition pattern and whether $\mathcal{A}$ has an f-ambiguous transition pattern for a final state f.*

3.3 Characterizations of Degrees of Ambiguity

The next two propositions characterize bounded and finite ambiguity.

▶ **Proposition 17** (Bounded ambiguity). *The following are equivalent:*

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

When "Büchi tree automaton $\mathcal{A}$" is replaced by "an automaton $\mathcal{A}$ on finite trees" in Proposition 17 we obtain an instance of a theorem proved by Seidl in [12]. A proof of Proposition 17 is a simple variation of the proof in [12]. It will be sketched in the full paper.

▶ **Proposition 18** (Finite ambiguity). *The following are equivalent:*

1. *Büchi tree automaton $\mathcal{A}$ is not finitely ambiguous.*
2. *At least one of the following conditions holds:*
   a. *$\mathcal{A}$ is not finite branch ambiguous.*
   b. *$\mathcal{A}$ has an f-ambiguous transition pattern for a final state f.*

In order to characterize countable ambiguity, we first introduce branching patterns.

▶ **Definition 19** (A branching pattern for $\mathcal{A}$ over (Q, f)). *Let $\mathcal{A}$ be a Büchi tree automaton, f a final state of $\mathcal{A}$ and $Q \subseteq Q_{\mathcal{A}} \setminus \{f\}$, where $Q_{\mathcal{A}}$ are the states of $\mathcal{A}$. A branching pattern M for $\mathcal{A}$ over (Q, f) is a function $\tau_M : Q \rightarrow Q \times Q$ and a tuple $(q_1, q_2) \in Q \times Q$.*

▶ **Definition 20** (Realizable branching pattern). *Let t be a full binary tree and $u \perp v$ two nodes of t. A branching pattern M for $\mathcal{A}$ over (Q, f) is realized in t at u, v by computations ϕ_1, ϕ_2, $\{\phi_q \mid q \in Q\}$ if the following holds:*

1. *$\phi_1, \phi_2 \in ACC(\mathcal{A}_f, t)$ and $\phi_1(u) = f = \phi_2(v)$, $\phi_1(v) = q_1$ and $\phi_2(u) = q_2$.*
2. *For each $q \in Q$: $\phi_q \in ACC(\mathcal{A}_q, t)$ and $\tau_M(q) = (\phi_q(v), \phi_q(u))$ and ϕ_q visits an accepting state on both paths from the root of t to u and from the root of t to v.*

In Sects. 5 and 7 we prove the next two propositions. Their proof is more complicated than the proofs of Propositions 17 and 18.

▶ **Proposition 21** (Countable ambiguity). *The following are equivalent:*

1. *Büchi tree automaton $\mathcal{A}$ is not countably ambiguous.*
2. *At least one of the following conditions holds:*
   a. *$\mathcal{A}$ is not countable branch ambiguous.*
   b. *$\mathcal{A}$ has an f-ambiguous transition pattern for a final state f.*
   c. *A branching pattern for $\mathcal{A}$ is realizable.*

▶ **Proposition 22.** *It is computable in polynomial time whether there is a realizable branching pattern for a Büchi tree automaton $\mathcal{A}$.*

▶ **Theorem 23** (Main). *It is computable in polynomial time whether a Büchi tree automaton is unambiguous, bounded ambiguous, finitely ambiguous, or countably ambiguous.*

Proof. For unambiguity – by Lemma 5. For bounded ambiguity by Proposition 17, Lemma 16 and Proposition 12. For finite ambiguity by Proposition 18, Lemma 16 and Proposition 12. For countable ambiguity by Propositions 21, 12, 22 and Lemma 16. ◀

Road map of the proofs. Sect. 4 and Sect. 5 deal with structural characterizations of finite and countable ambiguity and prove Propositions 18 and 21. Sects 6 and 7 deal with computability of degrees of ambiguity. Proposition 12 and Lemma 16 are proved in Sect. 6, and Proposition 22 is proved in Sect. 7. The proofs of Lemmas 11, 14 and 15 are given in the full version of the paper.

4 Finite Ambiguity

In this section we prove Proposition 18 - a structural characterization of finite ambiguity. (2) ⇒ (1) follows from Lemma 11 and Lemma 15. Below we prove the (1) ⇒ (2) direction.

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

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

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

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

If $|ACC(\mathcal{A}, t, \pi)|$ is infinite, then by the definition of branch ambiguity we have that $\mathcal{A}$ is not finite branch ambiguous, and 2(a) holds. Otherwise, there exist $\phi_1, \dots, \phi_k \in ACC(\mathcal{A}, t)$ such that $ACC(\mathcal{A}, t, \pi) = \{\phi_i(\pi) \mid 1 \leq i \leq k\}$. Choose n such that for all $1 \leq i < j \leq k : \phi_i(v_0 \dots v_n) \neq \phi_j(v_0 \dots v_n)$. One of these computations, say ϕ_1, holds that $\forall i \leq n : \phi_1(v_i) = q_i$. Hence, $\forall i : \phi_1(v_i) = q_i$.

