Theoretical Informatics and Applications Will be set by the publisher
Informatique Théorique et Applications

COMPLEXITY RESULTS FOR PREFIX GRAMMARS

Markus Lohrey[1] and Holger Petersen[1]

Abstract. Resolving an open problem of Ravikumar and Quan, we show that equivalence of prefix grammars is complete in PSPACE. We also show that membership for these grammars is complete in P (it was known that this problem is in P) and characterize the complexity of equivalence and inclusion for monotonic grammars. For grammars with several premises we show that membership is complete in EXPTIME and hard for PSPACE for monotonic grammars.

1991 Mathematics Subject Classification. 03D03, 68Q17, 68Q42, 68Q45.

1. Introduction

The computational complexity of decision problems concerning languages depends on the formalisms that are chosen to represent them. As an example take the equivalence problem for regular sets. If languages are given by deterministic finite automata, then equivalence can be tested in almost linear time [10]. The problem becomes complete in PSPACE if languages are described by regular expressions [14]. If squaring is allowed as an operation in regular expressions, the problem is complete in exponential space [14]. Adding complementation leads to an equivalence problem that is not even elementary [18].

The number of characterizations of regular sets is especially impressive and includes powerful devices like several variants of two-way automata. Almost four decades ago, J. Richard Büchi added another formalism to the list by proving that systems of rewriting rules which are applied to a prefix of a string generate exactly the regular languages when starting from a finite set of axioms [1]. These systems were therefore called *regular canonical systems*. Greibach gave a different proof and

Keywords and phrases: Rewriting systems, regular languages, computational complexity

[1] University of Stuttgart, FMI
Universitätsstr. 38
70569 Stuttgart, Germany,
e-mail: {`lohrey, petersen`}`@informatik.uni-stuttgart.de`

extended the result to any regular set of axioms [9]. Büchi had already announced in [1] that rules with several premises still generate the regular languages. The proof was published by Büchi and Hosken in [3] and generalized to mixed systems with prefix and suffix rewriting rules (the suffix rules have to be restricted to a single premise, otherwise all recursively enumerable sets are obtained). In an article based on [1] Kratko showed the same results [12]. A more recent exposition can be found in [2].

Independently of the cited earlier work, Frazier and Page showed the original result for pure regular canonical systems (without non-terminal symbols) [8]. They called these systems *prefix grammars*, a terminology we will also use here. Petersen [16] considered the descriptional complexity of these grammars, while Ravikumar and Quan [17] mainly investigated the computational complexity of problems related to prefix grammars. There is a well-known correspondence between prefix grammars and pushdown systems, see [4, 9]. The latter model has recently been investigated in the area of model checking with an emphasis on efficient algorithms [6, 7]. Here we consider decision problems for prefix grammars in the more general framework of complexity classes.

In the present paper we will solve one of the questions left open in [17] concerning the complexity of the equivalence problem for prefix grammars6. Additional results are related to the membership problem of these grammars and to monotonic grammars. For grammars with several premises we show that the membership problem (which is solvable in polynomial time for one premise rules) is complete in EXPTIME. For monotonic grammars with several premises membership is still hard for PSPACE.

2. Preliminaries

For a finite alphabet Σ, let Σ^* denote the set of all finite words over Σ. The empty word is denoted by ε. The length of a word $w \in \Sigma^*$ is $|w|$.

We assume that the reader has some basic background in complexity theory, see e.g. [15]. We will use the standard complexity classes P, NP, PSPACE, and EXPTIME. All hardness results in this paper hold with respect to logspace reductions. This fact will not be mentioned explicitly in the rest of the paper.

3. Prefix grammars

Definition 3.1. A *prefix grammar* G is a triple $G = (\Sigma, S, P)$, where Σ is a finite alphabet, S is a finite set of strings over Σ called *axioms* or *base strings*, and P is a finite set of *productions* of the form $\alpha \to \beta$ with finite strings $\alpha, \beta \in \Sigma^*$.

