DECIDING EQUIVALENCE OF FINITE TREE AUTOMATA

by

Helmut Seidl

Fachbereich Informatik
Universität des Saarlandes
D-6600 Saarbrücken
West Germany

Abstract

We show: 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 w.r.t. logspace reductions, and PSPACE-complete w.r.t. logspace reductions, if the automata in question are supposed to accept only finite languages. For finite tree automata with coefficients in a field R we give a polynomial time algorithm for deciding ambiguity-equivalence provided R-operations and R-tests for 0 can be performed in constant time. We apply this algorithm for deciding ambiguity-inequivalence of finite tree automata in randomized polynomial time.
Furthermore, for every constant m we show that it can be decided in polynomial time whether or not a given finite tree automaton is m-ambiguous.

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 analyse the equivalence problem for finite tree automata. Two finite tree automata A_1 and A_2 are called equivalent, iff they accept the same tree language. We find, that deciding inequivalence of finite tree automata is logspace complete in DEXPTIME, and that deciding inequivalence of finite tree automata which accept only finite languages still is logspace complete in PSPACE. Note that these problems for finite word automata are PSPACE- and NP-complete w.r.t. 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 in some restricted case. One parameter which especially attracts attention in this context is the degree of ambiguity of the corresponding automata. 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 show that for m-ambiguous finite word automata, m a fixed constant, equivalence can be decided in polynomial time. They employ difference equations for their proof. Kuich in [Kui87] simplifies their constructions. Although not stated explicitly, Kuich's proof can be used to show that deciding equivalence of m-ambiguous finite word automata even is in NC. Kuich uses semiring automata and the formalism of formal power series. Semiring automata are gained from ordinary automata by giving weights in some semiring to transitions and initial states. Especially, Kuich gives a method how deciding equivalence of m-ambiguous word automata can be reduced to deciding ambiguity-equivalence of unambiguous $\mathbb{Q}$-automata.
For tree languages the concept of formal power series seems to be much more involved than for word languages. So we avoid to use this concept, but show that some basic ideas of Kuich can be carried over to tree automata. Thus we introduce finite tree automata with coefficients in some semiring R and show how deciding equivalence for m-ambiguous finite tree automata can be reduced to deciding ambiguity-equivalence for unambiguous finite tree automata with coefficients in $\mathbb{Q}$ or even in $\mathbb{Z}_p$ for some prime number p>m.
Then we consider the problem of deciding ambiguity-equivalence for finite tree automata with coefficients in a field R. We are able to prove the appropriate generalization of Eilenberg's equality theorem [Ei74, theorem 8.1] to tree automata, i.e. we give an explicit upper bound for the depth of a witness for ambiguity-inequivalence. Furthermore, analysing the proof we get a polynomial time algorithm for deciding ambiguity-equivalence of finite tree automata with coefficients 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 for deciding ambiguity-equivalence for ordinary finite tree automata (viewed as automata with coefficients in $\mathbb{Q}$) since the only upper bound for the lengthes of occurring integers given by the algorithm is exponential in the input size. However, we get a polynomial time algorithm for deciding equivalence for m-ambiguous tree automata. As another consequence we are able to construct a randomized polynomial algorithm for deciding ambiguity-inequivalence of arbitrary finite tree automata.
Finally, we show: for any fixed constant m it can be decided in polynomial time whether or not a given finite tree automaton has a degree of ambiguity less than m. A subsequent paper will consider 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 deciding equivalence or ambiguity-equivalence for finite tree automata it suffices to consider finite tree automata of rank ≤2.

