SIAM J. COMPUT.
Vol. 19, No. 3, pp. 424-437, June 1990

DECIDING EQUIVALENCE OF FINITE TREE AUTOMATA*

HELMUT SEIDL†

Abstract. It is shown that for every constant m it can be decided in polynomial time whether or not two m-ambiguous finite tree automata are equivalent. In general, inequivalence for finite tree automata is DEXPTIME-complete with respect to logspace reductions, and PSPACE-complete with respect to logspace reductions, if the automata in question are supposed to accept only finite languages. For finite tree automata with weights in a field R, a polynomial time algorithm is presented for deciding ambiguity-equivalence, provided R-operations and R-tests for 0 can be performed in constant time. This result is used to construct an algorithm deciding ambiguity-inequivalence of finite tree automata in randomized polynomial time. Finally, for every constant m it is shown that it can be decided in polynomial time whether or not a given finite tree automaton is m-ambiguous.

Key words. finite tree automata, equivalence, complexity, ambiguity, semirings

AMS(MOS) subject classifications. 68D35, 68D30, 68F10, 68C25, 20M35

0. Introduction. Finite tree automata were defined in the late sixties by Thatcher and Wright and Doner as generalizations of finite automata accepting word languages to finite state devices for tree languages [ThaWri68], [Do70]. Their main interest in tree automata was a logical point of view. Finite tree automata can describe classes of (finite or infinite) models for formulas of monadic theories with multiple successors and therefore can be used to get effective decision procedures for these theories [ThaWri68], [Do70], [Ra69], [Tho84]. For other possible applications see [GeStei84].

In this paper we investigate finite tree automata accepting tree languages of finite trees from a complexity theoretical point of view. Especially, we want to analyze the equivalence problem for finite tree automata. Two finite tree automata A_1 and A_2 are called equivalent, if and only if they accept the same tree language. We find that the inequivalence problem for finite tree automata is logspace complete in DEXPTIME, and the inequivalence problem for finite tree automata that accept only finite languages is still logspace complete in PSPACE. These problems for finite word automata are known to be PSPACE- and NP-complete with respect to logspace reductions, respectively. Actually, our proofs extend proofs of the corresponding results for finite word automata.

Since the equivalence problem for finite tree automata is provably difficult in general, it seems natural to look for adequate subclasses such that equivalence can be decided in polynomial time, at least for automata of such a restricted class. One parameter that attracts attention in this context is the degree of ambiguity. A finite automaton is called m-ambiguous, if for every input there are at most m accepting computations. In [SteHu81], [SteHu85] Stearns and Hunt III give a polynomial time algorithm that decides equivalence of m-ambiguous finite word automata for every fixed constant m. They employ difference equations for their decision procedure. Kuich [Kui88] simplifies their constructions. Although not stated explicitly, Kuich's proof can be used to show that the equivalence problem for m-ambiguous finite word automata is even in NC. Kuich uses semiring automata and the formalism of formal power series. Semiring automata are obtained from ordinary automata by giving weights

* Received by the editors April 25, 1988; accepted for publication (in revised form) May 22, 1989. A previous version of my paper appeared in the proceedings of STACS'89 (Springer, Lecture Notes in Computer Science 349, pp. 480-492).

† Fachbereich Informatik, Universität des Saarlandes, D-6600 Saarbrücken, Federal Republic of Germany.

in some semiring to transitions and initial states. In particular, Kuich shows how the equivalence problem for m-ambiguous word automata can be reduced to the ambiguity-equivalence problem for unambiguous automata with weights in the field of rational numbers $\mathbb{Q}$.

For tree languages the concept of formal power series seems to be much more involved than for word languages [BeReu82]. Therefore, we avoid using this concept, but show how some basic ideas of Kuich can be carried over to tree automata. Thus, we introduce finite tree automata with weights in some semiring R and show how to reduce the equivalence problem for m-ambiguous finite tree automata to the ambiguity-equivalence problem for unambiguous finite tree automata with weights in $\mathbb{Q}$ or even in $\mathbb{Z}_p$ for some prime number $p > m$.

We then consider the problem of deciding ambiguity-equivalence for finite tree automata with weights in a field R. We are able to prove the appropriate generalization of Eilenberg's equality theorem [Ei74, Thm. 8.1] to tree automata, i.e., we give an explicit upper bound for the depth of a witness for ambiguity-inequivalence. Furthermore, analyzing the proof we get a polynomial time algorithm that decides ambiguity-equivalence of finite tree automata with weights in R, provided we are allowed to perform R-operations and R-tests for 0 in constant time. Note that this does not automatically lead to a polynomial time algorithm deciding ambiguity-equivalence for ordinary finite tree automata (viewed as automata with weights in $\mathbb{Q}$), since the only upper bound for the lengths of occurring integers given by the algorithm is exponential in the input size. However, we get a polynomial time algorithm deciding equivalence of m-ambiguous tree automata. As another consequence, we are able to construct a randomized polynomial algorithm deciding ambiguity-inequivalence of arbitrary finite tree automata.

For reasons of completeness we finally show that it is also possible to detect whether or not our polynomial equivalence test is applicable, i.e., it can be decided in polynomial time for every constant m whether or not a given finite tree automaton has a degree of ambiguity less than m.

A subsequent paper [Se89] considers the finite degree of ambiguity for its own sake, showing that it can be decided in polynomial time whether or not the degree of ambiguity of a finite tree automaton is finite. Furthermore, we will give a tight upper bound for the maximal degree of ambiguity of a finitely ambiguous finite tree automaton.

1. General notations and concepts. In this section we give basic definitions and state some fundamental properties. Especially, we show that for the equivalence or ambiguity-equivalence problems of finite tree automata, it suffices to consider finite tree automata of rank $\leqq 2$.

A ranked alphabet Σ is the disjoint union of alphabets $\Sigma_0, \cdots, \Sigma_L$. For $a \in \Sigma$, the rank of a, $rk(a)$, equals m if and only if $a \in \Sigma_m$. 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 $t_0, \cdots, t_{m-1} \in T$, then $a(t_0, \cdots, t_{m-1}) \in T$. Note that (i) can be viewed as the subcase of (ii) with $m = 0$.

The depth of a tree t, $depth(t)$, is defined by $depth(t) = 0$ if $t \in \Sigma_0$, and $depth(t) = 1 + \max\{depth(t_0), \cdots, depth(t_{m-1})\}$ if $t = a(t_0, \cdots, t_{m-1})$ for some $a \in \Sigma_m$, $m > 0$. Let $\mathbb{N}_0$ denote the set of all nonnegative integers, and let $\mathbb{N}_0^*$ denote the set of all finite sequences of nonnegative integers. 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 $t = a(t_0, \cdots, t_{m-1})$ for some $a \in \Sigma_m$ with $m \geqq 0$. t defines a map $\lambda_t(\_): S(t) \to \Sigma$, mapping the nodes of t to their labels. We have

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

A finite tree automaton (abbreviated: FTA) is a quadruple $A = (Q, \Sigma, Q_I, \delta)$, where
Q is a finite set of states,
$Q_I \subseteq Q$ is the set of initial states,
Σ is a ranked alphabet, and
$\delta \subseteq \bigcup_{m \geqq 0} Q \times \Sigma_m \times Q^m$ is the set of transitions of A.
$rk(A) = \max\{rk(a) \mid a \in \Sigma\}$ is called the rank of A.

