Acta Informatica 26, 527–542 (1989)

On the Finite Degree of Ambiguity of Finite Tree Automata

Helmut Seidl
Fachbereich Informatik, Universität des Saarlandes, D-6600 Saarbrücken, Federal Republic of Germany

Summary. The degree of ambiguity of a finite tree automaton A, $\mathrm{da}(A)$, is the maximal number of different accepting computations of A for any possible input tree. We show: it can be decided in polynomial time whether or not $\mathrm{da}(A)<\infty$. We give two criteria characterizing an infinite degree of ambiguity and derive the following fundamental properties of an finite tree automaton A with n states and rank $L>1$ having a finite degree of ambiguity: for every input tree t there is a input tree t_1 of depth less than $2^{2^n}\cdot n!$ having the same number of accepting computations; the degree of ambiguity of A is bounded by $2^{2^{2\cdot\log(L+1)\cdot n}}$

0. Introduction

Generalizing a result of [5, 8, 9] from finite word automata to finite tree automata we showed in [7] that, for any fixed constant m it can be decided in polynomial time whether or not two m-ambiguous finite tree automata are equivalent. Since the equivalence problem of finite tree automata is logspace complete in deterministic exponential time in general, this result justifies our special interest in the class of finitely ambiguous finite tree automata. In this paper we continue the investigations of [7].

In [11] it is shown that it can be decided in polynomial time whether or not the degree of ambiguity of a finite word automaton is finite. For this a criterion (IDA) is given characterizing an infinite degree of ambiguity. Moreover, this paper proves an upper bound $5^{n/2}\cdot n^n$ for the maximal degree of ambiguity of a finitely ambiguous finite word automaton A having n states. Using an estimation of Baron [2] Kuich slightly improves this upper bound [5]. In [12] the analysis of finitely ambiguous finite word automata is completed by proving a non-ramification lemma which allows for every word w to construct a word w' of length less than $2^{2^n}\cdot n!$ having the same number of accepting computation paths.

In this paper we extend the methods of [11, 12] to finite tree automata. For a finite tree automaton A we employ the branch automaton A_B. A_B is a finite word automaton canonically constructed from A which accepts the set of all branches of trees in $L(A)$. A_B allows to formulate two reasons (T1) and (T2) for A to be infinitely ambiguous. The second one originates in an appropriate extension of the criterion (IDA) of [11] whereas the first one has no analogon in the word case. We prove a non-ramification lemma for finite tree automata. We apply this lemma to prove: if the branch automaton A_B of a finite tree automaton A with n states neither complies with (T1) nor with (T2) then for every input tree t there is a input tree t_1 of depth less than $2^{2n} \cdot n!$ having the same number of accepting computations as t. Since the number of computations for a tree of bounded depth is bounded, this proves: $\mathrm{da}(A) < \infty$ iff A_B doesn't comply with (T1) or (T2). Since the criteria (T1) and (T2) are testable in polynomial time, it follows that it can be decided in polynomial time whether or not the degree of ambiguity of a finite tree automaton is finite.

Finally, we investigate the maximal number of accepting computations of a finitely ambiguous finite tree automaton A for a given tree t. Now, it no longer suffices to analyse the set of traces of the set of accepting computations for t on a single branch. We estimate the number of nodes in t where an accepting computation of A for t "leaves" the first strong connectivity component of the state set of A. This allows to perform an induction on the number of strong connectivity components yielding $\mathrm{da}(A) < \infty$ iff $\mathrm{da}(A) < 2^{2^{2 \cdot \log(L+1) \cdot n}}$ where n is the number of states and L is the rank of A. (As usual, log denotes the logarithm with base 2). A simple example shows that this upper bound is tight up to a constant factor in the highest exponent.

1. General Notations and Concepts

In this section we give basic definitions and state some fundamental properties. A ranked alphabet Σ is the disjoint union of alphabets $\Sigma_0, \ldots, \Sigma_L$. The rank of $a \in \Sigma$, $\mathrm{rk}(a)$, equals m iff $a \in \Sigma_m$. T_Σ denotes the free Σ-algebra of (finite ordered Σ-labeled) trees, i.e. T_Σ is the smallest set T satisfying (i) $\Sigma_0 \subseteq T$, and (ii) if $a \in \Sigma_m$ and $t_0, \ldots, t_{m-1} \in T$, then $a(t_0, \ldots, t_{m-1}) \in T$. Note: (i) can be viewed as the subcase of (ii) where $m=0$.

The depth of a tree $t \in T_\Sigma$, $\mathrm{depth}(t)$, is defined by $\mathrm{depth}(t)=0$ if $t \in \Sigma_0$, and $\mathrm{depth}(t) = 1 + \max\{\mathrm{depth}(t_0), \ldots, \mathrm{depth}(t_{m-1})\}$ if $t = a(t_0, \ldots, t_{m-1})$ for some $a \in \Sigma_m$, $m > 0$.

The set of nodes of t, $S(t)$ is the subset of $\mathbb{N}_0^*$ defined by $S(t) = \{\varepsilon\} \cup \bigcup_{j=0}^{m-1} j \cdot S(t_j)$ where $t = a(t_0, \ldots, t_{m-1})$ for some $a \in \Sigma_m$, $m \geqq 0$. t defines maps $\lambda_t(\_)\colon S(t) \to \Sigma$ and $\sigma_t(\_)\colon S(t) \to T_\Sigma$ mapping the nodes r of t to their labels or the subtrees of t with root r, respectively. We have

$$\lambda_t(r) = \begin{cases} a & \text{if } r = \varepsilon \\ \lambda_{t_j}(r') & \text{if } r = j \cdot r' \end{cases}$$

and

$$\sigma_t(r) = \begin{cases} t & \text{if } r = \varepsilon \\ \sigma_{t_j}(r') & \text{if } r = j \cdot r'. \end{cases}$$

We need the notion of substitution of subtrees. Let t, $t_1 \in T_\Sigma$ and $r \in S(t)$. Then $t[t_1/r]$ denotes the tree obtained from t by replacing the subtree with root r with t_1.

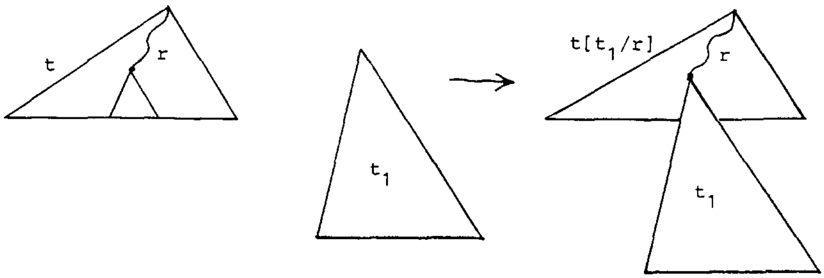

Fig. 1

A finite tree automaton (abbreviated: FTA) is a quadrupole $A=(Q, \Sigma, Q_I, \delta)$ where:

Q is a finite set of states,
$Q_I \subseteq Q$ is the set of initial states,
$\Sigma = \Sigma_0 \cup \ldots \cup \Sigma_L$ is a ranked alphabet, and
$\delta \subseteq \bigcup_{m=0}^{L} Q \times \Sigma_m \times Q^m$ is the set of transitions of A.

$\mathrm{rk}(A) = \max\{\mathrm{rk}(a) \mid a \in \Sigma\}$ is called the rank of A.