Definition 3.2. Let $G = (\Sigma, S, P)$ be a prefix grammar. A string v is produced from u, denoted $u \Rightarrow_G v$, if there exist a production $\alpha \to \beta$ in P and $w \in \Sigma^*$ such that $u = \alpha w$ and $v = \beta w$. The language $L(G)$ generated by G is the smallest set of words satisfying $S \subseteq L(G)$ and closed with respect to $\Rightarrow_G$, i.e., $u \in L(G)$ and $u \Rightarrow_G v$ implies also $v \in L(G)$.

The above defined prefix grammars coincide with *pure regular systems* in the sense of Büchi [1, 2]. Ravikumar and Quan [17] also investigate monotonic prefix grammars defined in the following way.

Definition 3.3. A prefix grammar is called *monotonic*, if every production $\alpha \to \beta$ satisfies $|\alpha| \geq |\beta|$.

We also consider prefix grammars with several premises, where the generation of a string depends on more than one string.

Definition 3.4. A *prefix grammar with several premises* G is a triple $G = (\Sigma, S, P)$ where Σ and S have the same meaning as in Definition 3.1 and P is a finite set of *productions* of the form $\alpha_1, \ldots, \alpha_k \to \beta$, where $k \geq 1$ and $\alpha_1, \ldots, \alpha_k, \beta \in \Sigma^*$ (k may be different for different productions). The grammar G is called monotonic if for every production $\alpha_1, \ldots, \alpha_k \to \beta$ we have $|\alpha_i| \geq |\beta|$ for $1 \leq i \leq k$.

A string v is produced from strings $u_1, \ldots, u_k$, denoted $u_1, \ldots, u_k \Rightarrow_G v$, if there exist a production $\alpha_1, \ldots, \alpha_k \to \beta$ in P and $w \in \Sigma^*$ such that $u_i = \alpha_i w$ for all $1 \leq i \leq k$ and $v = \beta w$. The language $L(G)$ generated by G is the smallest set of words satisfying $S \subseteq L(G)$ and closed with respect to $\Rightarrow_G$, i.e., $u_1, \ldots, u_k \in L(G)$ and $u_1, \ldots, u_k \Rightarrow_G v$ implies also $v \in L(G)$.

Example 3.5. Consider the prefix grammar with several premises

$$G = (\{2, 3, 5, a\}, \{2, 3, 5\}, P),$$

where P contains the productions $2 \to 2a^2$, $3 \to 3a^3$, $5 \to 5a^5$, and $2, 3, 5 \to \varepsilon$. Then $L(G)$ is the language $\bigcup_{i \in \{2,3,5\}} i(a^i)^* \cup (a^{2 \cdot 3 \cdot 5})^*$.

Let $\mathcal{C}$ be one of the four grammar classes introduced above (prefix grammars, monotonic prefix grammars, prefix grammar with several premises, monotonic prefix grammar with several premises). We will consider the following computational problems for the class $\mathcal{C}$:

Membership for $\mathcal{C}$:
INPUT: $G \in \mathcal{C}$ and a string $w \in \Sigma^*$
QUESTION: $w \in L(G)$?

Equivalence for $\mathcal{C}$:
INPUT: $G_1, G_2 \in \mathcal{C}$
QUESTION: $L(G_1) = L(G_2)$?

Inclusion for $\mathcal{C}$:
INPUT: $G_1, G_2 \in \mathcal{C}$
QUESTION: $L(G_1) \subseteq L(G_2)$?

Inequivalence (resp. non-inclusion) for $\mathcal{C}$ is the complementary problem of equivalence (resp. inclusion) for $\mathcal{C}$. Concerning the membership problem we emphasize that the grammar is part of the input. For a fixed prefix grammar (with possibly several premises) G, $L(G)$ is a fixed regular language. Thus, the membership problem for $L(G)$ belongs to the parallel complexity class NC^1, see e.g. [19].

4. Results

We begin our investigation with problems left open by Ravikumar and Quan [17].

Theorem 4.1. *Equivalence and inclusion for prefix grammars are both complete in PSPACE.*

Proof. The upper bound follows from the effective translation of prefix grammars into NFA of polynomial size [4,17] and the fact that equivalence and inclusion are in PSPACE for NFA [18].