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

$n_A(t)_q = \#\Phi_{A,q}(t)$ is the number of different q-computations of A for t, the Q-tuple $(n_A(t)_q)_{q \in Q}$ is denoted by $n_A(t)$; finally $da_A(t) = \#\Phi_{A,Q_I}(t)$, the number of different accepting computations of A for t, is called the ambiguity of A for t.

The following proposition is an easy consequence of these definitions.

PROPOSITION 1.1. *Assume* $t = a(t_0, \cdots, t_{m-1}) \in T_\Sigma$. *Then*

$$(1) \quad n_A(t)_q = \sum_{(q, a, q_0 \cdots q_{m-1}) \in \delta} n_A(t_0)_{q_0} \cdot \cdots \cdot n_A(t_{m-1})_{q_{m-1}}$$

$$(2) \quad da_A(t) = \sum_{q \in Q_I} n_A(t)_q. \qquad \square$$

The (tree) language accepted by A, $L(A)$, is defined by $L(A) = \{t \in T_\Sigma \mid \Phi_{A,Q_I}(t) \neq \emptyset\}$. The degree of ambiguity of A, $da(A)$, is defined by $da(A) = \sup\{da_A(t) \mid t \in T_\Sigma\}$. A is called
-unambiguous, if $da(A) \leqq 1$;
-ambiguous, if $da(A) > 1$;
-m-ambiguous, if $da(A) \leqq m$;
-finitely ambiguous, if $da(A) < \infty$; and
-infinitely ambiguous, if $da(A) = \infty$.

Two FTAs A_1, A_2 are called
(1) ambiguity-equivalent (written $A_1 \equiv A_2$), iff $da_{A_1}(t) = da_{A_2}(t)$ for all $t \in T_\Sigma$.
(2) equivalent, iff $L(A_1) = L(A_2)$.

Clearly, $A_1 \equiv A_2$ implies $L(A_1) = L(A_2)$. Moreover, if A_1 and A_2 are unambiguous, then $A_1 \equiv A_2$ if and only if $L(A_1) = L(A_2)$.

For describing our algorithms we will mostly use Random Access Machines (RAMs) with the uniform cost criterion (see [Aho74] or [Paul78] for precise definitions and basic properties). If we allow multiplications, divisions, or the manipulation of

registers not containing nonnegative integers, we will state this explicitly. For measuring the computational costs of our algorithms relative to the size of the input automata in question, we define

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

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

$-\forall m \geqq 0,\ a \in \Sigma_m$: $Q \times \{a\} \times Q^m \cap \delta \neq \varnothing$.

$-\forall q \in Q \quad \exists t \in T_\Sigma,\ \phi \in \Phi_{A,Q_I}(t)$: $q \in \operatorname{im}(\phi)$.[1]

The following proposition is well known:

PROPOSITION 1.2. *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*; *and*

(3) $A_r \equiv A$.

A_r *can be constructed from* A *by a* RAM (*without multiplications*) *in time* $O(|A|)$. □

Actually, the construction of A_r is analogous to the reduction of a context-free grammar.

Next, we observe that we may restrict our attention without loss of generality to automata of rank $\leqq 2$ (nonetheless we always will give the constructions for arbitrary rank L).

Consider the (injective) tree homomorphism θ that maps symbols of rank >2 onto a tree of binary symbols as shown in Fig. 1. One easily proves:

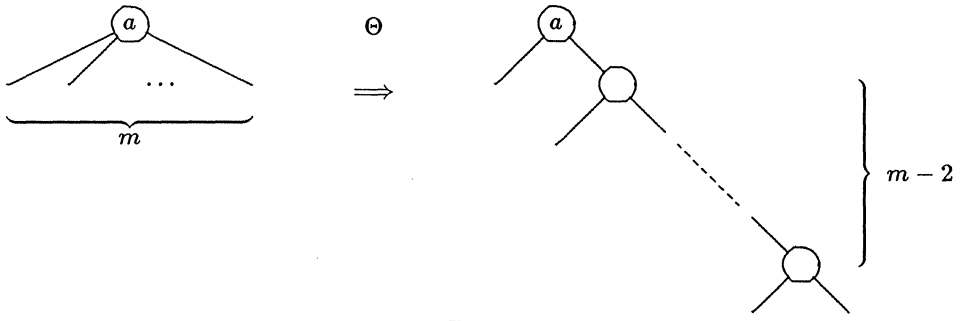

FIG. 1

PROPOSITION 1.3. *For every* FTA $A=(Q, \Sigma, Q_I, \delta)$ *exists an* FTA A' *with the following properties*:

(1) $da_A(t) = da_{A'}(\theta(t))$ *for all* $t \in T_\Sigma$;

(2) $L(A') = \theta(L(A))$.

(3) A' *can be computed from* A *in time* $O(|A|)$. □

2. The complexity of the inequivalence problem. Before we give a polynomial time algorithm that decides equivalence of m-ambiguous FTAs, we analyze the complexity of the inequivalence problems for unrestricted FTAs and FTAs accepting finite languages, respectively. For nondeterministic finite word automata these two problems are known to be PSPACE-complete and NP-complete (with respect to logspace reductions), respectively [MeSto72], [StoMe73]. We find these problems for FTAs to be DEXPTIME-complete and PSPACE-complete, respectively.

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

For the upper bounds we construct Turing machines that simulate the computations of the subset automata corresponding to the FTAs to be tested for inequivalence. For an FTA $A=(Q, \Sigma, Q_I, \delta)$ the corresponding subset automaton $P(A)$ is defined by

$P(A)=(P(Q), \Sigma, \bar{Q}_I, \bar{\delta})$, where

$P(Q)$ denotes the power set of Q,

$\bar{Q}_I=\{Q' \subseteq Q \mid Q' \cap Q_I \neq \emptyset\}$, and

$(Q', a, Q'_0 \cdots Q'_{m-1}) \in \bar{\delta}$ iff $Q'=\{q \in Q \mid \exists q_0 \in Q'_0, \cdots, q_{m-1} \in Q'_{m-1}: (q, a, q_0 \cdots q_{m-1}) \in \delta\}$.

Let $t \in T_\Sigma$. Define $Q(t)=\{q \in Q \mid \Phi_{A,q}(t) \neq \emptyset\}$. Then $Q(t)$ is the unique subset Q' of Q such that there is a Q'-computation of $P(A)$ for t. This uniqueness is usually called "bottom-up determinism." It is well known that $L(P(A))=L(A)$ [Do70]. Especially, $t \notin L(A)$ if and only if $Q(t) \cap Q_I=\emptyset$.