Let $t = a(t_0, \ldots, t_{m-1}) \in T_\Sigma$ and $q \in Q$. A q-computation of A for t consists of a transition $(q, a, q_0 \ldots q_{m-1}) \in \delta$ for the root and q_j-computations of A for the subtrees t_j, $j \in \{0, \ldots, m-1\}$. Especially, for $m=0$, there is a q-computation of A for t iff $(q, a, \varepsilon) \in \delta$. Formally, a q-computation ϕ of A for t can be viewed as a map ϕ: $S(t) \to Q$ satisfying (i) $\phi(\varepsilon) = q$ and (ii) if $\lambda_t(r) = a \in \Sigma_m$, then $(\phi(r), a, \phi(r \cdot 0) \ldots \phi(r \cdot (m-1))) \in \delta$. ϕ is called accepting computation of A for t, if ϕ is a q-computation of A for t with $q \in Q_I$. For $t \in T_\Sigma$ and $q \in Q$ $\Phi_{A,q}(t)$ denotes the set of all q-computations of A for t, $\Phi_{A,Q_I}(t)$ denotes the set of all accepting computations of A for t. If A is known from the context, we will omit A in the index of Φ.

For any $r \in S(t)$ and any q-computation $\phi \in \Phi_q(t)$ let ϕ_r denote the subcomputation of A for the subtree $\sigma_t(r)$ of t induced by ϕ, i.e. ϕ_r is defined by $\phi_r(r') = \phi(r r')$. Furthermore, we need the notion of a partial q-computation. Assume $t \in T_\Sigma$, $r \in S(t)$ and q, $q_1 \in Q$. A map ϕ: $(S(t) \setminus r \cdot S(\sigma_t(r)) \cup \{r\} \to Q$ is called partial q-computation of A for t relative to q_1 at node r, if

- $\phi(\varepsilon) = q$; $\phi(r) = q_1$; and
- $\lambda_t(r') = a \in \Sigma_m$ implies $(\phi(r'), a, \phi(r' 0) \ldots \phi(r'(m-1)) \in \delta$ for all $r' \notin r \cdot S(\sigma_t(r))$.

If $q \in Q_I$, then ϕ is called accepting partial computation of A for t relative to q_1 at r. The set of all partial q-computations of A for t relative to q_1 at r is denoted by $\Phi^p_{A,q,q_1}(t, r)$. The set of all accepting partial computations of A

for t relative to q_1 at r is denoted by $\Phi^p_{A,Q_I,q_1}(t, r)$. Again, if A is known from the context we omit A in the index.

Finally, we define the ambiguity of A for a tree t, $\mathrm{da}_A(t)$, as the number of different accepting computations of A for t.
Note: $\mathrm{da}_A(t)$ is finite for every $t \in T_\Sigma$.

The (tree) language accepted by A, $L(A)$ is defined by

$$L(A) = \{t \in T_\Sigma \mid \mathrm{da}_A(t) \neq 0\}.$$

The degree of ambiguity of A, $\mathrm{da}(A)$ is defined by

$$\mathrm{da}(A) = \sup\{\mathrm{da}_A(t) \mid t \in T_\Sigma\}.$$

A is called

- unambiguous, if $\mathrm{da}(A) \leqq 1$;
- ambiguous, if $\mathrm{da}(A) > 1$;
- finitely ambiguous, if $\mathrm{da}(A) < \infty$; and
- infinitely ambiguous, if $\mathrm{da}(A) = \infty$.

For describing our algorithms we use Random Access Machines (RAM's) with the uniform cost criterion, see [1] or [6] for precise definitions and basic properties. For measuring the computational costs of our algorithms relative to the size of an input automaton, we define the size of A, $|A|$, by

$$|A| = \sum_{(q, a, q_0 \ldots q_{m-1}) \in \delta} (m+2).$$

An FTA $A = (Q, \Sigma, Q_I, \delta)$ is called reduced, if

- $Q \times \{a\} \times Q^m \cap \delta \neq \emptyset$ for all $m \geqq 0$ and $a \in \Sigma_m$, and
- $\exists t \in T_\Sigma$, $\phi \in \Phi_{Q_I}(t)$: $q \in \mathrm{im}(\phi)$ for all $q \in Q$.[1]

The following fact is wellknown:

Proposition 1.1. *For every* FTA $A = (Q, \Sigma, Q_I, \delta)$ *there is an* FTA $A_r = (Q_r, \Sigma_r, Q_{r,I}, \delta_r)$ *with the following properties:*

(1) $Q_r \subseteq Q$, $Q_{r,I} \subseteq Q_I$, $\delta_r \subseteq \delta$;

(2) A_r *is reduced*;

(3) $L(A_r) = L(A)$; *and*

(4) $\mathrm{da}(A_r) = \mathrm{da}(A)$.

A_r *can be constructed from* A *by a* RAM *in time* $O(|A|)$. □

Actually, the construction of A_r is analogous to the reduction of a contextfree grammar.

[1] $\mathrm{im}(\phi)$ denotes the image of the map ϕ

Proposition 1.1 can be used to decide in polynomial time whether or not $L(A)$ is empty. The next proposition shows that it also can be decided in polynomial time whether or not A is unambiguous.

Proposition 1.2. *Given* FTA A, *one can decide in time* $O(|A|^2)$ *whether or not* $\mathrm{da}(A) > 1$.

Proof. Assume $A = (Q, \Sigma, Q_I, \delta)$. Define an FTA $A^{(2)} = (Q^{(2)}, \Sigma, Q_I^{(2)}, \delta^{(2)})$ by

$$Q^{(2)} = Q^2 \cup Q \times \{\#\},$$

$$Q_I^{(2)} = \{(p, q) \in Q_I^2 \mid p \neq q\} \cup \{(q, \#) \mid q \in Q_I\},$$

$$\begin{aligned}\delta^{(2)} = {} & \{((p, q), a, (p_0, q_0) \dots (p_{m-1}, q_{m-1})) \mid (p, a, p_0 \dots p_{m-1}), (q, a, q_0 \dots q_{m-1}) \in \delta\} \\ & \cup \{((q, \#), a, (q_0, q_0) \dots (q_{j-1}, q_{j-1})(q_j, \#)(q_{j+1}, q_{j+1}) \dots (q_{m-1}, q_{m-1})) \mid \\ & \quad (q, a, q_0 \dots q_{m-1}) \in \delta, 0 \leqq j \leqq m-1\} \\ & \cup \{((q, \#), a, (p_0, q_0) \dots (p_{m-1}, q_{m-1})) \mid \\ & \quad (q, a, p_0 \dots p_{m-1}), (q, a, q_0 \dots q_{m-1}) \in \delta, p_0 \dots p_{m-1} \neq q_0 \dots q_{m-1}\}.\end{aligned}$$

An accepting computation ϕ of $A^{(2)}$ for some $t \in T_\Sigma$ behaves as follows:

– ϕ simulates two accepting computations of A for t; meanwhile

– $\#$ is "pushed down" along a branch of t; $\#$ disappears at the first node where a difference between the two simulated computations of A occurs.

Therefore,

$$(*) \quad L(A^{(2)}) = \{t \in T_\Sigma \mid \mathrm{da}_A(t) > 1\}.$$

A formal proof of $(*)$ is omitted. Proposition 1.1 can be used to decide whether or not $L(A^{(2)})$ is empty. We have: $|A^{(2)}| \leqq 3|A|^2$, and $A^{(2)}$ can be constructed in time $O(|A|^2)$. Thus, the result follows. □

Note: The construction in the proof of Proposition 1.2 is a simplified version (of a special case) from the construction in [7, Theorem 5.1]. Especially, we don't need any assumption about the rank of A.

As usual, a finite word automaton is defined as a 5-tuple $M = (Q, \Gamma, \delta, Q_I, Q_F)$ where

Q is a finite set of states;
Γ is a finite alphabet;
$Q_I \subseteq Q$ is the set of initial states;
$Q_F \subseteq Q$ is the set of final states; and
$\delta \subseteq Q \times \Gamma \times Q$ is the transition relation of M.

