Word problems on compressed words

Markus Lohrey

Universität Stuttgart, FMI, Universitätsstr. 38, 70569 Stuttgart, Germany
lohrey@fmi.uni-stuttgart.de

Abstract. We consider a compressed form of the word problem for finitely presented monoids, where the input consists of two compressed representations of words over the generators of a monoid $\mathcal{M}$, and we ask whether these two words represent the same monoid element of $\mathcal{M}$. For compression we use straight-line programs. For several classes of monoids we obtain completeness results for complexity classes in the range from P to EXPSPACE. As a by-product of our results on compressed word problems we obtain a fixed deterministic context-free language with a PSPACE-complete membership problem. The existence of such a language was open so far. Finally, we investigate the complexity of the compressed membership problem for various circuit complexity classes.

1 Introduction

During the last decade, the massive increase in the volume of data has motivated the investigation of algorithms on *compressed data*, like for instance compressed strings, trees, or pictures. The general goal is to develop algorithms that directly work on compressed data without prior decompression. Let us mention here the work on compressed pattern matching, see, e.g., [21]. In this paper we investigate two classes of computational problems on compressed data that are of central importance in theoretical computer science since its very beginning: the *word problem* and the *membership problem*. In its most general form, the word problem asks whether two terms over an algebraic structure represent the same element of the structure. Here, we consider the word problem for *finitely presented monoids*, which are given by finite sets of generators and defining relations. In this case the input consists of two finite words over the set of generators and we ask whether these two words represent the same monoid element. The undecidability results concerning the word problem for finitely presented monoids/groups are among the first undecidability results that touched "real mathematics", see [23] for references. Moreover, these negative results motivated a still ongoing investigation of decidable subclasses of word problems and their complexity. In particular, monoids that can be presented by terminating and confluent semi-Thue systems (i.e., string rewriting systems) received a lot of attention [8]. These monoids have decidable word problems, and sharp complexity bounds are known for various subclasses [7, 15, 16].

In its compressed variant, the input to the word problem for a finitely presented monoid consists of two compressed representations of words over the generators. We choose *straight-line programs*, or equivalently context-free grammars

that generate exactly one word, for compression. Straight-line programs turned out to be a very flexible compressed representation of strings. Several other compressed representations, like for instance Lempel-Ziv factorizations [28], can be efficiently converted into straight-line programs and vice versa [19], which implies that most of our complexity results hold for Lempel-Ziv factorizations as well. Moreover, by using straight-line programs for representing inputs, the compressed word problem becomes equivalent to the circuit equivalence problem (a generalization of the well-known circuit evaluation problem), where we ask whether two circuits over a finitely presented monoid $\mathcal{M}$ (i.e., acyclic directed graphs with leafs labeled by generators of $\mathcal{M}$ and internal nodes labeled by the monoid operation) evaluate to the same element of $\mathcal{M}$. So far this problem was only investigated for finite monoids [5]. In Section 3–5 we study the complexity of compressed word problems for several subclasses of monoids presented by terminating and confluent semi-Thue systems. We obtain completeness results for various complexity classes between P and EXPSPACE. The general phenomenon that we observe when moving from the (uncompressed) word problem to its compressed variant is an exponential jump with respect to complexity. This exponential jump is well known also from other work on the complexity of succinct problems [12, 25, 26].

As a by-product of our investigation of compressed word problems we obtain several new results concerning *compressed membership problems*. Here, the problem is to decide for a fixed language L, whether a given compressed representation of a word belongs to L [19]. We show that there exists a deterministic context-free (even deterministic linear) language with a PSPACE-complete compressed membership problem, which solves an open problem from [9, 19]. This result is also interesting in light of recent attempts to use straight-line programs for compressing control flow traces of procedural programming languages [27]. At a certain level of abstraction, the set of all valid control flow traces is a context-free language. We also present a context-sensitive language with an EXPSPACE-complete compressed membership problem. Finally, in Section 6 we investigate the complexity of the compressed membership problem for various circuit complexity classes. We show that the levels of the logtime hierarchy [22] correspond in a compressed setting to the levels of the polynomial time hierarchy. A full version of this paper can be obtained from the author.

