On Degrees of Ambiguity for Büchi Tree Automata

Alexander Rabinovich, Doron Tiferet*

The Blavatnik School of Computer Science, Tel Aviv University, Israel

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.

Keywords: Büchi automata, automata ambiguity
2010 MSC: 68Q45

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 [1, 2] 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 [2]. 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 [3, 4].

Some problems are easier for unambiguous than for nondeterministic automata. As shown by Stearns and Hunt [5], the equivalence and inclusion problems for unambiguous automata are in polynomial time, while these problems are PSPACE-complete for nondeterministic automata.

*Corresponding author
Email addresses: rabinoa@tauex.tau.ac.il (Alexander Rabinovich), sdoron5.t2@gmail.com (Doron Tiferet)

Figure 1: Finitely ambiguous and 2-ambiguous automata

The complexity of basic regular operations on languages represented by unambiguous finite automata was investigated in [6], and tight upper bounds on state complexity of intersection, concatenation, and many other operations on languages represented by unambiguous automata were established.

It is well-known that the tight bound on the state complexity of the complementation of nondeterministic automata is 2^n. In [6], it was shown that the complement of the language accepted by an n-state unambiguous automaton is accepted by an unambiguous automaton with $2^{0.79n+\log n}$ states.

Many other notions of ambiguity were suggested and investigated. A recent paper [2] 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 (see Figure 1 for examples).

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

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 [7] we proved that the problem whether a parity tree automaton is ambiguous is co-NP-complete.

Related Works. Weber and Seidl [8] 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.

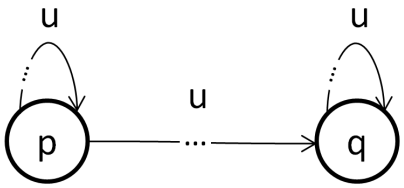

Figure 2: Forbidden pattern for bounded ambiguity

Forbidden Pattern for Bounded Ambiguity: There are distinct useful[1] 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 (see Figure 2).

Weber and Seidl [8] proved that an NFA is not boundedly ambiguous iff it contains the forbidden pattern for bounded ambiguity. This pattern is testable in polynomial time; hence, it can be decided in polynomial time whether the degree of ambiguity of an NFA is bounded.

Seidl [9] 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 [10] and Rabinovich [11] 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 [12], 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 conclude

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

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.

An extended abstract of this paper was published in [13].

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. 4-9. The last section presents further problems.

2. Preliminaries

In Sections 2.1 and 2.2, we recall standard terminology and notations about trees and automata [14, 15]. In Section 2.3 some simple lemmas are stated, and it is proved that there is a polynomial time algorithm that checks whether a Büchi tree automaton is unambiguous. In Section 2.4 we recall a forbidden pattern characterization of degrees of ambiguity of automata on ω-words [10, 11].

2.1. Trees

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

We define a tree order "$\leq$" as a partial order such that $\forall u, v \in \{l, r\}^* : u \leq v$ iff u is a prefix of v. Nodes u and v are incomparable - denoted by $u \perp v$ - if neither $u \leq v$ nor $v \leq u$; a set U of nodes is an **antichain**, if its elements are incomparable with each other.

We say that an infinite sequence $\pi = v_0, v_1, \ldots$ is a **tree branch** if $v_0 = \epsilon$ and $\forall i \in \mathbb{N} : v_{i+1} = v_i \cdot l$ or $v_{i+1} = v_i \cdot r$. We sometimes refer to π as the set $\{v_i \mid i \in \mathbb{N}\}$. It will be clear from the context whether π is interpreted as a set or a sequence.

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

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

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

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

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

Finite Tree. A finite tree is a finite set $U \subseteq \{l, r\}^*$ which is closed under prefix relation.

Node Depth. The depth of a node $u \in \{l, r\}^*$, denoted $Depth(u)$, is defined as the length of u. For a finite set of nodes $U \subseteq \{l, r\}^*$ we define $Depth(U) := \max\{Depth(u) \mid u \in U\}$.

2.2. Automata

2.2.1. ω-word Automata

Büchi ω-word Automata (BWA). A BWA $\mathcal{A}$ is a tuple $(Q_\mathcal{A}, \Sigma, Q_I, \delta, F)$ where Σ is a finite alphabet, $Q_\mathcal{A}$ is a finite set of states, $Q_I \subseteq Q_\mathcal{A}$ is a set of initial states, $\delta \subseteq Q_\mathcal{A} \times \Sigma \times Q_\mathcal{A}$ is a transition relation, and $F \subseteq Q_\mathcal{A}$ is a set of final states. A run of $\mathcal{A}$ on a ω-word $y = a_0 a_1 \dots$ is an infinite sequence $\rho = q_0 q_1 \dots$ such that $q_0 \in Q_I$, and $(q_i, a_i, q_{i+1}) \in \delta$ for all $i \in \mathbb{N}$. We say that ρ is accepting if there is a state $f \in F$ which appears infinitely often in ρ.

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

2.2.2. Infinite Tree Automata

Büchi Tree Automata (BTA). A BTA is a tuple $(Q_\mathcal{A}, \Sigma, Q_I, \delta, F)$ where Σ is a finite alphabet, $Q_\mathcal{A}$ is a finite set of states, $Q_I \subseteq Q_\mathcal{A}$ is a set of initial states, $\delta \subseteq Q_\mathcal{A} \times \Sigma \times Q_\mathcal{A} \times Q_\mathcal{A}$ is a transition relation, and $F \subseteq Q_\mathcal{A}$ is a set of final states. A computation of $\mathcal{A}$ on a tree t is a function $\phi : \{l, r\}^* \to Q_\mathcal{A}$ such that $\phi(\epsilon) \in Q_I$, and $\forall v \in \{l, r\}^* : (\phi(v), t(v), \phi(v \cdot l), \phi(v \cdot r)) \in \delta$. We say that ϕ is accepting if for each tree branch $\pi = v_0, v_1, \dots$ there is a state $f \in F$ such that the sequence $\phi(v_0), \phi(v_1), \dots$ contains infinitely many occurrences of f. When $\mathcal{A}$ is known from the context, we will omit the subscript, and refer to the set of states as Q.

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

Given an automaton (either a BWA or a BTA) $\mathcal{A} = (Q_\mathcal{A}, \Sigma, Q_I, \delta, F)$ and a state $q \in Q$, we denote by $\mathcal{A}_q$ the automaton $(Q_\mathcal{A}, \Sigma, \{q\}, \delta, F)$ which is obtained from $\mathcal{A}$ by replacing the set of initial states Q_I with $\{q\}$.

A state $q \in Q$ of a BTA $\mathcal{A}$ is called useful if there is a tree $t \in L(\mathcal{A})$, a computation $\phi \in ACC(\mathcal{A}, t)$, and a node $v \in \{l, r\}^*$ such that $\phi(v) = q$. For BTA, it is computable in polynomial time whether a state is useful. Hence, throughout the paper we will assume all states of BTA to be useful.

Degree of Ambiguity of an Automaton. We denote by $|X|$ the cardinality of a set X. An automaton $\mathcal{A}$ is k-ambiguous if $|ACC(\mathcal{A}, t)| \leq k$ for all $t \in L(\mathcal{A})$; $\mathcal{A}$ is unambiguous if it is 1-ambiguous; $\mathcal{A}$ is boundedly ambiguous if there is $k \in \mathbb{N}$ such that $\mathcal{A}$ is k-ambiguous; $\mathcal{A}$ is finitely ambiguous if $ACC(\mathcal{A}, t)$ is finite for all t; $\mathcal{A}$ is countably ambiguous if $ACC(\mathcal{A}, t)$ is countable for all t.

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

Example 2 *Consider the BTA $\mathcal{A}_i = (Q_i, \Sigma_i, Q^i_I, \delta_i, F_i)$ for $1 \leq i \leq 4$.*

1. *$Q_1 = Q^1_I = F_1 = \{q\}$, $\Sigma_1 = \{1\}$, and $\delta_1 = \{(q, 1, q, q)\}$. $\mathcal{A}_1$ is deterministic and therefore unambiguous.*
2. *$Q_2 = Q^2_I = F_2 = \{q_1, q_2\}$, $\Sigma_2 = \{1\}$, and $\delta_2 = Q_2 \times \Sigma_2 \times Q_2 \times Q_2$. $\mathcal{A}_2$ is uncountably ambiguous.*

3. $Q_3 = \{q, f\}$, $Q_I^3 = \{q\}$, $F_3 = \{f\}$, $\Sigma_3 = \{1\}$, *and* $\delta_3 = \{(q, 1, p_1, p_2) \mid p_1, p_2 \in Q_3\} \cup \{(f, 1, f, f)\}$. $\mathcal{A}_3$ *is countably ambiguous.*
4. $Q_4 = Q_I^4 = F_4 = \{q_1, q_2\}$, $\Sigma_4 = \{1, 2, 3\}$, *and* $\delta_4 = \{(q_1, a, q_1, q_1) \mid a \neq 1\} \cup \{(q_2, a, q_2, q_2) \mid a \neq 2\}$. *A tree* t *is accepted by* $\mathcal{A}_4$ *iff* t *has no node labeled by 1 or* t *has no node labeled by* 2. *An accepting run of* $\mathcal{A}_4$ *is either assigns* q_1 *to all nodes or* q_2 *to all nodes. Hence,* $\mathcal{A}_4$ *has two accepting runs on the tree with all nodes labeled by* 3, *and at most one accepting run on other trees. Therefore,* $\mathcal{A}_4$ *is 2-ambiguous. Moreover, in [16] we proved that every automaton that accepts* $L(\mathcal{A}_4)$ *has at least two accepting runs on the tree with all nodes labeled by* 3.

A computation ϕ of $\mathcal{A}$ on a Σ-labeled tree t can be considered as a $Q_{\mathcal{A}}$-labeling of t. Given two computations ϕ, ϕ' and a node $v \in \{l, r\}^*$, the grafting of ϕ' on v in ϕ (denoted by $\phi \circ_v \phi'$) is defined as for the corresponding $Q_{\mathcal{A}}$-labeled trees.

2.3. Automata Properties

We often use implicitly the following simple Lemma.

Lemma 3 (Grafting) *Let* $\mathcal{A}$ *be an automaton,* t, t_1 *trees,* $v \in \{l, r\}^*$, *and* $\phi \in ACC(\mathcal{A}, t)$, *and* $\phi_1 \in ACC(\mathcal{A}_q, t_1)$. *If* $\phi(v) = q$, *then* $\phi \circ_v \phi_1$ *is an accepting computation of* $\mathcal{A}$ *on* $t \circ_v t_1$.

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

Lemma 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 of the lemma in the rest of this subsection.

Lemma 5 *It is computable in polynomial time whether a BTA is unambiguous.*

We first prove the following lemma:

Lemma 6 *Let* $\mathcal{A} = (Q, \Sigma, Q_I, \delta, F)$ *be a BTA, and assume all states in* Q *are useful. Then* $da(\mathcal{A}) > 1$ *iff at least one of the following holds:*

- *There are states* $p, q \in Q_I$ *such that* $L(\mathcal{A}_p) \cap L(\mathcal{A}_q) \neq \emptyset$, *or*
- *There are two different transitions* $(q, a, q_1, q_2), (q, a, q_1', q_2') \in \delta$ *such that* $L(\mathcal{A}_{q_1}) \cap L(\mathcal{A}_{q_1'}) \neq \emptyset$ *and* $L(\mathcal{A}_{q_2}) \cap L(\mathcal{A}_{q_2'}) \neq \emptyset$.