For the lower bound notice that equivalence reduces to inclusion and thus it suffices to consider equivalence. We will make use of the fact that the equivalence problem for NFA is complete in PSPACE (can be shown with the technique from [14]).

For the reduction of NFA equivalence to prefix grammar equivalence, consider two NFA $M_1 = (Q_1, \Sigma, \delta_1, q_0^1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_0^2, F_2)$, where Q_1 and Q_2 are disjoint sets of states, Σ is the w.l.o.g. common alphabet, δ_1 and δ_2 are transition relations, and F_1, F_2 are sets of final states.

First we construct a prefix grammar $G = (\Sigma \cup Q_1 \cup Q_2, F_1 \cup F_2, P)$ by letting

$$P = \{q \to q'x \mid (q', x, q) \in \delta_1 \cup \delta_2\}.$$

By induction $qw \in L(G)$ iff starting from q string w is accepted (by M_1 or M_2 depending on whether $q \in Q_1$ or $q \in Q_2$).

Now we form $G_1 = (\Sigma \cup Q_1 \cup Q_2, F_1 \cup F_2, P \cup \{q_0^1 \to \varepsilon\})$ and $G_2 = (\Sigma \cup Q_1 \cup Q_2, F_1 \cup F_2, P \cup \{q_0^2 \to \varepsilon\})$. By construction $L(G_1)$ and $L(G_2)$ coincide on strings from $(Q_1 \cup Q_2)\Sigma^*$. For $w \in \Sigma^*$ we have $w \in L(G_1) \iff q_0^1 w \in L(G) \iff w \in L(M_1)$ and $w \in L(G_2) \iff q_0^2 w \in L(G) \iff w \in L(M_2)$. Therefore $L(G_1) = L(G_2)$ iff M_1 and M_2 are equivalent.

The transformation is obviously possible in logspace. □

Remark: In the above proof, the straight-forward translation of M_1 and M_2 into prefix grammars, i.e., taking $G_i = (\Sigma \cup Q_i, F_i, P_i)$ with $P_i = \{q \to q'x \mid (q', x, q) \in \delta_i\} \cup \{q_0^i \to \varepsilon\}$, will result in grammars that are inequivalent in general, even if the NFA are equivalent.

Theorem 4.2. *Membership for prefix grammars is complete in P.*

Proof. The upper bound is Theorem 6.1 of [17].

Hardness follows from the result that emptiness for context-free grammars is complete in P [11]. Notice that it suffices to consider context-free grammars that either generate $\emptyset$ or $\{\varepsilon\}$ by replacing every occurrence of a terminal symbol by ε.

Now take the context-free productions as productions of a prefix grammar and observe that there is no loss of generality in considering left-most derivations. The prefix grammar will derive ε from the initial symbol iff the initial context-free grammar generated a nonempty language. □

Theorem 4.3. *Inequivalence and non-inclusion for monotonic prefix grammars are both complete in NP.*

Proof. Inequivalence trivially reduces to non-inclusion. Given two monotonic prefix grammars G_1 and G_2, the length of a string $w \in (L(G_1) - L(G_2))$ is bounded by the longest base string. Such a string can therefore be guessed in polynomial time. Then the two membership problems are tested in deterministic polynomial time according to Theorem 6.1 from [17].

For the lower bound we define a reduction from CNF-SAT to inequivalence for monotonic prefix grammar (in [17] a similar construction is applied to *inclusion*, but since inclusion does not reduce to equivalence, we cannot use the result from [17] directly).

Let $F = F_1 \wedge F_2 \wedge \cdots \wedge F_k$ be a CNF-formula over the variables $x_1, \ldots, x_n$ with clauses $F_1, F_2, \ldots, F_k$. We view every F_i as a subset of $\{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$. We assume w.l.o.g. that no clause contains both a variable and its complement. A truth assignment to the variables $x_1, x_2, \ldots, x_n$ will be identified with the string $a_1 a_2 \cdots a_n$, where $a_i = t$ if x_i is true and $a_i = f$ if x_i is false.

