On the Parallel Complexity of Tree Automata

Markus Lohrey

Universität Stuttgart, Institut für Informatik
Breitwiesenstr. 20–22, 70565 Stuttgart, Germany
lohreyms@informatik.uni-stuttgart.de

Abstract. We determine the parallel complexity of several (uniform) membership problems for recognizable tree languages. Furthermore we show that the word problem for a fixed finitely presented algebra is in DLOGTIME-uniform NC^1.

1 Introduction

Tree automata are a natural generalization of usual word automata to terms. Tree automata were introduced in [11, 12] and [27] in order to solve certain decision problems in logic. Since then they were successfully applied to many other decision problems in logic and term rewriting, see e.g. [7]. These applications motivate the investigation of decision problems for tree automata like emptiness, equivalence, and intersection nonemptiness. Several complexity results are known for these problems, see [28] for an overview. Another important decision problem is the membership problem, i.e, the problem whether a given tree automaton accepts a given term. It is easily seen that this problem can be solved in deterministic polynomial time [7], but up to now no precise bounds on the complexity are known.

In this paper we investigate the complexity of several variants of membership problems for tree automata. In Section 3 we consider the membership problem for a fixed tree automaton, i.e, for a fixed tree automaton $\mathcal{A}$ we ask whether a given input term is accepted by $\mathcal{A}$. We prove that this problem is contained in the parallel complexity class DLOGTIME-uniform NC^1, and furthermore that there exists a fixed tree automaton for which this problem is complete for DLOGTIME-uniform NC^1. Using these results, in Section 4 we prove that the word problem for a fixed finitely presented algebra is in DLOGTIME-uniform NC^1. This result nicely contrasts a result of Kozen that the uniform word problem for finitely presented algebras is P-complete [21]. Finally in Section 5 we investigate uniform membership problems for tree automata. In these problems the input consists of a tree automaton $\mathcal{A}$ from some fixed class $\mathcal{C}$ of tree automata and a term t, and we ask whether $\mathcal{A}$ accepts t. For the class $\mathcal{C}$ we consider the class of all deterministic top-down, deterministic bottom-up, and nondeterministic (bottom-up) tree automata, respectively. The complexity of the corresponding uniform membership problem varies between the classes log-space and LOGCFL, which is the class of all languages that can be reduced in log-space to a context free language. Again we prove several completeness results. Table 1 at the end of this paper summarizes the presented complexity results for membership problems.

2 Preliminaries

In the following let Σ be a finite alphabet. The empty word is denoted by ϵ. The set of all finite words over Σ is Σ^*. We set $\Sigma^+ = \Sigma^* \backslash \{\epsilon\}$. For $\Gamma \subseteq \Sigma$ we denote by $|s|_\Gamma$ the number of occurrences of symbols from Γ in s. We set $|s| = |s|_\Sigma$. For a binary relation $\to$ on some set we denote by $\stackrel{+}{\to}$ ($\stackrel{*}{\to}$) the transitive (reflexive and transitive) closure of $\to$. *Context-free grammars* are defined as usual. If $G = (N, \Sigma, S, P)$ is a context-free grammar then N is the set of *non-terminals*, Σ is the set of *terminals*, $S \in N$ is the *initial non-terminal*, and $P \subseteq N \times (N \cup \Sigma)^*$ is the finite set of *productions*. With $\to_G$ we denote the derivation relation of G. The language generated by G is denoted by $L(G)$. A context-free grammar is *ϵ-free* if it does not contain productions of the form $A \to \epsilon$.

We assume that the reader is familiar with the basic concepts of computational complexity, see for instance [23]. We just recall a few definitions concerning parallel complexity theory, see [30] for more details. It is not necessary to be familiar with this field in order to understand the constructions in this paper. L denotes deterministic logarithmic space. The definition of DLOGTIME-uniformity and DLOGTIME-reductions can be found in [2]. An important subclass of L is DLOGTIME-uniform NC^1, briefly uNC^1. More general, for $k \geq 1$ the class uNC^k contains all languages K such that there exists a DLOGTIME-uniform family $(\mathcal{C}_n)_{n \geq 0}$ of Boolean circuits with the following properties: (i) for some constant c the depth of the circuit $\mathcal{C}_n$ is bounded by $c \cdot \log(n)^k$, (ii) for some polynomial $p(n)$ the size of $\mathcal{C}_n$, i.e., the number of gates in $\mathcal{C}_n$, is bounded by $p(n)$, (iii) all gates in $\mathcal{C}_n$ have fan-in at most two, and (iv) the circuit $\mathcal{C}_n$ recognizes exactly the set of all words in K of length n. By [25] uNC^1 is equal to ALOGTIME.

An important subclass of uNC^1 is DLOGTIME-uniform-TC^0, briefly uTC^0. A language K is in uTC^0 if there exists a DLOGTIME-uniform family $(\mathcal{C}_n)_{n \geq 0}$ of circuits built up from Boolean gates and majority gates (or equivalently arbitrary threshold-gates) with the following properties: (i) for some constant c the depth of the circuit $\mathcal{C}_n$ is bounded by c, (ii) for some polynomial $p(n)$ the size of $\mathcal{C}_n$ is bounded by $p(n)$, (iii) all gates in $\mathcal{C}_n$ have unbounded fan-in, and (iv) the circuit $\mathcal{C}_n$ recognizes exactly the set of all words in K of length n. For more details see [2]. In this paper we will use a more convenient characterization of uTC^0 using first-order formulas with majority quantifiers, briefly FOM-formulas. Let Σ be a fixed finite alphabet of symbols. An FOM-formula is built up from the unary predicate symbols Q_a ($a \in \Sigma$) and the binary predicate symbols $<$ and BIT, using Boolean operators, first-order quantifiers, and the majority quantifier M. Such formulas are interpreted over words from Σ^+. Let $w = a_1 \cdots a_m$, where $m \geq 1$ and $a_i \in \Sigma$ for $i \in \{1, \ldots, m\}$. If we interpret an FOM-formula over w then all variables range over the interval $\{1, \ldots, m\}$, $<$ is interpreted by the usual order on this interval, $\mathrm{BIT}(n, i)$ is true if the i-th bit in the binary representation of n is one (we will not need this predicate any more), and $Q_a(x)$ is true if $a_x = a$. Boolean connectives and first-order quantifiers are interpreted as usual. Finally the formula $Mx\,\varphi(x)$ evaluates to true if $\varphi(x)$ is true for at least half of all $x \in \{1, \ldots, m\}$. The language defined by an FOM-sentence φ is the set of words from Σ^+ for which the FOM-sentence φ evaluates to true. For instance the FOM-sentence

$$Mx\, Q_a(x) \ \wedge\ Mx\, Q_b(x) \ \wedge\ \forall x, y\, \{x < y \ \rightarrow\ \neg(Q_b(x) \ \wedge\ Q_a(y))\}$$

defines the language $\{a^n b^n \mid n \geq 1\}$. It is well-known that uTC^0 is the set of languages that can be defined by an FOM-sentence [2].