Proof $\Rightarrow$: *Assume that* $da(\mathcal{A}) > 1$. *Therefore, there is a tree* $t \in L(\mathcal{A})$ *such that* $|ACC(\mathcal{A}, t)| > 1$. *Let* $\phi_1, \phi_2 \in ACC(\mathcal{A}, t)$ *such that* $\phi_1 \neq \phi_2$, *and let* $v \in \{l, r\}^*$ *be a node of minimal depth such that* $\phi_1(v) \neq \phi_2(v)$. *If* $v = \epsilon$, *then* $\phi_1(v), \phi_2(v) \in Q_I$ *and* $t \in L(\mathcal{A}_{\phi_1(v)}) \cap L(\mathcal{A}_{\phi_2(v)})$ *- hence, item (1) holds. Otherwise, let* v' *be the parent node of* v. *Notice that* ϕ_1 *and* ϕ_2 *use different transitions* $(q, a, q_1, q_2), (q, a, q_1', q_2') \in \delta$ *from* v'. *if* v *is the left child of* v' *then* $q_1 \neq q_1'$, *and otherwise* $q_2 \neq q_2'$. ϕ_1 *and* ϕ_2 *are both accepting and therefore* $t_{\geq v \cdot l} \in L(\mathcal{A}_{q_1}) \cap L(\mathcal{A}_{q_1'})$ *and* $t_{\geq v \cdot r} \in L(\mathcal{A}_{q_2}) \cap L(\mathcal{A}_{q_2'})$, *and we obtain* $L(\mathcal{A}_{q_1}) \cap L(\mathcal{A}_{q_1'}) \neq \emptyset$ *and* $L(\mathcal{A}_{q_2}) \cap L(\mathcal{A}_{q_2'}) \neq \emptyset$ *- hence, item (2) holds.*

$\Leftarrow$: If there are $p, q \in Q_I$ such that $L(\mathcal{A}_p) \cap L(\mathcal{A}_q) \neq \emptyset$ then there is a tree $t \in L(\mathcal{A}_p) \cap L(\mathcal{A}_q)$. Therefore, there are two computations $\phi_1, \phi_2 \in ACC(\mathcal{A}, t)$ such that $\phi_1(\epsilon) = p$ and $\phi_2(\epsilon) = q$, and we obtain $da(\mathcal{A}) \geq |ACC(\mathcal{A}, t)| \geq |\{\phi_1, \phi_2\}| = 2$. Otherwise, there are two different transitions $(q, a, q_1, q_2), (q, a, q_1', q_2') \in \delta$ such that $L(\mathcal{A}_{q_1}) \cap L(\mathcal{A}_{q_1'}) \neq \emptyset$ and $L(\mathcal{A}_{q_2}) \cap L(\mathcal{A}_{q_2'}) \neq \emptyset$. Therefore, there are $t_1 \in L(\mathcal{A}_{q_1}) \cap L(\mathcal{A}_{q_1'})$ and $t_2 \in L(\mathcal{A}_{q_2}) \cap L(\mathcal{A}_{q_2'})$, and computations $\phi_{q_1} \in ACC(\mathcal{A}_{q_1}, t_1), \phi_{q_1'} \in ACC(\mathcal{A}_{q_1'}, t_1), \phi_{q_2} \in ACC(\mathcal{A}_{q_2}, t_2)$, and $\phi_{q_2'} \in ACC(\mathcal{A}_{q_2'}, t_2)$. Since all states of $\mathcal{A}$ are useful, there is a tree $t \in L(\mathcal{A})$, a computation $\phi \in ACC(\mathcal{A}, t)$, and a node $v \in \{l, r\}^*$ such that $\phi(v) = q$. Let $t' := (t \circ_{v \cdot l} t_1) \circ_{v \cdot r} t_2$, and define two computations $\phi' := \phi \circ_{v \cdot l} \phi_{q_1} \circ_{v \cdot r} \phi_{q_2}$ and $\phi'' := \phi \circ_{v \cdot l} \phi_{q_1'} \circ_{v \cdot r} \phi_{q_2'}$. By grafting lemma, $\phi', \phi'' \in ACC(\mathcal{A}, t')$. Since $(q, a, q_1, q_2) \neq (q, a, q_1', q_2')$, we conclude that $q_1 \neq q_1'$ or $q_2 \neq q_2'$. Therefore $\phi'(v \cdot l) \neq \phi''(v \cdot l)$ or $\phi'(v \cdot r) \neq \phi''(v \cdot r)$, and we obtain $\phi' \neq \phi''$. We conclude that $|ACC(\mathcal{A}, t')| \geq |\{\phi', \phi''\}| = 2$ and therefore $da(\mathcal{A}) \geq 2$, as requested. □

Lemma 7 *Let $\mathcal{A}$ and $\mathcal{B}$ be two BTA. Then the emptiness of $L(\mathcal{A}) \cap L(\mathcal{B})$ is decidable in polynomial time.*

Proof *Given two BTA $\mathcal{A}$ and $\mathcal{B}$, it is well-known that there is a BTA $\mathcal{C}$ such that $L(\mathcal{C}) = \emptyset$ iff $L(\mathcal{A}) \cap L(\mathcal{B}) = \emptyset$ and $\mathcal{C}$ can be constructed in $O(|\mathcal{A}| \cdot |\mathcal{B}|)$ time. Since the emptiness of BTA could be determined in polynomial time, that concludes the proof.* □

Now, we are ready to prove Lemma 5. Since the number of pairs of states in Q is polynomial in $\mathcal{A}$, it follows from Lemma 6 that it is sufficient to show that the emptiness of $L(\mathcal{A}_q) \cap L(\mathcal{A}_p)$ can be decided in polynomial time for all $p, q \in Q$. The latter follows from Lemma 7, and that concludes our proof.

2.4. Degree of Ambiguity for Automata on ω-words

The next definition and theorem are taken from [10, 11]. They provide a forbidden pattern characterization of degrees of ambiguity of automata on ω-words.

Definition 8 (Forbidden pattern for BWA) *Let $\mathcal{B}$ be a BWA 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 from q to f and a run of $\mathcal{B}_f$ on u from f to f (see Figure 3).*

Theorem 9 *Let $\mathcal{B}$ be a BWA.*

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.*

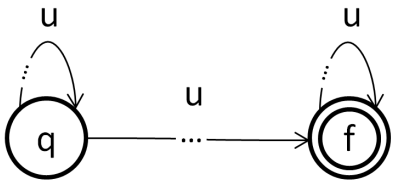

Figure 3: Forbidden pattern for finite ambiguity of BWA

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

Definition 11 (Branch ambiguity) *$\mathcal{A}$ is at most k branch-ambiguous if $|ACC(\mathcal{A}, t, \pi)| \leq k$ for every t and branch π. $\mathcal{A}$ is boundedly branch ambiguous if it is at most n branch ambiguous for some n. $\mathcal{A}$ is finitely (respectively, countably) branch ambiguous if $|ACC(\mathcal{A}, t, \pi)|$ is finite (respectively, countable) for every t and π.*

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

Definition 12 (Branch automaton) *For a BTA $\mathcal{A} = (Q, \Sigma, Q_I, \delta, F)$, the corresponding branch automaton $\mathcal{A}_B$ is an BWA $(Q, \Sigma_B, Q_I, \delta_B, F)$, 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 $\exists p \in S : (q, \sigma, (p, q')) \in \delta$.*

The following lemma, which is proved in Subsection 4.1, states the connection between branch ambiguity and ambiguity of branch automaton.

Lemma 13 *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).*

In Sect. 8 we will show:

Proposition 14 (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 15 (Ambiguous transition pattern) *Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, F)$ be a BTA with a 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 $L(\mathcal{A}_{q_1}) \cap L(\mathcal{A}_{q_2}) \neq \emptyset$ (see Figure 4), 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.*

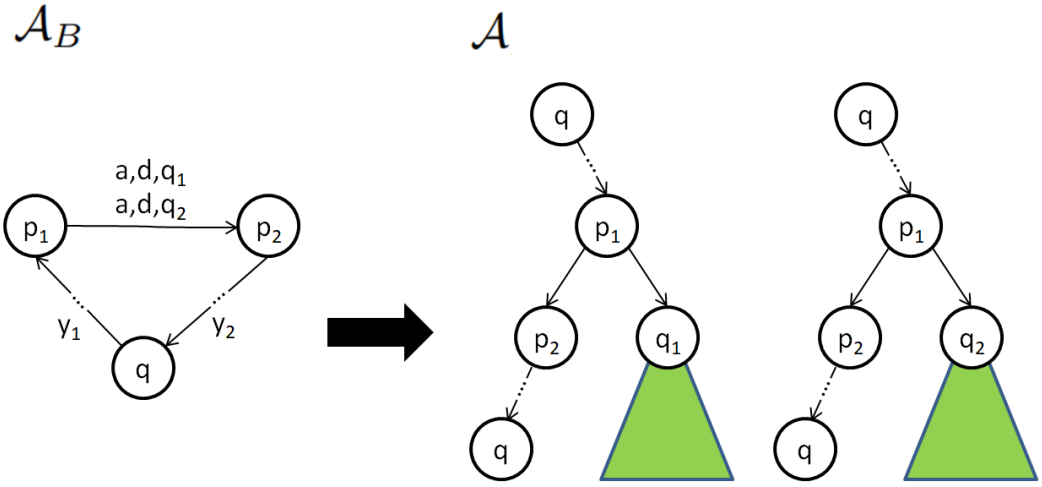

Figure 4: Ambiguous transition pattern. Notice that the green tree is accepted from both q_1 and q_2.

Lemma 16 *(1) If $\mathcal{A}$ has an ambiguous transition pattern then its ambiguity degree is not bounded. (2) If $\mathcal{A}$ has an f-ambiguous transition pattern (for a final state f) then its ambiguity degree is not countable.*

Lemma 16 is proven in Subsection 4.2. The next lemma, which deals with the computability of the transition pattern, is proven in Sect. 8.

Lemma 17 *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. Their proofs are presented in Sect. 5 and 6.

Proposition 18 (Bounded ambiguity) *The following are equivalent:*

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

Proposition 19 (Finite ambiguity) *The following are equivalent:*

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

The above characterizations of bounded and finite ambiguity are based on properties of $\mathcal{A}_B$. In order to characterize countable ambiguity we first introduce branching patterns for $\mathcal{A}$ (which are not captured by $\mathcal{A}_B$).

Definition 20 (A branching pattern for $\mathcal{A}$ over (R, f)) *Let $\mathcal{A}$ be a BTA, f a final state of $\mathcal{A}$, and $R \subseteq Q_{\mathcal{A}} \setminus \{f\}$, where $Q_{\mathcal{A}}$ are the states of $\mathcal{A}$. A branching pattern M for $\mathcal{A}$ over (R, f) is a function $\tau_M : R \to R \times R$ and a tuple $(q_1, q_2) \in R \times R$.*

Definition 21 (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 (R, f) is realized in t at u, v by computations ϕ_1, ϕ_2, $\{\phi_q \mid q \in R\}$ iff the following hold:*

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 $q \in R$: $\phi_q \in ACC(\mathcal{A}_q, t)$, $\tau_M(q) = (\phi_q(v), \phi_q(u))$ and ϕ_q visits a final state on both paths from the root of t to u and from the root of t to v.*

In Sects. 7 and 9 we prove the next two propositions. Their proofs are more complicated than the proofs of Propositions 18 and 19.

Proposition 22 (Countable ambiguity) *The following are equivalent:*

1. *A BTA $\mathcal{A}$ is not countably ambiguous.*
2. *At least one of the following conditions holds:*
   (a) *$\mathcal{A}$ is not countably 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 23 *It is computable in polynomial time whether there is a realizable branching pattern for a BTA $\mathcal{A}$.*

Theorem 24 (Main) *It is computable in polynomial time whether a BTA is unambiguous, bounded ambiguous, finitely ambiguous, or countably ambiguous.*

Proof *For unambiguity - by Lemma 5. For bounded ambiguity by Proposition 18, Lemma 17, and Proposition 14. For finite ambiguity by Proposition 19, Lemma 17, and Proposition 14. For countable ambiguity by Propositions 22, 14, 23 and Lemma 17.* □

Road map of the proofs: In Sect. 4 we prove Lemmas 13 and 16, and present a couple of useful lemmas. Sect. 5, Sect. 6, and Sect. 7 deal with structural characterizations of bounded, finite, and countable ambiguity of BTA and prove Propositions 18, 19 and 22, respectively. Sect. 8 deals with computability of branch ambiguity and of the ambiguous transition pattern, and Lemma 17 and Proposition 14 are proved there. Sect. 9 deals with computability of a branching pattern and proves Proposition 23.

4. Ambiguous Transition Pattern and Branch Ambiguity

In the first subsection we prove Lemma 13, and in the second subsection we prove Lemma 16. In the last subsection we prove two useful lemmas which provide sufficient conditions for an ambiguous pattern and for uncountable branch ambiguity.

4.1. Proof of Lemma 13

Recall that Lemma 13 states that the branch ambiguity of a BTA $\mathcal{A}$ is the same as the ambiguity of the corresponding branch automaton $\mathcal{A}_B$.