Now we form a grammar $G_1 = (\Sigma, S_1, R_1)$ with $\Sigma = \{t, f, 1, \ldots, k\}$ and

$$S_1 = \{is_1 s_2 \cdots s_n \mid 1 \leq i \leq k,\ \forall 1 \leq m \leq n : s_m = f \text{ if } x_m \in F_i, \text{ else } s_m = t\}.$$

Let

$$\begin{aligned} R = \{ & is_1 s_2 \cdots s_{r-1} t \to is_1 s_2 \cdots s_{r-1} f \mid \\ & 1 \leq i \leq k, \neg x_r \notin F_i, \\ & \forall 1 \leq m < r : s_m = f \text{ if } x_m \in F_i,\ \text{else } s_m = t\}. \end{aligned}$$

We define $R_1 = R \cup \{i \to \varepsilon \mid 1 \leq i \leq k\}$.

The grammar to be compared with G_1 is $G_2 = (\Sigma, S_2, R_2)$, where $S_2 = S_1 \cup \{t^n\}$. We define $R_2 = R_1 \cup \{t^m \to t^{m-1} f \mid 1 \leq m \leq n\}$.

Since $S_1 \subseteq S_2$ and $R_1 \subseteq R_2$, we certainly have $L(G_1) \subseteq L(G_2)$. The additional base string and productions of G_2 will generate all strings from $\{t, f\}^n$. In order to be inequivalent to G_2, at least one string of this form has to be missing from $L(G_1)$. Now observe that a truth assignment to $x_1, \ldots, x_n$ does not satisfy F iff at least one clause F_i is not satisfied. All truth assignments with this property are generated by G_1. Therefore F is satisfiable iff $L(G_1) \neq L(G_2)$. □

Theorem 4.4. *Membership for monotonic prefix grammars with several premises is hard for PSPACE.*

Proof. We describe a reduction from QBF (where formulas are in 3CNF) to membership for monotonic prefix grammars with several premises.

In the same way as in the proof of Theorem 4.3, $F = F_1 \wedge F_2 \wedge \cdots \wedge F_k$ is a CNF-formula over the variables $x_1, \ldots, x_n$ and the quantified formula to be tested has the form

$$Q_n x_n Q_{n-1} x_{n-1} \cdots Q_1 x_1 F$$

with $Q_i \in \{\forall, \exists\}$ for $1 \leq i \leq n$.

For $1 \leq m \leq k$ and $1 \leq j \leq n$ let

$$\phi_m(j) = \begin{cases} t & \text{if } x_j \in F_m \\ f & \text{otherwise.} \end{cases}$$

We define a grammar $G = (\Sigma, S, R')$ with $\Sigma = \{t, f, 1, \dots, k, \tau_1, \dots, \tau_{n+1}\}$ and

$$\begin{aligned} S \quad = \quad & \{is_1 s_2 \cdots s_n \mid \\ & \qquad 1 \leq i \leq k, \exists j : s_j = \phi_i(j) \text{ and } (x_j \in F_i \text{ or } \neg x_j \in F_i) \\ & \qquad \forall 1 \leq m \leq n : s_m = t \text{ if } (x_m \notin F_i \text{ and } \neg x_m \notin F_i)\}. \end{aligned}$$

Notice that for each clause F_i seven axioms will be included in S (one axiom for each satisfying truth assignment to the three literals appearing in F_i).

Let

$$\begin{aligned} R = \{is_1 s_2 \cdots s_{r-1} t \rightarrow is_1 s_2 \cdots s_{r-1} f \mid 1 \leq i \leq k,\ x_r \notin F_i,\ \neg x_r \notin F_i, \\ is_1 s_2 \cdots s_{r-1} t \text{ is a prefix of some } \alpha \in S\}. \end{aligned}$$

With the productions from R we can derive for every i all strings of the form iw, where w satisfies the clause F_i. Finally, we define

$$\begin{aligned} R' = R \cup \{1, \dots, k \rightarrow \tau_1\} \cup \\ \{\tau_i u \rightarrow \tau_{i+1} \mid 1 \leq i \leq n, Q_i = \exists, u \in \{t, f\}\} \cup \\ \{\tau_i t, \tau_i f \rightarrow \tau_{i+1} \mid 1 \leq i \leq n, Q_i = \forall\}. \end{aligned}$$