Let f be an accepting state which occurs infinitely often in $\phi_1(\pi)$. Choose $N > n$ such that $\phi_1(v_N) = q_N = f$. By selection of q_N, there are infinitely many accepting computations of $\mathcal{A}_f$ on $t_{\geq v_N}$. Take two different accepting computations $\phi', \phi'' \in ACC(\mathcal{A}_f, t_{\geq v_N})$. By selection of ϕ_1, $\forall i \geq N : \phi_1(v_i) = \phi'(v_i) = \phi''(v_i) = q_i$. Therefore, ϕ' and ϕ'' differ at some node $w \notin \pi$, and there exist $i > N$ such that $\phi_1(v_i) = f = \phi'(v_i) = \phi''(v_i)$ and $v_i \perp w$. Applying Lemma 14(2) on ϕ', ϕ'' and $v_i \perp w$, we obtain that $\mathcal{A}_f$ has an f-ambiguous transition pattern, and 2(b) holds.

5 Countable Ambiguity

In this section we prove Proposition 21 – a structural characterization of countable ambiguity.

5.1 Direction (2) ⇒ (1) of Proposition 21

2(a)⇒(1) follows by definition of branch ambiguity, and 2(b)⇒(1) follows by Lemma 15. Below 2(c)⇒(1) is proved.

▶ **Definition 24** (Corresponding automaton $\mathcal{A}_M$ for pattern M). *Let M be a branching pattern for $\mathcal{A}$ over (Q, f) (see Definition 19). We define a Büchi tree automaton $\mathcal{A}_M$ over the unary alphabet with the set of states $Q \cup \{f\}$; all states are final, the initial state is f, and the transition relation is $\Delta_M := \{(q, q', q'') \mid q \in Q \text{ and } (q', q'') = \tau_M(q)\} \cup \{(f, q_1, f), (f, f, q_2)\}$.*

The following simple lemma states the properties of accepting computations of $\mathcal{A}_M$. It will be useful in showing that if a branching pattern for $\mathcal{A}$ is realized, then $\mathcal{A}$ is not countably ambiguous.

▶ **Lemma 25** (Accepting computations of $\mathcal{A}_M$).

1. *Let ϕ be an accepting computation of $\mathcal{A}_M$. Then the set of nodes $\{v \mid \phi(v) = f\}$ is a branch.*
2. *For every branch π there is an accepting computation ϕ of $\mathcal{A}_M$ such that $\forall v \in \pi(\phi(v) = f)$.*
3. *The set of accepting computations of $\mathcal{A}_M$ is uncountable.*

▶ **Lemma 26.** *Let $\mathcal{A} = (Q_{\mathcal{A}}, \Sigma, Q_I, \delta, F)$ be a Büchi tree automaton such that a branching pattern for $\mathcal{A}$ is realizable. Then $\mathcal{A}$ is not countably ambiguous.*

Proof. Assume that a branching pattern M over (Q, f) is realized in t at $u \perp v$ by ϕ_1, ϕ_2, $\{\phi_q \mid q \in Q\}$. We construct a sequence of trees: $t_1 := t$, and $\forall i \geq 1 : t_{i+1} := t_i \circ_{A_i} t$, where $A_i = \{u, v\}^i$. We graft t at every node in A_i of t_i. This operation is well defined as A_i is an antichain ($\forall a_1 \neq a_2 \in A_i : a_1 \perp a_2$, since $u \perp v$).

For each $y \in \{l, r\}^*$ we define $k_y := max\{i \mid y \in \{u, v\}^i \cdot z, z \in \{l, r\}^*\}$. Notice that by the construction, if $\sigma_{t_{1+k_y}}(y) = a$ then $\forall i > k_y : \sigma_{t_i}(y) = a$. Define t^ω as $\sigma_{t^\omega}(y) := t_{1+k_y}(y)$.

We now proceed to show that the set of accepting computations of $\mathcal{A}_f$ on t^ω is not countable, by defining an injective map from the set of accepting computations of $\mathcal{A}_M$ (on the tree over the unary alphabet) to the set of accepting computations of $\mathcal{A}_f$ on t^ω.