In FOM-formulas we will often use constants and relations that can be easily defined in FOM, like for instance the equality of positions or the constants 1 and max, which denote the first and last position in a word, respectively. Furthermore by [2, Lemma 10.1] also the predicates $x + y = z$ and $x = \#y\ \varphi(y)$, i.e, the number of positions y that satisfy the FOM-formula $\varphi(y)$ is exactly x, can be expressed in FOM. Finally let us mention that uNC^1 also has a logical characterization similar to uTC^0. The only difference is that instead of majority quantifiers so called group quantifiers for a non-solvable group are used, see [2] for the details. Of course the resulting logic is at least as expressive as FOM.

In this paper we also use reductions between problems that can be defined within FOM. Formally let $f : \Sigma^+ \rightarrow \Gamma^+$ be a function such that for some constant k we have $|f(w)| \leq k \cdot |w|$ for all $w \in \Sigma^+$. [1] Then we say that f is *FOM-definable* if there exist formulas $\phi(x)$ and $\phi_a(x)$ for $a \in \Gamma$ such that when interpreted over a word w and $i \in \{1, \ldots, k \cdot |w|\}$ then $\phi(i)$ evaluates to true if and only if $i = |f(w)|$, and $\phi_a(i)$ evaluates to true if and only if the i-th symbol in $f(w)$ is a (here also all quantified variables in ϕ and ϕ_a range over the interval $\{1, \ldots, |w|\}$). We say that ϕ and ϕ_a $(a \in \Gamma)$ define f.

Lemma 1. *Let $f : \Sigma^+ \rightarrow \Gamma^+$ be FOM-definable and let $L \subseteq \Sigma^+$, $K \subseteq \Gamma^+$ such that $w \in L$ if and only if $f(w) \in K$ (in this case we say that L is FOM-reducible to K). If K is in uTC^0 (resp. uNC^1) then also L is in uTC^0 (resp. uNC^1).*

Proof. Let ϕ, ϕ_a $(a \in \Gamma)$ be FOM-formulas that define the function f and let K be in uTC^0, i.e., it can be defined by an FOM-sentence ψ. Let $|f(w)| \leq k \cdot |w|$ for all $w \in \Sigma^+$. In the following we restrict to the case $k = 2$, the generalization to an arbitrary k is obvious. In principle we can define the language L by the sentence that results from ψ by replacing every subformula $Q_a(x)$ by the formula $\phi_a(x)$. The only problem is that if we interpret this sentence over a word w then the variables quantified in ψ have to range over the interval $\{1, \ldots, |f(w)|\}$. Hence we define L by the FOM-sentence $\exists z\ \{(\phi(z) \ \wedge\ \psi^{z,0}) \ \vee\ (\phi(\max + z) \ \wedge\ \psi^{z,1})\}$, where the sentence $\psi^{z,i}$ is inductively defined as follows:

– $(\exists x\ \varphi(x))^{z,i} \ \equiv\ \exists x\ \{(x \leq i \cdot \max \wedge \varphi(x)^{z,i}) \ \vee\ (x \leq z \ \wedge\ \varphi(x + i \cdot \max)^{z,i})\}$

– $$(Mx\ \varphi(x))^{z,i} \ \equiv\ \exists x_1, x_2 \left\{ \begin{array}{l} x_1 = \#y\ (y \leq i \cdot \max \wedge \varphi(y)^{z,i}) \wedge \\ x_2 = \#y\ (y \leq z \ \wedge\ \varphi(y + i \cdot \max)^{z,i}) \wedge \\ \exists y\, \{2(x_1 + x_2) = y - 1 + z + i \cdot \max\} \end{array} \right\}$$

– $Q_a(x)^{z,i} \equiv \phi_a(x)$ and $Q_a(x + \max)^{z,i} \equiv \phi_a(x + \max)$

[1] This linear length-bound may be replaced by a polynomial bound, but this is not necessary for this paper.

If K belongs to uNC^1 the arguments are similar using the logical characterization of uNC^1. $\square$

LOGCFL (respectively LOGDCFL) is the class of all languages that are log-space reducible to a context-free language (respectively deterministic context-free language) [26]. In [26] it was shown that LOGCFL is the class of all languages that can be recognized in polynomial time on a log-space bounded auxiliary push-down automaton, whereas the deterministic variants of these machines precisely recognize all languages in LOGDCFL. The following inclusions are well-known and it is conjectured that they are all proper.

$$\mathrm{uTC}^0 \subseteq \mathrm{uNC}^1 = \mathrm{ALOGTIME} \subseteq \mathrm{L} \subseteq \mathrm{LOGDCFL} \subseteq \mathrm{LOGCFL} \subseteq \mathrm{uNC}^2 \subseteq \mathrm{P}$$

A *ranked alphabet* is a pair $(\mathcal{F}, \mathrm{arity})$ where $\mathcal{F}$ is a finite set of function symbols and arity is a function from $\mathcal{F}$ to $\mathbb{N}$ which assigns to each $\alpha \in \mathcal{F}$ its arity $\mathrm{arity}(\alpha)$. A function symbol a with $\mathrm{arity}(a) = 0$ is called a *constant*. In all examples we will use function symbols a and f, where $\mathrm{arity}(a) = 0$ and $\mathrm{arity}(f) = 2$. Mostly we omit the function arity in the description of a ranked alphabet. With $\mathcal{F}_i$ we denote the set of all function symbols in $\mathcal{F}$ of arity i. In FOM-formulas we use $Q_{\mathcal{F}_i}(x)$ as an abbreviation for $\bigvee_{\alpha \in \mathcal{F}_i} Q_\alpha(x)$. Let $\mathcal{X}$ be a countably infinite set of *variables*. Then $T(\mathcal{F}, \mathcal{X})$ denotes the set of *terms* over $\mathcal{F}$ and $\mathcal{X}$, it is defined as usual. The word *tree* is used as a synonym for term. We use the abbreviation $T(\mathcal{F}, \emptyset) = T(\mathcal{F})$, this set is called the set of *ground terms* over $\mathcal{F}$. We identify the set $T(\mathcal{F})$ with the corresponding *free term algebra* over the signature $\mathcal{F}$. In computational problems terms will be always represented by their prefix-operator notation, which is a word over the alphabet $\mathcal{F}$. The set of all these words is known as the Lukasiewicz-language $\L(\mathcal{F})$ for the ranked alphabet $\mathcal{F}$. For instance $ffafaafaa \in \L(\mathcal{F})$ but $fafaf \notin \L(\mathcal{F})$. When we write terms we will usually use the prefix-operator notation including brackets and commas in order to improve readability.

Lemma 2. *For every ranked alphabet $\mathcal{F}$ the language $L(\mathcal{F}) \subseteq \mathcal{F}^+$ is in uTC^0.*

A similar result for Dyck-languages was shown in [1].