We claim that $\tau_{n+1} \in L(G)$ iff the quantified formula $Q_n x_n \cdots Q_1 x_1 F$ evaluates to true. Before we prove this claim, we first consider an example. Let us take the quantified formula

$$\forall x_3 \forall x_2 \exists x_1 : (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3),$$

which evaluates to true. Then S contains the following axioms:

$$\begin{array}{ccccccc} 1ttt & 1ttf & 1tft & 1tff & 1ftt & 1ftf & 1fff \\ 2ftt & 2ftf & 2fft & 2fff & 2ttt & 2ttf & 2tft \end{array}$$

The set R is empty because every clause contains either x_i or $\neg x_i$ for $i \in \{1, 2, 3\}$. Using the production $1, 2 \rightarrow \tau_1$ we can derive the strings

$$\tau_1 ttt \qquad \tau_1 ttf \qquad \tau_1 tft \qquad \tau_1 ftt \qquad \tau_1 ftf \qquad \tau_1 fff.$$

These are all truth assignments that satisfy $(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3)$. Using the productions $\tau_1 t \rightarrow \tau_2$ and $\tau_1 f \rightarrow \tau_2$ (note that x_1 is quantified existentially), we obtain the strings

$$\tau_2 tt \qquad \tau_2 tf \qquad \tau_2 ft \qquad \tau_2 ff.$$

Then $\tau_2 t, \tau_2 f \to \tau_3$ yields the strings $\tau_3 t$ and $\tau_3 f$, from which we finally obtain τ_4 using the production $\tau_3 t, \tau_3 f \to \tau_4$.

In order to prove "$\tau_{n+1} \in L(G) \iff Q_n x_n \cdots Q_1 x_1 F$ evaluates to true", we will show that a string of the form $\tau_i w$ with $w \in \{t, f\}^{n-i+1}$ can be derived iff the quantified formula $Q_{i-1} x_{i-1} \cdots Q_1 x_1 F$ is satisfied by the truth assignment to $x_i, \ldots, x_n$ corresponding to w. We prove this claim by induction over i. For $i = 1$ there are no quantifiers in $Q_{i-1} x_{i-1} \cdots Q_1 x_1 F$. By our construction a string of the form $\tau_1 w$ can only be generated by a derivation $1w, \ldots, kw \Rightarrow_G \tau_1 w$. Then every clause has to contain a satisfied literal, and consequently the conjunction F of all clauses is satisfied by the truth assignment corresponding to w. For $i > 1$ assume that the claim holds for $i - 1$. Thus, for all $w' \in \{t, f\}^{n-i+2}$, $\tau_{i-1} w'$ can be derived if and only if $Q_{i-2} x_{i-2} \cdots Q_1 x_1 F$ is satisfied by the truth assignment to $x_{i-1}, \ldots, x_n$ corresponding to w'. If $Q_{i-1} = \exists$, then the last derivation step was $\tau_{i-1} tw \Rightarrow_G \tau_i w$ or $\tau_{i-1} fw \Rightarrow_G \tau_i w$. By the induction hypothesis at least one of tw or fw satisfies $Q_{i-2} x_{i-2} \cdots Q_1 x_1 F$ and therefore w satisfies $Q_{i-1} x_{i-1} \cdots Q_1 x_1 F$. If $Q_{i-1} = \forall$, then the last step was necessarily $\tau_{i-1} fw, \tau_{i-1} tw \Rightarrow_G \tau_i w$ and by the hypothesis both tw and fw satisfy $Q_{i-2} x_{i-2} \cdots Q_1 x_1 F$. This argument shows for $i = n + 1$ that $\tau_{n+1} \in L(G)$ iff the quantified formula $Q_n x_n \cdots Q_1 x_1 F$ evaluates to true. □

We will finally show in this section that membership for arbitrary (not necessarily monotonic) prefix grammars with several premises is complete in EXPTIME. Our proof of this result is based on a strong relationship between prefix grammars with several premises and alternating pushdown automata.