2 Preliminaries

We assume that the reader has some basic background in complexity theory [17]. The reflexive and transitive closure of a binary relation $\to$ is $\overset{*}{\to}$. Let Γ be a finite alphabet. The *empty word* over Γ is denoted by ε. For a word $s = a_1 a_2 \cdots a_n \in \Gamma^*$ $(a_i \in \Gamma)$ let $w^{\mathrm{rev}} = a_n a_{n-1} \cdots a_1$, $\mathrm{alph}(s) = \{a_1, \ldots, a_n\}$, $|s| = n$, $|s|_a = |\{i \mid a_i = a\}|$ (for $a \in \Gamma$), $s[i] = a_i$ (for $1 \leq i \leq n$), and $s[i,j] = a_i a_{i+1} \cdots a_j$ (for $1 \leq i \leq j \leq n$). If $i > j$ we set $s[i,j] = \varepsilon$. An *involution* $\overline{\ }$ on Γ is a function $\overline{\ } : \Gamma \to \Gamma$ with $\overline{\overline{a}} = a$ for all $a \in \Gamma$. It can be extended to an involution on Γ^* by setting $\overline{a_1 \cdots a_n} = \overline{a_n} \cdots \overline{a_1}$. With $\overline{\Gamma} = \{\overline{a} \mid a \in \Gamma\}$

we always denote a disjoint copy of the alphabet Γ. Then we can define an involution $^-$ on $\Delta = \Gamma \cup \overline{\Gamma}$ by setting $\overline{\overline{a}} = a$; this involution will be extended to Δ^* in the above way. A *weight-function* is a homomorphism $f : \Gamma^* \to \mathbb{N}$ from the free monoid Γ^* to the natural numbers (with $+$) such that $f^{-1}(0) = \{\varepsilon\}$. Given a linear oder $\succ$ on the alphabet Γ, we extend $\succ$ to a linear order on Γ^*, called the *lexicographic extension of* $\succ$, as follows: $u \succ v$ if v is a prefix of u or $u = wau'$ and $v = wbv'$ with $a, b \in \Gamma$ and $a \succ b$.

Semi-Thue systems and finitely presented monoids For more details and references on the topic of this section see [8]. Let Γ be a finite alphabet. A *semi-Thue system* R over Γ is a finite subset $R \subseteq \Gamma^* \times \Gamma^*$; its elements are called rules. A rule $(s,t) \in R$ is also written as $s \to t$. The pair (Γ, R) is a *presentation*. Let $\mathrm{dom}(R) = \{s \mid \exists t : (s,t) \in R\}$. We define the binary relation $\to_R$ on Γ^* as follows: $s \to_R t$ if there exist $u, v \in \Gamma^*$ and $(\ell, r) \in R$ with $s = u\ell v$ and $t = urv$. Moreover, let ${}_R\leftarrow = (\to_R)^{-1}$, $\leftrightarrow_R = (\to_R \cup {}_R\leftarrow)$, and $\mathrm{IRR}(R) = \Gamma^* \backslash \Gamma^* \mathrm{dom}(R) \Gamma^*$ (the set of *irreducible words*). We say that (Γ, R) is *terminating* if there do not exist $s_i \in \Gamma^*$ for $i \in \mathbb{N}$ with $s_i \to_R s_{i+1}$ for all $i \in \mathbb{N}$. We say that (Γ, R) is *confluent* (resp. *locally confluent*) if for all $s, t, u \in \Gamma^*$ with $t \,{}_R\overset{*}{\leftarrow} s \overset{*}{\to}_R u$ (resp. $t \,{}_R\leftarrow s \to_R u$) there exists $v \in \Gamma^*$ such that $t \overset{*}{\to}_R v \,{}_R\overset{*}{\leftarrow} u$. By Newman's lemma, a terminating presentation is confluent if and only if it is locally confluent. Moreover, for a terminating presentation, local confluence (and hence confluence) can be checked effectively using *critical pairs*, which result from overlapping left-hand sides. The reflexive and transitive closure $\overset{*}{\leftrightarrow}_R$ is a congruence on the free monoid Γ^*, hence we can define the quotient monoid $\Gamma^*/\overset{*}{\leftrightarrow}_R$, which we denote by $\mathcal{M}(\Gamma, R)$. It is called a *finitely presented monoid*, and we say that $\mathcal{M}(\Gamma, R)$ is the *monoid presented by* (Γ, R). The *word problem* for the fixed presentation (Γ, R) is the following decision problem:

INPUT: Two words $s, t \in \Gamma^*$.
QUESTION: Does $s \overset{*}{\leftrightarrow}_R t$ hold?