Proof. Let $m = \max\{\mathrm{arity}(\alpha) \mid \alpha \in \mathcal{F}\}$. For $s \in \mathcal{F}^+$ define

$$\|s\| = \sum_{i=0}^{m} (i-1) \cdot |s|_{\mathcal{F}_i}.$$

Then for $s \in \mathcal{F}^+$ it holds $s \in \L(\mathcal{F})$ if and only if $\|s\| = -1$ and $\|t\| \geq 0$ for every prefix $t \neq s$ of s, see [17, p 323]. This characterization can be easily converted into an FOM-sentence:

$$\exists x_0, \ldots, x_m \{ \bigwedge_{i=0}^{m} x_i = \#z\, Q_{\mathcal{F}_i}(z) \ \wedge \ \sum_{i=0}^{m} (i-1) \cdot x_i = -1 \} \ \wedge$$
$$\forall y < \max \exists x_0, \ldots, x_m \{ \bigwedge_{i=0}^{m} x_i = \#z\, (Q_{\mathcal{F}_i}(z) \wedge z \leq y) \ \wedge \ \sum_{i=0}^{m} (i-1) \cdot x_i \geq 0 \}$$

$\square$

If the ranked alphabet $\mathcal{F}$ is clear from the context then in the following we will always write L instead of $\mathrm{L}(\mathcal{F})$. From the formula above it is straight forward to construct a formula $\mathrm{L}(i,j)$ which evaluates to true for a word $\alpha_1 \cdots \alpha_n \in \mathcal{F}^+$ and two positions $i,j \in \{1,\ldots,n\}$ if and only if $i \leq j$ and $\alpha_i \cdots \alpha_j \in \mathrm{L}$. The height $\mathrm{height}(t)$ of the term $t \in T(\mathcal{F})$ is inductively defined by $\mathrm{height}(\alpha(t_1,\ldots,t_n)) = 1 + \max\{\mathrm{height}(t_1),\ldots,\mathrm{height}(t_n)\}$, where $\mathrm{arity}(\alpha) = n \geq 0$ and $t_1,\ldots,t_n \in T(\mathcal{F})$ (here $\max(\emptyset) = 0$).

A *term rewriting system*, briefly TRS, over a ranked alphabet $\mathcal{F}$ is a finite set $\mathcal{R} \subseteq T(\mathcal{F},\mathcal{X}) \times T(\mathcal{F},\mathcal{X})$ such that for all $(s,t) \in \mathcal{R}$ every variable that occurs in t also occurs in s and furthermore $s \notin \mathcal{X}$. With a TRS $\mathcal{R}$ the *one-step rewriting relation* $\to_{\mathcal{R}}$ over $T(\mathcal{F},\mathcal{X})$ is associated as usual, see any text on term rewriting like for instance [10]. A *ground term rewriting system* $\mathcal{P}$ is a finite subset of $T(\mathcal{F}) \times T(\mathcal{F})$, i.e., the rules only contain ground terms. The symmetric, transitive, and reflexive closure of the one-step rewriting relation $\to_{\mathcal{P}}$ of a ground TRS is the smallest congruence relation on the free term algebra $T(\mathcal{F})$ that contains all pairs in $\mathcal{P}$, it is denoted by $\equiv_{\mathcal{P}}$. The corresponding quotient algebra $T(\mathcal{F})/\equiv_{\mathcal{P}}$ is denoted by $A(\mathcal{F},\mathcal{P})$, it is a *finitely presented algebra*.

For a detailed introduction into the field of tree automata see [14, 7]. A *top-down tree automaton*, briefly TDTA, is a tuple $\mathcal{A} = (Q,\mathcal{F},q_0,\mathcal{R})$, where Q is a finite set of states, $Q \cup \mathcal{F}$ is a ranked alphabet with $\mathrm{arity}(q) = 1$ for all $q \in Q$, $q_0 \in Q$ is the initial state, and $\mathcal{R}$ is a TRS such that all rules of $\mathcal{R}$ have the form $q(\alpha(x_1,\ldots,x_n)) \to \alpha(q_1(x_1),\ldots,q_n(x_n))$, where $q,q_1,\ldots,q_n \in Q$, $x_1,\ldots,x_n \in \mathcal{X}$, $\alpha \in \mathcal{F}$, and $\mathrm{arity}(\alpha) = n$. $\mathcal{A}$ is a *deterministic TDTA* if there are no two rules in $\mathcal{R}$ with the same left-hand side. The language that is accepted by a TDTA $\mathcal{A}$ is defined by

$$T(\mathcal{A}) = \{t \in T(\mathcal{F}) \mid q_0(t) \xrightarrow{*}_{\mathcal{R}} t\}.$$

A *bottom-up tree automaton*, briefly BUTA, is a tuple $\mathcal{A} = (Q,\mathcal{F},q_f,\mathcal{R})$, where Q is a finite set of states, $Q \cup \mathcal{F}$ is a ranked alphabet with $\mathrm{arity}(q) = 1$ for all $q \in Q$, $q_f \in Q$ is the final state, and $\mathcal{R}$ is a TRS such that all rules of $\mathcal{R}$ have the form $\alpha(q_1(x_1),\ldots,q_n(x_n)) \to q(\alpha(x_1,\ldots,x_n))$, where $q,q_1,\ldots,q_n \in Q$, $x_1,\ldots,x_n \in \mathcal{X}$, $\alpha \in \mathcal{F}$, and $\mathrm{arity}(\alpha) = n$. $\mathcal{A}$ is a *deterministic BUTA* if there are no two rules in $\mathcal{R}$ with the same left-hand side. The language that is accepted by a BUTA $\mathcal{A}$ is defined by

$$T(\mathcal{A}) = \{t \in T(\mathcal{F}) \mid t \xrightarrow{*}_{\mathcal{R}} q_f(t)\}.$$

It is well known that TDTAs, BUTAs, and deterministic BUTAs, respectively, all recognize the same subsets of $T(\mathcal{F})$. These subsets are called *recognizable tree languages* over $\mathcal{F}$. On the other hand deterministic TDTAs cannot recognize all recognizable tree languages.

As already remarked, if a term is part of the input for a Turing machine then the term will be encoded by its corresponding word from $\mathrm{L}(\mathcal{F})$, where the symbols from $\mathcal{F}$ are binary coded. A tree automaton will be encoded by basically listing its rules, we omit the formal details. The *membership problem* for a fixed TDTA $\mathcal{A}$, defined over a ranked alphabet $\mathcal{F}$, is the following decision problem:

INPUT: A term $t \in T(\mathcal{F})$.
QUESTION: Does $t \in T(\mathcal{A})$ hold?

If the TDTA $\mathcal{A}$ is also part of the input we speak of the *uniform membership problem* for TDTAs. It is the following decision problem:

INPUT: A TDTA $\mathcal{A} = (Q, \mathcal{F}, q_0, \mathcal{R})$ and a term $t \in T(\mathcal{F})$.
QUESTION: Does $t \in T(\mathcal{A})$ hold?

(Uniform) membership problems for other classes of automata or grammars are defined analogously. Note that the uniform membership problem for TDTAs can be reduced trivially to the uniform membership problem for BUTAs (and vice versa) by reversing the rules. Thus these two problems have the same computational complexity.

3 Membership problems

In this section we will study the membership problem for a fixed recognizable tree language. First we need some preliminary results.

A *parenthesis grammar* is a context-free grammar $G = (N, \Sigma, S, P)$ that contains two distinguished terminal symbols (and) such that all productions of G are of the form $A \to (s)$, where $A \in N$ and $s \in (N \cup \Sigma \backslash \{(,)\})^*$. A language that is generated by a parenthesis grammar is called a *parenthesis language*. Parenthesis languages where first studied in [22]. In [5] it was shown that every parenthesis language is in uNC^1.