THEOREM 2.1. (1) *For two* FTAs A_1, A_2 *it can be decided in time* $O(2^{c \cdot n \cdot L})$ *whether or not* $L(A_1) \neq L(A_2)$, *where* L *is the maximal rank,* n *is the maximal number of states of* A_1 *and* A_2, *and* $c>0$ *is a suitable constant independent of* n *and* L.

(2) *The inequivalence problem for* FTAs *is hard in* DEXPTIME *with respect to* logspace *reductions.*

Proof. The proof is an appropriate generalization of a proof showing PSPACE-completeness for deciding inequivalence of finite word automata. Where in the word case, the computations of the subset automata could be simulated by a nondeterministic polynomially space bounded Turing machine, we need an alternating Turing machine in the tree case. For the hardness part instead of coding the word problem for nondeterministic linear space bounded Turing machines into the inequivalence problem, we can, in the tree case, encode the word problem for linear space bounded alternating Turing machines into the inequivalence problem.

Ad(1): Construct an alternating Turing machine M that on input (A_1, A_2) simulates topdown the computation of $P(A_1)$ and $P(A_2)$ for a witness for inequivalence. M needs space $O(L \cdot n)$ on its worktape to verify that for the current pair of state sets $(\bar{Q}^{(1)}, \bar{Q}^{(2)})$, $\bar{Q}^{(i)} \subseteq Q_i$, $i=1,2$, guessed $a \in \Sigma$ and guessed tuple of pairs $(\bar{Q}^{(1)}_0, \bar{Q}^{(2)}_0) \cdots (\bar{Q}^{(1)}_{m-1}, \bar{Q}^{(2)}_{m-1})$, $(\bar{Q}^{(i)}, a, \bar{Q}^{(i)}_0 \cdots \bar{Q}^{(i)}_{m-1})$ is a transition of $P(A_i)$, $i=1,2$.

Since by [ChaKoSto81] $\mathrm{ASPACE}(L \cdot n)=\bigcup_{c>0} \mathrm{DTIME}(2^{c \cdot L \cdot n})$, assertion (1) follows.

Ad(2): Assume M is an alternating n-space bounded Turing machine with tape alphabet Γ, and $w \in \Gamma^N$ for a fixed natural number N. We construct a ranked alphabet Σ and an FTA A with alphabet Σ having $O(N)$ states such that $L(A) \neq T_\Sigma$ if and only if M has an accepting computation for w. This construction will be possible in logspace. Thus, every problem in $\mathrm{ASPACE}(n)$ can be reduced to the inequivalence problem of FTAs. By the simulations given in [ChaKoSto81] the result follows.

Without loss of generality we assume that

- M has only one accepting state f and no transitions in state f;
- M has no negating states;
- in M, universal states have always exactly two successors.

Without loss of generality the work tape of M on a computation for w always contains N symbols. Therefore, configurations of M can be denoted by $w_1 \uparrow w_2 q$ where $w_1 w_2 \in \Gamma^N$ is the current inscription of the work tape, $\uparrow \notin \Gamma$ stands left to the position of the read/write head of M, and q represents the current state.

Define Σ by
$\Sigma_0 = \{f\}$,
$\Sigma_1 = \Gamma \cup \{\uparrow\} \cup \{q \mid q \text{ existential state of } M\}$,
$\Sigma_2 = \{q \mid q \text{ universal state of } M\}$, and
$\Sigma_m = \emptyset$ for all $m > 2$.

Note that every accepting computation tree of M for w can be represented by some $t \in T_\Sigma$.

We construct an FTA A such that $L(A) = \{t \in T_\Sigma \mid t \text{ is not an accepting computation tree for } w\}$. Given an input tree t, A nondeterministically performs one of the following tasks:

(1) Test whether t does not start with the initial configuration $\uparrow wq_0$ of a computation of M for w;

(2) Test whether t contains an "ill-formed configuration," i.e., whether t contains a sequence of unary symbols such that there are more or less than N symbols of Γ and more or less than one $\uparrow$ between two states;

(3) Test whether t contains an incorrect transition between two configurations, i.e., whether one of the tape symbols is not copied correctly, whether $\uparrow$ has moved more than one step, or whether there is no transition of M to cause the change of current input symbol, state, or head position;

(4) Test whether t contains a configuration of M with universal state such that the two following configurations in t do not correspond to two different transitions of M.

Each of these tasks can be implemented with $O(N)$ states. Since (a description of) A can be constructed from w in logspace, and from $\text{ASPACE}(n) = \bigcup_{c>0} \text{DTIME}(2^{cn})$ [ChaKoSto81], the result follows. □

THEOREM 2.2. *The inequivalence problem for* FTAs *accepting finite languages is* PSPACE*-complete with respect to* logspace *reductions.*

Similar to the proof of Theorem 2.1, the proof of Theorem 2.2 will be an appropriate generalization of a proof for the NP-completeness of deciding inequivalence of finite word automata accepting finite languages. In the case of finite word automata the bound on the length of the witness for inequivalence allows one to construct an NP-algorithm; in the case of tree automata the bound on the depth of a witness for inequivalence allows one to construct a polynomially space bounded algorithm. Accordingly for the hardness part, instead of satisfiability of conjunctive normal forms that can be encoded into the inequivalence problem for finite word automata accepting finite languages, we can encode arbitrarily quantified conjunctive normal forms into the inequivalence problem for finite tree automata accepting finite languages.

Before giving a detailed proof of Theorem 2.2, we state an immediate corollary.

COROLLARY 2.3. *The inequivalence problem for finitely ambiguous* FTAs *is* PSPACE*-hard.*

Proof. FTAs accepting finite languages form a subclass of the class of finitely ambiguous FTAs. □

Proof of Theorem 2.2. We need the notion of branches of trees. For a ranked alphabet Σ define Σ_B as the (ordinary) alphabet

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

Assume $t \in T_\Sigma$, $r = j_1 \cdots j_k \in S(t)$ is a leaf of t, and $a_1, \cdots, a_k$ is the sequence of labels of the nodes on the path from the root of t to r (omitting the label of the leaf itself), i.e., $a_\kappa = \lambda_t(j_1 \cdots j_{\kappa-1})$ for $\kappa = 1, \cdots, k$. Then $(a_1, j_1) \cdots (a_k, j_k)$ is called the branch of t corresponding to r.

Now assume $A_i = (Q_i, \Sigma, Q_{i,0}, \delta_i)$, $i = 1, 2$, are two FTAs accepting finite languages, $\# Q_i \leqq n$ and $rk(\Sigma) = L$. The computations of $P(A_1)$ and $P(A_2)$ for a witness for inequivalence can be simulated by a nondeterministic Turing machine M that uses its worktape as a pushdown store (with entries of size $O(n)$ to hold pairs of sets of states). M accepts if and only if it has simulated correctly the computations of $P(A_1)$ and $P(A_2)$ on some t in the symmetric difference of $L(A_1)$ and $L(A_2)$. Since $L(A_1)$ and $L(A_2)$ are finite, depth $(t) < n$. Therefore, during an accepting computation of M, the pushdown store never contains more than $L \cdot n$ pairs of sets of states. It follows that M has only polynomial space complexity.