A word $\pi = q_0 x_1 q_1 \dots q_{m-1} x_m q_m \in Q(\Gamma Q)^*$ with $x_j \in \Gamma$ and $q_j \in Q$ is called computation path of M for $w = x_1 \dots x_m$ from q_0 to q_m if $(q_{j-1}, x_j, q_j) \in \delta$ for all $j \in \{1, \dots, m\}$. π is said to start in q_0 and end in q_m. The set of all computation paths of M for w from q_0 to q_m is denoted by $\Pi_{M, q_0, q_m}(w)$. A computation path π of M for w is called accepting, if π starts in an initial state and ends in a final state. The set of all accepting paths of M for w is denoted by $\Pi_{M, Q_I, Q_F}(w)$. If M is known from the context, we omit M in the index of Π.

Two computation paths π_1, π_2 of M for w_1 and w_2 respectively can be composed to a computation path $\pi=\pi_1\cdot\pi_2$ of M for $w_1 w_2$ if π_2 starts in the same state in which π_1 ends. Accordingly, if $w\in\Gamma^*$ and $w=w_1 w_2$ is a factorization of w, then every computation path π of M for w can (uniquely) be broken up into computation paths π_1 for w_1 and π_2 for w_2 where $\pi=\pi_1\cdot\pi_2$.

The language $L(M)$ accepted by M is defined by

$$L(M)=\{w\in\Gamma^* \mid \text{there is an accepting computation path of } M \text{ for } w\}.$$

For the ranked alphabet Σ let Σ_B be the (ordinary) alphabet

$$\Sigma_B=\{(a,j)\mid m>0,\ a\in\Sigma_m,\ j\in\{0,\ldots,m-1\}\}.$$

The set $B(t)$ of branches of a tree t is defined by $B(t)=\{\varepsilon\}$ if $t=a\in\Sigma_0$ and $B(t)=\bigcup_{j=0}^{m-1}(a,j)\cdot B(t_j)$ if $t=a(t_0,\ldots,t_{m-1})$ for some $a\in\Sigma_m$, $m>0$.

Note: The sequence of the second components of the symbols of a branch forms a leaf, whereas the sequence of the first components gives the labels on the path in t from the root of t to this leaf (omitting the label of the leaf itself). A prefix $w=(a_1,j_1)\ldots(a_k,j_k)$ of a branch of t is called path in t. A subtree $\sigma_t(rj)$ of t is called associated to the path w if $r=j_1\ldots j_\kappa$ for some $\kappa<k$ and $j\neq j_{\kappa+1}$.

For a given reduced FTA $A=(Q,\Sigma,Q_I,\delta)$ we define the branch automaton A_B. A_B is the finite word automaton defined by $A_B=(Q,\Sigma_B,\delta_B,Q_I,Q_F)$ where $Q_F=\{q\in Q\mid \exists\, a\in\Sigma_0\colon (q,a,\varepsilon)\in\delta\}$, and the transition relation δ_B is obtained from δ by: $(q,a,q_0\ldots q_{k-1})\in\delta$ implies $(q,(a,j),q_j)\in\delta_B$ for all $j\in\{0,\ldots,k-1\}$.

Since A is assumed to be reduced, it follows that every $q\in Q$ also lies on an accepting computation path of A_B. By [3, Prop. 4.9] we have

$$L(A_B)=\{v\in\Sigma_B^*\mid \exists\, t\in L(A)\colon v \text{ branch of } t\}.$$

Assume $t\in T_\Sigma$, ϕ is a q-computation of A for t, and $w=(a_1,j_1)\ldots(a_k,j_k)$ is a path in t. The trace of ϕ on w is the computation path ϕ_w of A_B for w with $\phi_w=q_0(a_1,j_1)\,q_1\ldots(a_k,j_k)\,q_k$ where $q_\kappa=\phi(j_1\ldots j_\kappa)$ for all $\kappa\in\{0,\ldots,k\}$.

2. Characterizing an Infinite Degree of Ambiguity

In this section we give a complete characterization of those FTA's having an infinite degree of ambiguity. In terms of the branch automaton corresponding to a FTA A we state two reasons (T1) and (T2) for an infinite degree of ambiguity of A. Both (T1) and (T2) are decidable in polynomial time. We formulate a non-ramification lemma for FTA's. This lemma enables us to prove: if the branch automaton of a FTA neither satisfies (T1) nor (T2), then for every tree t there is a tree of depth less than $2^{2^n}\cdot n!$ having the same number of accepting computations. Since the number of different accepting computations for a tree of depth at most $2^{2^n}\cdot n!$ is bounded by some constant, we conclude that (T1) and (T2) precisely characterize an infinite degree of ambiguity.

For the following, $A=(Q, \Sigma, Q_I, \delta)$ is a fixed reduced FTA with n states. For an arbitrary state $q\in Q$, A_q denotes the FTA $A_q=(Q, \Sigma, \{q\}, \delta)$.

Proposition 2.1. *If A_B satisfies* (T 1), *then* $\mathrm{da}(A)=\infty$:

(T 1) $\exists p, q, q_j\in Q\ \exists w_1, w_2\in\Sigma_B^*, (a, j)\in\Sigma_B\ \exists \pi_1\in\Pi_{p,q}(w_1), \pi_2\in\Pi_{q_j,p}(w_2)$:
(T 1.1) *or* (T 1.2) *is true:*
(T 1.1) *There exist two different transitions*
$(q, a, q_0^{(i)} \dots q_{j-1}^{(i)}\, q_j\, q_{j+1}^{(i)} \dots q_{k-1}^{(i)})\in\delta$,
$i=1, 2$, *with* $L(A_{q_{j'}^{(1)}})\cap L(A_{q_{j'}^{(2)}})\neq\emptyset$ *for all* $j'\neq j$.
(T 1.2) *There exists a transition* $(q, a, q_0 \dots q_j \dots q_{k-1})\in\delta$ *with* $\mathrm{da}(A_{q_{j'}})>1$ *for some* $j'\neq j$.

Whether or not A_B satisfies (T 1) *can be decided in polynomial time.*

Proof. Assume A_B satisfies (T 1). Since A is reduced, we can construct a tree $t\in T_\Sigma$, $r_0=r_1 r_2\in S(t)$, $r_2\neq\varepsilon$ and $\phi^{(0)}, \phi^{(1)}\in\Phi_{Q_I}(t)$ such that:

(1) $\phi^{(0)}(r_1)=\phi^{(0)}(r_1 r_2)=\phi^{(1)}(r_1)=\phi^{(1)}(r_1 r_2)$ and
(2) $\exists r'j$ prefix of $r_2\ \exists j'\neq j$: $\phi^{(0)}_{r_1 r' j'}\neq\phi^{(1)}_{r_1 r' j'}$.

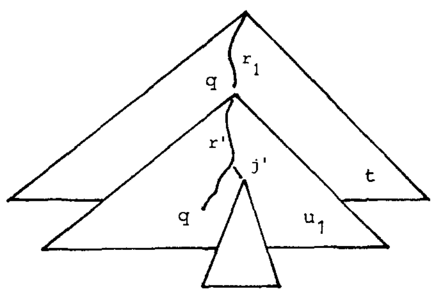

Fig. 2

Define $u_1=\sigma_t(r_1)$ and $u_k=u_1[u_{k-1}/r_2]$ for $k>1$. Let $t_k=t[u_k/r_1]$. We show: $\mathrm{da}(t_k)\geqq 2^k$. Intuitively, t_k is obtained from t by iterating "u_1 minus $\sigma_{u_1}(r_2)$" k times. By (2), $\phi^{(0)}$ and $\phi^{(1)}$ differ at the iterated part. Furthermore, we can mend the corresponding subcomputations of $\phi^{(0)}$ and $\phi^{(1)}$ together to obtain accepting computations for t_k. Since for different occurrences of the iterated part we can independently choose subcomputations either according to $\phi^{(0)}$ or according to $\phi^{(1)}$, we get at least 2^k accepting computations for t_k.