▶ **Notations 27.** *Let h be a homomorphism $h : \{l, r\}^* \to \{l, r\}^*$, where $h(l) = v$ and $h(r) = u$. Since $u \perp v$, it follows that h is a bijection from $\{l, r\}^*$ onto $\{u, v\}^*$.*

For each accepting computation ϕ of $\mathcal{A}_M$ we assign an accepting computation $\widehat{\phi}$ of $\mathcal{A}_f$ on t^ω. If $w \in \{u, v\}^*$ then $\widehat{\phi}(w) := \phi(h^{-1}(w))$ (hence, the map is injective). Otherwise, let $w = y \cdot z$ where $y \in \{u, v\}^{k_w}$ and $z \in \{l, r\}^+$. If $\phi(h^{-1}(y)) = q \neq f$ then $\widehat{\phi}(w) := \phi_q(z)$. Else, if $\phi(h^{-1}(y \cdot u)) = f$ then $\widehat{\phi}(w) := \phi_1(z)$; otherwise, $\widehat{\phi}(w) := \phi_2(z)$ (recall that ϕ_1, ϕ_2, ϕ_q are computations on t that realize M).

It is routine to verify that $\widehat{\phi}$ is an accepting computation of $\mathcal{A}_f$ on t^ω. By Lemma 25, $\mathcal{A}_M$ has uncountably many accepting computations and we defined an injective map from these computations to accepting computations of $\mathcal{A}_f$. Hence, $\mathcal{A}_f$ is not countably ambiguous. Therefore, by Lemma 4, $\mathcal{A}$ is not countably ambiguous. ◀

5.2 Direction (1) ⇒ (2) of Proposition 21

▶ **Definition 28** (q-path and q-computation). *Given a Büchi tree automaton $\mathcal{A} = (Q, \Sigma, Q_I, \delta, F)$, a state $q \in Q$ and a tree $t \in L(\mathcal{A})$, we define the following:*

- *A **q-path** (of an accepting computation ϕ) is an ω-path $\pi := v_0 \dots v_i \dots$ of t such that v_0 is the root, $\phi(v_0) = q$ and there exist infinitely many nodes v_i such that $\phi(v_i) = q$.*
- *A **q-computation** is an accepting computation ϕ such that ϕ has a q-path in t.*

The next lemma reduces the question whether the cardinality of accepting computations is uncountable to the question whether the cardinality of f-computations is uncountable.

▶ **Lemma 29.** *A Büchi tree automaton $\mathcal{A} = (Q, \Sigma, Q_I, \delta, F)$ has uncountably many accepting computations on t iff there is a state $f \in F$, a node $u \in t$ and an accepting computation $\phi_0 \in ACC(\mathcal{A}, t)$ such that $\phi_0(u) = f$ and $\mathcal{A}_f$ has uncountably many f-computations on $t_{\geq u}$.*

Proof. $\Leftarrow$ direction is trivial.

$\Rightarrow$: Assume that the set $\Phi := ACC(\mathcal{A}, t)$ of accepting computations of $\mathcal{A}$ on t is uncountable. For each computation $\phi \in \Phi$ define a tree t'_ϕ by pruning the tree t as follows: for every node $v \in t$, if $\phi(v) \in F$ and ϕ has an $\phi(v)$-path on $t_{\geq v}$, prune the descendants of v. Hence, t'_ϕ is a subtree of t over the set of nodes $V_\phi := \{v \mid \forall u < v : \phi(u) \in F \to \phi \text{ has no } \phi(u)\text{-paths on } t_{\geq u}\}$. If u is a leaf of t'_ϕ, then $\phi(u) \in F$ and ϕ has an $\phi(u)$-path on $t_{\geq u}$.

Observe that t'_ϕ is finite. Otherwise, by the König Lemma, it would have an infinite branch $\pi = v_0 \dots v_i \dots$ such that $\phi(v_0) \dots \phi(v_i) \dots$ has finitely many occurrences of states from F which contradicts that ϕ is an accepting run on t.

Therefore, to each computation $\phi \in \Phi$ corresponds a finite tree t'_ϕ. The set of all possible finite trees is countable, and since there are uncountably many computations in Φ, we conclude that there is a finite tree t_0 and an uncountable set $\Phi_{t_0} \subseteq \Phi$ such that $\forall \phi \in \Phi_{t_0} : t_0 = t'_\phi$. Since there are finitely many assignments of states to the nodes of t_0, we conclude that there is a computation ϕ_0 and an uncountable set $\Phi' \subseteq \Phi_{t_0}$ such that $\forall v \in t_0 \forall \phi \in \Phi' : \phi(v) = \phi_0(v)$. For each leaf $u \in t_0$ define Φ_u as the set of restrictions Φ' on $t_{\geq u}$. Notice that the cardinality of Φ' is bounded by the product of the cardinalities of Φ_u. Hence, there is u such that Φ_u is uncountable. Each computation $\phi \in \Phi_u$ has originated from a computation with an $\phi_0(u)$-path on $t_{\geq u}$, and therefore Φ_u is the set of f-computations of $\mathcal{A}_f$ on $t_{\geq u}$ for $f = \phi_0(u)$. ◀

We are going to prove that if $\mathcal{A}$ is not countably ambiguous and has at most countable branch ambiguity and no f-ambiguous transition pattern, then it has a branching pattern. The main technical lemma uses the following definition.