An *alternating pushdown automaton* (alternating PDA for short) [5] is a pushdown automaton $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, F, \perp, \delta)$, where Q is the set of states, Σ is the input alphabet, Γ is the pushdown alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, $\perp \in \Gamma$ is the bottom symbol, and $\delta : Q \times \Gamma \times \Sigma \cup \{\varepsilon\} \to 2^{Q \times \Gamma^*}$ is the transition function. In addition to an ordinary nondeterministic pushdown automaton, the set of non-final states $Q \setminus F$ is partitioned into two sets $Q_\exists$ and $Q_\forall$. A configuration of $\mathcal{A}$ is a triple $(q, u, v) \in Q \times \Gamma^* \times \Sigma^*$. The notion of an accepting configuration is defined as follows:

- every configuration from $F \times \Gamma^* \times \{\varepsilon\}$ is accepting.
- a configuration of the form (q, Au, av) with $q \in Q_\exists$, $A \in \Gamma$, and $a \in \Sigma \cup \{\varepsilon\}$ is accepting if there exists $(p, \alpha) \in \delta(q, A, a)$ such that $(p, \alpha u, v)$ is accepting.
- a configuration of the form (q, Au, av) with $q \in Q_\forall$, $A \in \Gamma$, and $a \in \Sigma \cup \{\varepsilon\}$ is accepting if for all $(p, \alpha) \in \delta(q, A, a)$ the configuration $(p, \alpha u, v)$ is accepting.

The alternating PDA $\mathcal{A}$ accepts the string $v \in \Sigma^*$ if the configuration $(q_0, \perp, v)$ is accepting. The language $L(\mathcal{A})$ accepted by $\mathcal{A}$ consists of all strings accepted by

$\mathcal{A}$. In [5] it is shown that the class of all languages that can be accepted by an alternating PDA is precisely ETIME $= \bigcup_{c>0} \mathrm{DTIME}(2^{cn})$. Since ETIME contains EXPTIME-complete languages[1], we obtain the following lemma:

Lemma 4.5. *There exists a fixed alternating pushdown automaton that accepts an EXPTIME-complete language.*

An *alternating auxPDA* [13] is an alternating PDA with a two-way input tape that has an additional two-way read-write tape of size $O(\log(n))$, where n is the input length. The class of all languages that can be accepted by an alternating auxPDA is precisely EXPTIME $= \bigcup_{c>0} \mathrm{DTIME}(2^{(n^c)})$ [13].

Theorem 4.6. *Membership for prefix grammars with several premises is complete in EXPTIME.*

Proof. In order to show membership in EXPTIME it suffices to show that membership for prefix grammars with several premises can be solved on an alternating auxPDA $\mathcal{A}$. Thus, we will construct an alternating auxPDA $\mathcal{A}$ such that

$$L(\mathcal{A}) = \{(w, G) \mid G \text{ is a prefix grammar with several premises and } w \in L(G)\}.$$

Here, the grammar G is encoded in some canonical way as a string, for instance by listing its productions separated by some distinguished symbol. We only sketch the automaton $\mathcal{A}$. Let $G = (\Sigma, S, P)$ be a prefix grammar with several premises and let $w \in \Sigma^*$ be the word that has to be tested for membership in $L(G)$. The input tape of $\mathcal{A}$ contains both, the word w and a description of G. The pushdown of $\mathcal{A}$ stores a word over the alphabet Σ. Since Σ is part of the input for $\mathcal{A}$, a single symbol from Σ has to be stored by a binary sequence of length $O(\log(n))$, where n is the total length of the input of $\mathcal{A}$. Two of these blocks have to be separated by a special symbol. The automaton $\mathcal{A}$ starts by pushing the word w on the pushdown by reading w from right to left. This ensures that the topmost block of the pushdown represents the first symbol of w. Then $\mathcal{A}$ repeats the following step, where each time $\mathcal{A}$ chooses nondeterministically (in a state from $Q_\exists$) between one of the following two alternatives:

- $\mathcal{A}$ checks, whether the pushdown contains an axiom from the set of words S. To check this, $\mathcal{A}$ guesses nondeterministically one of the axioms $u \in S$ and tries to pop $u\bot$ from the pushdown. If this succeeds, then $\mathcal{A}$ accepts, otherwise $\mathcal{A}$ rejects.
- $\mathcal{A}$ tries to apply one of the productions from P. For this, $\mathcal{A}$ guesses nondeterministically a production $\alpha_1, \ldots, \alpha_k \to \beta$. Then, $\mathcal{A}$ tries to pop β from the pushdown. If this does not succeed, then $\mathcal{A}$ rejects. Otherwise $\mathcal{A}$ branches universally (i.e., in a state from $Q_\forall$) into k different branches. In the i-th branch, $\mathcal{A}$ pushes the word α_i onto the pushdown.