Formally, the 2^k different accepting computations for t_k are constructed as follows. For every $\mu\in\{0, \dots, 2^k-1\}$ with binary representation $\mu_{k-1} \dots \mu_0$ define $\bar\phi^{(\mu)}: S(t_k)\to Q$ by

$$\bar\phi^{(\mu)}(r)=\begin{cases}\phi^{(\mu_j)}(r_1 r') & \text{if } r=r_1 r_2^j r' \text{ and } j<k\\ \phi^{(0)}(r_1 r_2 r') & \text{if } r=r_1 r_2^k r'\\ \phi^{(0)}(r) & \text{else}\end{cases}$$

where j in the exponent of line 1 is the maximal number j' such that $r_1 r_2^{j'}$ is a prefix of r.

By the assumptions under (1), $\bar{\phi}^{(\mu)} \in \Phi_{Q_I}(t)$ for all μ. If $\mu \neq \mu'$, then there is some $\kappa \in \{0, \ldots, k-1\}$ such that μ and μ' differ at the digits μ_κ and μ'_κ of their binary representations. For every prefix $r'j$ of r_2 and $j' \neq j$ we have: $\bar{\phi}^{(\mu)}_{r_1 r_2^\kappa r' j'} = \phi^{(\mu_\kappa)}_{r_1 r' j'}$ & $\bar{\phi}^{(\mu')}_{r_1 r_2^\kappa r' j'} = \phi^{(\mu'_\kappa)}_{r_1 r' j'}$. Therefore by (1), $\bar{\phi}^{(\mu)} \neq \bar{\phi}^{(\mu')}$.

Our algorithm testing (T 1) works as follows:

(1) Mark all pairs $(q_1, q_2) \in Q^2$ with $L(A_{q_1}) \cap L(A_{q_2}) \neq \emptyset$; time: $O(|A|^2)$.

(2) For all pairs of different transitions $(q, a, q_0^{(i)} \ldots q_{k-1}^{(i)}) \in \delta$, $i = 1, 2$, mark all $(q, (a, j), q_j^{(1)}) \in \delta_B$ such that $q_j^{(1)} = q_j^{(2)}$ and $L(A_{q_{j'}^{(1)}}) \cap L(A_{q_{j'}^{(2)}}) \neq \emptyset$ for all $j' \neq j$; time: $O(|A|^2)$.

(3) Mark every $q \in Q$ where A_q is ambiguous; time: $O(|A|^2)$.

(4) For all $(q, a, q_0 \ldots q_{k-1}) \in \delta$ mark all transitions $(q, (a, j), q_j) \in \delta_B$ where $\exists j' \neq j$: $A_{q'}$ is ambiguous; time: $O(|A|)$.

(5) Test whether there is a cyclic computation path of A_B which contains a marked transition; time: $O(|A|)$.

Together we get an $O(|A|^2)$-algorithm. Therefore, the result follows. □

A set of transitions $\{(q^{(i)}, (a, j), q_j^{(i)}) \in \delta_B \mid i \in I\}$ for some index set I, is said to match if there are transitions $(q^{(i)}, a, q_0^{(i)} \ldots q_j^{(i)} \ldots q_{m-1}^{(i)}) \in \delta$, $i \in I$, such that $\bigcap_{i \in I} L(A_{q_{j'}^{(i)}}) \neq \emptyset$ for all $j' \neq j$.

A set of computation paths $\{q_0^{(i)}(a_1, j_1)\, q_1^{(i)} \ldots (a_k, j_k)\, q_k^{(i)} \mid i \in I\}$ is said to match if the sets of transitions $\{(q_{\kappa-1}^{(i)}, (a_\kappa, j_\kappa), q_\kappa^{(i)}) \mid i \in I\}$ match for all $\kappa \in \{1, \ldots, k\}$.

Proposition 2.2. *If A_B satisfies* (T 2), *then* $\mathrm{da}(A) = \infty$:

(T 2) $\exists p, q \in Q$, $p \neq q$, $w \in \Sigma_B^+$: $\exists \pi_1 \in \Pi_{p,p}(w)$, $\pi_2 \in \Pi_{p,q}(w)$, $\pi_3 \in \Pi_{q,q}(w)$: π_1, π_2, π_3 *match.*

Whether or not A_B satisfies (T 2) *can be decided in polynomial time.*

Proof. If (T 2) is fullfilled we can construct $t \in T_\Sigma$, $r_0 = r_1 r_2 \in S(t)$ with $r_2 \neq \varepsilon$ and $u_1 = \sigma_t(r_1)$ such that there are:

$$\phi_0 \in \Phi_{Q_I}(t) \quad \text{with} \quad \phi_0(r_1) = p \text{ and } \phi_0(r_1 r_2) = q;$$
$$\phi_1 \in \Phi^P_{p,p}(u_1, r_2), \text{ and}$$
$$\phi_2 \in \Phi^P_{q,q}(u_1, r_2).$$

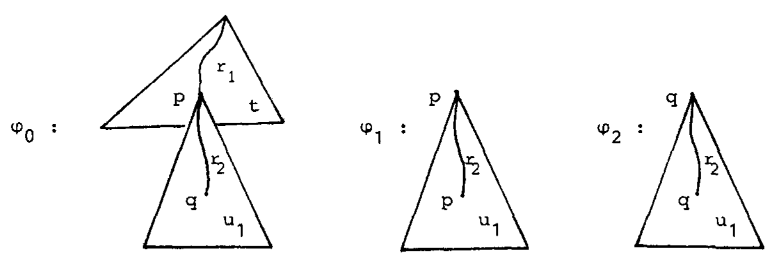

Fig. 3

Define $t_k = t[u_k/r_1]$ where, for $k>1$, $u_k = u_1[u_{k-1}/r_2]$. We show: $\mathrm{da}_A(t_k) \geqq k$. Intuitively, one can construct accepting computations $\phi^{(\kappa)}$, $\kappa \in \{1, \ldots, k\}$, for t_k which accept the first $\kappa-1$ occurrences of "u_1 minus $\sigma_{u_1}(r_2)$" according to ϕ_1, the next occurrence according to ϕ_0 and the remaining $k-\kappa$ occurrences according to ϕ_2. Formally, for $\kappa \in \{1, \ldots, k\}$ we define $\phi^{(\kappa)}$ by

$$\phi^{(\kappa)}(r) = \begin{cases} \phi_1(r') & \text{if } r = r_1 r_2^j r' \text{ and } j < \kappa \\ \phi_0(r_1 r') & \text{if } r = r_1 r_2^\kappa r' \\ \phi_2(r') & \text{if } r = r_1 r_2^j r' \text{ and } \kappa < j < k \\ \phi_0(r_1 r_2 r') & \text{if } r = r_1 r_2^k r' \\ \phi_0(r) & \text{else} \end{cases}$$

where the exponents j and κ in the first three lines are the maximal numbers j' such that $r_1 r_2^{j'}$ is a prefix of r.
Note: $\phi^{(\kappa)} \in \Phi_{Q_I}(t_k)$ for all κ, and if $\kappa > \kappa'$ then $\phi^{(\kappa)}(r_1 r_2^{\kappa-1}) = p \neq q = \phi^{(\kappa')}(r_1 r_2^{\kappa-1})$, and hence $\phi^{(\kappa)} \neq \phi^{(\kappa')}$.