Let $\Sigma=\Sigma_0\cup..\cup\Sigma_L$ be a ranked alphabet. For $a\in\Sigma$: the rank of a, rk(a) , equals m iff $a\in\Sigma_m$. T_Σ denotes the free Σ-algebra of (finite ordered Σ-labelled) 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 that (i) can be viewed as a subcase of (ii) if we allow m to equal 0.
Let $t\in T_\Sigma$. Then the depth of t, depth(t) is defined by depth(t)=0 if $t\in\Sigma_0$, and depth(t)=1+max{depth(t_0),..,depth(t_{m-1})} if $t=a(t_0,\ldots,t_{m-1})$ for some $a\in\Sigma_m$.
Let $t=a(t_0,\ldots,t_{m-1})$ for some $a\in\Sigma_m$ with m≥0. The set of nodes of t , S(t) is the

subset of $\mathbb{N}_0^*$ defined by $S(t)=\{\varepsilon\}\cup\bigcup_{j=0}^{m-1} j\cdot S(t_j)$. 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,
$\Sigma=\Sigma_0\cup..\cup\Sigma_L$ 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.
$rk(A)=\max\{rk(a)\,|\,a\in\Sigma\}$ is called the rank of A.
Let $t=a(t_0,..,t_{m-1})\in T_\Sigma$ and $q\in Q$. A q-computation of A for t consists of a transition $(q,a,q_0..q_{m-1})\in\delta$ for the root and q_j-computations of A for the subtrees t_j, $j\in\{0,..,m-1\}$. 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 $\varphi:S(t)\to Q$ satisfying (i) $\varphi(\varepsilon)=q$ and (ii) if $\lambda_t(r)=a\in\Sigma_m$, then $(\varphi(r),a,\varphi(r\cdot 0)..\varphi(r\cdot(m-1)))\in\delta$. 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 $q\in Q$ let $\Phi_{A,q}(t)$ denote the set of all q-computations of A for t, and let $\Phi_{A,Q_I}(t)$ denote 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
Let $t=a(t_0,..,t_{m-1}) \in T_\Sigma$. Then

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

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

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

$$L(A) = \{t\in T_\Sigma\,|\,\Phi_{A,Q_I}(t)\neq\phi\}$$

The degree of ambiguity of A, da(A) is defined by

$$da(A) = \sup\{da_A(t)\,|\,t\in T_\Sigma\}$$

A is called
- unambiguous, if $da(A)\le 1$;
- ambiguous, if $da(A)>1$;
- m-ambiguous, if $da(A)\le m$;
- finitely ambiguous, if $da(A)<\infty$; and
- infinitely ambiguous, if $da(A)=\infty$.

Two FTA's A_1,A_2 are called
(1) ambiguity-equivalent (written $A_1\equiv A_2$), iff $\forall\ t\in T_\Sigma : da_{A_1}(t) = da_{A_2}(t)$;
(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 $A_1\equiv A_2$ iff $L(A_1)=L(A_2)$.

For describing our algorithms we will mostly use Random Access Machines (RAM's) with the uniform cost criterion, see [Aho74] or [Paul78] for precise definitions and basic properties. If we allow multiplications or divisions or the manipulation of registers not containing natural numbers, we will state this explicitly. For measureing the computational costs of our algorithms relative to the size of the input automata in question, we define

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

Let $A=(Q,\Sigma,Q_I,\delta)$ be an FTA. A is called reduced, if

$$\forall\ m\geq 0\ \forall\ a\in\Sigma_m : Q\times\{a\}\times Q^m\cap\delta\neq\phi$$

and

$$\forall\ q\in Q\ \exists\ t\in T_\Sigma, \varphi\in\Phi_{A,Q_I}(t) : q\in im(\varphi) \quad ^{1)}$$

The following fact is wellknown:

Proposition 1.2
For every FTA $A=(Q,\Sigma,Q_I,\delta)$ there is an FTA $A_r=(Q_r,\Sigma,Q_{r,I},\delta_r)$ 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 contextfree grammar.