Lemma 3. *Every recognizable tree language is FOM-reducible to a parenthesis language. Furthermore the uniform membership problem for TDTAs is log-space reducible to the uniform membership problem for parenthesis grammars.*

Proof. Let $\mathcal{A} = (Q, \mathcal{F}, q_0, \mathcal{R})$ be a TDTA. Let G be the parenthesis grammar $G = (Q, \mathcal{F} \cup \{(,)\}, q_0, P)$ where

$$P = \{q \to (f q_1 \cdots q_m) \mid q(f(x_1, \ldots, x_m)) \to f(q_1(x_1), \ldots, q_m(x_m)) \in \mathcal{R}\}.$$

Let us define a function $\beta : \text{Ł}(\mathcal{F}) \to (\mathcal{F} \cup \{(,)\})^+$ inductively by $\beta(f t_1 \cdots t_m) = (f \beta(t_1) \cdots \beta(t_m))$ for $f \in \mathcal{F}_m$ and $t_1, \ldots, t_m \in \text{Ł}(\mathcal{F})$. Then we have $t \in T(\mathcal{A})$ if and only if $\beta(t) \in L(G)$. Thus by Lemma 1 it suffices to show that the function β is FOM-definable. Let $t = \alpha_1 \cdots \alpha_n$, where $\alpha_j \in \mathcal{F}$. Then in order to construct $\beta(t)$ from t, an opening bracket has to be inserted in front of every symbol in t. Furthermore for $j \in \{1, \ldots, n\}$ the number of closing brackets following α_j in $\beta(t)$ is precisely the number of positions $i \leq j$ such that $\alpha_i \cdots \alpha_j \in \text{Ł}$. Hence β can be defined by the following formulas, where $\alpha \in \mathcal{F}$:

$$\begin{aligned}
\phi(x) &\equiv x = 3 \cdot \max \\
\phi_\alpha(x) &\equiv \exists y, z \, \{ \, Q_\alpha(y) \, \wedge \, z = \#i(\exists j(j < y \, \wedge \, \text{Ł}(i, j))) \, \wedge \, x = 2y + z \, \} \\
\phi_((x) &\equiv \bigvee_{\alpha \in \mathcal{F}} \phi_\alpha(x + 1) \\
\phi_)(x) &\equiv \neg \phi_((x) \, \wedge \, \bigwedge_{\alpha \in \mathcal{F}} \neg \phi_\alpha(x)
\end{aligned}$$

For the second statement note that in the uniform case all constructions can be easily done in log-space. □

Theorem 1. *Let T be a fixed recognizable tree language. Then the membership problem for T is in uNC^1. Furthermore there exists a fixed deterministic TDTA $\mathcal{A}$ such that the membership problem for $T(\mathcal{A})$ is uNC^1-complete under DLOGTIME-reductions.*

Proof. The first statement follows from Lemma 3 and the results of [5]. For the hardness part let $L \subseteq \Sigma^*$ be a fixed regular word language, whose membership problem is uNC^1-complete under DLOGTIME-reductions. By [2, Proposition 6.4] such a language exists. If we define $\mathrm{arity}(a) = 1$ for all $a \in \Sigma$ and let $\# \notin \Sigma$ be a constant then we can identify a word $a_1 a_2 \cdots a_n \in \Sigma^*$ with the ground term $a_1 a_2 \cdots a_n \# \in T(\Sigma \cup \{\#\})$, and the language L can be recognized by a fixed deterministic TDTA. □

4 Word problems for finitely presented algebras

In this section we present an application of Theorem 1 to the word problem for a finitely presented algebra. The *uniform word problem* for finitely presented algebras is the following problem:

INPUT: A ranked alphabet $\mathcal{F}$, a ground TRS $\mathcal{P}$ over $\mathcal{F}$, and $t_1, t_2 \in T(\mathcal{F})$.
QUESTION: Does $t_1 \equiv_{\mathcal{P}} t_2$ hold?

In [21] it was shown that the uniform word problem for finitely presented algebras is P-complete. Here we will study the *word problem* for a fixed finitely presented algebra $A(\mathcal{F}, \mathcal{P})$, where $\mathcal{F}$ is a fixed ranked alphabet, and $\mathcal{P}$ is a fixed ground TRS over $\mathcal{F}$:

INPUT: Two ground terms $t_1, t_2 \in T(\mathcal{F})$.
QUESTION: Does $t_1 \equiv_{\mathcal{P}} t_2$ hold?

For the rest of this section let us fix two ground terms $t_1, t_2 \in T(\mathcal{F})$. We want to decide whether $t_1 \equiv_{\mathcal{P}} t_2$. The following definition is taken from [9]. Let Ω be a new constant. Let $\Delta = (\mathcal{F} \cup \{\Omega\}) \times (\mathcal{F} \cup \{\Omega\}) \backslash \{(\Omega, \Omega)\}$ and define the arity of $[\alpha, \beta] \in \Delta$ by $\max\{\mathrm{arity}(\alpha), \mathrm{arity}(\beta)\}$. We define the function $\sigma : T(\mathcal{F}) \times T(\mathcal{F}) \to T(\Delta)$ inductively by

$$\begin{aligned}&\sigma(f(u_1, \ldots, u_m), g(v_1, \ldots, v_n)) = \\ &\qquad [f, g](\sigma(u_1, v_1), \ldots, \sigma(u_n, v_n), \sigma(u_{n+1}, \Omega), \ldots, \sigma(u_m, \Omega))\end{aligned}$$

if $m \geq n$ plus the symmetric rules for the case $m < n$. The term $\sigma(t_1, t_2)$ is a kind of parallel superposition of t_1 and t_2.

Example 1. Let $t_1 = faffafaaa$, $t_2 = fffaafaafaa$. Then $\sigma(t_1, t_2)$ is the term $[f,f][a,f][\Omega,f][\Omega,a][\Omega,a][\Omega,f][\Omega,a][\Omega,a][f,f][f,a][a,\Omega][f,\Omega][a,\Omega][a,\Omega][a,a]$.

In [9] the was shown that the set $T_{\mathcal{P}} = \{\sigma(t_1, t_2) \mid t_1, t_2 \in T(\mathcal{F}), t_1 \equiv_{\mathcal{P}} t_2\}$ is recognizable. Since $\mathcal{P}$ is a fixed ground TRS, $T_{\mathcal{P}}$ is also a fixed recognizable tree language. Thus by Theorem 1 we can decide in uNC^1 whether a term $t \in T(\Delta)$ belongs to $T_{\mathcal{P}}$. Therefore in order to put the word problem for $A(\mathcal{F}, \mathcal{P})$ into uNC^1 it suffices by Lemma 1 to prove the following lemma:

Lemma 4. *The function σ is FOM-definable.*

Proof. We assume that the input for σ is given as $t_1 t_2$. Let $x \in \{1, \ldots, |t_i|\}$. Then the x-th symbol of t_i corresponds to a node in the tree associated with t_i, and we denote the sequence of numbers that labels the path from the root to this node by $p_i(x)$, where each time we descend to the k-th child we write k. For instance if $t_1 = faffafaaa$ then $p_1(7) = 2121$. This sequence can be also constructed as follows. Let us fix some constant $a \in \mathcal{F}_0$. Let s be the prefix of t_i of length $x - 1$. Now we replace in s an arbitrary subword which belongs to $\text{Ł}\backslash\{a\}$ by a and repeat this as long as possible. Formally we define a function Π inductively by $\Pi(s) = s$ if $s \notin \mathcal{F}^*(\text{Ł}\backslash\{a\})\mathcal{F}^*$ and $\Pi(vtw) = \Pi(vaw)$ if $t \in \text{Ł}\backslash\{a\}$. We have for instance $\Pi(faff) = faff$ and $\Pi(fffaafaafa) = fafa$. Then it is easy to see that $p_i(x) = k_1 \cdots k_m$ if and only if $\Pi(s) = f_1 a^{k_1 - 1} \cdots f_m a^{k_m - 1}$ where $\mathrm{arity}(f_j) > 0$ for $j \in \{1, \ldots, m\}$.