The algorithm:

(1) Construct the labeled graph δ_B^3 with Q^3 as set of vertices and

$$\{(p_1 p_2 p_3, x, q_1 q_2 q_3) \mid (p_i, x, q_i) \in \delta_B \text{ for } i = 1, 2, 3\}$$

as set of edges; time: $O(|A|^3)$.

(2) Mark all $q_1 q_2 q_3 \in Q^3$ where $L(A_{q_1}) \cap L(A_{q_2}) \cap L(A_{q_3}) \neq \emptyset$; time: $O(|A|^3)$.

(3) Mark the edges $((q^{(1)} q^{(2)} q^{(3)}, (a, j), q_j^{(1)} q_j^{(2)} q_j^{(3)})$ in δ_B^3 where the transitions $(q^{(i)}, (a, j), q_j^{(i)})$, $i = 1, 2, 3$, match; time: $O(|A|^3)$.

(4) Construct the subgraph of δ_B^3 which contains only edges corresponding to matching triples of transitions; time: $O(|A|^3)$.

(5) For every pair (p, q) of different states decide whether (p, q, q) is accessible from (p, p, q) w.r.t. to the resulting subgraph of δ_B^3; a straight foreward implementation yields a time bound $O(n^2 \cdot |A|^3)$; however by using the same algorithmic idea as in [10] for deciding the criterion (IDA) for finite word automata one gets a time bound of $O(|A|^3)$.

Together we have an $O(|A|^3)$-time algorithm. □

Thus, (T 1) and (T 2) give two polynomially decidable reasons for the infinite degree of ambiguity of A.
(T 2) is the extension of the criterion (IDA) in [11] characterizing the infinite degree of ambiguity of finite word automata (additionally we demand the three computation paths of A_B from q to q, q to p and p to p to match), whereas (T 1) solely arises from the tree structure. We now formulate the non-ramification lemma for FTA's.

Assume $t \in T_\Sigma$, and $w = (a_1, j_1) \ldots (a_K, j_K)$ is a branch of t. By $G_t(w)$ we denote the acyclic digraph which describes all traces of accepting computations of A for t on w. $G_t(w) = (V, E)$ is defined as follows.

Vertices:
$V \subseteq Q \times \{0, \dots, K\}$ is the set of all (q, k) such that

$$\exists \phi \in \Phi_{Q_I}(t): \phi(j_1 \dots j_k) = q.$$

Edges:
$E \subseteq V \times V$ is the set of all pairs $((q, k), (q', k+1))$ such that

$$\exists \phi \in \Phi_{Q_I}(t): \phi(j_1 \dots j_k) = q \;\&\; \phi(j_1 \dots j_k j_{k+1}) = q'.$$

If $((q_0, k), (q_1, k+1))((q_1, k+1), (q_2, k+2)) \dots ((q_{d-1}, k+d-1)), (q_d, k+d))$ is a path in $G_t(w)$ where $k \in \{0, \dots, K-1\}$ and $d > 0$, then the following holds:

(1) $q_0(a_{k+1}, j_{k+1})\, q_1(a_{k+2}, j_{k+2}) \dots q_{d-1}(a_{k+d}, j_{k+d})\, q_d$ is a computation path of A_B for $(a_{k+1}, j_{k+1}) \dots (a_{k+d}, j_{k+d})$;

(2) there is a partial q_0-computation ϕ of A for $\sigma_t(j_1 \dots j_k)$ relative to q_d at node $j_{k+1} \dots j_{k+d}$ such that $\phi(j_{k+1} \dots j_{k+\kappa}) = q_\kappa$, $\kappa \in \{0, \dots, d\}$.

Proposition 2.3 (Non-Ramification Lemma for FTA's). *Assume A_B does not comply with* (T2). *Let $t \in T_\Sigma$, let w be a branch of t, and $G_t(w) = (V, E)$. For $k \in \{0, \dots, |w|\}$ define $D_k = \{q \in Q \mid (q, k) \in V\}$. If $D_k = D_{k+d}$ for some $d \geqq 1$, then*

(1) *for every vertex (q, k) in V there is exactly one path in $G_t(w)$ starting in (q, k), and*

(2) *for every vertex $(q', k+d)$ in V there is exactly one path in $G_t(w)$ ending in $(q', k+d)$.*

Proof. Assertion (2) is an immediate consequence of (1). Therefore, it suffices to prove (1). Let $D = D_k = D_{k+d}$, $w = (a_1, j_1) \dots (a_K, j_K)$, and $y = (a_{k+1}, j_{k+1}) \dots (a_{k+d}, j_{k+d})$.
For every k and $d > 0$ all paths in $G_t(w)$ from $D \times \{k\}$ to $D \times \{k+d\}$ describe matching computation paths of A_B for y. Let Π denote the set of these computation paths. For a contradiction assume there are two different paths $\bar{\pi}_1$ and $\bar{\pi}_2$ from (q, k) to the vertices $(\bar{q}_1, k+d)$ and $(\bar{q}_2, k+d)$ respectively. We will use the computation paths for y in Π to construct the forbidden situation of (T2).

Since for every state q' in D there is a computation path in Π ending in q' we can "follow the way back" from q, i.e. we can find a sequence $(\pi^{(j)})_{j \in \mathbb{N}}$ of computation paths $\pi^{(j)}$ in Π such that $\pi^{(1)}$ ends in q, and for all $j \in \mathbb{N}$, $\pi^{(j+1)}$ ends in the same state in which $\pi^{(j)}$ starts. Since $\# Q < \infty$, there are $s < s'$ such that $\pi^{(s)}$ and $\pi^{(s')}$ start in the same state. Call this state p. Define $\pi_0 = \pi^{(s')} \pi^{(s'-1)} \dots \pi^{(s+2)} \pi^{(s+1)}$ and $j_0 = s' - s$.

Accordingly, since for every state q' in D there is a computation path in Π starting in q' we can "prolong" the paths $\bar{\pi}_1$ and $\bar{\pi}_2$ beyond $\bar{q}_1$ and $\bar{q}_2$ respectively, i.e. we can find sequences $(\pi_i^{(j)})_{j \in \mathbb{N}}$, $i = 1, 2$, of computation paths $\pi_i^{(j)}$ in Π such that $\pi_i^{(1)}$ start in $\bar{q}_i$, and for all $j \in \mathbb{N}$ $\pi_i^{(j)}$ end in the same state in which $\pi_i^{(j+1)}$ start. Since $\# Q < \infty$, there are $s_i < s_i'$ such that $\pi_i^{(s_i)}$ and $\pi^{(s_i')}$ end in the same state. Call this state q_i. Define $j_i = s + s_i + 1$ and $j_{ii} = s_i' - s_i$, $\pi_i = \pi^{(s)} \dots \pi^{(1)} \bar{\pi}_i \pi_i^{(1)} \dots \pi_i^{(s_i)}$ and $\pi_{ii} = \pi_i^{(s_i+1)} \dots \pi_i^{(s_i')}$. We have:

- π_0 is a computation path for y^{j_0} from p to p;
- π_i is a computation path for y^{j_i} from p to q_i, $i=1, 2$;
- π_{ii} is a computation path for $y^{j_{ii}}$ from q_i to q_i, $i=1, 2$.

By appropriate pumping of π_0 and pumping and "shifting" of the cyclic computation paths π_{ii} we may assume w.l.o.g. $j_0=j_1=j_2=j_{11}=j_{22}=j$. Thus, we have the following situation.