We show, that without loss of generality we may restrict our attention to automata of rank ≤ 2 (nontheless we always will give the constructions for arbitrary rank L).
Consider the tree transformation tr which transforms a symbol a of rank >2 into a tree of binary symbols. Let Σ_{tr} be the ranked alphabet with

$$\Sigma_{tr,j} = \begin{cases} \Sigma_j & \text{if } j<2 \\ \{o\}\cup\bigcup_{m\geq 2}\Sigma_m & \text{if } j=2 \\ \phi & \text{else} \end{cases}$$

where o is a new symbol of rank 2.
Define $tr: T_\Sigma\to T_{\Sigma_{tr}}$ as follows. Let $t=a(t_0,..,t_{m-1})$ for some $a\in\Sigma_m$, $m\geq 0$. Then

$$tr(t) = \begin{cases} a & \text{if } m=0 \\ a(tr(t_0),..,tr(t_{m-1})) & \text{if } m\in\{1,2\} \\ a(tr(t_0),o(tr(t_1),..o(tr(t_{m-2}),tr(t_{m-1}))..)) & \text{if } m>2 \end{cases}$$

One has:

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

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

(2) $L(A') = tr(L(A))$ ▫

Furthermore,

---

1) $im(\varphi)$ denotes the image of the map φ.

(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 for deciding equivalence of m-ambiguous FTA's, m a fixed constant, we analyse the complexity of the inequivalence problems for unrestricted FTA's and FTA's accepting only finite languages, respectively. Note that for nondeterministic finite word automata these two problems are known to be PSPACE-complete and NP-complete (w.r.t. logspace reductions), respectively [MeySto72,StoMey73]. We find that for FTA's these problems become DEXPTIME-complete and PSPACE-complete, respectively.

Theorem 2.1

(1) Let $A_i=(Q_i,\Sigma,Q_{i,I},\delta_i)$, $i=1,2$, be two FTA's with rank L and $\#Q_i \leq n$. Deciding whether $L(A_1) \neq L(A_2)$ is in $DTIME(2^{c \cdot n \cdot L})$ for a suitable constant $c>0$.

(2) The inequivalence problem for FTA's is hard in DEXPTIME w.r.t. 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 even encode the word problem for linear space bounded alternating Turing machines into the inequivalence problem. ▫

Theorem 2.2
The inequivalence problem for FTA's accepting finite languages is PSPACE-complete w.r.t. logspace reductions.

Similar to the proof of theorem 2.1, the proof of theorem 2.2 is an appropriate generalization of a proof for the NP-completeness of deciding inequivalence of finite word automata accepting finite languages. Where in the case of finite word automata the bound on the length of the witness for inequivalence allows to construct an NP-algorithm, the bound on the depth of a witness for inequivalence allows us in the case of tree automata to construct a polynomially space bounded algorithm. Accordingly for the hardness part, instead of satisfiability of conjunctive normal forms which 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. ▫

As an immediate corollary we get:

Corollary 2.3
Deciding inequivalence for finitely ambiguous FTA's is PSPACE-hard.

Proof:
FTA's accepting finite languages form a subclass of the class of finitely ambiguous FTA's. ▫

3. Semiring Automata

We extend our notion of a finite tree automaton by allowing the transitions and initial states to have additional weights in some semiring R. The advantage of this extension is twofold.
On the one hand the resulting automata have nice algebraic properties which make it possible to "eliminate" finite ambiguity. These are studied in this section.
On the other hand it enables us to use methods from linear algebra to decide ambiguity-equivalence. This will be investigated in the next section.

Let R be an arbitrary (commutative) semiring with 0 and 1. A finite tree automaton with coefficients in R (short: R-FTA) is a quadriple $A = (Q,\Sigma,I,\delta)$ 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 multiplicities.
Let $V=R^Q$ be 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 $n_A(t) = (n_A(t))_q$ where $n_A(t)_q = \sum_{q_0..q_{m-1}\in Q^m} \delta(q,a,q_0..,q_{m-1})$ $n_A(t_0)_{q_0}\cdot\ldots\cdot n_A(t_{m-1})_{q_{m-1}}$ for $t=a(t_0,..,t_{m-1})$, and

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

$n_A(t)$ 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, $j=0,..,m-1$, the ambiguity vector of A for $a(t_0,..,t_{m-1})$.
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 FTA's, the size of the R-FTA A, |A|, is defined by

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

An R-FTA A is called unambiguous, iff $\forall\ t\in T_\Sigma : da_A(t)\in\{0,1\}$.
Let A_1, A_2 be two R-FTA's. A_1, A_2 are called equivalent, iff $L(A_1) = L(A_2)$. They are called ambiguity-equivalent (denoted: $A_1\equiv A_2$), iff $da_{A_1}(t) = da_{A_2}(t)$ for every tree t.