First we construct an FOM-formula $c(x_1, x_2)$ that evaluates to true for two positions $x_1 \in \{1, \ldots, |t_1|\}$ and $x_2 \in \{1, \ldots, |t_2|\}$ if and only if $p_1(x_1) = p_2(x_2)$. For this we formalize the ideas above in FOM. In the following formulas we use the constants $o_1 = 0$ and o_2, where o_2 is uniquely defined by the formula $\text{Ł}(1, o_1)$. Thus, if interpreted over the word $t_1 t_2$, we have $o_2 = |t_1|$ and $\max - o_2 = |t_2|$. Furthermore let $I_1 = \{1, \ldots, o_2\}$ and $I_2 = \{1, \ldots, \max - o_2\}$. Quantification over these intervals can be easily done in FOM. If $t_i = \alpha_1 \cdots \alpha_n$ and $1 \leq x \leq n$ then the formula $\varphi_i(\ell, r, x)$ evaluates to true if $r < x$, $\alpha_\ell \cdots \alpha_r \in \text{Ł}$, and the interval between the positions ℓ and r is maximal with these two properties. The formula $\pi_i(u, x)$ evaluates to true if $u = |\Pi(\alpha_1 \cdots \alpha_{x-1})|$ and finally $f_i(u, x)$ evaluates to true if the u-th symbol of $\Pi(\alpha_1 \cdots \alpha_{x-1})$ has a nonzero arity. Formally for $i \in \{1, 2\}$ we define:

$$\varphi_i(\ell, r, x) \;\equiv\; \left\{ \begin{array}{l} r < x \;\wedge\; \text{Ł}(\ell + o_i, r + o_i) \;\wedge \\ \neg \exists y, z \in I_i \,\{y < \ell \;\wedge\; r \leq z < x \;\wedge\; \text{Ł}(y + o_i, z + o_i)\} \end{array} \right\}$$

$$\pi_i(u, x) \;\equiv\; x = u + 1 + \#z \,(\exists \ell, r \in I_i \,\{\varphi_i(\ell, r, x) \;\wedge\; \ell < z \leq r\})$$

$$f_i(u, x) \;\equiv\; \exists z \,\{z < x \;\wedge\; \neg \exists \ell, r \in I_i (\varphi_i(\ell, r, x) \;\wedge\; \ell \leq z \leq r) \;\wedge\; \pi_i(u - 1, z)\}$$

$$c(x_1, x_2) \;\equiv\; \exists u \left\{ \begin{array}{l} \pi_1(u, x_1) \;\wedge\; \pi_2(u, x_2) \;\wedge \\ \forall y \,(1 \leq y \leq u \;\rightarrow\; (f_1(y, x_1) \;\leftrightarrow\; f_2(y, x_2))) \end{array} \right\}$$

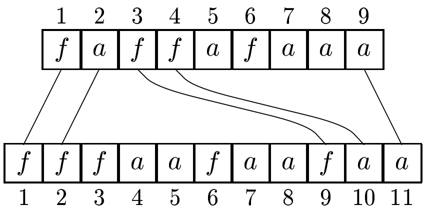
Finally we can define the functions σ by the following formulas, where $\alpha, \beta \in \mathcal{F}$ (the formulas $\phi_{[\Omega, \alpha]}(x)$ and $\phi_{[\alpha, \Omega]}(x)$ can be defined similarly to $\phi_{[\alpha, \beta]}(x)$):

$$\phi(x) \;\equiv\; \max = x + \#y \in I_1 \,(\exists y \in I_2 \,(c(y_1, y_2)))$$

$$\phi_{[\alpha, \beta]}(x) \;\equiv\; \exists y \in I_1, z \in I_2 \left\{ \begin{array}{l} c(y, z) \;\wedge\; Q_\alpha(y) \;\wedge\; Q_\beta(z + o_2) \;\wedge \\ y + z = x + \#y' \,(y' \leq y \;\wedge\; \exists z' \leq z \,(c(y', z'))) \end{array} \right\}$$

$\square$

Example 2. Let t_1, t_2 be from Example 1. In the following picture two positions satisfy the formula $c(y, z)$ if they are connected by a line. If $x = 15$ then the formula $\phi_{[a,a]}(x)$ is satisfied if we choose $y = 9$ and $z = 11$. Indeed, the 15-th symbol of $\sigma(t_1, t_2)$ is $[a, a]$.



Corollary 1. *For every finitely presented algebra the word problem is in uNC^1.*

Clearly there are also finitely presented algebras whose word problems are uNC^1-complete, like for instance the Boolean algebra $(\{0, 1\}, \wedge, \vee)$ [5]. An interesting open problem might be to find criteria for a finitely presented algebra $A(\mathcal{F}, \mathcal{P})$ which imply that the word problem is uNC^1-complete. For similar work in the context of finite groupoids see [4].

We should also say a few words concerning the input representation. In Theorem 1 and Corollary 1 we represent the input terms as strings over the alphabet $\mathcal{F}$. This is in fact crucial for the uNC^1-upper bounds. If we would represent input terms by their pointer representations then the problems considered would be in general L-complete. For instance if Boolean expressions are represented by their pointer representations then the expression evaluation problem becomes L-complete [3]. For other problems on trees for which it is crucial whether the string or the pointer representation is chosen see [6, 19]. For the uniform membership problems in the next section the encoding of the input terms is not crucial for the complexity since these problems are at least L-hard regardless of the chosen encoding.

5 Uniform membership problems

In this section we will investigate uniform membership problems for TDTAs. First we need some preliminary results.

Remark 1. The uniform membership problem for the class of all ϵ-free context-free grammars is in LOGCFL.

This fact seems to be folklore. In fact the usual algorithm for recognizing a context-free language on a push-down automaton can be implemented on a log-space bounded auxiliary push-down automaton also if the context-free grammar is part of the input. Furthermore if the grammar does not contain ϵ-productions then this automaton runs in polynomial time. Thus Remark 1 follows from [26]. The next lemma is stated in a similar form in [24, Lemma 3].