▶ **Definition 30.** *Let T be a subset of nodes of a tree t. We consider T as a substructure of t with the ancestor relation. In particular, u is a **T-leaf** if $u < v$ for no $v \in T$; u is a **T-successor** of v if $u, v \in T$, $u > v$ and no T-node is between u and v; T is a **full binary subset-tree** of t, if T has a minimal node and every node of T has two T-successors.*

▶ **Lemma 31** (Main). *Assume f is a final state and there are uncountably many f-computations of $\mathcal{A}$ on t, and conditions 2(a) and 2(b) of Proposition 21 do not hold. Then, there is a full binary subset-tree X of t such that for every $u \in X$ there is an f-computation ϕ_u on t such that if $v \in X$ and $v \leq u$ then $\phi_u(v) = f$.*

The lemma is proved in the full paper, where X is constructed level by level. However, in order to carry out such a construction we need a much stronger inductive assertion. In particular, our construction implies that for every $u \in X$, $\mathcal{A}_f$ has uncountably many f-computations on $t_{\geq u}$. The next lemma is easily derived from the König Lemma.

▶ **Lemma 32.** *If T is a full binary subset-tree of t, then there is a full binary subset-tree $T' \subseteq T$ such that if v_1, v_2 are the T'-successors of u, and $\mathcal{A}_q$ accepts $t_{\geq u}$, then $\mathcal{A}_q$ has an accepting computation on $t_{\geq u}$ which passes through F on the paths of $t_{\geq u}$ from u to v_1 and from u to v_2.*

▶ **Lemma 33.** *Let $(T, \leq)$ be the full binary tree and lab be a labeling of its nodes by a finite alphabet. Then, there are $v_1, v_2 > u$ such that $v_1 \perp v_2$ and $lab(v_1) = lab(u) = lab(v_2)$.*

Proof. Choose a node u such that the cardinality of $\Sigma_{\geq u} := \{lab(w) \mid u \leq w\}$ is minimal. Then for every $w' \geq u$ and every $\sigma \in \Sigma_{\geq w}$ there is $v' \geq w'$ with $lab(v') = \sigma$. ◀

The next Lemma, together with Lemma 29, shows that if a tree automaton is uncountably ambiguous and 2(a) and 2(b) of Proposition 21 do not hold, then 2(c) holds. This implies the (1) ⇒ (2) direction of Proposition 21.

▶ **Lemma 34.** *Let $\mathcal{A}$ be a Büchi tree automaton and f be a final state of $\mathcal{A}$. Assume that there are uncountably many f-computations of $\mathcal{A}_f$ on t and conditions 2(a) and 2(b) of Proposition 21 do not hold. Then, there exist three nodes $u, v_1, v_2 \in t$ such that a branching pattern for $\mathcal{A}_f$ is realized at v_1, v_2 in $t_{\geq u}$.*

Proof. Let X be the full binary subset-tree of t, guaranteed by Lemma 31. By applying Lemma 32 on X, we obtain a full binary subset-tree $T \subseteq X$. Define a labeling of T by $lab(v) = \{\phi(v) | \phi \in ACC(\mathcal{A}_f, t)\}$ for each $v \in T$. This is a labeling by a finite alphabet. Therefore, by Lemma 33, we have nodes $v_1, v_2 > u$ such that $v_1 \perp v_2$ and $lab(u) = lab(v_1) = lab(v_2) = Q'$. We are going to define computations that realize a branching pattern over $(Q' \setminus \{f\}, f)$ at v_1, v_2 in $t_{\geq u}$.

For $i = 1, 2$, set ϕ_i to be the restriction of ϕ_{v_i} to $t_{\geq u}$, where ϕ_{v_i} is as in Main Lemma. This gives immediately that $\phi_i \in ACC(\mathcal{A}_f, t_{\geq u})$ and $\phi_1(v_1) = \phi_2(v_2) = f$. Since A_f is ambiguous, by Lemma 14(1) and the assumption that $\mathcal{A}$ has no f-ambiguous transition pattern, we obtain $\phi_1(v_2) \neq f \neq \phi_2(v_1)$.

By Lemma 32, for each $q \in Q' \setminus \{f\}$ there is $\phi_q \in ACC(\mathcal{A}_q, t_{\geq u})$ which visits F on the paths (in $t_{\geq u}$) from u to the children of u in T. Hence, it visits F on the paths from u to v_1 and from u to v_2. Next, observe that $\phi_q(v_1), \phi_q(v_2) \in Q'$ by the definition of labeling. We are going to show that $\phi_q(v_1) \neq f$ and $\phi_q(v_2) \neq f$. This will show that ϕ_1, ϕ_2 and ϕ_q for $q \in Q' \setminus \{f\}$ realize a branching pattern, and thus finish the proof.