Let $A=(Q,\Sigma,Q_I,\delta)$ be an FTA. A 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

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

We can state the following observations:

Proposition 3.1
$\forall\ t\in T_\Sigma$:

(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) $da_{A_R}(t) = da_A(t)$

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

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

Proof:
Note that 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 don't destinguish any longer between an FTA A and its corresponding $\mathbb{N}_0$-FTA $A_{\mathbb{N}_0}$.

Let R be a semiring. Let $A_i=(Q_i,\Sigma,I^{(i)},\delta_i)$, $i=1,2$, be two R-FTA's and $\mu_1,\mu_2\in R$.
$\mu_1 A_1+\mu_2 A_2 = (\overline{Q},\Sigma,\overline{I},\overline{\delta})$ is the R-FTA where
Q is the disjoint union of Q_1 and Q_2;

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

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

Then the following holds:

Proposition 3.2
$\forall\ t\in T_\Sigma$:
(1)

$$n_{\mu_1A_1+\mu_2A_2}(t)_q = \begin{cases} n_{A_1}(t)_q & \text{if } q\in Q_1 \\ n_{A_2}(t)_q & \text{if } q\in Q_2 \end{cases}$$

(2) $da_{\mu_1A_1+\mu_2A_2}(t) = \mu_1 da_{A_1}(t)+\mu_2 da_{A_2}(t)$. ▫

Moreover, define the R-FTA $A_1\times A_2$ by $A_1\times A_2 = (\overline{\overline{Q}},\Sigma,\overline{\overline{I}},\overline{\overline{\delta}})$ where
$\overline{\overline{Q}} = Q_1\times Q_2$;
$\overline{\overline{I}}_{(p,q)} = I_p^{(1)}\cdot I_q^{(2)}$ for $p\in Q_1$, $q\in Q_2$; and
$\overline{\overline{\delta}}((p,q),a,(p_0,q_0)..(p_{m-1},q_{m-1})) = \delta_1(p,a,p_0..p_{m-1})\cdot\delta_2(q,a,q_0..q_{m-1})$
Then the following holds:

Proposition 3.3
$\forall\ t\in T_\Sigma$:

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

Finally, for every $j \in R$ we need the special R-FTA $\mathbf{j} = (\{q\}, \Sigma, j, \delta^1)$ with $\delta^1(q, a, q^m) = 1$ for all $a \in \Sigma_m$, $m \geq 0$.
Clearly, we have:

Proposition 3.4
$\forall\, t \in T_\Sigma$:
(1) $n_{\mathbf{j}}(t) = n_{\mathbf{j}}(t)_q = 1$
(2) $da_{\mathbf{j}}(t) = j$. ▫

In [Kui87] Kuich gives a construction for transforming an m-ambiguous finite word automaton into an unambiguous $\mathbb{Q}$-automaton A' such that L(A)=L(A'). The only properties he needs are the presence of constructions for the linear combination and product of automata (as we have proved to exist for FTA's 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
Let A be 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 $[A]_j = A \times (A-1) \times .. \times (A-(j-1))$.
Furthermore, L(un(A))=L(A).

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

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

Let $da_A(t)=0$. 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 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 occurring numbers small.
So, let p be a prime number greater than m. Note that by Bertrand's postulat (see [HaWri60, theorem 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 $j \in \{1,..,m\}$. 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 3.1 (2) and 3.5 we get:

Theorem 3.6
Let A be an m-ambiguous FTA and p>m a prime number. Then $un(A)_p$ is an unambiguous $\mathbb{Z}_p$-FTA such that $L(A) = L(un(A)_p)$. ▫

Since for unambiguous R-FTA's equivalence coincides with ambiguity-equivalence, theorem 3.6 can be used to reduce deciding equivalence for m-ambiguous FTA's to deciding ambiguity-equivalence for unambiguous $\mathbb{Z}_p$-FTA's.