Lemma 5. *Let $G = (N, \Sigma, S, P)$ be a context-free grammar in Chomsky normal form. Assume that $A \xrightarrow{*}_G s$, where $A \in N$, $s \in (N \cup \Sigma)^*$, and $|s| > 2$. Then there exist a factorization $s = u_1 v u_2$ and $B \in N$ such that $A \xrightarrow{*}_G u_1 B u_2$, $B \xrightarrow{*}_G v$ and $|v|, |u_1 B u_2| \leq \frac{8}{9} \cdot |s|$.*

Proof. Consider a derivation tree T for the derivation $A \xrightarrow{*}_G s$ and let $n = |s|$. Since G is in Chomsky normal form T is a binary tree. For a node ν of T let yield(ν) be the factor of s that labels the sequence of leafs from left to right of the subtree of T rooted at ν. Consider a path p in T with the following two properties: (i) p starts at the root of T and ends at a leaf of T. (ii) If an edge (ν, ν_1) of T belongs to p and ν_1 and ν_2 are the children of ν then $|\text{yield}(\nu_1)| \geq |\text{yield}(\nu_2)|$. Let ν be the first node on p with $\text{yield}(\nu) \leq \frac{8}{9} \cdot n$ and let ν' be the parent node of ν. Thus $\text{yield}(\nu') > \frac{8}{9} \cdot n$. Let ν be labeled with $B \in N$ and let $\text{yield}(\nu) = v$. Thus there exists a factorization $s = u_1 v u_2$ such that $A \xrightarrow{*}_G u_1 B u_2$, $B \xrightarrow{*}_G v$, and $|v| \leq \frac{8}{9} \cdot |s|$. Furthermore since $n > 2$ and $|v| \geq |\text{yield}(\nu')|/2 > \frac{4}{9} \cdot n$ we also have $|u_1 B u_2| = n - |v| + 1 < n - \frac{4}{9} \cdot n + 1 \leq \frac{8}{9} \cdot n$. □

Theorem 2. *The uniform membership problem for the class of all TDTAs is LOGCFL-complete under log-space reductions.*

Proof. By the second statement from Lemma 3 and Remark 1 the uniform membership problem for TDTAs is in LOGCFL. It remains to show LOGCFL-hardness. For this we will make use of a technique from [24, Proof of Theorem 2]. Let $G = (N, \Sigma, P, S)$ be an arbitrary fixed context-free grammar [2] and let $w \in \Sigma^*$. We may assume that G is in Chomsky normal form and that $\epsilon \notin L(G)$. Let $|w| = n$ and $\mathcal{F} = \{a, f\}$, where $\text{arity}(a) = 0$ and $\text{arity}(f) = 2$. We will construct a TDTA $\mathcal{A} = (Q, \mathcal{F}, q_0, \mathcal{R})$ and a term $t \in T(\mathcal{F})$ such that $t \in T(\mathcal{A})$ if and only if $w \in L(G)$. Furthermore $\mathcal{A}$ and t can be computed in log-space from w. Let

$$W = \{w_1 A_1 w_2 \cdots w_i A_i w_{i+1} \mid 0 \leq i \leq 3,\ A_1, \ldots, A_i \in N,\\ w \in \Sigma^* w_1 \Sigma^* w_2 \ldots w_i \Sigma^* w_{i+1} \Sigma^*\}.$$

Thus W is the set of all $s \in (N \cup \Sigma)^*$ with $|s|_N \leq 3$ such that a subword of w can be obtained by substituting terminal words for the non-terminals in s. Note that $|W|$ is bounded polynomially in $|w| = n$, more precisely $|W| \in O(n^8)$. The set Q of states of $\mathcal{A}$ is $Q = \{\langle A, s\rangle \mid A \in N, s \in W\}$. The state $\langle A, s\rangle$ may be seen as the assertion that $A \xrightarrow{*}_G s$ holds. The initial state q_0 is $\langle S, w\rangle$. Finally the set $\mathcal{R}$ contains all rules of the following form, where $A \in N$, $s \in W$, and $v, u_1, u_2 \in (N \cup \Sigma)^*$ such that $u_1 v u_2 \in W$.

(1) $\langle A, s\rangle(a) \to a$ if $(A, s) \in P$
(2) $\langle A, s\rangle(f(x, y)) \to f(\langle A, s\rangle(x), \langle A, s\rangle(y))$ if $(A, s) \in P$
(3) $\langle A, u_1 v u_2\rangle(f(x, y)) \to f(\langle A, u_1 B u_2\rangle(x), \langle B, v\rangle(y))$ if $|u_1 v u_2|_N < 3$ or ($|v|_N = 2$ and $|u_1 v u_2|_N = 3$).

[2] In fact we may choose for G Greibach's hardest context-free grammar [16].

Note that in (3), if u_1vu_2 contains three non-terminals then we must choose a factorization u_1vu_2 such that v contains exactly two non-terminals. Then also u_1Bu_2 contains exactly two non-terminals. On the other hand if u_1vu_2 contains less then three non-terminals in (3) then we may choose any factorization. In this case both v and u_1Bu_2 contain at most three non-terminals. This concludes the description of $\mathcal{A}$. Note that $\mathcal{A}$ can be constructed in log-space from w. For the definition of the term t we need some further notations. In the following let $\gamma = 9/8 > 1$ and for $m > 0$ let $g_m = 2 \cdot \lceil \log_\gamma(m) \rceil + 2$. Furthermore for $m > 0$ let $\mathrm{bal}(m) \in T(\{a, f\})$ be a fully balanced term of height m, i.e., if $m = 1$ then $\mathrm{bal}(m) = a$, otherwise $\mathrm{bal}(m) = f(\mathrm{bal}(m-1), \mathrm{bal}(m-1))$. Now let $t = \mathrm{bal}(g_n)$. Since g_n is logarithmic in $n = |w|$, the size of t is polynomially bounded in n and t can be constructed from w in log-space. We claim that $w \in L(G)$ if and only if $t \in T(\mathcal{A})$. For the if-direction it suffices to prove the following more general claim for all $A \in N$, $s \in W$, and $t' \in T(\{a, f\})$:

$$\text{If } \langle A, s\rangle(t') \xrightarrow{*}_{\mathcal{R}} t' \text{ then } A \xrightarrow{*}_G s.$$

This statement can be shown easily by an induction on the structure of the term t'. For the (only if)-direction we will first show the following two statements, where $A \in N$, $s \in W$, $A \xrightarrow{*}_G s$, and $|s| = m > 0$ (note that $\epsilon \notin L(G)$).

(1) If $|s|_N < 3$ then $\langle A, s\rangle(t') \xrightarrow{*}_{\mathcal{R}} t'$ for some t' with $\mathrm{height}(t') \leq g_m - 1$.
(2) If $|s|_N = 3$ then $\langle A, s\rangle(t') \xrightarrow{*}_{\mathcal{R}} t'$ for some t' with $\mathrm{height}(t') \leq g_m$.