Aiming for a contradiction, assume $\phi_q(v_1) = f$. There is $\phi' \in ACC(A_f, t)$ such that $\phi'(u) = q$. Let ϕ'_q be a grafting of ϕ_q on ϕ' at u. It reaches v_1 in state f. Consider the branch automaton computation from the root of t to v_1 which correspond to ϕ'_q and ϕ_{v_1}. These are different computations (since they differ at u) from f to f. Hence, $\mathcal{A}_B$ contains the forbidden pattern for countable ambiguity (see Def. 6), and by Theorem 7 we have that $\mathcal{A}_B$ is not countably ambiguous. Therefore, $\mathcal{A}$ is not countable branch ambiguous - contradiction to the assumptions of the lemma. The proof of $\phi_q(v_2) \neq f$ is similar. ◀

6 Computability of branch ambiguity and the ambiguous transition pattern

Here we describe algorithms to test the degree of ambiguity of branch automata and to test if an automaton has an ambiguous transition pattern. The following Lemma easily follows from Definition 10 of the branch automaton.

▶ **Lemma 35.** *Let $\mathcal{A}_B$ be the branch automaton of $\mathcal{A}$. Assume that $r_i \in Q^{l+1}$ for $i = 1, \ldots, k$ are runs of $\mathcal{A}_B$ on $u = (\sigma_1, d_1, S_1) \ldots (\sigma_l, d_l, S_l) \in \Sigma_B^*$. Then for $i = 1, \ldots, l$ there are $S'_i \subseteq S_i$ such that $|S'_i| \leq k$ and r_i for $i = 1, \ldots, k$ are runs of $\mathcal{A}_B$ on $u = (\sigma_1, d_1, S'_1) \ldots (\sigma_l, d_l, S'_l)$.*

A letter $(\sigma, d, S) \in \Sigma_B$ is called a k-state letters if S has at most k states. If $\mathcal{A}$ has n states, then the alphabet Σ_B of the branch automaton $\mathcal{A}_B$ might be of size $2|\Sigma| \times 2^n$, yet the number of k-state letter is bounded by $2|\Sigma| \times \sum_{i=1}^{k} \binom{n}{i} \leq 2|\Sigma| n^k$. To test whether a k-state letter (σ, d, S) is in Σ_B, we can check whether the intersection of the tree languages $L(\mathcal{A}_q)$ for $q \in S$ is non-empty. This can be done in $O(n^k)$ time (checking non-emptiness of the intersection Büchi language). We denote by $\mathcal{A}_B^{(k)}$ the restriction of the branch automaton $\mathcal{A}_B$ to the k state letters. It is computable in $O(|\mathcal{A}|^k)$ time from $\mathcal{A}$.

Now, we are ready to prove Lemma 16 and Proposition 12.

Proof of Lemma 16. For each p_1 and p_2, items 1 and 2 of Definition 13 can be tested in polynomial time. There is a q-ambiguous pattern, if there is a run of $\mathcal{A}_B^{(1)}$ from q to p_1 and from p_2 to q for a pair p_1 and p_2 which passed the test. This is reduced to the reachability problem. ◀

Proof Sketch of Proposition 12. The degree of ambiguity of ω-word Büchi automata is characterized by the forbidden patterns in Theorem 7. Each of these patterns involves conditions on at most three runs on the same word and can be tested for an automaton $\mathcal{B}$ in polynomial time. Hence, by Lemma 35, $\mathcal{A}_B$ has these patterns iff $\mathcal{A}_B^{(3)}$ has them, and can be tested in time $p(|\mathcal{A}_B^{(3)}|)$ for a polynomial p. Since $\mathcal{A}_B^{(3)}$ is computable in polynomial time from $\mathcal{A}$, we obtain a polynomial time algorithm. ◀

7 Computability of a branching pattern

Here we prove Proposition 22. In Sect. 7.1 we show that if $\mathcal{A}$ has a branching pattern, then it has a branching pattern over (Q, f), where Q has at most two states. Sect. 7.2 presents a polynomial time algorithm to verify if $\mathcal{A}$ has a branching pattern with at most two states.

7.1 Reduction to small branching patterns

In Sect. 5.1 we assigned to each branching pattern M a tree automaton $\mathcal{A}_M$ over the unary alphabet. This automaton is almost deterministic, in the sense that from every state $q \neq f$ it has a unique transition and it does not enter f. Hence, $\mathcal{A}_M$ has a unique accepting computation from every $q \neq f$. From f it has two transitions. A transition function defined next will help to describe properties of accepting computations of $\mathcal{A}_M$.