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

Lemma 25 *Let $t \in L(\mathcal{A})$, and let $\pi = v_0, v_1, \ldots$ be a tree branch. Then there exists $y \in L(\mathcal{A}_B)$ such that $ACC(\mathcal{A}, t, \pi) \subseteq ACC(\mathcal{A}_B, y)$.*

Proof *Let $y = (a_1, d_1, S_1) \ldots (a_i, d_i, S_i) \ldots$ be a word over the alphabet Σ_B, such that:*

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

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

Lemma 26 *Let $\mathcal{A}$ be a BTA with the corresponding branch automaton $\mathcal{A}_B$. If $y = (a_1, d_1, S_1) \ldots (a_i, d_i, S_i) \ldots$ is an ω-word over the alphabet Σ_B such that $y \in L(\mathcal{A}_B)$ then there exist a tree $t \in L(\mathcal{A})$ and a branch $\pi = v_0, v_1, \ldots$ such that:*

- *$t(v_i) = a_{i+1}$*
- *v_{i+1} is the left child of v_i iff $d_i = l$*

- *For each run $\rho \in ACC(\mathcal{A}_B, y)$ there is a computation $\phi \in ACC(\mathcal{A}, t)$ such that $\phi(\pi) = \rho$.*

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

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

We define an infinite Σ-labeled tree t by $t(u) := \begin{cases} a_{i+1} & \exists i : u = v_i \\ t_{i+1}(w) & \exists i : u = v'_i \cdot w \end{cases}$
(see Figure 5 for an illustration).

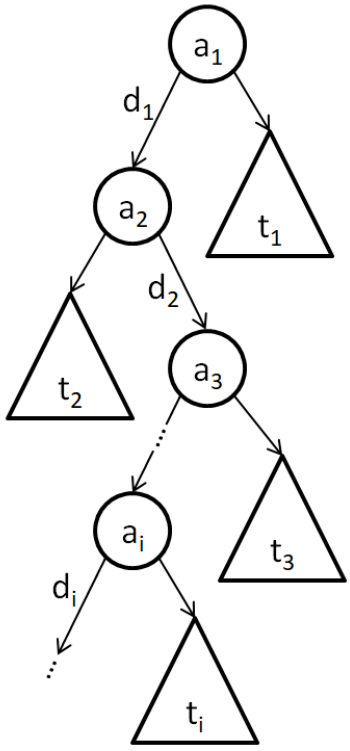

Figure 5: The tree t constructed in the proof of Lemma 26

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

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

It is easy to see that ϕ is a computation of $\mathcal{A}$ on t. We will show that ϕ is accepting. For each tree branch π', if $\pi' = \pi$ then $\phi(\pi') = \phi(\pi) = \rho$ and since $\rho \in ACC(\mathcal{A}_B, y)$ we conclude that $\phi(\pi')$ has infinitely many occurrences of states in F. Otherwise, by

the definition of t, there is $i \in \mathbb{N}$ such that $v'_i \in \pi'$. By the definition of ϕ we obtain $\phi(u) = \phi_i(w)$ for all nodes $u = v'_i \cdot w$, and since ϕ_i is accepting we conclude that $\phi(\pi')$ has infinitely many occurrences of states in F. Hence, $\phi \in ACC(\mathcal{A}, t)$ as requested. □

We are now ready to prove Lemma 13.

⇒: By Lemma 25, for each tree $t \in L(\mathcal{A})$ and a tree branch π there is an ω-word $y \in L(\mathcal{A}_B)$ such that $ACC(\mathcal{A}, t, \pi) \subseteq ACC(\mathcal{A}_B, y)$. Therefore, if $\mathcal{A}$ is not boundedly (respectively, finitely, countably) branch ambiguous then $\mathcal{A}_B$ is not boundedly (respectively, finitely, countably) ambiguous.

⇐: By Lemma 26, for each $y \in L(\mathcal{A}_\mathcal{B})$ there is a tree $t \in L(\mathcal{A})$ and a tree branch π such that $ACC(\mathcal{A}_B, y) \subseteq ACC(\mathcal{A}, t, \pi)$. Therefore, if $\mathcal{A}_B$ is not boundedly (respectively, finitely, countably) ambiguous then $\mathcal{A}$ is not boundedly (respectively, finitely, countably) branch ambiguous.

4.2. Proof of Lemma 16

Fix a BTA $\mathcal{A}$, and let $\mathcal{A}_B = (Q, \Sigma_B, Q_I, \delta_B, F)$ be the corresponding branch automaton of $\mathcal{A}$. By the definition of q-ambiguous transition pattern, there exist $p'_1, p'_2 \in Q$ and $z_1 \in \Sigma_B^*$, $z_2 \in \Sigma_B^+$ such that there is a run ρ_1 of $(\mathcal{A}_B)_q$ on z_1 from q to p'_1, and a run ρ_2 of $(\mathcal{A}_B)_{p'_2}$ on z_2 from p'_2 to q.

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

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

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

(1) We will show that for each $k \in \mathbb{N}$ there is a tree $t_k \in L(\mathcal{A})$ such that $|ACC(\mathcal{A}, t_k)| \geq 2^k$.

Denote by ρ' the run ρ without the last state. By the definition of ρ we conclude that $(\rho')^k \cdot q$ is a run of $\mathcal{A}_B$ on y^k from q to q.

All states in Q are useful and therefore there is an ω-word $\hat{y} \in L((\mathcal{A}_B)_q)$ and an accepting run $\hat{\rho} \in ACC((\mathcal{A}_B)_q, \hat{y})$. We conclude that $(\rho')^k \cdot \hat{\rho}$ is an accepting run of $(\mathcal{A}_B)_q$ on $y^k \cdot \hat{y}$.

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

By Lemma 26, there is a tree $t \in L(\mathcal{A}_q)$, a computation $\phi \in ACC(\mathcal{A}_q, t)$, and a tree branch $\pi = v_0, v_1, \dots$ such that $\phi(\pi) = (\rho')^k \cdot \hat{\rho}$; and for each $i \in \mathbb{N}$ we have $t(v_i) = a_{i+1}$, and v_{i+1} is the left child of v_i iff $d_i = l$.

Let $J := \{i \mid \text{the } i\text{-th transition of } (\rho')^k \cdot \hat{\rho} \text{ is from } p'_1 \text{ to } p'_2 \text{ over } z'\}$. By the definition of ρ we conclude that $|J| \geq k$. Denote by v'_i the child of v_i which is not v_{i+1}, and define $A := \{v'_i \mid i \in J\}$. Notice that A is an antichain, and therefore $t_k := t \circ_A t_z$ - i.e., the grafting of t_z in t at all nodes in A - is well-defined.

For each $B \subseteq A$, let $\phi_B : \{l,r\}^* \to Q$ such that:

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

Notice that $\phi_B = (\phi \circ_{A \setminus B} \phi_1) \circ_B \phi_2$. It is easy to verify that, similarly to Lemma 3, ϕ_B is an accepting computation of $\mathcal{A}_q$ on t_k.

For completeness, we will show that ϕ_B respects δ by proving that $(\phi_B(u), t_k(u), \phi_B(u \cdot l), \phi_B(u \cdot r)) \in \delta$ for all $u \in \{l,r\}^*$:

- If $u = v \cdot w$ for $v \in A \setminus B$, then $(\phi_B(u), t_k(u), \phi_B(u \cdot l), \phi_B(u \cdot r)) = (\phi_1(w), t_1(w), \phi_1(w \cdot l), \phi_1(w \cdot r))$ which is in δ by the definition of ϕ_1.
- Otherwise, if $u = v \cdot w$ for $v \in B$, then $(\phi_B(u), t_k(u), \phi_B(u \cdot l), \phi_B(u \cdot r)) = (\phi_2(w), t_2(w), \phi_2(w \cdot l), \phi_2(w \cdot r))$ which is in δ by the definition of ϕ_2.
- If $u = v_i$ for $i \in J$, then assume w.l.o.g. that v_{i+1} is the left child of v_i and $v'_i \in A \setminus B$ (the other cases are symmetrical). By the definition of ϕ and z' we obtain $(\phi_B(u), t_k(u), \phi_B(u \cdot l), \phi_B(u \cdot r)) = (\phi(u), t_k(u), \phi(u \cdot l), \phi_1(\epsilon)) = (p'_1, a_z, p'_2, \phi_1(\epsilon))$, and since $\phi_1(\epsilon) \in S_z$, we conclude that $(p'_1, (a_z, d_z, S_z), p'_2) \in \delta_B$ for $d_z = l$. Hence, $(p'_1, a_z, p'_2, \phi_1(\epsilon)) \in \delta$, as requested.
- Otherwise, by the definition of ϕ_B we conclude that $u \cdot l \not\geqslant v'_j$ and $u \cdot r \not\geqslant v'_j$ for all $j \in \mathbb{N}$, and therefore we obtain $(\phi_B(u), t_k(u), \phi_B(u \cdot l), \phi_B(u \cdot r)) = (\phi(u), t_k(u), \phi(u \cdot l), \phi(u \cdot r))$ which is in δ by the definition of ϕ.

Let $B_1, B_2 \subseteq A$ such that there exists $b \in B_1$, $b \notin B_2$. By the definition of ϕ_{B_1} and ϕ_{B_2}, we have $\forall w \in \{l,r\}^* : \phi_{B_1}(b \cdot w) = \phi_2(w)$ and $\phi_{B_2}(b \cdot w) = \phi_1(w)$. $\phi_1 \neq \phi_2$ and therefore there is a node w such that $\phi_1(w) \neq \phi_2(w)$ and therefore $\phi_{B_1}(b \cdot w) \neq \phi_{B_2}(b \cdot w)$. Similarly, if there is $b \in B_2 \setminus B_1$, then $\phi_{B_1} \neq \phi_{B_2}$. Hence, $B_1 \neq B_2 \to \phi_{B_1} \neq \phi_{B_2}$.

We obtain $da(\mathcal{A}_q) \geq |ACC(\mathcal{A}_q, t_k)| \geq |\{B \mid B \subseteq A\}| \geq 2^k$, and by Lemma 4 we conclude that $da(\mathcal{A}) \geq da(\mathcal{A}_q) \geq 2^k$. Therefore, $\mathcal{A}$ is not boundedly ambiguous.

(2) Assume that $f := q$ is a final state of $\mathcal{A}$.

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

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

By Lemma 26, there is a tree $t \in L(\mathcal{A}_f)$, a computation $\phi \in ACC(\mathcal{A}_f, t)$, and a tree branch $\pi = v_0, v_1, \dots$ such that $\phi(\pi) = (\rho')^\omega$; and for each $i \in \mathbb{N}$ we have $t(v_i) = a_i$, and v_{i+1} is the left child of v_i iff $d_i = l$.

Let $J := \{i \mid \text{the } i\text{-th transition of } (\rho')^\omega \text{ is from } p'_1 \text{ to } p'_2 \text{ over } z'\}$. By the definition of ρ we conclude that J is an infinite subset of $\mathbb{N}$. Denote by v'_i the child of v_i which is not v_{i+1}, and define $A := \{v'_i \mid i \in J\}$. Notice that A is an antichain, and therefore $t' := t \circ_A t_z$ is a well-defined grafting of t_z in t at all nodes in A.

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

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

Similarly to the proof of (1), we conclude that ϕ_B is a computation of $\mathcal{A}_f$ on t', and that $B_1 \neq B_2 \rightarrow \phi_{B_1} \neq \phi_{B_2}$. Since the number of subsets of A is uncountable, and each subset B yields a unique accepting computation ϕ_B of $\mathcal{A}$ on t', it follows that $ACC(\mathcal{A}_f, t')$ is not countable. By Lemma 4, we conclude that $\mathcal{A}$ is not countably ambiguous.

4.3. Two Useful Lemmas

Here we prove two simple but useful lemmas - Lemma 27, which provides sufficient conditions for an ambiguous transition pattern; and Lemma 28, which provides sufficient conditions for uncountable branch ambiguity.