4. Deciding Ambiguity–Equivalence

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

Main Lemma 4.1
Let R be a field. Let $A = (Q, \Sigma, I, \delta)$ an R-FTA with n states and rank L.

(1) $A \equiv 0$ iff $da_A(t) = 0$ for all $t \in T_\Sigma$ with depth(t)<n .

(2) Whether or not $A \equiv 0$ 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 +,-,·,: and R-tests for 0 in constant time.

As an immediate consequence we get:

Theorem 4.2
Let R be a field. Let A_1, A_2 be R-FTA's with n_1 and n_2 as the numbers of states.

(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 +,-,·,: 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:
Let V denote the n-dimensional R-vector space $V = R^Q$. Let $k \geq 0$. Define

$$V_k = \langle n_A(t) \mid \text{depth}(t) \leq k \rangle$$

i.e. V_k is the subspace of V generated by the ambiguity vectors of trees t with depth(t)≤k .
Clearly, $V_0 \subseteq V_1 \subseteq .. \subseteq V_k \subseteq V_{k+1} .. \subseteq V$.
Recall that every $a \in \Sigma$ defines a multilinear map $a: V^m \to V$; and we have:

$$V_{k+1} = \langle \bigcup_{a \in \Sigma} a(V_k, \ldots, V_k) \rangle$$

Thus, we can conclude:

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

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

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

Now 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):
Let L=rk(A). 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 $O(n^{L+2} \#\Sigma)$ steps. W.l.o.g. we may assume $n \leq |A|$. 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 that $L\le 2$. Hence, assertion (2) follows. ▫

It must be noted that the Random Access Machine performing our test for $A\equiv 0$ 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 $n^{L+L^2+..+L^n}$. Thus, for L>1 the occurring integers may have length $O(L^n\cdot \mathrm{ld}\, n)$. However, we can restrict the lengthes of occurring numbers by employing the fields $\mathbb{Z}_p$, p a prime number, instead of $\mathbb{Q}$ as domains for the coefficients of the FTA's in question. Thus, for testing equivalence of m-ambiguous FTA's we can construct a deterministic polynomial algorithm; whereas for testing ambiguity-inequivalence of arbitrary FTA's we are able tp give an at least randomized polynomial algorithm.

Theorem 4.3
Let m>0 be a fixed constant. Deciding equivalence for two m-ambiguous FTA's is P-complete w.r.t. logspace reductions.

Proof:
Let A_i , i=1,2 , be the two m-ambiguous FTA's. By theorem 3.6 it suffices to decide whether $\mathrm{un}(A_1)_p \equiv \mathrm{un}(A_2)_p$ for a prime number p>m. The $\mathbb{Z}_p$–FTA's $\mathrm{un}(A_i)_p$, i=1,2 , can be constructed in polynomial time, and the algorithm of theorem 4.2 (2) needs only the $\mathbb{Z}_p$-operations +,-,·. Therefore, deciding equivalence of m-ambiguous FTA's is in P. Since deciding emptiness of a contextfree language can be reduced in logspace to deciding equivalence even of unambiguous FTA's, the result follows. ▫

Let A_1, A_2 be FTA's with rank L and $\le$n states. To avoid trivial subcases we assume n>1. For k>0 let p_k denote the k-th prime number.

Proposition 4.4
Let $K = \mathrm{ld}\, n\, (L+1)^{2n}$.

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

(2) If $K_0\ge 2K$ and $p\in\{p_1,..,p_{K_0}\}$ is randomly chosen (w.r.t. the equal distribution) then:

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

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

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 $da_{A_1}(t)\ne da_{A_2}(t)$. The bound on the depth of t implies that $da_{A_1}(t) \le n^{1+L+..+L^{2n}} < n^{(L+1)^{2n}}$. Since $\prod_{k\le K} p_k \ge 2^K = n^{(L+1)^{2n}}$, it follows from the Chinese Remainder Theorem that there must be a prime number $p\in\{p_k \mid k\le K\}$ such that $da_{A_1}(t) \bmod p \ne da_{A_2}(t) \bmod p$ 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 $t\in T_\Sigma$ sueh that $z := da_{A_1}(t) - da_{A_2}(t) \ne 0$ and $|z| \le n^{(L+1)^{2n}}$. Since z contains at most K primes as a factor, assertion (2.2) follows. ▫

Theorem 4.5
Deciding ambiguity-inequivalence for two FTA's A_1, A_2 is in RP, i.e. the class of problems with randomized polynomial algorithms.