PSPACE-hardness: By [StoMe73] the following set $\mathbf{B}$ is known to be PSPACE-complete with respect to logspace reductions:

$\mathbf{B} = \{\langle x_1, y_1, \cdots, x_k, y_k, B\rangle \mid k \geqq 0, B$ conjunctive normal form containing variables from $x_1, y_1, \cdots, x_k, y_k$ such that $\exists x_1\, \forall y_1 \cdots \exists x_k\, \forall y_k\, B(x_1, y_1, \cdots, x_k, y_k)\}$.

Therefore, fix some $k \geqq 0$, pairwise different variables $x_1, \cdots, x_k, y_1, \cdots, y_k$ and a conjunctive normal form $B(x_1, y_1, \cdots, x_k, y_k)$. We want to construct FTAs A_1, A_2 accepting finite languages such that

$$L(A_1) \neq L(A_2) \quad \text{iff} \quad \langle x_1, y_1, \cdots, x_k, y_k, B\rangle \in \mathbf{B}.$$

Let Σ denote the ranked alphabet with $\Sigma_0 = \{\#\}$, $\Sigma_2 = \{0, 1\}$ and $\Sigma_j = \varnothing$ otherwise. Both the integers 0 and 1 and the corresponding two elements in Σ_2 will be used to represent the truth values "false" and "true," respectively.

Define a set $T^{(k)} \subseteq T_\Sigma$ inductively by

$$T^{(0)} = \{\#\}, \quad \text{and} \quad T^{(k)} = \{a(t_1, t_2) \mid a \in \Sigma_2, t_1, t_2 \in T^{(k-1)}\} \quad \text{for } k > 0.$$

Obviously, $depth(t) = k$ for every $t \in T^{(k)}$. Define an FTA A_2 of size $O(k)$ such that $T^{(k)} = L(A_2)$. Clearly, A_2 can be constructed by a logspace Turing machine.

We construct an FTA A_1 with $L(A_1) \subseteq T^{(k)}$ and $L(A_1) \neq T^{(k)}$ iff $\exists x_1\, \forall y_1 \cdots \exists x_k\, \forall y_k\, B(x_1, y_1, \cdots, x_k, y_k)$.

Given an input tree t, A_1 behaves as follows:

(1) A_1 checks whether t is in $T^{(k)}$ and rejects any other tree. So, without loss of generality assume $t \in T^{(k)}$.

(2) A_1 guesses a clause c of B and a branch $(\xi_1, \eta_1) \cdots (\xi_k, \eta_k)$ and accepts whenever

$$c(\xi_1, \eta_1, \cdots, \xi_k, \eta_k) = 0.$$

A_1 can be constructed in logspace as well.

For the correctness of the construction we observe that $\exists x_1\, \forall y_1 \cdots \exists x_k\, \forall y_k\, B(x_1, y_1, \cdots, x_k, y_k)$ if and only if there is a tree $t \in T^{(k)}$ such that t satisfies (*):

(*) $\quad B(\xi_1, \eta_1, \cdots, \xi_k, \eta_k) = 1$ for all branches $(\xi_1, \eta_1) \cdots (\xi_k, \eta_k)$ of t.

However, by the construction of A_1, A_1 accepts exactly those trees in $T^{(k)}$ that do not satisfy (*). Thus,

$$L(A_1) \neq L(A_2) \quad \text{iff } \exists x_1\, \forall y_1 \cdots \exists x_k\, \forall y_k\, B(x_1, y_1, \cdots, x_k, y_k). \qquad \square$$

3. Semiring automata. We extend our notion of a finite tree automaton by allowing the transitions and initial states to have weights in some semiring R. The advantage of this extension is twofold.

On the one hand the resulting automata have nice algebraic properties that make it possible to "eliminate" finite ambiguity. These properties are studied in this section. On the other hand, the extension enables us to use methods from linear algebra to decide ambiguity-equivalence. This will be investigated in the next section.

A commutative semiring with 0 and 1 is a structure $(R, +, \cdot)$, where 0 and 1 are two different elements of R; $+$ and $\cdot$ are commutative and associative operations on R; and the following equations hold for arbitrary $a, b, c \in R$:

$$a \cdot (b+c) = (a \cdot b) + (a \cdot c), \qquad 1 \cdot a = a, \qquad 0 + a = a, \qquad 0 \cdot a = 0.$$

If $+$ and $\cdot$ are understood, we write R instead of $(R, +, \cdot)$. Moreover, since all semirings we will use are commutative and have 0 and 1, we will call them semirings for short. Assume R is such a semiring. A finite tree automaton with weights in R (short: R-FTA) is a quadruple $A = (Q, \Sigma, I, \delta)$, where

Q is a finite set of states;

Σ is a ranked alphabet;

$I = (I_q)_{q \in Q} \in R^Q$ is the Q-tuple of initial ambiguities; and

δ is a map $\delta : \bigcup_{m \geq 0} Q \times \Sigma_m \times Q^m \to R$ denoting the transition multiplicites.

Let $V = R^Q$ denote the set of Q-tuples of semiring elements. Taking equations (1) and (2) of Proposition 1.1 as a definition, we define

(1) a map $n_A : T_\Sigma \to V$ by $n_A(t) = (n_A(t))_{q \in Q}$, where

$$n_A(t)_q = \sum_{q_0 \cdots q_{m-1} \in Q^m} \delta(q, a, q_0, \cdots, q_{m-1}) \cdot n_A(t_0)_{q_0} \cdot \cdots \cdot n_A(t_{m-1})_{q_{m-1}}$$

for $t = a(t_0, \cdots, t_{m-1})$, and

(2) a map $da_A : T_\Sigma \to R$ by $da_A(t) = \sum_{q \in Q} I_1 \cdot n_A(t)_q$.

$n_A(t)$ is called ambiguity vector of A for t in V; $da_A(t)$ is called the ambiguity of A for t in R. Note that by (1), every $a \in \Sigma_m$ defines a multilinear map $a : V^m \to V$ which in particular yields, when applied to the ambiguity vectors of A for the subtrees t_j, $j = 0, \cdots, m-1$, the ambiguity vector of A for $a(t_0, \cdots, t_{m-1})$.

Finally, the language $L(A)$ accepted by A is defined by $L(A) = \{t \in T_\Sigma \mid da_A(t) \neq 0\}$. Similar to the case of FTAs, the size of the R-FTA A, $|A|$, is defined by

$$|A| = \sum_{\delta(q,a,q_0 \cdots q_{m-1}) \neq 0} (m+2).$$

An R-FTA A is called unambiguous if and only if $da_A(t) \in \{0, 1\}$ for all $t \in T_\Sigma$.

Two R-FTAs A_1, A_2 are called equivalent if and only if $L(A_1) = L(A_2)$. They are called ambiguity-equivalent (denoted: $A_1 \equiv A_2$) if and only if $da_{A_1}(t) = da_{A_2}(t)$ for every tree t.