[1]If L is an EXPTIME-complete language that can be decided in time $2^{p(n)}$ for some polynomial $p(n)$, then the language $K = \{w\$^{p(|w|)} \mid w \in L\}$, where \$ is a new symbol, belongs to ETIME and L can be reduced to K in logspace.

The intuition behind the automaton $\mathcal{A}$ is the following: Instead of trying to derive the word w from the axioms in S by using the productions from P, $\mathcal{A}$ tries to construct a proof tree for the fact that $w \in L(G)$. This proof tree is constructed by starting from the root of the tree (i.e., the word w) and by applying the productions of P in reverse order. Automaton $\mathcal{A}$ chooses a specific production that is applied to the current pushdown contents by branching existentially. The several words that result by applying a production from P in reverse order are verified by branching universally. Note that the additional logspace bounded read-write tape is actually not needed. On the other hand, the possibility of auxPDA to read the input tape in a two-way mode is crucial for the above construction.

To sum up, the alternating auxPDA $\mathcal{A}$ accepts its input (w, G) if and only if $w \in L(G)$. This shows that membership for prefix grammars with several premises belongs to EXPTIME.

In order to prove EXPTIME-hardness, it suffices by Lemma 4.5 to simulate a specific alternating PDA by a prefix grammar with several premises. Let us fix an alternating PDA $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, F, \bot, \delta)$ that accepts an EXPTIME-complete language. Let $w \in \Sigma^*$ be an input for $\mathcal{A}$. We will construct a prefix grammar $G(w)$ with several premises such that $wq_0\bot \in L(G(w))$ if and only if $w \in L(\mathcal{A})$. W.l.o.g. we may assume that if $\mathcal{A}$ enters a final state from F, then the pushdown only contains the bottom symbol $\bot$ and the input word w is completely read. The alphabet of $G(w)$ is $\Sigma \cup \Gamma \cup Q$ (here we assume that the three sets Σ, Γ, and Q are pairwise disjoint). The set of axioms of $G(w)$ is $S = \{q\bot \mid q \in F\}$. To define the set of productions of $G(w)$, let us fix a triple $(q, A, a) \in Q \times \Gamma \times \Sigma \cup \{\varepsilon\}$ and let $\delta(q, A, a) = \{(p_1, \alpha_1), \ldots, (p_k, \alpha_k)\}$. If $q \in Q_\exists$, then for every $1 \le i \le k$ and every factorization $w = uav$ of the word w we put the production $vp_i\alpha_i \to avqA$ into the set of productions. Now assume that $q \in Q_\forall$. Then for every factorization $w = uav$ we put the production $vp_1\alpha_1, \ldots, vp_k\alpha_k \to avqA$ into the set of productions. For the resulting grammar $G(w)$ one can easily prove by induction that a configuration $(q, u, v) \in Q \times \Gamma^* \times \Sigma^*$, which is reachable from the initial configuration $(q_0, \bot, w)$ of $\mathcal{A}$, is accepting if and only if $vqu \in L(G(w))$. Thus, $wq_0\bot \in L(G(w))$ if and only if $w \in L(\mathcal{A})$. □

5. Conclusion

We have shown that equivalence and inclusion for prefix grammars are complete in PSPACE, a property shared by the corresponding problems for regular expressions and nondeterministic finite automata. Membership however is complete in P for prefix grammars, which appears to be harder than for the two other formalisms, where the problem is complete in NL. With several premises membership is even complete in EXPTIME.