Lemma 27 (*q*-ambiguous transition pattern) *Let $\mathcal{A}$ be a BTA and let $v \perp w$. 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 $\phi(v) = q$ and $\phi(w) = p$, where $\mathcal{A}_p$ is ambiguous.*
2. *There are $\phi, \phi' \in ACC(\mathcal{A}_q, t)$ such that $\phi(v) = q$ and $\forall v' : (v' \leq v) \rightarrow (\phi(v') = \phi'(v')$, and $\phi(w) \neq \phi'(w)$.*

Proof *Let u be the node of maximal depth on the path from the root of t to v such that $w > u$. Let u', u'' be the children of u such that $w \geq u'$ and $v \geq u''$. Assume w.l.o.g. that u' is the left child of u.*

(1) By the definition of ϕ, there is a transition $(\phi(u), t(u), \phi(u'), \phi(u'')) \in \delta$. Since $da(\mathcal{A}_{\phi(w)}) > 1$, we can use Lemma 4 to obtain $da(\mathcal{A}_{\phi(u')}) > 1$. Hence, condition 2 of the definition of q-ambiguous transition pattern applies.

(2) Look at transitions $(\phi(u), t(u), \phi(u'), \phi(u'')), (\phi'(u), t(u), \phi'(u'), \phi'(u'')) \in \delta$. Since $u'' \leq v$ we have $\phi(u'') = \phi'(u'')$. If $\phi(u') = \phi'(u')$ then the restriction of ϕ and ϕ' on $t_{\geq u'}$ are two different computations in $ACC(\mathcal{A}_{\phi(u')}, t_{\geq u'})$ and therefore $da(\mathcal{A}_{\phi(u')}) > 1$ and condition 2 of q-ambiguous transition pattern definition applies. Otherwise, we have $\phi(u') \neq \phi'(u')$ and $t_{\geq u'} \in L(\mathcal{A}_{\phi(u')}) \cap L(\mathcal{A}_{\phi'(u')})$ and therefore condition 1 of q-ambiguous transition pattern definition applies. □

Lemma 28 (uncountable branch ambiguity) *Let $\mathcal{A}$ be a BTA with the corresponding branch automaton $\mathcal{A}_B$. Let f be a final state of $\mathcal{A}$, and let $u < w < v$ be nodes in $\{l, r\}^*$. If there are $\phi_1, \phi_2 \in ACC(\mathcal{A}, t)$ such that $\phi_1(u) = \phi_1(v) = \phi_2(u) = \phi_2(v) = f$ and $\phi_1(w) \neq \phi_2(w)$, then $\mathcal{A}$ is not countably branch ambiguous.*

Proof *Assume that there are $\phi_1, \phi_2 \in ACC(\mathcal{A}, t)$ such that $\phi_1(u) = \phi_1(v) = \phi_2(u) = \phi_2(v) = f$ and $\phi_1(w) \neq \phi_2(w)$. Let $\pi = v_0, v_1, \ldots$ be a tree branch such that $u, v, w \in \pi$. By Lemma 25, there is an ω-word $y \in L(\mathcal{A}_B)$ such that for each computation $\phi \in ACC(\mathcal{A}, t)$ the projection of ϕ on π is an accepting run of $\mathcal{A}_B$ on y. Therefore, we obtain $\phi_1(\pi), \phi_2(\pi) \in ACC(\mathcal{A}_B, y)$.*

By the definition of π, there are $i < j < k$ such that $v_i = u$, $v_j = w$ and $v_k = v$. Let $\rho_1 := \phi_1(v_i)\phi_1(v_{i+1}) \ldots \phi_1(v_k)$, and $\rho_2 := \phi_2(v_i)\phi_2(v_{i+1}) \ldots \phi_2(v_k)$. By the definition of ϕ_1 and ϕ_2 we have $\phi_1(v_i) = \phi_2(v_i) = f$, $\phi_1(v_k) = \phi_2(v_k) = f$ and $\phi_1(v_j) \neq \phi_2(v_j)$.

Assume that $y = z_1 z_2 \ldots$ for $z_i \in \Sigma_B$. By the definition of y we conclude that ρ_1 and ρ_2 are two distinct runs of $(\mathcal{A}_B)_f$ on $z_i, z_{i+1}, \ldots, z_{k-1}$ from f to f. Hence, $\mathcal{A}_B$ contains the forbidden pattern for countable ambiguity, and by Theorem 9 and Lemma 13 we conclude that $\mathcal{A}$ is not countably branch ambiguous. □

5. Bounded Ambiguity

In this section we prove Proposition 18 - a structural characterization of bounded ambiguity. (2) ⇒ (1) follows from Lemma 13 and Lemma 16. Below we prove the (1) ⇒ (2) direction. Our proof is based on the results shown in [9], where Seidl gave two criteria characterizing not boundedly ambiguous finite tree automata.

We first introduce labeled finite binary trees, and automata over finite binary trees.

Definition 29 *A finite prefix closed set $U \subseteq \{l, r\}^*$ is called a finite **binary** tree if for each $u \in U$, if u is not maximal in U then $u \cdot l, u \cdot r \in U$. A node $u \in U$ is called a **leaf** if it is maximal in U. Otherwise, u is called an **internal node**. As labels of the finite binary trees we use a finite alphabet Σ which is partitioned into two sets: Σ_2 - labels of internal nodes, and Σ_0 - labels of leaves. A finite Σ-labeled binary tree is a function $t_U : U \to \Sigma$, where $U \subseteq \{l, r\}^*$ is a finite binary tree, $t_U(v) \in \Sigma_0$ if v is a leaf, and $t_U(v) \in \Sigma_2$ if v has children.*

Definition 30 *An automaton over finite binary trees is a tuple $\mathcal{B} = (Q, \Sigma, Q_I, \delta)$, where Q is a finite set of states, $\Sigma = \Sigma_0 \cup \Sigma_2$ is an alphabet, Q_I is a set of initial states, and $\delta \subseteq (Q \times \Sigma_0) \cup (Q \times \Sigma_2 \times Q \times Q)$ is a set of transitions.*

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

The following definitions are taken from [9]. Since our proof only uses finite binary trees, we simplify the notations where appropriate.

Definition 31 (Finite tree branch automaton) *The corresponding branch automaton $\mathcal{B}_B$ of a finite binary tree automaton $\mathcal{B} = (Q, \Sigma, Q_I, \delta)$ with $\Sigma = \Sigma_2 \cup \Sigma_0$ is a finite word automaton $(Q, \Sigma_B, Q_I, \delta_B, F)$, where:*

- $\Sigma_B := \Sigma_2 \times \{l, r\}$
- $F := \{q \in Q \mid \exists a \in \Sigma_0 : (q, a) \in \delta\}$
- *δ_B is the minimal set such that $(q, a, q_l, q_r) \in \delta$ implies $(q, (a, d), q_d) \in \delta_B$ for $d \in \{l, r\}$.*

Definition 32 (Conditions T1 and T2) *Let $\mathcal{B}$ be a finite binary tree automaton, with the corresponding branch automaton $\mathcal{B}_B$.*

T1: *$\mathcal{B}_B$ satisfies condition T1 if there are states $p, q, q_d \in Q$ and finite words $y_1, y_2 \in \Sigma_B^*$ such that there is a run of $(\mathcal{B}_B)_p$ on y_1 from p to q, a run of $(\mathcal{B}_B)_{q_d}$ on y_2 from q_d to p, and a transition $(q, (a, d), q_d) \in \delta$ for $a \in \Sigma$, such that at least one of the following hold:*

1. *There exist two different transitions* $(q, a, q_d, q_1), (q, a, q_d, q_2) \in \delta$ *(if* $d = l$*) or* $(q, a, q_1, q_d), (q, a, q_2, q_d) \in \delta$ *(if* $d = r$*) such that* $L(\mathcal{B}_{q_1}) \cap L(\mathcal{B}_{q_2}) \neq \emptyset$
2. *There exists a transition* $(q, a, q_d, q_1) \in \delta$ *(if* $d = l$*) or* $(q, a, q_1, q_d) \in \delta$ *(if* $d = r$*) such that* $da(\mathcal{B}_{q_1}) > 1$.

T2: *$\mathcal{B}_B$ satisfies condition T2 if there are two distinct states $p, q \in Q$, a finite word $y \in \Sigma_B^+$, and runs ρ_1 of $(\mathcal{B}_B)_p$ on y from p to p, ρ_2 of $(\mathcal{B}_B)_p$ on y from p to q, and ρ_3 of $(\mathcal{B}_B)_q$ on y from q to q such that the following holds:*

Let $\rho_i = p_0^i p_1^i \ldots, p_n^i$ for $1 \leq i \leq 3$, and $y = (a_1, d_1) \ldots (a_n, d_n)$. Then for all $1 \leq j \leq n$ there are transitions $(p_{j-1}^i, a_j, p_j^i, q_j^i) \in \delta$ for $1 \leq i \leq 3$ if $d_j = l$, or $(p_{j-1}^i, a_j, q_j^i, p_j^i) \in \delta$ for $1 \leq i \leq 3$ if $d_j = r$, such that $L(\mathcal{B}_{q_j^1}) \cap L(\mathcal{B}_{q_j^2}) \cap L(\mathcal{B}_{q_j^3}) \neq \emptyset$.

This definition is illustrated in Figure 6.

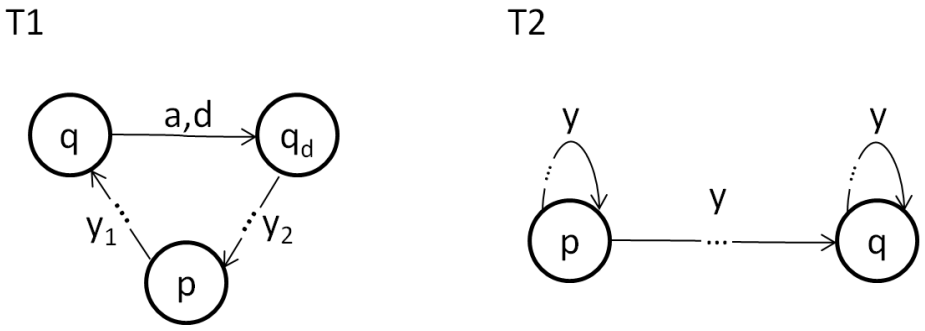

Figure 6: Conditions T1 and T2

The following theorem provides a structural characterization of bounded ambiguity for automata over finite trees:

Theorem 33 (Seidl [9]) *An automaton $\mathcal{B}$ over finite trees is not boundedly ambiguous if and only if $\mathcal{B}$ satisfies condition T1 or condition T2.*

We will now show a reduction from BTA to automata over finite binary trees. Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, F)$ be a BTA, and let $Q_{cons} := \{Q' \subseteq Q \mid \cap_{q \in Q'} L(\mathcal{A}_q) \neq \emptyset\}$. We define a finite binary tree automaton $\mathcal{A}^{fin} := (Q, \Sigma_{fin}, Q_I, \delta_{fin})$, where:

- $\Sigma_{fin} := \Sigma_2 \cup \Sigma_0$, where $\Sigma_2 := \Sigma$ is the set of labels of nodes with two children, and $\Sigma_0 := Q_{cons}$ is the set of labels of node without children (i.e. leaves).
- $\delta_{fin} := \delta \cup \{(q, Q') \mid q \in Q' \text{ and } Q' \in Q_{cons}\}$ (i.e., the transitions on inner nodes are according to δ, and a Q'-labeled leaf is accepted from a state q iff $q \in Q'$).

Lemma 34 *If $\mathcal{A}$ is not boundedly ambiguous, then $\mathcal{A}^{fin}$ is not boundedly ambiguous.*

Proof *We have to prove that for each $k \in \mathbb{N}$, $\mathcal{A}^{fin}$ has at least k accepting computations on some finite Σ_{fin}-labeled tree. Assume that $\mathcal{A}$ is not boundedly ambiguous. Therefore,*

there is an infinite tree $t \in L(\mathcal{A})$ such that $|ACC(\mathcal{A},t)| \geq k$. Let $\phi_1,\ldots,\phi_k \in ACC(\mathcal{A},t)$ be pairwise different computations of $\mathcal{A}$ on t. For each $1 \leq i < j \leq k$, let $u_{i,j} \in \{l,r\}^*$ be a node such that $\phi_i(u_{i,j}) \neq \phi_j(u_{i,j})$. Let m be the maximal depth of a node $u_{i,j}$, and let $U := \{u \in \{l,r\}^* \mid u \text{ is of depth} \leq m+1\}$. It is clear that U is a finite binary tree. Define a finite labeled tree $t_U : U \to \Sigma_{fin}$ as follows:

$$t_U(u) := \begin{cases} t(u) & u \text{ is not a leaf in } U \\ \{\phi(u) \mid \phi \in ACC(\mathcal{A},t)\} & u \text{ is a leaf in } U \end{cases}$$

Notice that for each leaf $u \in U$ and for each computation $\phi \in ACC(\mathcal{A},t)$ we have $t_{\geq u} \in L(\mathcal{A}_{\phi(u)})$. Therefore, $\cap_{q \in S_u} L(\mathcal{A}_q) \neq \emptyset$ for $S_u := \{\phi(u) \mid \phi \in ACC(\mathcal{A},t)\} \in Q_{cons}$ and we conclude that t_U is well-defined.