An FTA $A = (Q, \Sigma, Q_I, \delta)$ can be viewed as the definition of an R-FTA $A_R = (Q, \Sigma, 1_{Q_I}, \delta_R)$, where

$$1_{Q_I, q} = \begin{cases} 1 & \text{if } q \in Q_I \\ 0 & \text{else} \end{cases}$$

and

$$\delta_R(q, a, q_0 \cdots q_{m-1}) = \begin{cases} 1 & \text{if } (q, a, q_0 \cdots q_{m-1}) \in \delta \\ 0 & \text{else.} \end{cases}$$

We make the following observations.

PROPOSITION 3.1. *For all $t \in T_\Sigma$:*

(1) *If $\mathbb{N}_0$ is a subsemiring of R, then*

(1.1) $n_{A_R}(t)_q = n_A(t)_q$ *for all* $q \in Q$;

(1.2) $daA_R(t) = da_A(t)$.

(2) *If $p > 1$ is a natural number, then*

(2.1) $(n_{A_{\mathbb{Z}_p}}(t)_q = n_A(t)_q \bmod p)$ *for all* $q \in Q$;

(2.2) $da_{A_{\mathbb{Z}_p}}(t) = da_A(t) \bmod p$.

(3) *If $\mathbb{B}$ is the Boolean semiring defined by $1+1=1$, then*

(3.1) $(n_{A_{\mathbb{B}}}(t)_q = 1 \quad \text{iff} \quad n_A(t)_q > 0)$ *for all* $q \in Q$;

(3.2) $da_{A_{\mathbb{B}}}(t) = 1 \quad \text{iff} \quad da_A(t) > 0$.

Proof. In all three cases there are semiring homomorphisms $\rho_R : \mathbb{N}_0 \to R$ with $\rho_R(0) = 0$ and $\rho_R(1) = 1$. Thus, the subcases (_.2) follow directly from the subcases (_.1), and the subcases (_.1) follow by induction on the depth of t. □

For convenience we no longer distinguish between an FTA A and its corresponding $\mathbb{N}_0$-FTA $A_{\mathbb{N}_0}$.

Let R denote a semiring. The next three lemmas provide constructions for linear combination of R-FTAs, products of R-FTAs, and constant R-FTAs.

Assume $A_i = (Q_i, \Sigma, I^{(i)}, \delta_i)$, $i = 1, 2$, are two R-FTAs and $\mu_1, \mu_2 \in R$. By $\mu_1 A_1 + \mu_2 A_2$ we denote the R-FTA $(\bar{Q}, \Sigma, \bar{I}, \bar{\delta})$, where $\bar{Q}$ is the disjoint union of Q_1 and Q_2;

$$\bar{I}_q = \begin{cases} \mu_1 I_q^{(1)} & \text{if } q \in Q_1 \\ \mu_2 I_q^{(2)} & \text{if } q \in Q_2 \end{cases}$$

$$\bar{\delta}(q, a, q_0 \cdots q_{m-1}) = \begin{cases} \delta_1(q, a, q_0 \cdots q_{m-1}) & \text{if } q, q_0, \cdots, q_{m-1} \in Q_1 \\ \delta_2(q, a, q_0 \cdots q_{m-1}) & \text{if } q, q_0, \cdots, q_{m-1} \in Q_2 \\ 0 & \text{else} \end{cases}$$

By $A_1 \times A_2$ we denote the R-FTA $(\bar{\bar{Q}}, \Sigma, \bar{\bar{I}}, \bar{\bar{\delta}})$, where

$\bar{\bar{Q}} = Q_1 \times Q_2$;

$\bar{\bar{I}}_{(p,q)} = I_p^{(1)} \cdot I_q^{(2)}$ for $p \in Q_1$, $q \in Q_2$; and

$\bar{\bar{\delta}}((p, q), a, (p_0, q_0) \cdots (p_{m-1}, q_{m-1})) = \delta_1(p, a, p_0 \cdots p_{m-1}) \cdot \delta_2(q, a, q_0 \cdots q_{m-1})$.

Finally, for every $j \in R$ we define the constant R-FTA

$$\mathbf{j} = (\{q\}, \Sigma, j, \delta^1), \quad \text{where } \delta^1(q, a, q^m) = 1 \text{ for all } a \in \Sigma_m,\ m \geqq 0.$$

PROPOSITION 3.2. *For all $t \in T_\Sigma$:*

(1) $$n_{\mu_1 A_1 + \mu_2 A_2}(t)_q = \begin{cases} n_{A_1}(t)_q & \text{if } q \in Q_1 \\ n_{A_2}(t)_q & \text{if } q \in Q_2 \end{cases}$$

(2) $$da_{\mu_1 A_1 + \mu_2 A_2}(t) = \mu_1 da_{A_1}(t) + \mu_2 da_{A_2}(t).$$ □

PROPOSITION 3.3. *For all $t \in T_\Sigma$:*

(1) $n_{A_1 \times A_2}(t)_{(p,q)} = n_{A_1}(t)_p \cdot n_{A_2}(t)_q$ *for all* $p \in Q_1$ *and* $q \in Q_2$;

(2) $da_{A_1 \times A_2}(t) = da_{A_1}(t) \cdot da_{A_2}(t)$. □

PROPOSITION 3.4. *For all $t \in T_\Sigma$:*

(1) $n_{\mathbf{j}}(t) = n_{\mathbf{j}}(t)_q = 1$

(2) $da_j(t) = j$. □

The proofs of the Propositions 3.2–3.4 are omitted. Their assertions (1) can be verified by induction on the structure of terms, whereas assertions (2) immediately follow from the definitions and the assertions (1).

In [Kui88] Kuich describes a transformation transforming an m-ambiguous finite word automaton into an unambiguous $\mathbb{Q}$-automaton A' such that $L(A) = L(A')$. The

only properties that are needed are the presence of constructions for the linear combination and product of automata (as we have proved to exist for R-FTAs as well in 3.2 and 3.3) and the existence of automata accepting every word with a fixed ambiguity (similar to the R-FTA's **j**). Thus we get Proposition 3.5.

PROPOSITION 3.5. *Assume A is an m-ambiguous* FTA. *Then*

$$un(A) = \sum_{j=1}^{m} \frac{(-1)^{j+1}}{j!} [A]_j$$

is an unambiguous $\mathbb{Q}$-FTA *where* $[A]_j = A \times (A-1) \times \cdots \times (A-(\mathbf{j}-1))$. *Furthermore,* $L(un(A)) = L(A)$.

Proof. If $k = da_A(t) > 0$, then $da_{[A]_j}(t) = k \cdot (k-1) \cdot \cdots \cdot (k-j+1)$, and hence

$$da_{un(A)}(t) = \sum_{j=1}^{m} (-1)^{j+1} \binom{k}{j} = \sum_{j=1}^{k} (-1)^{j+1} \binom{k}{j} = 1.$$

If $da_A(t) = 0$, then $da_{[A]_j}(t) = 0$ for all $j > 0$, and hence also $da_{un(A)}(t) = 0$. □

Note that $un(A)$ can be constructed on a Random Access Machine (RAM) (without multiplications) in time $O(|A|^m)$.