▶ **Definition 36** (Transition function of branching pattern)**.** *Let M be a branching pattern for $\mathcal{A}$ over (Q, f) with $\tau_M : Q \to Q \times Q$ and a tuple $(q_1, q_2) \in Q \times Q$. Its transition function $\delta_M : (\{f\} \cup Q) \times \{l, r\} \to Q$ is defined as follows:*

$$\delta_M(f, d) := \begin{cases} q_1 & \text{if } d = l \\ q_2 & \text{if } d = r \end{cases} \quad \text{;For } p \neq f, \text{with } \tau_M(p) = (q', q'') : \quad \delta_M(p, d) := \begin{cases} q' & \text{if } d = l \\ q'' & \text{if } d = r \end{cases}$$

▶ **Lemma 37.**

1. *Let $q \neq f$ and ϕ^q be a (unique) accepting computation of $\mathcal{A}_M$ (on the tree over unary alphabet) from q. Then $\phi^q(w) = \delta_M(q, w)$ for every $w \in \{l, r\}^*$.*
2. *Let $s := d_1 \ldots d_k \in \{l, r\}^+$, and let ϕ_s be an accepting computation of $\mathcal{A}_M$ from f such that $\phi_s(d_1 \ldots d_i) = f$ for every $i \leq k$. Then for every $w \in \{l, r\}^*$: (a) if $d_i = l$ then $\phi_s(d_1 \ldots d_{i-1} r w) = \delta_M(f, lw)$ and (b) if $d_i = r$ then $\phi_s(d_1 \ldots d_{i-1} l w) = \delta_M(f, rw)$.*

▶ **Lemma 38.** *Assume a branching pattern M for $\mathcal{A}$ over (Q, f) is realized. Let $l_M(q) := \delta_M(q, l)$ and $r_M(q) := \delta_M(q, r)$ for all $q \in Q$. Then:*

1. *If l_M maps Q to $Q_0 \subsetneq Q$, then a branching pattern for $\mathcal{A}$ over (Q_0, f) is realized. Dually, if r_M maps Q to $Q_1 \subsetneq Q$ then a branching pattern for $\mathcal{A}$ over (Q_1, f) is realized.*
2. *If l_M and r_M are bijections, then there is Q' such that $|Q'| \leq 2$ and a branching pattern for $\mathcal{A}$ over (Q', f) is realized.*
3. *A branching pattern for $\mathcal{A}$ over (Q', f) is realized with $|Q'| \leq 2$.*

Proof. We will assume the branching pattern M for $\mathcal{A}$ over (Q, f) is realized in a tree t at nodes u, v by computations ϕ_1, ϕ_2, $\{\phi_q \mid q \in Q\}$.

(1) Assume l_M maps Q to $Q_0 \subsetneq Q$. Let $t' := (t \circ_u t) \circ_v t$. Define the following computations on t': $\phi'_1 := (\phi_1 \circ_u \phi_2) \circ_v \phi_{q_1}$, $\phi'_2 := (\phi_2 \circ_u \phi_{q_2}) \circ_v \phi_2$, and for each $q \in Q_0$ with $f_M(q) = (p_1, p_2)$ we set $\phi'_q := (\phi_q \circ_u \phi_{p_2}) \circ_v \phi_{p_1}$. Let $u' := uv$ and $v' = vv$ be two nodes of t'. By Lemma 3 we have that $\phi'_1, \phi'_2 \in ACC(\mathcal{A}_f, t)$ and $\forall q \in Q_0 : \phi'_q \in ACC(A_q, t')$. Notice that $\phi'_1(u') = \phi'_1(uv) = \phi_2(v) = f$, $\phi'_2(v') = \phi'_2(vv) = \phi_2(v) = f$, and from the construction it follows that $\phi'_1(v'), \phi'_2(u'), \phi'_q(u'), \phi'_q(v') \in \{\phi_q(v) \mid q \in Q\} = \{\delta_M(q, l) \mid q \in Q\} \subseteq Q_0$. Since ϕ_q visits F on both paths from the root to u and from the root to v, so does ϕ'_q on the path from the root to $u' = uv$ and from the root to $v' = vv$. It follows that a branching pattern for $\mathcal{A}$ over (Q_0, f) is realized in t' at u', v' by computations ϕ'_1, ϕ'_2 and $\{\phi'_q \mid q \in Q_0\}$. The proof of the dual case is symmetric.

(2) The set of bijections on a finite set is a finite group under the composition and the identity map is its identity element. If k is the cardinality of a finite group, then c^k is equal to the identity for every element c. Let $k > 0$ be such that both l_M^k and r_M^k are the identity map.