The following table collects our results together with the well-known bounds for NFA. For the equivalence problem, the same results as for the inclusion problem hold. With two exceptions the bounds are shown in Section 4 or follow immediately. The EXPSPACE upper bound on inclusion for prefix grammars with multiple premises can be derived by an analysis of the construction from [3]. It leads to NFA of exponential size which are equivalent to the original grammars. An inclusion test can then be applied to the automata. The EXPTIME upper bound on inclusion for monotonic prefix grammars with multiple premises is obtained by enumerating all possible words (their length is bounded by the longest axiom) and checking for membership.

	membership	inclusion
NFA	NL	PSPACE
prefix grammars	P	PSPACE
monotonic prefix grammars	NL $\cdots$ P	Co-NP
prefix grammars with multiple premises	EXPTIME	EXPTIME $\cdots$ EXPSPACE
monotonic prefix grammars with multiple premises	PSPACE $\cdots$ EXPTIME	PSPACE $\cdots$ EXPTIME

An entry with a single complexity class means completeness for that class. An entry $\mathcal{C}_1 \cdots \mathcal{C}_2$ for complexity classes $\mathcal{C}_1$ and $\mathcal{C}_2$ means that $\mathcal{C}_1$ is a lower bound for the corresponding problem, whereas $\mathcal{C}_2$ is an upper bound.

Acknowledgments

Helpful comments of the referees of DCFS 2003 on a preliminary version of this paper are gratefully acknowledged. This work has been supported by the Hungarian-German research project No. D 39/2000 of the Hungarian Ministry of Education and the German BMBF.

References

[1] J. R. Büchi. Regular canonical systems. *Archiv Math. Logik und Grundlagenforschung*, 6:91–111, 1964.

[2] J. R. Büchi. *Finite Automata, their Algebras and Grammars.* Springer, Berlin-Heidelberg-New York, 1989.

[3] J. R. Büchi and W. H. Hosken. Canonical systems which produce periodic sets. *Mathematical Systems Theory*, 4:81–90, 1970.

[4] D. Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 106:61–86, 1992.

[5] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, 1981.

[6] J. Esparza, D. Hansel, P. Rossmanith, and S. Schwoon. Efficient algorithms for model checking pushdown systems. In E. A. Emerson and A. P. Sistla, editors, *Proc. of 12th International Conference on Computer Aided Verification (CAV)*, number 1855 in Lecture Notes in Computer Science, pages 232–247. Springer, 2000.

[7] J. Esparza, A. Kucera, and S. Schwoon. Model checking LTL with regular valuations for pushdown systems. *Information and Computation*, 186:355–376, 2003.

[8] M. Frazier and C. D. Page Jr. Prefix grammars: An alternative characterization of the regular languages. *Information Processing Letters*, 51:67–71, 1994.

[9] S. A. Greibach. A note on pushdown store automata and regular systems. *Proceedings of the AMS*, 18:263–268, 1967.

[10] J. E. Hopcroft and R. M. Karp. A linear algorithm for testing the equivalence of finite automata. Report TR 71-114, Department of Computer Science, Cornell University, 1971.

[11] N. D. Jones and W. T. Laaser. Complete problems for deterministic polynomial time. *Theoretical Computer Science*, 3:105–117, 1977.

[12] M. Kratko. Formal post calculi and finite automata. *Problemy Kibernet.*, 17:41–65, 1966. In Russian.

[13] R. E. Ladner, R. J. Lipton, and L. J. Stockmeyer. Alternating pushdown and stack automata. *SIAM Journal on Computing*, 13(1):135–155, 1984.

[14] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th Annual IEEE Symposium on Switching and Automata Theory, College Park (Maryland)*, pages 125–129, 1972.

[15] C. H. Papadimitriou. *Computational Complexity.* Addison Wesley, 1994.

[16] H. Petersen. Prefix rewriting and descriptional complexity. *Journal of Automata, Languages and Combinatorics*, 5:245–254, 2000.

[17] B. Ravikumar and L. Quan. Efficient algorithms for prefix grammars. Available at: `http://www.cs.sonoma.edu/~ravi`, 2002.

[18] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *Proceedings of the 5th ACM Symposium on Theory of Computing (STOC'73), Austin (Texas)*, pages 1–9, 1973.

[19] H. Straubing *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, 1994.

Communicated by (The editor will be set by the publisher).
(The dates will be set by the publisher).