We relativize the construction of un (A) modulo a suitable prime number p. The reason for this is to keep the numbers occurring in our algorithms small. Assume p is a prime number greater than m. By Bertrand's postulate (see [HaWri60, Thm. 418]) such a prime p exists in the range between m and $2m$. Since $p > m$, the multiplicative inverse $(j! \bmod p)^{-1}$ is defined in $\mathbb{Z}_p$ for all $j \in \{1, \cdots, m\}$. Therefore, we can define a $\mathbb{Z}_p$-FTA $un(A)_p$ by

$$un(A)_p = \sum_{j=1}^{m} (-1)^{j+1} (j! \bmod p)^{-1} [A_{\mathbb{Z}_p}]_j.$$

As a consequence of Proposition 3.1(2) and Proposition 3.5 we get the following theorem.

THEOREM 3.6. *For every m-ambiguous* FTA, A, *and every prime number $p > m$, $un(A)_p$ is an unambiguous $\mathbb{Z}_p$-*FTA *with $L(A) = L(un(A)_p)$.*

Since for unambiguous R-FTAs equivalence coincides with ambiguity-equivalence, Theorem 3.6 can be used to reduce the equivalence problem for m-ambiguous FTAs to the ambiguity-equivalence problem for unambiguous $\mathbb{Z}_p$-FTAs.

4. Deciding ambiguity-equivalence. In this section we present an algorithm deciding ambiguity-equivalence for R-FTAs, provided R is a field. This algorithm relies on a generalization of Eilenberg's equality theorem [Ei74, Thm. 8.1] to finite tree automata.

MAIN LEMMA 4.1. *If R is a field, and A is an R-*FTA *with n states, then the following holds.*

(1) $A \equiv \mathbf{0}$ iff $da_A(t) = 0$ for all $t \in T_\Sigma$ with $depth(t) < n$. linear bound on depth

(2) Whether or not $A \equiv \mathbf{0}$ can be decided in polynomial time on a RAM which is allowed to hold elements of R in its registers and to perform the R-operations $+$, $-$, $\cdot$, : and R-tests for 0 in constant time.

As an immediate consequence we get Theorem 4.2.

THEOREM 4.2. *If R is a field, and A_1 and A_2 are R-*FTA*s with n_1 and n_2 states, respectively, then the following holds.*

(1) $A_1 \equiv A_2$ iff $da_{A_1}(t) = da_{A_2}(t)$ *for all* $t \in T_\Sigma$ *with* $depth(t) < n_1 + n_2$.

(2) *Whether $A_1 \equiv A_2$ can be decided in polynomial time on a* RAM *which can hold elements of R in its registers and performs the R-operations $+$, $-$, $\cdot$, : and R-tests for* 0 *in constant time.*

Proof. $A_1 \equiv A_2$ iff $A_1 - A_2 \equiv 0$. Thus, Theorem 4.2 follows from the Main Lemma 4.1. □

Proof of 4.1. Assume $A = (Q, \Sigma, \delta, I)$ and $rk(A) = L$. Let V denote the n-dimensional R-vector space $V = R^Q$. For $k \geqq 0$ define $V_k = \langle n_A(t) \mid depth(t) \leqq k \rangle$, i.e., V_k is the subspace of V generated by the ambiguity vectors of trees t with $depth(t) \leqq k$. Clearly, $V_0 \subseteq V_1 \subseteq \cdots \subseteq V_k \subseteq V_{k+1} \cdots \subseteq V$.

Recall that every $a \in \Sigma$ defines a multilinear map $a : V^m \to V$; and we have

$$V_{k+1} = \left\langle \bigcup_{a \in \Sigma} a(V_k, \cdots, V_k) \right\rangle.$$

We conclude

(i) If $V_k = V_{k+1}$ for some $k \geqq 0$, then $V_k = V_{k+1}$ for all $1 > 0$; and therefore, since $\dim(V) = n$

(ii) $V_{n-1} = V_n = \bigcup_{k \geqq 0} V_k$.

(iii) If B_k is a basis of V_k, then $B'_{k+1} = B_k \cup \bigcup_{m>0} \{a(v_0, \cdots, v_{m-1}) \mid a \in \Sigma_m, v_j \in B_k\}$ is a generating system for V_{k+1}.

(iv) The following three statements are equivalent:

(a) $\sum_{q \in Q} I_q n_A(t)_q = 0$ for all $t \in T_\Sigma$;

(b) $\sum_{q \in Q} I_q v_q = 0$ for all $v = (v_q)_{q \in Q} \in V_{n-1}$;

(c) $\sum_{q \in Q} I_q v_q = 0$ for all $v = (v_q)_{q \in Q} \in B_{n-1}$ of some basis B_{n-1} of V_{n-1}.

By (iii) there is a basis B_{n-1} of V_{n-1} consisting only of vectors $n_A(t)$ for some $t \in T_\Sigma$ of depth less than n. This proves (1).

Note that this proof results from Eilenberg's proof by using multilinear maps instead of linear maps.

Ad(2): For one $k > 0$, computing B'_k from B_k needs $O(\#\Sigma \cdot n^L \cdot |A|)$ operations ($\#\Sigma$ denotes the number of elements of Σ). B'_k contains $O(n^L \cdot \#\Sigma)$ elements. By the Gaussian elimination method we can compute from B'_k a basis for V_k in $O(n^{L+2} \cdot \#\Sigma)$ steps. Without loss of generality we may assume $n \leqq |A|$. Thus, a basis for V_{n-1} can be computed in time $O(n^{L+2}|A| \cdot \#\Sigma)$. Since testing whether $\sum_{q \in Q} I_q v_q = 0$ for all $v = (v_q)_{q \in Q} \in B_{n-1}$ can be done in time $O(n^2)$, this is already the final complexity. By Proposition 1.3 we may assume $L \leqq 2$. Hence, assertion (2) follows. □

Note that the RAM performing our test for $A \equiv \mathbf{0}$ makes intensive use of unrestricted multiplication. Especially, if A is an FTA the entries of the basis vectors to be considered may grow very rapidly. The only upper bound for these entries given by the algorithm is $n^{L+L^2+\cdots+L^n}$. Thus, for $L > 1$ the occurring integers may have length $O(L^n \cdot \log n)$. However, we can restrict the lengths of occurring numbers by employing the fields $\mathbb{Z}_p$ (p a prime number) instead of $\mathbb{Q}$ as domains for the weights of the FTAs in question. Thus, for testing equivalence of m-ambiguous FTAs we can construct a deterministic polynomial algorithm; whereas for testing ambiguity-inequivalence of arbitrary FTAs we are able to give at least a randomized polynomial algorithm.

THEOREM 4.3. *The equivalence problem for m-ambiguous* FTA*s is* P*-complete with respect to* logspace *reductions for every fixed constant* $m > 0$.