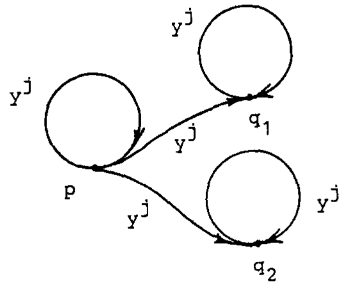

Fig. 4

Since $\pi_0, \pi_1, \pi_2, \pi_{11}, \pi_{22}$ are the composition of the same number of matching computation paths for y, $\pi_0, \pi_1, \pi_2, \pi_{11}, \pi_{22}$ match as well.

We distinguish two cases:

Case I. $p=q_1=q_2$. Since $\tilde{\pi}_1 \neq \tilde{\pi}_2$ we have $\pi_1 \neq \pi_2$. Therefore, there is a factorization $y^j=y_1 y_2$, states $\tilde{p} \neq \tilde{q}$ and decompositions $\pi_i=\tilde{\pi}_i^{(1)}\tilde{\pi}_i^{(2)}$ such that $\tilde{\pi}_1^{(1)} \in \Pi_{p,\tilde{p}}(y_1)$, $\tilde{\pi}_1^{(2)} \in \Pi_{\tilde{p},p}(y_2)$, $\tilde{\pi}_2^{(1)} \in \Pi_{p,\tilde{q}}(y_1)$, and $\tilde{\pi}_2^{(2)} \in \Pi_{\tilde{q},p}(y_2)$.

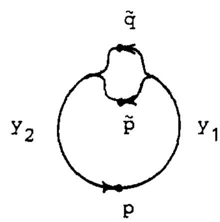

Fig. 5

Then we have $\tilde{\pi}_1^{(2)}\tilde{\pi}_1^{(1)} \in \Pi_{\tilde{p},\tilde{p}}(y_2 y_1)$; $\tilde{\pi}_1^{(2)}\tilde{\pi}_2^{(1)} \in \Pi_{\tilde{p},\tilde{q}}(y_2 y_1)$; and $\tilde{\pi}_2^{(2)}\tilde{\pi}_2^{(1)} \in \Pi_{\tilde{q},\tilde{q}}(y_2 y_1)$. Since $\tilde{p} \neq \tilde{q}$, it follows that A_B satisfies (T 2).

Case II. Case I is not true. Then at least one of the states q_1, q_2 is different from p. Assume this is q_i. Then the computation paths π_0, π_i and π_{ii} satisfy the assumptions of (T 2). Therefore, in both cases we arrive at a contradiction. □

Theorem 2.4. *Assume that A_B neither satisfies* (T1) *nor* (T2). *Then, for every tree $t \in T_\Sigma$, there is a tree t_1 with* $\mathrm{depth}(t_1) < 2^{2^n} \cdot n!$ *such that* $\mathrm{da}_A(t) = \mathrm{da}_A(t_1)$.

As a consequence of Theorem 2.4 we get the main theorem of this section:

Theorem 2.5. *Assume A is a reduced* FTA. *Then*

(1) $\mathrm{da}(A) = \infty$ *iff A_B satisfies* (T1) *or* (T2).

(2) *It can be decided in polynomial time whether or not* $\mathrm{da}(A) < \infty$.

(3) *If* $\mathrm{da}(A) < \infty$, *then there is a tree $t \in T_\Sigma$ with* $\mathrm{depth}(t) < 2^{2^n} \cdot n!$ *such that* $\mathrm{da}(A) = \mathrm{da}_A(t)$. □

Note: One can easily construct FTA's such that the corresponding branch automata satisfy any of the criteria (T1.1), (T1.2) or (T2) but none of the others. Therefore, the characterization given in (1) is irredundant. Note further: (3) implies a triple exponential upper bound on the degree of ambiguity of finitely ambiguous FTA's. However, we will prove a (tight) double exponential upper bound in Sect. 3.

Proof of 2.4. For every tree $t \in T_\Sigma$ and position $r \in S(t)$, define $\mathrm{ACC}_t(r)$ as the set of all states q for which there is a accepting partial computation of A for t relative to q at r, i.e.

$$\mathrm{ACC}_t(r) = \{q \in Q \mid \Phi^P_{I,q}(t, r) \neq \emptyset\}.$$

Define $\mathrm{DER}_t(r)$ as the set of all q for which there is a q-computation of A for $\sigma_t(r)$, i.e.

$$\mathrm{DER}_t(r) = \{q \in Q \mid \Phi_q(\sigma_t(r)) \neq \emptyset\}.$$

Assume A_B neither satisfies (T1) nor (T2). Let t denote an arbitrary tree of T_Σ. We show: if there is a branch $w = (a_1, j_1) \ldots (a_K, j_K)$ of t of length $K \geqq n! \cdot 2^{2^n}$, then we can find a tree $t_1 \in T_\Sigma$ with fewer nodes such that $\mathrm{da}_A(t) = \mathrm{da}_A(t_1)$. This implies the assertion of Theorem 2.4.

W.l.o.g. we assume $\mathrm{da}_A(t) > 0$, i.e. $\Phi_{Q_I}(t) \neq \emptyset$. Consider the acyclic graph $G_t(w) = (V, E)$ and for $k \in \{0, \ldots, K\}$ the sets $D_k = \{q \in Q \mid (q, k) \in V\}$.
Note: $D_k = \mathrm{ACC}_t(j_1 \ldots j_k) \cap \mathrm{DER}_t(j_1 \ldots j_k)$.

Assume $K \geqq 2^{2^n} \cdot n!$. Then there exist $B_1, B_2 \subseteq Q$ and a set $I \subseteq \{0, \ldots, K\}$ with $\# I \geqq n! + 1$ such that $B_1 = \mathrm{ACC}_t(j_1 \ldots j_k)$ and $B_2 = \mathrm{DER}_t(j_1 \ldots j_k)$ for all $k \in I$. It follows that there are $k_1 < k_2$ in I such that

$$B_1 = \mathrm{ACC}_t(j_1 \ldots j_{k_1}) = \mathrm{ACC}_t(j_1 \ldots j_{k_2})$$

and

$$B_2 = \mathrm{DER}_t(j_1 \ldots j_{k_1}) = \mathrm{DER}_t(j_1 \ldots j_{k_2})$$

and for every $q \in B_1 \cap B_2$ there is a unique path in $G_t(w)$ from (q, k_1) to (q, k_2). Define $r_1 = j_1 \ldots j_{k_1}$, $r_2 = j_{k_1+1} \ldots j_{k_2}$, $u = \sigma_t(r_1 r_2)$, and $t_1 = t[u/r_1]$. We prove: $\mathrm{da}_A(t) = \mathrm{da}_A(t_1)$.

$\mathrm{da}_A(t) \leqq \mathrm{da}_A(t_1)$: For every $\phi \in \Phi_{Q_I}(t)$, we have $\phi(r_1) = \phi(r_1 r_2)$. Therefore, ϕ gives rise to an accepting computation $\bar{\phi}$ for t_1 where $\bar{\phi}$ is defined by:

$$\bar{\phi}(r) = \begin{cases} \phi(r_1 r_2 r') & \text{if } r = r_1 r' \\ \phi(r) & \text{else.} \end{cases}$$

We have to show that this map is injective. Assume ϕ_1, ϕ_2 are two accepting computations of A for t with $\phi_1(r_1)=\phi_2(r_1)$. By the construction of r_1 and r_2, ϕ_1 and ϕ_2 agree at every node $r_1 r'j$ where $r'j$ is a prefix of r_2. Since A_B does not satisfy (T1), we furthermore have that ϕ_1 and ϕ_2 also agree at every subtree of t with root $r_1 r'j'$, $j' \neq j$. It follows: if $\bar{\phi}_1=\bar{\phi}_2$ then also $\phi_1=\phi_2$. This proves the injectivity.