It is easy to see that for two given presentations (Γ, R) and (Σ, S) such that $\mathcal{M}(\Gamma, R) \cong \mathcal{M} \cong \mathcal{M}(\Sigma, S)$, there exists a logspace reduction from the word problem for (Γ, R) to the word problem for (Σ, S). Thus, the decidability and complexity of the word problem do not depend on the chosen presentation and we may just speak of the word problem for the monoid $\mathcal{M}$.

If (Γ, R) is terminating and confluent, then every $s \in \Gamma^*$ has a unique *normal form* $\mathrm{NF}_R(s) \in \mathrm{IRR}(R)$ satisfying $s \overset{*}{\to}_R \mathrm{NF}_R(s)$. Moreover, $s \overset{*}{\leftrightarrow}_R t$ if and only if $\mathrm{NF}_R(s) = \mathrm{NF}_R(t)$. Thus, the word problem is decidable. On the other hand, the calculation of normal forms does not yield any upper bound on the complexity of the word problem [3]. Complexity results on word problems for restricted classes of finitely presented monoids can be found for instance in [7, 15, 16].

Grammar based compression Following [19], a *straight-line program (SLP) over the alphabet* Γ is a context-free grammar $G = (V, \Gamma, S, P)$, where V is the set of nonterminals, Γ is the set of terminals, $S \in V$ is the initial nonterminal, and $P \subseteq V \times (V \cup \Gamma)^*$ is the set of productions, such that (i) for every $X \in V$

there is exactly one $\alpha \in (V \cup \Gamma)^*$ with $(X, \alpha) \in P$ and (ii) there is no cycle in the relation $\{(X, Y) \in V \times V \mid \exists \alpha : (X, \alpha) \in P, Y \in \mathrm{alph}(\alpha)\}$.[1] The language generated by the SLP G contains exactly one word that is denoted by $\mathrm{eval}(G)$. More generally, every nonterminal $X \in V$ produces exactly one word that is denoted by $\mathrm{eval}_G(X)$. We omit the index G if the underlying SLP is clear from the context. We also write $P(G)$ for the set of productions P. The size of G is $|G| = \sum_{(X,\alpha)\in P} |\alpha|$. Every SLP can be transformed in polynomial time into an equivalent SLP that is in Chomsky normal form (as a context-free grammar). We may also allow exponential expressions of the form A^i for $A \in V$ and a binary coded integer $i \in \mathbb{N}$ in the right-hand sides of productions. Such a production can be replaced by $O(\log(i))$ many ordinary productions. The following tasks can be solved in polynomial time; the first two problems can be reduced to simple arithmetic, whereas the third problem requires more subtle techniques:

- Given a SLP G, calculate $|\mathrm{eval}(G)|$.
- Given a SLP G and a number $i \in \{1, \ldots, |\mathrm{eval}(G)|\}$, calculate $\mathrm{eval}(G)[i]$.
- Given SLPs G_1 and G_2, decide whether $\mathrm{eval}(G_1) = \mathrm{eval}(G_2)$ [18].

Let (Γ, R) be a fixed presentation. The *compressed word problem* for (Γ, R) is the following problem:

INPUT: Two SLPs G_1 and G_2 over the terminal alphabet Γ.
QUESTION: Does $\mathrm{eval}(G_1) \overset{*}{\leftrightarrow}_R \mathrm{eval}(G_2)$ hold?

Here, the input size is $|G_1| + |G_2|$. It is easy to see that also for the compressed word problem the complexity does not depend on the chosen presentation, which allows to speak of the compressed word problem for the monoid $\mathcal{M} = \mathcal{M}(\Gamma, R)$. We can view the compressed word problem also from another perspective. A *circuit $\mathcal{C}$ over $\mathcal{M}$* is a finite directed acyclic graph with exactly one node of outdegree 0. The nodes of indegree 0 are labeled with elements from Γ. All nodes of indegree greater than zero are labeled with the multiplication of $\mathcal{M}$. Such a circuit computes in a natural way an element of $\mathcal{M}$. Then, the question, whether two given circuits over $\mathcal{M}$ compute the same monoid element, is equivalent to the compressed word problem for $\mathcal{M}$. In [5], it was shown that for a finite non-solvable monoid the compressed word problem is P-complete, whereas for every finite solvable monoid the compressed word problem belongs to $\mathrm{DET} \subseteq \mathrm{NC}^2 \subseteq \mathrm{P}$. Our work can be seen as a first step towards extending the work from [5] to infinite monoids.