For each $\phi \in ACC(\mathcal{A},t)$, let ϕ' be the restriction of ϕ on U. For each node $u \in U$, if u is not a leaf then $(\phi(u),t(u),\phi(u\cdot l),\phi(u\cdot r)) = (\phi'(u),t_U(u),\phi'(u\cdot l),\phi'(u\cdot r)) \in \delta$, and since $\delta \subseteq \delta_{fin}$, we conclude that $(\phi'(u),t_U(u),\phi'(u\cdot l),\phi'(u\cdot r)) \in \delta_{fin}$. If u is a leaf then we have $\phi'(u) = \phi(u) \in S_u = t_U(u)$, and by the definition of δ_{fin} we conclude that $(\phi'(u),t_U(u)) \in \delta_B$. Hence, ϕ' is an accepting computation of $\mathcal{A}^{fin}$ on t_U. Notice that $\forall u \in U : \phi'(u) = \phi(u)$, and therefore $\phi'_i(u_{i,j}) = \phi_i(u_{i,j}) \neq \phi_j(u_{i,j}) = \phi'_j(u_{i,j})$ for each $1 \leq i < j \leq k$. We conclude that $\phi'_1,\phi'_2,\ldots,\phi'_k \in ACC(\mathcal{A}^{fin},t_U)$ are k distinct accepting computations, and therefore $|ACC(\mathcal{A}^{fin},t_U)| \geq k$. □

Lemma 35 *Let $U \subseteq \{l,r\}^*$ be a finite binary tree, and let $t_U : U \to \Sigma_{fin}$ be a finite binary Σ_{fin}-labeled tree. Then there is an infinite Σ-labeled binary tree t such that for each computation $\phi \in ACC(\mathcal{A}^{fin}_{\phi(\epsilon)},t_U)$ there is a computation $\phi' \in ACC(\mathcal{A}_{\phi(\epsilon)},t)$ such that $\phi'(u) = \phi(u)$ for all $u \in U$.*

Proof *By the definition of $\mathcal{A}^{fin}$, for each leaf $u \in U$ which is labeled by $S_u := t_U(u)$ we have $\cap_{q\in S_u} L(\mathcal{A}_q) \neq \emptyset$, and therefore there is a tree $t_u \in \cap_{q\in S_u} L(\mathcal{A}_q)$. We use t_u to define an infinite Σ-labeled tree t:*

$$t(u) := \begin{cases} t_U(u) & u \in U \text{ and } u \text{ is not a leaf in } U \\ t_v(w) & u = v\cdot w \text{ for a leaf } v \in U \text{ and } w \in \{l,r\}^* \end{cases}$$

Let $\phi \in ACC(\mathcal{A}^{fin}_{\phi(\epsilon)},t_U)$. Again by the definition of $\mathcal{A}^{fin}$, for every leaf $u \in U$ we have $\phi(u) \in S_u$ - hence, there is a computation $\phi_u \in ACC(\mathcal{A}_{\phi(u)},t_u)$. We define a computation ϕ' as follows:

$$\phi'(u) := \begin{cases} \phi(u) & u \in U \text{ and } u \text{ is not a leaf in } U \\ \phi_v(w) & u = v\cdot w \text{ for a leaf } v \in U \text{ and } w \in \{l,r\}^* \end{cases}$$

It is clear that ϕ' is a computation of $\mathcal{A}$ on t, and that $\forall u \in U : \phi'(u) = \phi(u)$. We will prove that ϕ' is accepting by showing that for each tree branch π there are infinitely many occurrences of final state in $\phi'(\pi)$. Let $\pi = v_0,\ldots,v_i,\ldots$. Since U is a finite tree, there is a leaf $u \in U$ such that $v_i = u$ for $i \in \mathbb{N}$. Therefore, π is of the form $v_0\ldots(v_{i-1})(u\cdot v'_1)(u\cdot v'_2)\ldots$. By the definition of ϕ' we obtain $\forall i \in \mathbb{N} : \phi'(u\cdot v'_i) = \phi_u(v'_i)$. Recall that ϕ_u is an accepting computation, and therefore $\phi'(\pi)$ contains infinitely many occurrences of a final state in F. We conclude that $\phi' \in ACC(\mathcal{A}_{\phi(\epsilon)},t)$, as requested. □

Corollary 36 *If $\mathcal{A}^{fin}$ is not boundedly ambiguous then $\mathcal{A}$ is not boundedly ambiguous.*

Remark (On Reduction of $\mathcal{A}$ to $\mathcal{A}^{fin}$) *By Lemma 34 and Corollary 36 we obtain that $\mathcal{A}$ is boundedly ambiguous iff $\mathcal{A}^{fin}$ is boundedly ambiguous. The alphabet of $\mathcal{A}^{fin}$*

might be exponential in the size of $\mathcal{A}$. Hence, this reduction is not polynomial. We use this reduction to obtain a structural characterization of bounded ambiguity for BTA from the structural characterization of bounded ambiguity for automata on finite trees (Theorem 33). In the rest of this section we show that if $\mathcal{A}^{fin}$ satisfies $T1$ (respectively, $T2$), then $\mathcal{A}$ has an ambiguous transition pattern (respectively, is not boundedly branch ambiguous). In Section 8 we show that it is testable in polynomial time whether $\mathcal{A}$ has an ambiguous transition pattern or is boundedly branch ambiguous. This will give us a polynomial algorithm to test whether a BTA is boundedly ambiguous.

Lemma 37 1. *If $L(\mathcal{A}_p^{fin}) \cap L(\mathcal{A}_q^{fin}) \neq \emptyset$ then $L(\mathcal{A}_p) \cap L(\mathcal{A}_q) \neq \emptyset$.*
2. *If $da(\mathcal{A}_q^{fin}) > 1$ then $da(\mathcal{A}_q) > 1$.*

Proof *(1) Let $t_U : U \to \Sigma_{fin}$ be a finite binary Σ_{fin}-labeled tree such that $t_U \in L(\mathcal{A}_p^{fin}) \cap L(\mathcal{A}_q^{fin})$, and let $\phi_1 \in ACC(\mathcal{A}_p^{fin}, t_U)$ and $\phi_2 \in ACC(\mathcal{A}_q^{fin}, t_U)$. By Lemma 35, there is an infinite tree t such that for each computation $\phi \in ACC(\mathcal{A}_{\phi(\epsilon)}^{fin}, t_U)$ there is a computation $\phi' \in ACC(\mathcal{A}_{\phi(\epsilon)}, t)$ such that $\phi'(u) = \phi(u)$ for all $u \in U$.*

Therefore, there are two computations $\phi'_1 \in ACC(\mathcal{A}_p, t)$ and $\phi'_2 \in ACC(\mathcal{A}_q, t)$, and we obtain $t \in L(\mathcal{A}_p) \cap L(\mathcal{A}_q)$. Hence, $L(\mathcal{A}_p) \cap L(\mathcal{A}_q) \neq \emptyset$, as requested.

(2) Let t_U be a finite binary Σ_{fin}-labeled tree, such that there are two distinct computations $\phi_1, \phi_2 \in ACC(\mathcal{A}_q, t_U)$. By Lemma 35, there is an infinite tree t and two computations $\phi'_1, \phi'_2 \in ACC(\mathcal{A}_q, t)$ such that $\forall u \in U : \phi'_1(u) = \phi_1(u)$ and $\phi'_2(u) = \phi_2(u)$. $\phi_1 \neq \phi_2$, and therefore there is a node $u \in U$ such that $\phi_1(u) \neq \phi_2(u)$. We conclude that $\phi'_1(u) = \phi_1(u) \neq \phi_2(u) = \phi'_2(u)$, and therefore $\phi'_1 \neq \phi'_2$, and we obtain $da(\mathcal{A}_q) > 1$, as requested. □

Lemma 38 *Let $\mathcal{B} := \mathcal{A}^{fin}$, and let $\mathcal{B}_B$ be the corresponding branch automaton of $\mathcal{B}$.*

1. *If $\mathcal{B}_B$ satisfies condition T1 then $\mathcal{A}$ has an ambiguous transition pattern.*
2. *If $\mathcal{B}_B$ satisfies condition T2 then $\mathcal{A}$ is not boundedly branch ambiguous*

Proof *Let $\mathcal{A}_B = (Q, \Sigma_{\mathcal{A}_B}, Q_I, \delta_{\mathcal{A}_B}, F_{\mathcal{A}_B})$ be the corresponding branch automaton of $\mathcal{A}$. We will first prove the following claim:*

Claim 38.1 *If there exists a run ρ of $(\mathcal{B}_B)_p$ on a word y from p to q, then there exists a word y' such that $(\mathcal{A}_B)_p$ has a run on y' from p to q.*

Proof of Claim 38.1 *Let $y = (a_1, d_1) \dots (a_n, d_n)$, and let $\rho = p_0 \dots p_n$. By the definition of $\mathcal{B}_B$, for each $1 \leq i \leq n$, there is a transition $(p_{i-1}, a_i, p_i, q_i) \in \delta$ if $d_i = l$, or a transition $(p_{i-1}, a_i, q_i, p_i) \in \delta$ if $d_i = r$. Let $S_i := \{q_i\}$, and define a word $y' := (a_1, d_1, S_1) \dots (a_n, d_n, S_n)$. By the definition of $\mathcal{A}_B$, we conclude that ρ is a run of $(\mathcal{A}_B)_p$ on y' from p to q, as requested.* ■

Now, we are ready to prove Lemma 38.

(1) By Claim 38.1, we conclude that there are words y'_1 and y'_2 such that there is a run of $(\mathcal{A}_B)_p$ on y'_1 from p to q, and a run of $(\mathcal{A}_B)_{q_d}$ on y'_2 from q_d to p.

Let $(q, (a, d), q_d) \in \delta$ as guaranteed by pattern T1, and assume w.l.o.g. that $d = l$ (the proof for $d = r$ is symmetrical). We will separate to cases:

Case 1: There exist two different transitions $(q, a, q_d, q_1), (q, a, q_d, q_2) \in \delta$ such that $L(\mathcal{B}_{q_1}) \cap L(\mathcal{B}_{q_2}) \neq \emptyset$. By the definition of $\mathcal{B}_B$ we conclude that there are two transitions

$(q,(a,d,\{q_1\}),q_d),(q,(a,d,\{q_2\}),q_d) \in \delta_{\mathcal{A}_B}$, *and by Lemma 37(1) we obtain* $L(\mathcal{A}_{q_1}) \cap L(\mathcal{A}_{q_2}) \neq \emptyset$, *and therefore* $\mathcal{A}_B$ *has a p-ambiguous transition pattern.*

Case 2: There exists a transition $(q,a,q_d,q_1) \in \delta$ *such that* $da(\mathcal{B}_{q_1}) > 1$. *By the definition of* $\mathcal{B}_B$ *we conclude that* $(q,(a,d,\{q_1\}),q_d) \in \delta_{\mathcal{A}_B}$ *and by Lemma 37(2) we obtain* $da(\mathcal{A}_{q_1}) > 1$, *and therefore* $\mathcal{A}_B$ *has a p-ambiguous transition pattern.*

(2) Let $y = (a_1,d_1)\dots(a_n,d_n)$, *and* $\rho_i = p_0^i p_1^i \dots p_n^i$ *for* $1 \leq i \leq 3$. *By the definition of T2, for all* $1 \leq j \leq n$ *there is* q_j^i *such that* $(p_{j-1}^i, a_j, p_j^i, q_j^i) \in \delta_B$ *for* $1 \leq i \leq 3$ *if* $d_j = l$, *or* $(p_{j-1}^i, a_j, q_j^i, p_j^i) \in \delta_B$ *for* $1 \leq i \leq 3$ *if* $d_j = r$. *Define* $S_j := \{q_j^1, q_j^2, q_j^3\}$ *for* $1 \leq j \leq n$. *Note that* S_j *are consistent, i.e.,* $L(\mathcal{A}_{q_j^1}) \cap L(\mathcal{A}_{q_j^2}) \cap L(\mathcal{A}_{q_j^3}) \neq \emptyset$ *for* $j \leq n$. *Let* $y' := (a_1,d_1,S_1)\dots(a_n,d_n,S_n)$. *By the definition of* $\mathcal{A}_B$ *we conclude that* ρ_1 *is a run of* $(\mathcal{A}_B)_p$ *on* y' *from* p *to* p, ρ_2 *is a run of* $(\mathcal{A}_B)_p$ *on* y' *from* p *to* p, *and* ρ_3 *is a run of* $(\mathcal{A}_B)_q$ *on* y' *from* q *to* q. *Hence,* $\mathcal{A}_B$ *contains the forbidden pattern for bounded ambiguity, and therefore* $\mathcal{A}$ *is not boundedly branch ambiguous.* □

We now proceed with the proof of the (1) ⇒ (2) direction of Proposition 18. $\mathcal{A}$ is not boundedly ambiguous, and therefore by Lemma 34 we conclude that $\mathcal{A}^{fin}$ is not boundedly ambiguous. Therefore, by Theorem 33, $\mathcal{A}^{fin}$ satisfies condition T1 or condition T2. Applying Lemma 38, we conclude that $\mathcal{A}$ has an ambiguous transition pattern, or that $\mathcal{A}$ is not bounded branch ambiguous, as requested.