Define $t_1^u := t$, $t_1^v := t$ and $\forall i > 1$ let $t_{i+1}^u := t \circ_u t_i^u$ and $t_{i+1}^v := t \circ_v t_i^v$. Finally, construct a tree $t' := (t \circ_u t_{k-1}^u) \circ_v t_{k-1}^v$.

Let $p_1 := \delta_M(f, l^k)$ and $p_2 := \delta_M(f, r^k)$. We will show that a branching pattern for $\mathcal{A}$ over $(\{p_1, p_2\}, f)$ is realized in t' at u^k, v^k.

The following are obtained using Lemma 3 and definition of δ_M:

- **i** $\alpha_i := \underbrace{\phi_1 \circ_u (\phi_1 \circ_u (\cdots \circ_u \phi_1) \dots)}_{i \text{ times}}$ is an accepting computation of $\mathcal{A}_f$ on t_i^u. It assigns f to node u^i.
- **ii** $\beta_i := \underbrace{\phi_2 \circ_v (\phi_2 \circ_v (\cdots \circ_v \phi_2) \dots)}_{i \text{ times}}$ is an accepting computation of $\mathcal{A}_f$ on t_i^v. It assigns f to node v^i.
- **iii** Let $q_0 \in Q$ and $q_i := \delta_M(q_0, r^i)$. Then $\phi_{q_0}^{r^i} := \phi_{q_0} \circ_u (\phi_{q_1} \circ_u (\cdots \circ_u \phi_{q_{i-1}}) \dots)$ is an accepting computation of $\mathcal{A}_{q_0}$ on t_i^u, which holds $\phi_{q_0}^{r^i}(u^j) = q_j$ for $j \leq i$.
- **iv** Let $q'_0 \in Q$ and $q'_i := \delta_M(q'_0, l^i)$. Then $\phi_{q'_0}^{l^i} := \phi_{q'_0} \circ_v (\phi_{q'_1} \circ_v (\cdots \circ_v \phi_{q'_{i-1}}) \dots)$ is an accepting computation of $\mathcal{A}_{q'_0}$ on t_i^v, which holds $\phi_{q'_0}^{l^i}(v^j) = q'_j$ for $j \leq i$.

Let $q' := \phi_1(v)$ and $q'' := \phi_2(u)$. From **i** and **iv**, it follows that $\phi'_1 := (\phi_1 \circ_u \alpha_{k-1}) \circ_v \phi_{q'}^{l^{k-1}}$ is an accepting computation of $\mathcal{A}_f$ on t', such that $\phi'_1(u^k) = f$, $\phi'_1(v^k) = \delta_M(f, l^k)$ and ϕ'_1 visits F on the path from the root to v^k (as it coincides with ϕ_1 on the path from the root to v, which visits F).

Using similar arguments from **ii** and **iii**, we obtain that $\phi'_2 := (\phi_2 \circ_u \phi_{q''}^{r^{k-1}}) \circ_v \beta_{k-1}$ is an accepting computation of $\mathcal{A}_f$ on t', such that $\phi'_2(v^k) = f$, $\phi'_2(u^k) = \delta_M(f, r^k)$ and ϕ'_2 visits F on the path from the root to u^k.

In addition, from **iii** and **iv** is follows that for all $p \in Q$ with $\tau_M(p) = (p', p'')$, the computation $\phi'_p := (\phi_p \circ_u \phi'^{r^{k-1}}_{p''}) \circ_v \phi'^{l^{k-1}}_{p'}$ is an accepting computation of $\mathcal{A}_p$ on t', such that $\phi'_p(u^k) = \delta_M(p, r^k)$ and $\phi'_p(v^k) = \delta_M(p, l^k)$. By selection of k we have $\delta_M(p, l^k) = \delta_M(p, r^k) = p$ and therefore we obtain that $\phi'_p(u^k) = p = \phi'_p(v^k)$.

Take $p_1 := \delta_M(f, l^k) = \phi'_1(v^k)$ and $p_2 := \delta_M(f, r^k) = \phi'_2(u^k)$. We have $\phi'_{p_1}(u^k) = \phi'_{p_1}(v^k) = p_1$ and $\phi'_{p_2}(u^k) = \phi'_{p_2}(v^k) = p_2$, and therefore we obtain that a branching pattern for $\mathcal{A}$ over $(\{p_1, p_2\}, f)$ is realized in t' at u^k, v^k by computations ϕ'_1, ϕ'_2, ϕ'_{p_1} and ϕ'_{p_2}, as requested.