We prove these two statements simultaneously by an induction on the length of the derivation $A \xrightarrow{*}_G s$. If $m = 1$ then the derivation $A \xrightarrow{*}_G s$ has length one. Choose $t' = a$, then $\langle A, s\rangle(t') \to_{\mathcal{R}} t'$ and $\mathrm{height}(t') = 1 = g_1 - 1$. If $m = 2$ then since G is in Chomsky normal form the derivation $A \xrightarrow{*}_G s$ has length at most 3. Let $t' = f(f(a, a), a)$. Then $\langle A, s\rangle(t') \xrightarrow{*}_{\mathcal{R}} t'$ and $\mathrm{height}(t') = 3 \leq g_2 - 1$. Now assume $m \geq 3$. First let $|s|_N = 3$. Then there exist a factorization $s = u_1vu_2$ and $B \in N$ with $A \xrightarrow{*}_G u_1Bu_2$, $B \xrightarrow{*}_G v$, and $|v|_N = |u_1Bu_2|_N = 2$. Let $m_1 = |u_1Bu_2|$ and $m_2 = |v|$, thus $m = m_1 + m_2 - 1$. W.l.o.g. assume $m_1 \geq m_2$. Now the induction hypothesis implies $\langle A, u_1Bu_2\rangle(t_1) \xrightarrow{*}_{\mathcal{R}} t_1$ and $\langle B, v\rangle(t_2) \xrightarrow{*}_{\mathcal{R}} t_2$ for some terms t_1 and t_2 with $\mathrm{height}(t_1) \leq g_{m_1} - 1$ and $\mathrm{height}(t_2) \leq g_{m_2} - 1$. It follows $\langle A, s\rangle(f(t_1, t_2)) \xrightarrow{*}_{\mathcal{R}} f(t_1, t_2)$ where $\mathrm{height}(f(t_1, t_2)) = 1 + \mathrm{height}(t_1) \leq g_{m_1} \leq g_m$ since $m_1 = m + 1 - m_2 \leq m$. Finally assume that $|s|_N < 3$. By Lemma 5 there exist a factorization $s = u_1vu_2$ and $B \in N$ such that $A \xrightarrow{*}_G u_1Bu_2$, $B \xrightarrow{*}_G v$, and $v, u_1Bu_2 \in W$, and $|u_1Bu_2|, |v| \leq m/\gamma$. Let $m_1 = |u_1Bu_2|$ and $m_2 = |v|$. The induction hypothesis implies $\langle A, u_1Bu_2\rangle(t_1) \xrightarrow{*}_{\mathcal{R}} t_1$ and $\langle B, v\rangle(t_2) \xrightarrow{*}_{\mathcal{R}} t_2$ for some terms t_1 and t_2 such that $\mathrm{height}(t_1) \leq g_{m_1} \leq 2 \cdot \lceil \log_\gamma(\frac{m}{\gamma}) \rceil + 2 = g_m - 2$ and similarly $\mathrm{height}(t_2) \leq g_m - 2$. It follows $\langle A, s\rangle(f(t_1, t_2)) \xrightarrow{*}_{\mathcal{R}} f(t_1, t_2)$ where $\mathrm{height}(f(t_1, t_2)) \leq g_m - 1$. This concludes the proof of the statements (1) and (2). Now assume that $w \in L(G)$. Then $\langle S, w\rangle(t') \xrightarrow{*}_{\mathcal{R}} t'$ for some t' with $\mathrm{height}(t') \leq g_n$. But then the first two groups of rules of $\mathcal{A}$ imply $\langle S, w\rangle(\mathrm{bal}(g_n)) \xrightarrow{*}_{\mathcal{R}} \mathrm{bal}(g_n)$, i.e, $t = \mathrm{bal}(g_n) \in T(\mathcal{A})$. This concludes the proof of the theorem. $\square$

Remark 1, Theorem 2 and the second statement from Lemma 3 immediately imply the following corollary.

Corollary 2. *The uniform membership problem for the class of all parenthesis grammars is LOGCFL-complete under log-space reductions.*

In [15] it was shown that the problem of evaluating acyclic Boolean conjunctive queries is LOGCFL–complete. In order to show LOGCFL–hardness, in [15] Venkateswaran's characterization [29] of LOGCFL in terms of semi–unbounded circuits is used. In fact the method from [15] may be modified in order to prove Theorem 2. On the other hand our proof does not use Venkateswaran's result and seems to be more elementary.

It should be also noted that since directed reachability in graphs is NL-complete [20], the uniform membership problem for usual nondeterministic word automata is NL-complete. Thus, since NL is supposed to be a proper subset of LOGCFL, for the nondeterministic case the complexity seems to increase when going from words to trees. The next theorem shows that this is not the case for the deterministic case if we restrict to TDTAs.

Theorem 3. *The uniform membership problem for the class of all deterministic TDTAs is L-complete under DLOGTIME-reductions.*

Proof. Hardness follows from the fact that the uniform membership problem for deterministic word automata is L-complete under DLOGTIME-reductions, see e.g. [8] and the remark in [18, Theorem 15]. For the upper bound we will use an idea that appeared in a similar form in [13, Section 4] in the context of tree walking automata with pebbles. Let $\mathcal{A} = (Q, \mathcal{F}, q_0, \mathcal{R})$ be a deterministic TDTA and let $t \in T(\mathcal{F})$. We will outline a high-level description of a deterministic log-space Turing machine that decides whether $t \in T(\mathcal{A})$. We use a result of [14], which roughly speaking says that in order to check whether a tree is accepted by a deterministic TDTA it suffices to check each path from the root to a leaf separately.

We assume that the input word $t \in \mathrm{L}$ is stored on the input tape starting at position 1. In the following we will identify a position $i \in \{1, \ldots, |t|\}$ on the input tape with the corresponding node of the tree t. For the term $t = ffaaffaaa$ for instance, position 1 is the root of t and $3, 4, 7, 8,$ and 9 are the leafs of t. In the high-level description we will use the following variables:

- $h_i \in \{1, \ldots, |t|\}$ $(i \in \{1, 2\})$ is a position on the input tape. With h_1 we visit all nodes of t. Each time h_1 visits a leaf of t, with h_2 we walk down the path from the root 1 to h_1 and check whether it is accepted by $\mathcal{A}$.
- $f_i \in \mathcal{F}$ $(i \in \{1, 2\})$ is the label of node h_i.
- $q \in Q$ is the state to which node h_2 evaluates under the automaton $\mathcal{A}$.

All these variables only need logarithmic space. We use the following routines:

- brother(h) returns the position of the right brother of h, or undefined if h does not have a right brother. This value can be calculated in log-space by counting, using the characterization of L from [17], see the proof of Lemma 2.

– $\delta(f, q, i)$, where $f \in \mathcal{F}$, $q \in Q$, and $i \in \{1, \ldots, \text{arity}(f)\}$, returns the state q' such that if $q(f(x_1, \ldots, x_n)) \to f(q_1(x_1), \ldots, q_n(x_n)) \in \mathcal{R}$ then $q' = q_i$.

For instance for the term t above we have brother$(2) = 5$. Finally we present the algorithm. It is clear that this algorithm runs in log-space.

```
for h1 := 1 to |t| do
    if arity(f1) = 0 then
        q := q0; h2 := 1;
        while h2 < h1 do
            f := f2; i := 1; h2 := h2 + 1;
            while brother(h2) is defined and brother(h2) ≤ h1 do
                i := i + 1; h2 := brother(h2)
            endwhile
            q := δ(f, q, i)
        endwhile
        if (q(f2) → f2) ∉ R then reject
endfor
accept
```

□

Finally we consider deterministic BUTAs. Note that the uniform membership problem for nondeterministic BUTAs was implicitly considered in Theorem 2, since the uniform membership problems for nondeterministic BUTAs and non-deterministic TDTAs, respectively, can be directly translated into each other.