$\mathrm{da}_A(t) \geqq \mathrm{da}_A(t_1)$: Assume $\bar{\phi}$ is an accepting computation of A for t_1 and $\bar{\phi}(r_1)=p$. Then $p \in \mathrm{ACC}_{t_1}(r_1) \cap \mathrm{DER}_{t_1}(r_1)$. Observe $\mathrm{ACC}_{t_1}(r_1)=\mathrm{ACC}_t(r_1)=B_1$ and $\mathrm{DER}_{t_1}(r_1)=\mathrm{DER}_t(r_1 r_2)=B_2$ which by the construction of r_1 and r_2 also equals $\mathrm{DER}_t(r_1)$. Thus, $p \in D_{k_1}$, and there is a path in $G_t(w)$ from (p, k_1) to (p, k_2). Therefore, there is a partial p-computation of A for $\sigma_t(r_1)$ relative to p at node r_2. It follows that we can extend $\bar{\phi}$ to an accepting computation ϕ for t. Clearly, two different accepting computations $\bar{\phi}_1$, $\bar{\phi}_2$ for t_1 give rise to two different accepting computations for t. This proves the stated inequality. □

3. A Tight Upper Bound for the Finite Degree of Ambiguity

In this section we prove the following theorem.

Theorem 3.1. *Assume A is a reduced* FTA *with n states and rank $L>1$. If A_B does not comply with* (T1) *or* (T2), *then* $\mathrm{da}(A) < 2^{2^{2\cdot\log(L+1)\cdot n}}$.

Theorem 3.1 gives an alternative proof for the correctness of our characterization of an infinite degree of ambiguity by the criteria (T1) and (T2). The following example shows that the upper bound for the maximal degree of ambiguity of a finitely ambiguous FTA given in Theorem 3.1 is optimal up to a constant factor in the highest exponent.

Theorem 3.2. *For every $n \geqq 3$ and $L \geqq 2$ there is a finitely ambiguous* FTA $A_{n,L}$ *with n states and rank L such that* $\mathrm{da}(A_{n,L})=2^{2^{\log(L)\cdot(n-2)}}$.

Proof. Define $A_{n,L}$ by $A_{n,L}=(\{1, \ldots, n\}, \Sigma, \{1\}, \delta_{n,L})$ where $\Sigma_0=\{\#\}$, $\Sigma_L=\{o\}$ and $\Sigma_m=\emptyset$ else, and

$$\delta_{n,L}=\{(i, o, (i+1)^L) \mid 1 \leqq i \leqq n-3\} \cup \{n-2\} \times \{o\} \times \{n-1, n\}^L$$
$$\cup \{(n-1, \#, \varepsilon), (n, \#, \varepsilon)\}.$$

Then $L(A_{n,L})=\{\Delta_{n,L}\}$ where $\Delta_{n,L}$ denotes the complete L-ary tree of depth $n-2$ whose inner nodes are labeled with o and whose leafs are labeled with $\#$. Since $L(A_{n,L})$ is finite, the degree of ambiguity of $A_{n,L}$ is finite, too. There is a bijection between $\Phi_{\{1\}}(\Delta_{n,L})$ and the set of all words of length L^{n-2} over a two letter alphabet. Therefore, $\mathrm{da}(A_{n,L})=2^{L^{n-2}}$. □

We now prove Theorem 3.1. Let $A=(Q, \Sigma, Q_I, \delta)$ be a fixed reduced FTA with $n>0$ states and rank $L>1$ (the case $n=0$ is trivial).

We partition the set Q according to accessibility. For states $p, q \in Q$, we say q is accessible from p (short: $p \rightarrow_A q$) iff there is a computation path of A_B from p to q. The equivalence relation $\leftrightarrow_A$ on Q is defined by $p \leftrightarrow_A q$ iff

$p \rightarrow_A q$ and $q \rightarrow_A p$. The equivalence classes of Q w.r.t. $\leftrightarrow_A$ are denoted by $Q_1, \ldots, Q_k$. They are also called the strong connectivity components of Q. W.l.o.g. we assume for $p \in Q_i$ and $q \in Q_j$, $p \rightarrow_A q$ impies $i \leq j$.

We first deal with FTA's having just one initial state. Define $d(k)$ to be the maximal degree of ambiguity of a reduced FTA A with 1 initial state, rank L, at most n states and at most k strong connectivity components such that A_B does not comply with (T 1) or (T 2). Observe: in order to prove Theorem 3.1 it suffices to compute an upper bound for $d(n)$.

So, for our FTA A assume $Q_I = \{q_I\}$. Since A is reduced, q_I is in Q_1. Let t be a fixed tree in $L(A)$. We classify the q_I-computations of A for t relative to Q_1. The following observation is crucial.

Fact 3.3. Assume A_B does not comply with (T 1), and $\text{da}(A) > 1$. Assume $\phi \in \Phi_{q_I}(t)$ and $r \in S(t)$. If $\phi(r) \in Q_1$, then there is at most one j such that $\phi(rj) \in Q_1$.

Proof. For a contradiction assume there are $j_1 \neq j_2$ such that $q_1 = \phi(rj_1) \in Q_1$ and $q_2 = \phi(rj_2) \in Q_1$. Since Q_1 is strongly connected, we have $q_1 \rightarrow_A q_I$. Therefore, since A_B does not comply with (T 1), A_{q_2} is unambiguous. Since also $q_2 \rightarrow_A q_I$, $A_{q_I} = A$ must be unambiguous as well: contradiction. □

Fact 3.3 already implies:

Fact 3.4. If $L > 1$, $d(1) = 1$. □

Thus, if A_B does not comply with (T 1) and $\text{da}(A) > 1$, then for every accepting computation ϕ of A for t, there is a unique maximal trace of ϕ such that every state on it lies in Q_1. This trace is denoted by $\pi_1(\phi)$.

The following fact is an easy consequence of Propositions 2.1 and 2,2:

Fact 3.5. Assume A_B does not comply with (T 1) or (T 2). Assume ϕ, ϕ' are two q_I-computations for t where $\pi_1(\phi)$ is a computation path for w, $\pi_1(\phi')$ is a computation path for w' and $v = (a_1, j_1) \ldots (a_K, j_K)$ is the maximal common prefix of w and w'. If $\phi(j_1 \ldots j_K) = \phi'(j_1 \ldots j_K)$, then the following holds:

(1) ϕ and ϕ' agree on v, i.e. $\phi_v = \phi'_v$;

(2) ϕ and ϕ' also agree on every subtree of t associated to v, i.e. if $\sigma_t(r)$ is a subtree of t associated to v then $\phi_r = \phi'_r$. □

Now assume $\text{da}(A) > 1$, and Q has $k > 1$ strong connectivity components. We want to perform an induction on k. Therefore, we calculate the cardinality of the set $\{\pi_1(\phi) \mid \phi \in \Phi_{q_I}(t)\}$. Let w be a branch of t and $G_t(w) = (V, E)$ be defined as in Sect. 2. Let $J(w)$ denote the set of all i such that $i = |w|$ or there is an edge $((q, i), (q', i+1))$ in $G_t(w)$ with $q \in Q_1$ and $q' \notin Q_1$. Applying the non-ramification lemma for FTA's we get:

Fact 3.6. Assume A_B does not comply with (T 2). Then for every branch w of t, $\# J(w) < 2^n$.