6. Finite Ambiguity

In this section we prove Proposition 19 - a structural characterization of finite ambiguity. (2) ⇒ (1) follows from Lemma 13 and Lemma 16. 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 finitely branch ambiguous, and 2(a) holds. Otherwise, there exist $\phi_1,\dots,\phi_k \in ACC(\mathcal{A},t)$ such that $ACC(\mathcal{A},t,\pi) = \{\phi_i(\pi) \mid 1 \leq i \leq k\}$. Choose n such that for all $1 \leq i < j \leq k : \phi_i(v_0)\dots\phi_i(v_n) \neq \phi_j(v_0)\dots\phi_j(v_n)$.

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

Let f be an accepting state which occurs infinitely often in $\phi_j(\pi)$. Choose $N > n$ such that $\phi_j(v_N) = q_N = f$. 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 $\phi', \phi'' \in ACC(\mathcal{A}_f, t_{\geq v_N})$. Note that $\phi \circ_{v_N} \phi'$ and $\phi \circ_{v_N} \phi''$ are accepting computations which coincide with ϕ on $v_0, \ldots v_N$ and hence on $v_0, \ldots v_n$. Therefore, they coincide with ϕ on π and $\forall i \geq N : \phi_j(v_i) = \phi'(v_i) = \phi''(v_i) = q_i$. Therefore, ϕ' and ϕ'' differ at some node $w \notin \pi$, and there exist $i > N$ such that $\phi_j(v_i) = f = \phi'(v_i) = \phi''(v_i)$ and $v_i \perp w$. Applying Lemma 27(2) on ϕ', ϕ'', and $v_i \perp w$, we conclude that $\mathcal{A}_f$ has an f-ambiguous transition pattern, and 2(b) holds.

7. Countable Ambiguity

In this section we prove Proposition 22 - a structural characterization of countable ambiguity.

7.1. Direction (2) ⇒ (1) *of Proposition 22*

$2(a) \Rightarrow (1)$ follows by the definition of branch ambiguity, and $2(b) \Rightarrow (1)$ follows by Lemma 16. Below $2(c) \Rightarrow (1)$ is proved.

Definition 39 (Corresponding automaton $\mathcal{A}_M$ for pattern M) *Let M be a branching pattern for $\mathcal{A}$ over (R, f) (see Definition 20). We define a BTA $\mathcal{A}_M$ over the unary alphabet with the set of states $R \cup \{f\}$; all states are final, the initial state is f, and the transition relation is $\Delta_M := \{(q, q', q'') \mid q \in R \text{ and } (q', q'') = \tau_M(q)\} \cup \{(f, q_1, f), (f, f, q_2)\}$.*

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 40 (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 $\phi(v) = f$ for all $v \in \pi$.*
3. *The set of accepting computations of $\mathcal{A}_M$ is uncountable.*

Lemma 41 *Let $\mathcal{A} = (Q_\mathcal{A}, \Sigma, Q_I, \delta, F)$ be a BTA 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 (R, f) is realized in t at $u \perp v$ by ϕ_1, ϕ_2, $\{\phi_q \mid q \in R\}$. 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 $t_{1+k_y}(y) = a$ then $\forall i > k_y : t_i(y) = a$. Define t^ω as $t^\omega(y) := t_{1+k_y}(y)$.

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.

Notation 42 *Let h be a function from $\{l,r\}^*$ into $\{l,r\}^*$ defined as follows: $h(l) := v$, $h(r) := u$, and $h(d_1 \dots d_m) := h(d_1) \dots h(d_m)$ for $d_i \in \{l,r\}$. 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 $\widehat{\phi}(w) := \phi_q(z)$. If $\phi(h^{-1}(y)) = f$ there are two cases: (1) $\phi(h^{-1}(y \cdot u)) = f$, in this case we define $\widehat{\phi}(w) := \phi_1(z)$; (2) $\phi(h^{-1}(y \cdot v)) = f$, in this case we define $\widehat{\phi}(w) := \phi_2(z)$ (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 40, $\mathcal{A}_M$ 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. □

7.2. Direction (1) ⇒ (2) of Proposition 22

Definition 43 (q-path and q-computation) *Given a BTA $\mathcal{A} = (Q, \Sigma, Q_I, \delta, F)$, a state $q \in Q$, and a tree $t \in L(\mathcal{A})$, we define the following:*

- *A q-**path** (of an accepting computation $\phi \in ACC(\mathcal{A}_q, t)$) is a tree branch π such that q occurs infinitely often in $\phi(\pi)$.*
- *A q-**computation** is an accepting computation ϕ of $\mathcal{A}_q$ on t such that ϕ has a q-path.*

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 for a final state $f \in F$.

Lemma 44 *A BTA $\mathcal{A} = (Q, \Sigma, Q_I, \delta, F)$ has uncountably many accepting computations on t iff there is a state $f \in F$, a node $u \in \{l,r\}^*$ 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 ⇐*: For each computation $\phi \in ACC(\mathcal{A}_f, t_{\geq u})$, define $g(\phi) := \phi_0 \circ_u \phi$. By the grafting lemma we obtain $g(\phi) \in ACC(\mathcal{A}, t)$. For each two computations $\phi_1, \phi_2 \in ACC(\mathcal{A}_f, t_{\geq u})$, $\phi_0 \circ_u \phi_1 = \phi_0 \circ_u \phi_2$ implies $\phi_1 = \phi_2$, and therefore the function $g : ACC(\mathcal{A}_f, t_{\geq u}) \to ACC(\mathcal{A}, t)$ is injective. We obtain $|ACC(\mathcal{A}_f, t_{\geq u})| \leq |ACC(\mathcal{A}, t)|$, as requested.*

⇒: 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 set of nodes $U_\phi := \{v \in \{l,r\}^* \mid \forall u < v$: if $\phi(u) \in F$ then ϕ has no $\phi(u)$-paths on $t_{\geq u}\}$. It is clear that U_ϕ is downward closed. If u is a leaf (maximal node) of U_ϕ, then $\phi(u) \in F$ and ϕ has a $\phi(u)$-path on $t_{\geq u}$.

Observe that U_ϕ is finite. Otherwise, by the König Lemma, there is an infinite branch $\pi = v_0, \dots, v_i, \dots$ such that $v_i \in U_\phi$ for all $i \in \mathbb{N}$. ϕ is accepting and therefore there is $i \in \mathbb{N}$ such that $\phi(v_i) \in F$ and $\phi(v_i)$ occurs infinitely often in $\phi(\pi)$ - a contradiction to $v_{i+1} \in U_\phi$.

Therefore, to each computation $\phi \in \Phi$ corresponds a finite set $U_\phi \subseteq \{l,r\}^*$. Since there are countably many finite subsets of $\{l,r\}^*$, and uncountably many computations in

Φ, we conclude that there is a finite set $U \subseteq \{l,r\}^*$ and an uncountable set $\Phi' \subseteq \Phi$ such that $\forall \phi \in \Phi' : U_\phi = U$. Since there are finitely many assignments of states to the nodes in U, we conclude that there is a function $\psi : U \to Q$ and an uncountable set $\Phi'' \subseteq \Phi'$ such that $\forall \phi \in \Phi'' \, \forall u \in U : \phi(u) = \psi(u)$. For each maximal node u in U, define Φ''_u as the set of restrictions of 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 a $\psi(u)$-path on $t_{\geq u}$, and therefore Φ''_u is the set of f-computations of $\mathcal{A}_f$ on $t_{\geq u}$ for $f := \psi(u)$. □

Let us state a simple lemma about f-paths:

Lemma 45 *Assume conditions 2(a) and 2(b) of Proposition 22 do not hold.*

1. *If $\phi_1 \neq \phi_2$ are f-computations of $\mathcal{A}$ on t and π is an f-path of ϕ_1, then $\phi_1(\pi) \neq \phi_2(\pi)$.*
2. *Let π be a tree branch and let $\Phi_\pi := \{\phi \mid \pi \text{ is an } f\text{-path of } \phi\}$. Then Φ_π is countable.*

Proof *(1) Assume, for the sake of contradiction, that $\phi_1(\pi) = \phi_2(\pi)$. Since $\phi_1 \neq \phi_2$, there is a node $w \notin \pi$ such that $\phi_1(w) \neq \phi_2(w)$. Since π is an f-path of ϕ_1, there is a node $v \in \pi$ such that $v \perp w$ and $\phi_1(v) = f$. By the assumption, $\forall v' \leq v : \phi_1(v') = \phi_2(v')$ and therefore by Lemma 27(2) we conclude that $\mathcal{A}$ has an f-ambiguous transition pattern, in contradiction to condition 2(b) not holding in $\mathcal{A}$.*

(2) By (1), if $\phi_1 \neq \phi_2 \in \Phi_\pi$, then $\phi_1(\pi) \neq \phi_2(\pi)$. Since $\mathcal{A}$ has countable branch ambiguity it follows that Φ_π is countable. □

Notations

1. For a final state f of $\mathcal{A}$, let $f\text{−}Comp(\mathcal{A}_f, t)$ be the set of f-computations of $\mathcal{A}_f$ on t.
2. For $\Phi \subseteq f\text{−}Comp(\mathcal{A}_f, t)$, define

$$B(\Phi, t) := \{v \mid \exists \phi \in \Phi : \phi(v) = f \text{ and } f\text{−}Comp(\mathcal{A}_f, t_{\geq v}) \text{ is uncountable}\}.$$

We are going to investigate the structure of $B(\Phi, t)$. Our main technical lemma implies that if $f\text{−}Comp(\mathcal{A}_f, t)$ is uncountable, then the full-binary tree can be embedded in $B(f\text{−}Comp(\mathcal{A}_f, t), t)$, i.e., there is an injective function $e : \{l,r\}^* \to B(f\text{−}Comp(\mathcal{A}_f, t), t)$ such that $s_1 < s_2$ iff $e(s_1) < e(s_2)$.

Let $\Phi := f\text{−}Comp(\mathcal{A}_f, t)$, and let g be a function from Φ to tree branches such that for each $\phi \in \Phi$, $g(\phi)$ is an f-path of ϕ.

Let π be a branch, and let $\Phi_\pi := \{\phi \mid \pi \text{ is an } f\text{-path of } \phi\}$. Define $\Gamma_\pi : (f\text{−}Comp(\mathcal{A}, t) \setminus \Phi_\pi) \to (\{l,r\}^+ \setminus \pi)$ as a function which, for each $\phi \in f\text{−}Comp(\mathcal{A}, t) \setminus \Phi_\pi$, returns an arbitrary node in $\{u \in g(\phi) \setminus \pi \mid \phi(u) = f\}$. Notice that the set $\{u \in g(\phi) \setminus \pi \mid \phi(u) = f\}$ is non-empty because $g(\phi) \neq \pi$ and ϕ assigns the state f to infinitely many nodes on $g(\phi)$ - hence, Γ_π is well-defined.