Proof:
Let $A_i = (Q_i, \Sigma, Q_{i,I}, \delta_i)$ and $\#Q_i \le n$, $i=1,2$. By proposition 1.3 we may assume w.l.o.g. that $L = \max\{rk(a) | a \in \Sigma\} \le 2$.
Let $K = \mathrm{ld}\ n\ (L+1)^{2n}$ as in proposition 4.4 . Since $p_k = O(k \cdot \mathrm{ld} k)$ (see [Ap76] or [RoSchoe62]), one can choose constants c, c' > 0 such that $p_{\lceil 2K \rceil} < 2^{cn}$ and for $\mathbb{P} = \{p \text{ prime} | p < 2^{cn}\}$, $\#\mathbb{P} \ge c' \frac{2^{cn}}{n}$. We construct a probabilistic polynomial algorithm A which 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"} $\ge \frac{c'}{2n}$.

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

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

(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 linearily independent set of vectors $B_{2n-1} \subseteq \{n_{\bar{A}}(t) | t \in T_\Sigma\}$ according to the algorithm given in the proof of 4.1 (2).
If the muliplicative inverse of some $q \in \mathbb{Z}_p$ has to be computed, then A tests whether $q^{p-1}=1$.
If $q^{p-1} \ne 1$, 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 $z(v) := \sum_{q \in Q_{1,I}} v_q - \sum_{q \in Q_{2,I}} v_q$ 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 destinguish two cases:
Case I:
A succeeds to compute B_{2n-1}. Since $B_{2n-1} \subseteq \{n_{\bar{A}}(t) | t \in T_\Sigma\}$, we have by the propositions 3.1 and 3.2 $\forall\, v \in B_{2n-1}\ \exists\, t \in T_\Sigma$:

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

Therefore, 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 $A_1 \not\equiv A_2$. Let $p \in \{0,..,2^{cn}-1\}$ be the integer randomly chosen in step (1). Then p is a prime number with probability $\ge \frac{c'}{n}$. 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", iff $A_{1,\mathbb{Z}_p} \neq A_{2,\mathbb{Z}_p}$. By proposition 4.4 (2), $A_{1,\mathbb{Z}_p} \neq A_{2,\mathbb{Z}_p}$ with probability $\geq \frac{1}{2}$. Therefore, A outputs "not ambiguity-equivalent" with probability $\geq \frac{c'}{2n}$. 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 yields a class of FTA's with a polynomial equivalence problem. Therefore, in this section we investigate finite ambiguity for its own sake. We show:

Theorem 5.1
Let $m \in \mathbb{N}$. Let $A = (Q, \Sigma, Q_I, \delta)$ be an FTA with n states and rank L.

(1) There is an FTA $A^{(m)} = (Q^{(m)}, \Sigma, Q_I^{(m)}, \delta^{(m)})$ such that $L(A^{(m)}) = \{t \in T_\Sigma \mid da_A(t) \geq m\}$. Especially, $L(A^{(m)}) = \phi$ 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:
Let $[0,m]$ denote the set $\{0,1,..,m\}$. Consider the semiring $([0,m], \oplus, *)$ with "absorbing upper bound" i.e. $\oplus$ and $*$ are defined by:

$$m_1 \oplus m_2 = \begin{cases} m & \text{if } m_1+m_2 \geq m \\ m_1+m_2 & \text{else} \end{cases}$$

and

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

Define $Q^{(m)} = \{v \in [0,m]^Q \mid \#\{q \mid v_q \neq 0\} \leq m\}$.
(As usual, for a Q-tuple v we adopt the convention that v_q denotes the q-th element in v).
Define $Q_I^{(m)} = \{v \in Q^{(m)} \mid \bigoplus_{q \in Q_I} v_q = m\}$