(3) Let M over (Q, f) be a realizable branching pattern for $\mathcal{A}$ such that the cardinality of Q is minimal. If either l_M or r_M is not a bijection, then by item 1, there is a realizable pattern over (Q_0, f), where $|Q_0| < |Q|$. Hence, both l_M and r_M are bijections. Therefore, by item 2 and minimality of $|Q|$, we obtain $|Q| \leq 2$. ◀

7.2 Small branching patterns are in P

For every t over Σ and $u_1, u_2 \in t$, define $t' := t(u_1, u_2)$ over $\Sigma' := \Sigma \times \Sigma_{u_1} \times \Sigma_{u_2}$ with $\Sigma_{u_i} := \{0, 1\}$, such that the projection of t' on Σ is t and the projection of t' on Σ_{u_i} is a tree t_{u_i} with $\sigma_{t_{u_i}}(w) = 1$ iff $w = u_i$ for $i = 1, 2$.

It is easy to construct Büchi automata over Σ' with the following properties in $O(|\mathcal{A}|)$ time.

- An automaton $\mathcal{A}_{nodes}$ which accepts t' iff $t' = t(u_1, u_2)$ and $u_1 \perp u_2$.
- An automaton $\mathcal{A}_{q,q_1,q_2}$ which accepts t' iff $t' = t(u_1, u_2)$ and there exists a computation $\phi \in ACC(\mathcal{A}_q, t)$ with $\phi(u_1) = q_1$, $\phi(u_2) = q_2$ and ϕ visits an accepting state on both paths from the root to u_1 and from the root to u_2.
- An automaton $\mathcal{A}^L_{f,q}$ which accepts t' iff $t' = t(u_1, u_2)$ and there exists a computation $\phi \in ACC(\mathcal{A}_f, t)$ such that $\phi(u_1) = f$ and $\phi(u_2) = q$.
- An automaton $\mathcal{A}^R_{f,q}$ which accepts t' iff $t' = t(u_1, u_2)$ and there exists a computation $\phi \in ACC(\mathcal{A}_f, t)$ such that $\phi(u_1) = q$ and $\phi(u_2) = f$.

By Lemma 38, $\mathcal{A}$ has a realizable branching pattern iff there exists a realizable branching pattern over (Q, f), $\tau_M : Q \to Q$, $(q_1, q_2) \in Q \times Q$ with $|Q| \leq 2$. For each such branching pattern we define:

$$L_M := L(\mathcal{A}_{nodes}) \cap \bigcap_{(p,p_1,p_2)|p\in Q, \tau_M(p)=(p_1,p_2)} L(\mathcal{A}_{p,p_1,p_2}) \cap L(\mathcal{A}^L_{f,q_1}) \cap L(\mathcal{A}^R_{f,q_2})$$

By construction of the automata we have that the branching pattern M is realizable iff $L_M \neq \emptyset$. This could be verified in polynomial time in $|Q_{\mathcal{A}}|$, as this is an intersection of at most five Büchi tree languages. Since the number of such patterns is polynomial in $|Q_{\mathcal{A}}|$ we obtain a polynomial time algorithm.

8 Conclusion and Further Results

We proved that the degree of ambiguity of Büchi automata on infinite trees is in PTIME. The Büchi acceptance conditions on trees are less expressive than parity, Rabin, Street and Muller conditions. Unfortunately, we have

▶ **Proposition 39.**

- *The problems of deciding whether a parity tree automaton is not unambiguous/boundedly ambiguous/finitely ambiguous are co-NP complete*
- *The problem of deciding weather a parity tree automaton is not countably ambiguous is co-NP hard.*

It is still unknown if the problem of deciding weather a parity tree automaton is not countably ambiguous is in co-NP, although we believe it is indeed the case.

The degree of ambiguity of a regular language is defined in a natural way. E.g., a language is k-ambiguous if it is accepted by a k-ambiguous automaton and no $k-1$-ambiguous automaton accepts it. Over finite words and finite trees every regular language is accepted by a deterministic automaton. Over ω-words every regular language is accepted by an unambiguous automaton [1]. Over infinite tree there are ambiguous languages [3]. We can show that over infinite trees there is a hierarchy of degrees of ambiguity [11]:

▶ **Proposition 40.** *There are k-ambiguous languages for every $k \in \mathbb{N}$. There are finitely, countably and uncountable ambiguous languages.*

We plan to investigate whether the degree of ambiguity of infinite tree language is decidable.

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 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.

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

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

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

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

8 Christof Löding and Anton Pirogov. On Finitely Ambiguous Büchi Automata. In *Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings*, pages 503–515, 2018. doi:10.1007/978-3-319-98654-8_41.

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

10 Alexander Rabinovich. Complementation of Finitely Ambiguous Büchi Automata. In *International Conference on Developments in Language Theory*, pages 541–552. Springer, 2018.

11 Alexander Rabinovich and Doron Tiferet. Degree of Ambiguity for Tree Automata and Tree Languages, forthcoming.

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

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

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