Proof. Assume A_i, $i = 1, 2$, are two m-ambiguous FTAs. By Theorem 3.6 it suffices to decide whether $un(A_1)_p \equiv un(A_2)_p$ for a prime number $p > m$. The $\mathbb{Z}_p$-FTAs $un(A_i)_p$,

$i = 1, 2$, can be constructed in polynomial time, and the algorithm of Theorem 4.2(2) needs only the $\mathbb{Z}_p$-operations $+, -, \cdot, :$. Therefore, deciding equivalence of m-ambiguous FTAs is in P. Since the emptiness problem for context-free grammars can be reduced in logspace to the equivalence problem even of unambiguous FTAs, the result follows. □

For the following proposition, again let A_1, A_2 denote two FTAs with rank L and $\leqq n$ states. To avoid trivial subcases, we assume $n > 1$. For $k > 0$ let p_k denote the kth prime number.

PROPOSITION 4.4. *Define* $K = \log(n) \cdot (L+1)^{2n}$ (*where* $\log(\_)$ *denotes the logarithm with base* 2).

(1) $A_1 \equiv A_2$ iff $(A_{1,\mathbb{Z}_{p_k}} \equiv A_{2,\mathbb{Z}_{p_k}}$ *for all* $k \leqq K)$.

(2) *If* $K_0 \geqq 2K$ *and* $p \in \{p_1, \cdots, p_{K_0}\}$ *is randomly chosen* (*with respect to the equal distribution*) *then*

(2.1) *if* $A_1 \equiv A_2$ *then* $\text{prob}\{A_{1,\mathbb{Z}_p} \equiv A_{2,\mathbb{Z}_p}\} = 1$; *and*

(2.2) *if* $A_1 \not\equiv A_2$ *then* $\text{prob}\{A_{1,\mathbb{Z}_p} \not\equiv A_{2,\mathbb{Z}_p}\} \geqq \frac{1}{2}$.

Proof:

Ad(1): If $A_1 \equiv A_2$, then by Proposition 3.1(2) $A_{1,\mathbb{Z}_p} \equiv A_{2,\mathbb{Z}_p}$. Now assume A_1 and A_2 are not ambiguity-equivalent. By Theorem 4.2 there is some $t \in T_\Sigma$ with $depth(t) < 2n$ such that $da_{A_1}(t) \neq da_{A_2}(t)$. The bound on the depth of t implies that $da_{A_i}(t) \leqq n^{1+L+\cdots+L^{2n}} < n^{(L+1)^{2n}}$. Since $\prod_{k \leqq K} p_k \geqq 2^K = n^{(L+1)^{2n}}$, it follows from the Chinese Remainder Theorem that a prime number $p \in \{p_k \mid k \leqq K\}$ exists such that $da_{A_1}(t) \bmod p \neq da_{A_2}(t) \bmod p$, and therefore $A_{1,\mathbb{Z}_p} \not\equiv A_{2,\mathbb{Z}_p}$.

Ad(2): Assertion (2.1) is again the immediate consequence of Proposition 3.1(2). By Theorem 4.2 there is some $t \in T_\Sigma$ such that $z := da_{A_1}(t) - da_{A_2}(t) \neq 0$ and $|z| \leqq n^{(L+1)^{2n}}$. Since z contains at most K primes as a factor, assertion (2.2) follows. □

THEOREM 4.5. *The ambiguity-inequivalence problem for* FTA*s is in* RP, *i.e., the class of problems with randomized polynomial algorithms.*

Proof. Assume $A_i = (Q_i, \Sigma, Q_{i,I}, \delta_i)$, $i = 1, 2$, are two FTAs with $\# Q_i \leqq n$. By Proposition 1.3 we may assume without loss of generality that $L = \max\{rk(a) \mid a \in \Sigma\} \geqq 2$.

Define $K = \log(n) \cdot (L+1)^{2n}$ as in Proposition 4.4. Since $p_k = O(k \cdot \log(k))$ (see [Ap76] or [RoSchoe62]), one can choose constants $c, c' > 0$ such that $p_{\lceil 2K \rceil} < 2^{cn}$ and the cardinality of the set $\{p \mid p \text{ prime}, p < 2^{cn}\}$ is at least $c'(2^{cn}/n)$. We construct a probabilistic polynomial algorithm **A** that on input A_1, A_2 behaves as follows:

(i) If $A_1 \equiv A_2$, then prob{**A** outputs: "ambiguity-equivalent"} = 1;

(ii) If $A_1 \not\equiv A_2$, then prob{**A** outputs: "not ambiguity-equivalent"} $\geqq c'/2n$.

If we repeat this algorithm N times, where $N \geqq 2n/c'$, we get an algorithm **A'** that satisfies (i) but outputs: "not ambiguity-equivalent" with probability $\geqq \frac{1}{2}$ if $A_1 \not\equiv A_2$. The algorithm **A** behaves as follows:

(1) **A** guesses a nonnegative integer $p \in \{0, \cdots, 2^{cn} - 1\}$.

(2) **A** constructs the $\mathbb{Z}_p$-FTA $\bar{A} = A_{1,\mathbb{Z}_p} - A_{2,\mathbb{Z}_p}$.

(3) **A** tries to compute the linearly independent set of vectors $B_{2n-1} \subseteq \{n_{\bar{A}}(t) \mid t \in T_\Sigma\}$ according to the algorithm given in the proof of 4.1(2).

If the multiplicative inverse of some $q \in \mathbb{Z}_p$ has to be computed, then A tests whether $q^{p-1} = 1$.

If $q^{p-1} \neq 1$, then **A** outputs: "ambiguity-equivalent".

If $q^{p-1}=1$, then **A** computes q^{p-2} which in this case is the multiplicative inverse of q.

(4) **A** computes $z(v) := \sum_{q\in Q_{1,l}} v_q - \sum_{q\in Q_{2,l}} v_q$ for all $v \in B_{2n-1}$. If $z(v)=0$ for all $v \in B_{2n-1}$, then **A** outputs "ambiguity-equivalent". If not, then **A** outputs: "not ambiguity-equivalent."

Clearly, this algorithm runs in polynomial time. We show that **A** has the properties (i) and (ii).

Assume $A_1 \equiv A_2$. We distinguish two cases:

Case I. **A** succeeds to compute B_{2n-1}. $B_{2n-1} \subseteq \{n_A(t) \mid t \in T_\Sigma\}$. Therefore, by Propositions 3.1-3.2 we have for all $v \in B_{2n-1}$ $\exists t \in T_\Sigma$:

$$\begin{aligned} z(v) &= da_{\bar{A}}(t) \\ &= (da_{A_1}(t) - da_{A_2}(t)) \bmod p \\ &= 0. \end{aligned}$$

Hence, **A** outputs "ambiguity-equivalent."

Case II. **A** does not succeed to compute B_{2n-1}. Then **A** always outputs "ambiguity-equivalent." Therefore, **A** has property (i).