Proof. For $i \in J(w)$ define $D_i = \{q \in Q \mid (q, i) \in V\}$. Assume $\# J(w) \geq 2^n$. Then there exist $i < i'$ such that $D_i = D_{i'}$. By the non-ramification Lemma 2.3 there is exactly one path in $G_t(w)$ starting in (q, i) for every q in D_i. Since $Q_1 \cap D_i = Q_1 \cap D_{i'}$ and every vertex (p', i') of $G_t(w)$ with $p' \in Q_1$ only can be reached from a vertex (p, i) with $p \in Q_1$, we conclude that for every edge $((q, i), (q', i+1))$ in $G_t(w)$, $q \in Q_1$

Fact 3.6 is the appropriate extension of a corresponding result in [11] for finite word automata. However, to apply Fact 3.6 we need the following additional observation.

Fact 3.7. Assume A_B does not comply with (T1). Assume ϕ, ϕ' are different q_I-computations of A for t where $\pi_1(\phi)$ is a computation path for v, $\pi_1(\phi')$ is a computation path for v', u is the maximal common prefix of v and v', and v is a prefix of the branch w. Then the following holds:

(1) $|v| \in J(w)$;

(2) $|u| \in J(w)$.

Proof. Assertion (1) is immediately clear from the definition of $\pi_1(\_)$.

Ad(2): W.l.o.g. $v \neq u \neq v'$. Assume $u = (a_1, j_1) \dots (a_m, j_m)$, $v = u(a, j)\, u_1$ and $v' = u(a, j')\, u'_1$. By Fact 3.3 there is at most one $\bar{j}$ such that $\phi(j_1 \dots j_m \bar{j}) \in Q_1$. Hence, $\phi(j_1 \dots j_m j') \notin Q_1$. □

Together the Facts 3.4, 3.5, 3.6 and 3.7 allow to estimate the cardinality of the set $\{\pi_1(\phi) \mid \phi \in \Phi_{q_I}(t)\}$.

Lemma 3.8. *Assume* A_B *does not comply with* (T1) *or* (T2). *Assume* da$(A) > 1$. *Then*

$$\#\{\pi_1(\phi) \mid \phi \in \Phi_{q_I}(t)\} < (L+1)^{2^n} \cdot n.$$

Proof. Define $T = \{u \in \Sigma_B^+ \mid \exists\, \phi \in \Phi_{q_I}(t)\colon\ \pi_1(\phi)$ is a computation path for $u\}$. By Fact 3.5, $\#\{\pi_1(\phi) \mid \phi \in \Phi_{q_I}(t)\} \leqq n \cdot \# T$. Consider the smallest superset $\bar{T}$ of T which for every two elements $v, v' \in T$ contains the maximal common prefix of v and v'. The set $\bar{T}$ can be viewed as the set of nodes of a tree $s = (\bar{T}, E)$ where $(v_1, v_2) \in E$ iff

(i) v_1 is a prefix of v_2 different from v_2; and

(ii) there is no v in $\bar{T}$ different from v_1 and v_2 such that v_1 is a prefix of v and v is a prefix of v_2.

By the Facts 3.6 and 3.7 depth$(s) \leqq 2^n$. Moreover, Fact 3.7 implies that every node of s has at most L successors. Therefore, s has less than $(L+1)^{2^n}$ nodes. From this, the result follows. □

Now we are able to prove:

Lemma 3.9. *For every* $k > 1$,

$$\log d(k) < \log(L+1) \cdot 2^n \cdot (L+1)^{k-2} + \log n \cdot (L+1)^{k-1}.$$

Proof. W.l.o.g. assume |da$(A) > 1$. Assume $t \in L(A)$, $w = (a_1, j_1) \dots (a_K, j_K)$ is a path in t, $r = j_1 \dots j_K$, $\sigma_t(r) = a(t_0, \dots, t_{m-1})$, and $q \in Q_1$. Let $\Phi^{(r,q)}$ denote the set of all accepting computations ϕ of A for t such that $\pi_1(\phi)$ is a computation path of A_B for w from q_I to q. By Lemma 3.4 all $\phi \in \Phi^{(r,q)}$ agree on w and on every subtree of t associated to w. They possibly differ in the transition chosen at node r and in the subcomputations chosen for the subtrees t_j, $0 \leqq j \leqq m-1$. By the definition of $\pi_1(\_)$ we may view the set of q'-computations $\{\phi_{rj} \mid \phi \in \Phi^{(r,q)}, \phi(rj) = q'\}$, $j \in \{0, \dots, m-1\}$, as the set of all accepting computations for t_j of a reduced FTA $A'_{q'} = (Q', \Sigma, \{q'\}, \delta')$ where $Q' \subseteq Q \backslash Q_1$ and $\delta' \subseteq \delta$

and Q' has at most $k-1$ strong connectivity components. Since there are at most n^m different transitions applicable at node r, we conclude that $\Phi^{(r,q)} \leqq n^m \cdot d(k-1)^m$. By Lemma 3.8 we get the following inductive inequation for $d(k)$:

$$d(k) < (L+1)^{2^n} \cdot n \cdot n^L \cdot d(k-1)^L.$$

Since by Fact 3.4 $d(1)=1$, the assertion follows. □

Proof of Theorem 3.1. Assume $A=(Q, \Sigma, Q_I, \delta)$ is a reduced FTA with n states and rank $L>1$. W.l.o.g. $n>1$. Assume A_B does not comply with (T 1) or (T 2). Since Q has at most n strong connectivity components and $\# Q_I \leqq n$, we have $\mathrm{da}(A) \leqq n \cdot d(n)$, and therefore by Lemma 3.9,

$$\begin{aligned} \log \mathrm{da}(A) &< \log(L+1) \cdot 2^n \cdot (L+1)^{n-2} + \log n \cdot [(L+1)^{n-1}+1] \\ &\leqq 2^n \cdot (L+1)^{n-1} \cdot \left[\frac{\log(L+1)}{L+1} + \frac{2 \cdot \log n}{2^n}\right] \\ &\leqq 2^n \cdot (L+1)^{n-1} \cdot 2 \\ &\leqq 2^{2 \cdot \log(L+1) \cdot n}. \quad \square \end{aligned}$$

Acknowledgement. I thank Andreas Weber for many fruitful discussions and carefully reading of an earlier version of this paper.

References

1. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The design and analysis of computer algorithms. New York: Addison-Wesley 1974
2. Baron, G.: Estimates for bounded automata. Technische Universität Graz und Österreichische Computer-Gesellschaft, Report 253, part 2, June 1988
3. Courcelle, B.: A representation of trees by languages, part II. Theor. Comput. Sci. **7**, 25–55 (1978)
4. Gecseg, F., Steinby, M.: Tree automata. Budapest: Akademiai Kiado 1984
5. Kuich, W.: Finite automata and ambiguity. Technische Universitaet Graz und Österreichische Computer Gesellschaft, Report 253, part 1, June 1988
6. Paul, W.: Komplexitaetstheorie. Stuttgart: Teubner 1978
7. Seidl, H.: Deciding equivalence of finite tree automata. Proc. STACS'88. Lect. Notes Comput. Sci **349**, 480–492 (1989)
8. Stearns, R., Hunt, H. III: On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. 22th FOCS, pp. 74–81 (1981)
9. Stearns, R., Hunt, H. III: On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. SIAM J. Comput. **14**, 598–611 (1985)
10. Weber, A.: Über die Mehrdeutigkeit und Wertigkeit von endlichen Automaten und Transducern. Doct. Thesis Frankfurt/Main 1987
11. Weber, A., Seidl, H.: On the degree of ambiguity of finite automata. MFCS 1986. Lect. Notes Comput. Sci. **233**, 620–629 (1986)
12. Weber, A., Seidl, H.: On finitely generated monoids of matrices with entries in $\mathbb{N}_0$. Preprint 1988

Received April 28, 1988