(here, $\bigoplus$ is related to $\oplus$ in the same way as $\sum$ to $+$).
If $m \geq n$, define $\delta^{(m)}$ as follows. For every $a \in \Sigma_k$ and $v_0, \ldots, v_{k-1} \in [0,m]^Q$, $(v, a, v_0..v_{k+1}) \in \delta^{(m)}$ where $v_q = \bigoplus_{(q,a,q_0..q_{k-1}) \in \delta} v_{0,q_0} * .. * v_{k-1,q_{k-1}}$. If $m<n$, define $\delta^{(m)}$ as follows. Let $a \in \Sigma_k$ and $\delta' \subseteq \delta \cap Q \times \{a\} \times Q^k$ with $\#\delta' \leq m$. For $\kappa \in \{0,..,k-1\}$ let $Q_\kappa = \{q' \in Q \mid \exists (q,a,q_0..q_{k-1}) \in \delta' : q' = q_\kappa\}$.
Let $v_0, \ldots, v_{k-1} \in [0,m]^Q$ be k vectors such that $v_{\kappa,q} \neq 0$ iff $q \in Q_\kappa$. For every such choice of a, δ' and $v_1, \ldots, v_{k-1}$, $(v, a, v_0..v_{k-1}) \in \delta^{(m)}$ where $v_q = \bigoplus_{(q,a,q_0..q_{k-1}) \in \delta'} v_{0,q_0} * .. * v_{k-1,q_{k-1}}$.
Note that by the restriction to the cardinality of δ', $\#\{q \mid v_q \neq 0\} \leq m$.
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 iff the corresponding reduced FTA has an empty state set. By proposition 1.2 the reduced FTA for $A^{(m)}$ can be constructed in time $O(|A^{(m)}|) \leq O(|A|^m \cdot m^{mL})$. 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 $L \leq 2$, it can be decided in polynomial time whether or not $da(A)<m$. ▫

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

References

[Aho74] A.V. Aho, J.E. Hopcroft, J.D. Ullman: The design and analysis of computer algorithms. Addison-Wesley 1974

[Ap76] T. Apostol: Introduction to analytic number theory. Springer Verlag New York, 1976

[ChaKoSto81] A.K. Chandra, D.C. Kozen, L.J. Stockmeyer: Alternation. JACM 28 (1981) pp. 114-133

[Do70] J. Doner: Tree acceptors and some of their applications. JCSS 4 (1970) pp. 406-451

[Ei74] S. Eilenberg: Automata, languages, and machines, Vol. A . Academic Press, New York, 1974

[GeStei84] F. Gecseg, M. Steinby: Tree automata. Akademiai Kiado, Budapest, 1984

[HaWri60] G.H. Hardy, E.M. Wright: An introduction to the theory of numbers. Oxford, 4th edition 1960

[Kui87] W. Kuich: Finite automata and ambiguity. Manuscript 1987

[MeSto72] A.R. Meyer, L.J. Stockmeyer: The equivalence problem for regular expressions with squaring requires exponential space. 13th SWAT (1972) pp. 125-129

[Paul78] W.J. Paul: Komplexitätstheorie. 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, L. Schoenfeld: Approximate formulas for some functions of prime numbers. Illinois J. of Mathematics 6 (1962) pp. 64-94

[SteHu81] R. Stearns, 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] R. Stearns, H. Hunt III: On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. SIAM J. Comp. 14 (1985) pp. 598-611

[StoMe73] L.J. Stockmeyer, A.R. Meyer: Word problems requiring exponential time. 5th ACM-STOC pp. 1-9, 1973

[ThaWri68] J.W. Thatcher, J.B. Wright: Generalized finite automata theory with an application to a decision problem of second order logic. Math. Syst. Th. 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 Univ. Press, 1984, pp. 31-49