Now assume $A_1 \not\equiv A_2$. Assume $p \in \{0, \cdots, 2^{cn}-1\}$ is the integer randomly chosen in step (1). Then p is a prime number with probability $\geqq c'/n$. If p is a prime number, then $\mathbb{Z}_p$ is a field and hence the algorithm in the proof of 4.1(2) works correctly. Especially, for every $q \in \mathbb{Z}_p$, $q^{p-1}=1$ and q^{p-2} is the multiplicative inverse of q; thus **A** outputs "not ambiguity-equivalent," if and only if $A_{1,\mathbb{Z}_p} \not\equiv A_{2,\mathbb{Z}_p}$. By Proposition 4.4(2), $A_{1,\mathbb{Z}_p} \not\equiv A_{2,\mathbb{Z}_p}$ with probability $\geqq \frac{1}{2}$. Therefore, **A** outputs "not ambiguity-equivalent" with probability $\geqq c'/2n$. Hence, **A** has also property (ii). This finishes the proof. □

5. Finite degree of ambiguity. We have seen that bounding the degree of ambiguity by some fixed constant m yields a class of FTAs with a polynomial equivalence problem. Therefore, for the sake of completeness we show in the following theorem:

THEOREM 5.1. *For every* FTA A and $m>0$ *the following holds.*

(1) *There is an* FTA $A^{(m)} = (Q^{(m)}, \Sigma, Q_I^{(m)}, \delta^{(m)})$ *such that* $L(A^{(m)}) = \{t \in T_\Sigma \mid da_A(t) \geqq m\}$. *Especially,* $L(A^{(m)}) = \emptyset$ iff $da(A) < m$.

(2) *If m is a fixed constant, then it can be decided in polynomial time whether or not* $da(A) < m$.

Proof. Assume $A = (Q, \Sigma, Q_I, \delta)$, where $\#Q = n$ and $rk(A) = L$. Let $[0, m]$ denote the set $\{0, 1, \cdots, m\}$. Consider the semiring $([0, m], (+), *)$ with "absorbing upper bound," i.e., $(+)$ and $*$ are defined by

$$m_1 (+)\, m_2 = \begin{cases} m & \text{if } m_1 + m_2 \geqq m \\ m_1 + m_2 & \text{else} \end{cases}$$

and

$$m_1 * m_2 = \begin{cases} m & \text{if } m_1 \cdot m_2 \geqq m \\ m_1 \cdot m_2 & \text{else} \end{cases}$$

Define $Q^{(m)} = \{v \in [0, m]^Q \mid \#\{q \mid v_q \neq 0\} \leqq m\}$.

(As usual, for a Q-tuple v we adopt the convention that v_q denotes the qth element in v.)

Define $Q_I^{(m)} = \{v \in Q^{(m)} \mid (+)_{q\in Q_I} v_q = m\}$.

If $m \geqq n$, define $\delta^{(m)}$ as follows. For every $a \in \Sigma_k$ and $v_0, \cdots, v_{k-1} \in [0, m]^Q$, $(v, a, v_0, \cdots, v_{k-1})$ is in $\delta^{(m)}$ iff $v_q = (+)_{(q,a,q_0 \cdots q_{k-1}) \in \delta} v_{0,q_0} * \cdots * v_{k-1,q_{k-1}}$ for all $q \in Q$.

If $m < n$, define $\delta^{(m)}$ as follows. For every $a \in \Sigma_k$, $\delta' \subseteq \delta$, with $\#\delta' \leqq m$ and $v_1, \cdots, v_k \in Q^m$, $(v, a, v_0 \cdots v_{k-1})$ is in $\delta^{(m)}$ if and only if $v_q = (+)_{(q,a,q_0 \cdots q_{k-1}) \in \delta'} v_{0,q_0} * \cdots * v_{k-1,q_{k-1}}$ for all $q \in Q$. By the restriction to the cardinality of δ', $\#\{q \mid v_q \neq 0\} \leqq m$.

We omit the proof that the above constructed automaton has the property stated in assertion (1).

If $m < n$ is a fixed constant, then $A^{(m)}$ can be constructed in time $O(|A|^m \cdot m^{mL})$. Recall that an FTA accepts the empty set if and only if the corresponding reduced FTA has an empty state set. By Proposition 1.2 the reduced FTA for $A^{(m)}$ can be constructed in time $O(|A^{(m)}|) \leqq O(|A|^m \cdot m^{mL})$. Therefore, it can be decided in time $O(|A|^m \cdot m^{mL})$, whether $L(A^{(m)})$ is empty or not. Since we may assume $L \leqq 2$, it can be decided in polynomial time whether or not $da(A) < m$. □

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

REFERENCES

[Aho74] A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[Ap76] T. APOSTOL, *Introduction to Analytic Number Theory*, Springer-Verlag, New York, Berlin, 1976.

[BeReu82] J. BERSTEL AND C. REUTENAUER, *Recognizable formal power series on trees*, TCS, 18 (1982), pp. 115-148.

[ChaKoSto81] A. K. CHANDRA, D. C. KOZEN, AND L. J. STOCKMEYER, *Alternation*, J. Assoc. Comput. Mach., 28 (1981), pp. 114-133.

[Do70] J. DONER, *Tree acceptors and some of their applications*, J. Comput. System Sci., 4 (1970), pp. 406-451.

[Ei74] S. EILENBERG, *Automata, Languages, and Machines*, Vol. A, Academic Press, New York, 1974.

[GeStei84] F. GECSEG AND M. STEINBY, *Tree automata*, Akad. Kiado, Budapest, 1984.

[HaWri60] G. H. HARDY AND E. M. WRIGHT, *An Introduction to the Theory of Numbers*, 4th ed., Oxford University Press, London, 1960.

[Kui88] W. KUICH, *Finite automata and ambiguity*, Technische Universitaet Graz und österreichische Computer Gesellschaft, Report 253, June 1988.

[MeSto72] A. R. MEYER AND L. J. STOCKMEYER, *The equivalence problem for regular expressions with squaring requires exponential space*, 13th SWAT, 1972, pp. 125-129.

[Paul78] W. J. PAUL, *Komplexittstheorie*, B. G. Teubner, Stuttgart, 1978.

[Ra69] M. O. RABIN, *Decidability of second-order theories and automata on infinite trees*, Trans. Amer. Math. Soc., 141 (1969), pp. 1-35.

[RoSchoe62] B. ROSSER AND L. SCHOENFELD, *Approximate formulas for some functions of prime numbers*, Illinois J. of Math., 6 (1962), pp. 64-94.

[Se89] H. SEIDL, *On the finite ambiguity of finite tree automata*, Acta Informatica, 26 (1989), pp. 527-542.

[SteHu81] R. STEARNS AND H. HUNT III, *On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata*, 22th FOCS, 1981, pp. 74-81.

[SteHu85] ———, *On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata*, SIAM J. Comput., 14 (1985), pp. 598-611.

[StoMe73] L. J. STOCKMEYER AND A. R. MEYER, *Word problems requiring exponential time*, 5th ACM-STOC, pp. 1-9, 1973.

[ThaWri68] J. W. THATCHER AND J. B. WRIGHT, *Generalized finite automata theory with an application to a decision problem of second order logic*, Math. Systems Theory, 2 (1968), pp. 57-81.

[Tho84] W. THOMAS, *Logical aspects in the study of tree languages*, in 9th Coll. on Trees in Algebra and Programming, B. Courcelle, ed., Cambridge University Press, 1984, pp. 31-49.