Lemma 46 (Properties of Γ_π) *Assume conditions 2(a) and 2(b) of Proposition 22 do not hold.*

1. *If $\Gamma_\pi(\phi_1) = u = \Gamma_\pi(\phi_2)$, then $\phi_1(w) = \phi_2(w)$ for every $w \in \{w' \mid \neg(w' > u)\}$.*

2. *If* $\Gamma_\pi(\phi_1) = u = \Gamma_\pi(\phi_2)$ *and* $\phi_1 \neq \phi_2$, *then* $\phi_1(v) \neq \phi_2(v)$ *for some* $v > u$.
3. *If* $\Gamma_\pi(\phi) = u$ *then* $\phi_{\geq u} \in f{-}Comp(\mathcal{A}_f, t_{\geq u})$.

Proof *(1) If* $\phi_1(w) \neq \phi_2(w)$ *for* $w < u$ *then by Lemma 28 we conclude that* $\mathcal{A}$ *is not countably branch ambiguous - a contradiction to condition 2(a) not holding. If* $\phi_1(w) \neq \phi_2(w)$ *for* $w \perp u$ *then, by Lemma 27(2),* $\mathcal{A}$ *has an* f*-ambiguous transition pattern - a contradiction to condition 2(b) not holding. Hence,* $\phi_1(w) = \phi_2(w)$ *for every* $w \in \{w' \mid \neg(w' > u)\}$.

(2) Immediately follows from (1).

(3) By the definition of Γ_π *we conclude that* $\phi(u) = f$. *Recall that* $g(\phi)$ *is an* f*-path of* ϕ, *and therefore* $g(\phi) \cap \{v \mid v \geq u\}$ *is an* f*-path of* $\phi_{\geq u}$.

Lemma 47 *Assume conditions 2(a) and 2(b) of Proposition 22 do not hold. If* $\Phi := f{-}Comp(\mathcal{A}_f, t)$ *is uncountable, then:*

1. *For every branch* π *there is a node* u *not on* π *such that* $u \in B(\Phi, t)$.
2. *There are* $v_1 \perp v_2$ *such that such that* $v_1, v_2 \in B(\Phi, t)$.

Proof *(1) The domain* $f{-}Comp(\mathcal{A}, t) \setminus \Phi_\pi$ *of* Γ_π *is uncountable (by the assumption and Lemma 45(2)), and its range* $\{l, r\}^+ \setminus \pi$ *is countable. Therefore, there is* $u \notin \pi$ *such that* $\Psi := \{\phi \in f{-}Comp(\mathcal{A}, t) \setminus \Phi_\pi \mid \Gamma_\pi(\phi) = u\}$ *is uncountable. By Lemma 46 (2)-(3), we conclude that* $\Psi_{\geq u} := \{\phi_{\geq u} \mid \phi \in \Psi\}$ *is an uncountable set of* f*-computations on* $t_{\geq u}$, *and therefore* $u \in B(\Phi, t)$.

(2) Assume, for the sake of contradiction, that there are no $v_1 \perp v_2$ *such that* $v_1, v_2 \in B(\Phi, t)$. *Therefore, there is a branch* π *such that* $B(\Phi, t) \subseteq \pi$. *However, by (1), there is* $u \notin \pi$ *such that* $u \in B(\Phi, t)$ *- a contradiction.* □

The main technical lemma uses the following definition.

Definition 48 *Let* $T \subseteq \{l, r\}^*$ *be a set of nodes. We say that* $u \in T$ *is a* T***-leaf*** *if* $\forall v \in T : \neg(v > u)$; u *is a* T***-successor*** *of* v *if* $u > v$ *and there is no node* $w \in T$ *such that* $v < w < u$.

We call T *a* ***binary subset-tree*** *if* T *has a minimal node, and each node in* T *is either a* T*-leaf, or has two* T*-successors.*

We call T *a* ***full-binary subset-tree*** *if* T *is a binary subset-tree with no leaves.*

Lemma 49 (Main) *Assume* f *is a final state, there are uncountably many* f*-computations of* $\mathcal{A}$ *on* t, *and conditions 2(a) and 2(b) of Proposition 22 do not hold. Then, there is a full-binary subset-tree* T *of* t *such that for every* $u \in T$ *there is an* f*-computation* ϕ_u *on* t *such that if* $v \in T$ *and* $v \leq u$ *then* $\phi_u(v) = f$.

Proof *First we define a sequence* $T_0, T_1, \ldots$ *of finite binary subset-trees of* t *such that*

1. T_{i+1} *is obtained from* T_i *by adding two children to a leaf of* T_i *of minimal depth.*
2. *For every* $u \in T_i$, *there are uncountably many* f*-computations of* $\mathcal{A}_f$ *on* $t_{\geq u}$.
3. *if* $v \in T_i$ *is a* T_i*-successor of* $u \in T_i$, *then there is an* f*-computation* ϕ' *of* $\mathcal{A}_f$ *on* $t_{\geq u}$ *such that* $\phi'(v) = f$.

Let T_0 be a set which consists of the root of t. It is clear that (2)-(3) hold. Assume we have defined T_i such that (2)-(3) hold. Let u be a leaf of T_i of minimal depth. We apply Lemma 47(2) to $t_{\geq u}$ and obtain $v_1, v_2 \in B(f\text{-}Comp(\mathcal{A}_f, t_{\geq u}), t_{\geq u})$ such that $v_1 \perp v_2$. Define T_{i+1} as $T_i \cup \{v_1, v_2\}$. It is clear that (1)-(3) holds for T_{i+1}.

Let $T := \bigcup T_i$. It is clear that T is a (infinite) full-binary subset-tree of t.

In order to complete the proof of Lemma 49, we have to construct ϕ_u for $u \in T$. We construct ϕ_u by induction on the depth of u in T. For the root, take an arbitrary f-computation as ϕ_u. For other nodes, let v be the predecessor of u in T. By the definition of T, there is i such that $u \in T_{i+1} \setminus T_i$. and by our construction there exists an f-computation ϕ' on $t_{\geq v}$ such that $\phi'(u) = f$. By induction assumption there exists ϕ_v which assigns f to all nodes $w : w \in T, w \leq v$. The computation $\phi_u := \phi_v \circ_u \phi'$ fulfills the requirement of Lemma 49. □

The next lemma is easily derived from the König Lemma.

Lemma 50 *If T is a full-binary subset-tree of t, then there is a full-binary subset-tree $T' \subseteq T$ such that for each $u \in T'$ with T'-successors v_1, v_2, and for each $q \in Q$ such that $t_{\geq u} \in L(\mathcal{A}_q)$, there is a computation $\phi \in ACC(\mathcal{A}_q, t_{\geq u})$ which passes through F on the paths of $t_{\geq u}$ from u to v_1 and from u to v_2.*

Proof *We first prove the following claim:*

Claim 50.1 *Let $\mathcal{A} = (Q, \Sigma, Q_I, \delta, F)$ be a BTA, $t \in L(\mathcal{A}_q)$, and $\phi \in ACC(\mathcal{A}_q, t)$. Then there is $d := d(\phi) \in \mathbb{N}$ such that for every node v' of depth greater than d, ϕ enters F at the path from the root to v'.*

Proof of Claim 50.1 *Let $U := \{u \in \{l, r\}^* \mid \forall w \leq u : \phi(w) \notin F\}$ and assume, for the sake of contradiction, that U is infinite. It is clear that U is downward closed under ancestor relation, and therefore, by the König Lemma, U contains an infinite branch π such that $\forall w \in \pi : \phi(w) \notin F$ - a contradiction to ϕ being an accepting computation. Therefore U is finite, and we conclude that Claim 50.1 holds for $d := Depth(U) + 1$.* ■

We are now ready to prove Lemma 50, by constructing a binary subset-tree $T' \subseteq T$ which fulfills the requirement. We will first define T'_k for all $k \in \mathbb{N}$, and then define $T' := \bigcup_{k \in \mathbb{N}} T'_k$. Let T'_0 be the tree which consists of the root of T (a single node). T'_{k+1} will be constructed from T'_k in the following way:

Let u be a leaf of T'_k of minimal depth, and let $Q' := \{q \mid t_{\geq u} \in L(\mathcal{A}_q)\}$. For each $q \in Q'$, let $\phi_q \in ACC(\mathcal{A}_q, t_{\geq u})$. By Claim 50.1, for each $q \in Q'$ there is $d_{u,q} \in \mathbb{N}$ such that the computation ϕ_q enters F on the paths from u to each node of depth greater than $d_{u,q}$. Define $d_u = \max\{d_{u,q} \mid q \in Q'\}$.

Let $v_1, v_2 \in T$ such that $v_1, v_2 > u$, $v_1 \perp v_2$ and the distances of v_1 and v_2 from u are greater than d_u. Notice that ϕ_q visits a final state on the paths from u to v_1 and from u to v_2. We now define $T'_{k+1} = T'_k \cup \{v_1, v_2\}$.

Notice that for each i, T'_{i+1} is obtained from T'_i by adding two children to the minimal leaf of T'_i. Therefore, $T' := \bigcup_{k \in \mathbb{N}} T'_k$ is a full-binary subset tree.

Let $u, v_1, v_2 \in T'$ such that v_1, v_2 are T'-successors of u. By the definition of T', v_1, v_2 were both added in the same iteration k, and for each q such that $t_{\geq u} \in L(\mathcal{A}_q)$, there is an accepting computation of $\mathcal{A}_q$ on $t_{\geq u}$ which passes through F on both paths from u to v_1 and from u to v_2. Therefore, T' fulfills the requirement. □

Lemma 51 *Let $(T, \leq)$ be the full-binary tree, Σ be a finite alphabet, and $\sigma : \{l, r\}^* \to \Sigma$ be a labeling function. Then, there are $v_1, v_2 > u$ such that $v_1 \perp v_2$ and $\sigma(v_1) = \sigma(u) = \sigma(v_2)$.*

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

The next Lemma, together with Lemma 44, shows that if a BTA is not countably ambiguous and 2(a) and 2(b) of Proposition 22 do not hold, then 2(c) holds. This implies the (1) ⇒ (2) direction of Proposition 22.

Lemma 52 *Let $\mathcal{A}$ be a BTA 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 22 do not hold. Then, there are three nodes $u, v_1, v_2 \in \{l, r\}^*$ such that a branching pattern for $\mathcal{A}_f$ is realized at v_1, v_2 in $t_{\geq u}$.*

Proof *Let T be the full-binary subset-tree of t, guaranteed by Lemma 49. By applying Lemma 50 to T, we obtain a full-binary subset-tree $T' \subseteq T$. Define a labeling of T' by $\sigma(v) = \{\phi(v) | \phi \in ACC(\mathcal{A}_f, t)\}$ for each $v \in T'$. This is a labeling by a finite alphabet. Therefore, by Lemma 51, we have nodes $v_1, v_2 > u$ such that $v_1 \perp v_2$ and $\sigma(u) = \sigma(v_1) = \sigma(v_2) = Q'$. 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 Lemma 49. 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 27(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 50, 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 the 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. Let ϕ_{v_1} be as in Lemma 49. $\phi_{v_1}(v) = f$ if $v_1 \geq v \in T$. We have $\phi'(v_1) = \phi_{v_1}(v_1) = f = \phi'(\epsilon) = \phi_{v_1}(\epsilon)$, $\phi'(u) = q \neq f = \phi_{v_1}(u)$, and $\epsilon < u < v_1$. Therefore, by Lemma 28, we conclude that $\mathcal{A}$ is not countably branch ambiguous - contradiction. The proof of $\phi_q(v_2) \neq f$ is similar.□

8. 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 a BTA has an ambiguous transition pattern. The following Lemma easily follows from Definition 12 of the branch automaton.

Lemma 53 *Let $\mathcal{A}_B$ be the branch automaton of $\mathcal{A}$. Assume that $r_i \in Q^{l+1}$ for $i = 1, \ldots, k$ are runs of $\mathcal{A}_B$ on $u = (\sigma_1, d_1, S_1) \ldots (\sigma_l, d_l, S_l) \in \Sigma_B^*$. Then for $i = 1, \ldots, l$ there are $S'_i \subseteq S_i$ such that $|S'_i| \leq k$ and r_i for $i = 1, \ldots, k$ are runs of $\mathcal{A}_B$ on $u = (\sigma_1, d_1, S'_1) \ldots (\sigma_l, d_l, S'_l)$.*

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

Now, we are ready to prove Lemma 17 and Proposition 14.