Theorem 4. *The uniform membership problem for the class of all deterministic BUTAs is in LOGDCFL.*

Proof. Let $\mathcal{A} = (Q, \mathcal{F}, q_f, \mathcal{R})$ be a deterministic BUTA and let $t \in T(\mathcal{F})$. Let $\# \notin \{0, 1\}$ be an additional symbol. By [26] it suffices to outline a deterministic log-space bounded auxiliary push-down automaton $\mathcal{M}$ that checks in polynomial time whether $t \in T(\mathcal{A})$. The input word t is scanned from from right to left. A sequence of the form $\#\text{bin}(q_1)\#\text{bin}(q_2)\cdots\#\text{bin}(q_m)$ is stored on the push-down, where $\text{bin}(q_i)$ is the binary coding of the state $q_i \in Q$ and the top-most push-down symbol corresponds to the right-most symbol in this word. The length of this coding is bounded logarithmically in the input length. If $\mathcal{M}$ reads the symbol f from the input, where $\text{arity}(f) = n$, then $\mathcal{M}$ replaces the sequence $\#\text{bin}(q_1)\#\text{bin}(q_2)\cdots\#\text{bin}(q_n)$ by the sequence $\#\text{bin}(q)$ on top of the push-down, where $f(q_1(x_1), \ldots, q_n(x_n)) \to q(f(x_1, \ldots, x_n))$ is a rule in $\mathcal{R}$. The auxiliary tape is used for storing binary coded states. □

The precise complexity of the uniform membership problem for deterministic BUTAs remains open. For the lower bound we can only prove L-hardness. This problem has also an interesting reformulation in terms of finite algebras. A deterministic BUTA $\mathcal{A}$ corresponds in a straight-forward way to a finite algebra A. The carrier set of A is the set Q of states of $\mathcal{A}$ and every function symbol f

of arity n is interpreted as an n-ary function on Q. Now the question whether a term t is accepted by $\mathcal{A}$ is equivalent to the question whether the expression t evaluates in the algebra A to a distinguished element q (namely the final state of $\mathcal{A}$). Thus the uniform membership problem for deterministic BUTAs is equivalent to the uniform expression evaluation problem for finite algebras. In the case of a fixed groupoid, the complexity of the expression evaluation problem was considered in [4].

Table 1 summarizes the complexity results for tree automata shown in this paper.

Table 1. Complexity results for tree automata

	det. TDTA	det. BUTA	TDTA (BUTA)
membership	uNC^1-complete	uNC^1-complete	uNC^1-complete
uniform membership	L-complete	LOGDCFL	LOGCFL-complete

Acknowledgments I would like to thank the referees for valuable comments.

References

1. D. A. M. Barrington and J. Corbet. On the relative complexity of some languages in NC^1. *Information Processing Letters*, 32:251–256, 1989.
2. D. A. M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.
3. M. Beaudry and P. McKenzie. Circuits, matrices, and nonassociative computation. *Journal of Computer and System Sciences*, 50(3):441–455, 1995.
4. J. Berman, A. Drisko, F. Lemieux, C. Moore, and D. Thérien. Circuits and expressions with non–associative gates. In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity, Ulm (Germany)*, pages 193–203. IEEE Computer Society Press, 1997.
5. S. R. Buss. The Boolean formula value problem is in ALOGTIME. In *Proceedings of the 19th Annual Symposium on Theory of Computing (STOC 87)*, pages 123–131. ACM Press, 1987.
6. S. R. Buss. Alogtime algorithms for tree isomorphism, comparison, and canonization. In *Kurt Gödel Colloquium 97*, pages 18–33, 1997.
7. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 1997.
8. S. A. Cook and P. McKenzie. Problems complete for deterministic logarithmic space. *Journal of Algorithms*, 8:385–394, 1987.
9. M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *Proceedings of the 5th Annual IEEE Symposium on Logic in Computer Science (LICS '90)*, pages 242–256. IEEE Computer Society Press, 1990.

10. N. Dershowitz and J.-P. Jouannaud. Rewriting systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 243–320. Elsevier Publishers, Amsterdam, 1990.
11. J. E. Doner. Decidability of the weak second-order theory of two successors. *Notices Amer. Math. Soc.*, 12:365–468, 1965.
12. J. E. Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4:406–451, 1970.
13. J. Engelfriet and H. J. Hoogeboom. Tree-walking pebble automata. In J. Karhumäki, H. Maurer, G. Paun, and G. Rozenberg, editors, *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 72–83. Springer, 1999.
14. F. Gécseg and M. Steinby. *Tree automata.* Akadémiai Kiadó, 1984.
15. G. Gottlob, N. Leone, and F. Scarcello. The complexity of acyclic conjunctive queries. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS 98, Palo Alto, California, USA)*, pages 706–715. IEEE Computer Society Press, 1998.
16. S. Greibach. The hardest context-free language. *SIAM Journal on Computing*, 2(4):304–310, 1973.
17. M. A. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, 1978.
18. M. Holzer and K.-J. Lange. On the complexities of linear LL(1) and LR(1) grammars. In Z. Ésik, editor, *Proceedings of the 9th International Symposium on Fundamentals of Computation Theory (FCT'93, Szeged, Hungary)*, number 710 in Lecture Notes in Computer Science, pages 299–308. Springer, 1993.
19. B. Jenner, P. McKenzie, and J. Torán. A note on the hardness of tree isomorphism. In *Proceedings of the 13th Annual IEEE Conference on Computational Complexity*, pages 101–105. IEEE Computer Society Press, 1998.
20. N. D. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences*, 11(1):68–85, 1975.
21. D. C. Kozen. Complexity of finitely presented algebras. In *9th Annual Symposium on Theory of Computing (STOC 77)*, pages 164–177. ACM Press, 1977.
22. R. McNaughton. Parenthesis grammars. *Journal of the Association for Computing Machinery*, 14(3):490–500, 1967.
23. C. H. Papadimitriou. *Computational Complexity.* Addison Wesley, 1994.
24. W. L. Ruzzo. Tree–size bounded alternation. *Journal of Computer and System Sciences*, 21:218–235, 1980.
25. W. L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22:365–383, 1981.
26. I. H. Sudborough. On the tape complexity of deterministic context–free languages. *Journal of the Association for Computing Machinery*, 25(3):405–414, 1978.
27. J. W. Thatcher and J. B. Wright. Generalized finite automata with an application to a decision problem of second order logic. *Mathematical Systems Theory*, 2:57–82, 1968.
28. M. Veanes. On computational complexity of basic decision problems of finite tree automata. Technical Report 133, Uppsala Computing Science Department, 1997.
29. H. Venkateswaran. Properties that characterize LOGCFL. *Journal of Computer and System Sciences*, 43:380–404, 1991.
30. H. Vollmer. *Introduction to Circuit Complexity.* Springer, 1999.