Proof (Proof of Lemma 17) *For each p_1 and p_2, items 1 and 2 of Definition 15 can be tested in polynomial time. There is a q-ambiguous pattern in $\mathcal{A}$, 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 (Proof Sketch of Proposition 14) *The degree of ambiguity of BWA is characterized by the forbidden patterns in Theorem 9. 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 53, $\mathcal{A}_B$ has these patterns iff $\mathcal{A}_B^{(3)}$ has them, and this 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.* □

9. Computability of a Branching Pattern

Here we prove Proposition 23. In Sect. 9.1 we show that if $\mathcal{A}$ has a branching pattern, then it has a branching pattern over (R, f), where R has at most two states. Sect. 9.2 presents a polynomial time algorithm to verify if $\mathcal{A}$ has a branching pattern with at most two states.

9.1. Reduction to Small Branching Patterns

In Sect. 7.1 we assigned to each branching pattern M a BTA $\mathcal{A}_M$ 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. The transition function defined next will help to describe the properties of the accepting computations of $\mathcal{A}_M$.

Definition 54 (Transition function of branching pattern) *Let M be a branching pattern for $\mathcal{A}$ over (R, f) with $\tau_M : R \to R \times R$ and a tuple $(q_1, q_2) \in R \times R$. Its transition function $\delta_M : (\{f\} \cup R) \times \{l, r\} \to R$ is defined as follows:*

$$\delta_M(f, d) := \begin{cases} q_1 & \text{if } d = l \\ q_2 & \text{if } d = r \end{cases}$$

For $p \neq f$ with $\tau_M(p) = (q', q'')$ we define:

$$\delta_M(p, d) := \begin{cases} q' & \text{if } d = l \\ q'' & \text{if } d = r \end{cases}$$

δ_M is naturally extended to a function $\delta_M : R \times \{l, r\}^+ \to R$ by $\delta_M(q, d \cdot w) := \delta_M(\delta_M(q, d), w)$ for all $w \in \{l, r\}^*$ and $d \in \{l, r\}$.

The following lemma follows from the definition of the transition relation of $\mathcal{A}_M$:

Lemma 55 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 $s = d_1 \dots d_k \in \{l, r\}^+$, and let ϕ_s be an accepting computation of $\mathcal{A}_M$ from f such that $\phi_s(d_1 \dots d_i) = f$ for every $i \leq k$. Then for every $w \in \{l, r\}^*$: (a) if $d_i = l$ then $\phi_s(d_1 \dots d_{i-1} r w) = \delta_M(f, lw)$ and (b) if $d_i = r$ then $\phi_s(d_1 \dots d_{i-1} l w) = \delta_M(f, rw)$.*

Lemma 56 *Assume a branching pattern M for $\mathcal{A}$ over (R, f) is realized. Let $l_M(q) := \delta_M(q, l)$ and $r_M(q) := \delta_M(q, r)$ for all $q \in R$. Then:*

1. *If l_M maps R to $Q_0 \subsetneq R$, then a branching pattern for $\mathcal{A}$ over (Q_0, f) is realized. Dually, if r_M maps R to $Q_1 \subsetneq R$ 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 (R, f) is realized in a tree t at nodes u, v by computations ϕ_1, ϕ_2, $\{\phi_q \mid q \in R\}$.*

(1) Assume l_M maps R to $Q_0 \subsetneq R$. Let $t' := (t \circ_u t) \circ_v t$. Define the following computations on t': $\phi'_1 := (\phi_1 \circ_u \phi_2) \circ_v \phi_{q_1}$ and $\phi'_2 := (\phi_2 \circ_u \phi_{q_2}) \circ_v \phi_2$. For each $q \in Q_0$ with $\tau_M(q) = (p_1, p_2)$, let $\phi'_q := (\phi_q \circ_u \phi_{p_2}) \circ_v \phi_{p_1}$. Let $u' := u \cdot v$ and $v' = v \cdot v$ be two nodes of t'. By Lemma 3 we have $\phi'_1, \phi'_2 \in ACC(\mathcal{A}_f, t)$ and $\forall q \in Q_0 : \phi'_q \in ACC(A_q, t')$. Notice that $\phi'_1(u') = \phi'_1(u \cdot v) = \phi_2(v) = f$, $\phi'_2(v') = \phi'_2(v \cdot v) = \phi_2(v) = f$, and from the construction it follows that $\phi'_1(v'), \phi'_2(u'), \phi'_q(u'), \phi'_q(v') \in \{\phi_q(v) \mid q \in R\} = \{\delta_M(q, l) \mid q \in R\} \subseteq Q_0$. 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 $u' = u \cdot v$ and from the root to $v' = v \cdot v$. 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 the definition of δ_M:

i $\alpha_i := \underbrace{\phi_1 \circ_u (\phi_1 \circ_u (\dots \circ_u \phi_1) \dots)}_{i \text{ times}}$ *is an accepting computation of $\mathcal{A}_f$ on t_i^u. It assigns f to node u^i.*

ii $\beta_i := \underbrace{\phi_2 \circ_v (\phi_2 \circ_v (\dots \circ_v \phi_2) \dots)}_{i \text{ times}}$ *is an accepting computation of $\mathcal{A}_f$ on t_i^v. It assigns f to node v^i.*

iii *Let* $q_0 \in R$ *and* $q_i := \delta_M(q_0, r^i)$. *Then* $\phi^{r^i}_{q_0} := \phi_{q_0} \circ_u (\phi_{q_1} \circ_u (\cdots \circ_u \phi_{q_{i-1}}) \dots)$ *is an accepting computation of* $\mathcal{A}_{q_0}$ *on* t^u_i, *and* $\phi^{r^i}_{q_0}(u^j) = q_j$ *for* $j \leq i$.

iv *Let* $q'_0 \in R$ *and* $q'_i := \delta_M(q'_0, l^i)$. *Then* $\phi^{l^i}_{q'_0} := \phi_{q'_0} \circ_v (\phi_{q'_1} \circ_v (\cdots \circ_v \phi_{q'_{i-1}}) \dots)$ *is an accepting computation of* $\mathcal{A}_{q'_0}$ *on* t^v_i, *and* $\phi^{l^i}_{q'_0}(v^j) = q'_j$ *for* $j \leq i$.

Let $q' := \phi_1(v)$ *and* $q'' := \phi_2(u)$. *From* ***i*** *and* ***iv***, *it follows that* $\phi'_1 := (\phi_1 \circ_u \alpha_{k-1}) \circ_v \phi^{l^{k-1}}_{q'}$ *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 conclude that* $\phi'_2 := (\phi_2 \circ_u \phi^{r^{k-1}}_{q''}) \circ_v \beta_{k-1}$ *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* $p \in R$ *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 conclude 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 conclude that a branching pattern for* $\mathcal{A}$ *over* $(\{p_1, p_2\}, f)$ *is realized in* t' *at* u^k, v^k *by the computations* ϕ'_1, ϕ'_2, ϕ'_{p_1}, *and* ϕ'_{p_2}, *as requested.*

(3) Let M *over* (R, f) *be a realizable branching pattern for* $\mathcal{A}$ *such that the cardinality of* R *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* $|Q_0| < |R|$. *Hence, both* l_M *and* r_M *are bijections. Therefore, by item 2 and minimality of* $|R|$, *we obtain* $|R| \leq 2$. □

9.2. Small Branching Patterns are Computable in Polynomial Time

In this subsection we will show a polynomial time algorithm for deciding whether there are R and f such that $|R| \leq 2$, and a branching pattern for $\mathcal{A}$ over (R, f) is realized.

For every t over Σ and $u_1, u_2 \in \{l, r\}^*$, define a tree $t' := t(u_1, u_2)$ over the alphabet $\Sigma' := \Sigma \times \Sigma_{u_1} \times \Sigma_{u_2}$ 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 $t_{u_i}(w) = 1$ iff $w = u_i$ for $i = 1, 2$

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

- A BTA $\mathcal{A}_{nodes}$ which accepts t' iff $t' = t(u_1, u_2)$ and $u_1 \perp u_2$.
- A BTA $\mathcal{A}_{q,q_1,q_2}$ 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.
- A BTA $\mathcal{A}^l_{f,q}$ 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 $\phi(u_2) = q$.
- A BTA $\mathcal{A}^r_{f,q}$ 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 56, $\mathcal{A}$ has a realizable branching pattern iff there exists a realizable branching pattern over (R, f), $\tau_M : R \to R \times R$, $(q_1, q_2) \in R \times R$ with $|R| \leq 2$. For each such branching pattern we define:

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

By the 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.

10. Conclusion and Further Results

We proved that the degree of ambiguity of Büchi tree automata (BTA) is computable in polynomial time. The Büchi acceptance conditions on trees are less expressive than parity, Rabin, Streett, and Muller conditions. Unfortunately, the degrees of ambiguity problem for parity tree automata is co-NP hard [7].

The complementation of finitely ambiguous Büchi automata over ω-words is easier than the complementation of non-deterministic Büchi automata over ω-words [11]. It is interesting to find natural problems for Büchi Tree Automata which are easier for Büchi Tree Automata with small degrees of ambiguity than for arbitrary Büchi Tree Automata.

The degree of ambiguity of a regular language is defined in a natural way. For example, 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 [17]. Over infinite trees there are ambiguous languages [18]. In [16] we proved that over infinite trees there is a hierarchy of degrees of ambiguity: There are k-ambiguous languages for every $k \in \mathbb{N}$; and there are finitely, countably, and uncountably ambiguous languages. The question whether the degree of ambiguity of an infinite tree language is decidable is still open.

Acknowledgement

We would like to thank anonymous referees for their helpful suggestions. Supported in part by Len Blavatnik and the Blavatnik Family foundation.

References

[1] T. Colcombet, Unambiguity in automata theory, in: International Workshop on Descriptional Complexity of Formal Systems, Springer, 2015, pp. 3–18.

[2] Y.-S. Han, A. Salomaa, K. Salomaa, Ambiguity, nondeterminism and state complexity of finite automata, Acta Cybernetica 23 (1) (2017) 141–157.

[3] E. Leiss, Succinct representation of regular languages by Boolean automata, Theoretical computer science 13 (3) (1981) 323–330.

[4] H. Leung, Descriptional complexity of NFA of different ambiguity, International Journal of Foundations of Computer Science 16 (05) (2005) 975–984.

[5] R. E. Stearns, H. B. Hunt III, On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata, SIAM Journal on Computing 14 (3) (1985) 598–611.
[6] J. Jirásek, G. Jirásková, J. Šebej, Operations on unambiguous finite automata, in: International Conference on Developments in Language Theory, Springer, 2016, pp. 243–255.
[7] A. Rabinovich, D. Tiferet, Degrees of ambiguity for parity tree automata, in: C. Baier, J. Goubault-Larrecq (Eds.), 29th EACSL Annual Conference on Computer Science Logic (CSL 2021), Vol. 183 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2021, pp. 36:1–36:20.
[8] A. Weber, H. Seidl, On the degree of ambiguity of finite automata, Theoretical Computer Science 88 (2) (1991) 325–349.
[9] H. Seidl, On the finite degree of ambiguity of finite tree automata, Acta Informatica 26 (6) (1989) 527–542.
[10] C. Löding, A. Pirogov, On finitely ambiguous Büchi automata, in: Developments in Language Theory - 22nd International Conference, DLT 2018, Tokyo, Japan, September 10-14, 2018, Proceedings, 2018, pp. 503–515.
[11] A. Rabinovich, Complementation of finitely ambiguous Büchi automata, in: International Conference on Developments in Language Theory, Springer, 2018, pp. 541–552.
[12] V. Bárány, Ł. Kaiser, A. Rabinovich, Expressing cardinality quantifiers in monadic second-order logic over trees, Fundamenta Informaticae 100 (1-4) (2010) 1–17.
[13] A. Rabinovich, D. Tiferet, Degrees of ambiguity of Büchi tree automata, in: 39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2019), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
[14] W. Thomas, Automata on infinite objects, in: Formal Models and Semantics, Elsevier, 1990, pp. 133–191.
[15] D. Perrin, J.-É. Pin, Infinite words: automata, semigroups, logic and games, Vol. 141, Academic Press, 2004.
[16] A. Rabinovich, D. Tiferet, Ambiguity hierarchy of regular infinite tree languages, in: 45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, Vol. 170 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 80:1–80:14.
[17] A. Arnold, Rational ω-languages are non-ambiguous, Theoretical Computer Science 26 (1-2) (1983) 221–223.
[18] A. Carayol, C. Löding, D. Niwiński, I. Walukiewicz, Choice functions and well-orderings over the infinite binary tree, Open Mathematics 8 (4) (2010) 662–682.