For a given language $L \subseteq \Gamma^*$ we also consider the *compressed membership problem* for the language L, which is the following problem:

INPUT: A SLP G over the terminal alphabet Γ.
QUESTION: Does $\mathrm{eval}(G) \in L$ hold?

Most of our complexity results can be also transfered to other compression schemes, like for instance Lempel-Ziv 77 (LZ77) [28]. If G is a SLP of size n with $\mathrm{eval}(G) = w$, then $\mathrm{LZ}(w)$ (the LZ77-compressed representation of w) has

[1] Usually, the term "straight-line program" is used in order to denote a linear sequence of instructions. In our context, the only instruction is the concatenation of words.

size $O(n)$ and can be constructed in polynomial time [19]. On the other hand, if n is the size of $\mathrm{LZ}(w)$, then we can construct in polynomial time a SLP of size $O(n^2 \cdot \log(n))$ generating w [19]. Thus, if we allow polynomial time reductions, the completeness results from Section 4-6 also hold, if we use LZ77 for compression. P-hardness results cannot be transfered directly, because the transformation from a SLP to the LZ77-compressed representation might be P-hard.

3 Polynomial time cases

It is obvious that for every finite monoid the compressed word problem belongs to P. In this section we present a class of infinite monoids with polynomial time solvable compressed word problems. This class contains all free groups. In fact, it turns out that for every non-abelian free group the compressed word problem is P-complete.

A presentation (Γ, R) is 2-homogeneous if for every $(\ell, r) \in R$: $|\ell| = 2$ and $r = \varepsilon$ [6]. In [16] it was shown that for every 2-homogeneous presentation the word problem is in logspace. Moreover, the uniform variant of the word problem for 2-homogeneous presentations, where the presentation is part of the input, is complete for symmetric logspace [16]. The following result was shown in [6]:

Proposition 1. *For every 2-homogeneous presentation (Γ, R) there exists a 2-homogeneous and confluent presentation (Σ, S) with $\mathcal{M}(\Gamma, R) \cong \mathcal{M}(\Sigma, S)$.*

For the further consideration let us fix a 2-homogeneous presentation (Γ, R). By Prop. 1 we may assume that (Γ, R) is confluent. Then we have:

Lemma 1 (cf. [16]). *There exist pairwise disjoint sets $\Sigma_\ell, \Sigma_r, \Delta \subseteq \Gamma$, an involution $\overline{\ } : \Delta \to \Delta$, and a semi-Thue system $S \subseteq \{(ab, \varepsilon) \mid a \in \Sigma_\ell,\ b \in \Sigma_r\}$ such that $\Gamma = \Sigma_\ell \cup \Sigma_r \cup \Delta$ and $R = S \cup \{(a\overline{a}, \varepsilon) \mid a \in \Delta\}$.*

We say that (Γ, R) is *N-free*, if $a, b \in \Sigma_\ell$, $c, d \in \Sigma_r$ (where Σ_ℓ and Σ_r result from the previous lemma), and $ac, ad, bc \in \mathrm{dom}(R)$ imply $bd \in \mathrm{dom}(R)$.

Theorem 1. *If (Γ, R) is 2-homogeneous, confluent, and N-free, then the compressed word problem for $\mathcal{M}(\Gamma, R)$ is in P.*

In the next section we will see that Thm. 1 cannot be extended to the non-N-free case unless P = NP. For the proof of Thm. 1 we need a generalization of straight-line programs from [9]: A *composition system* $G = (V, \Gamma, S, P)$ is defined analogously to a SLP, but in addition to ordinary productions it may also contain productions of the form $A \to B[i, j]$ for $B \in V$ and $i, j \in \mathbb{N}$. For such a production we define $\mathrm{eval}_G(A) = \mathrm{eval}_G(B)[i, j]$.[2] As for SLPs we define $\mathrm{eval}(G) = \mathrm{eval}_G(S)$. In [9] it was shown that for two given composition systems G_1 and G_2, the equality $\mathrm{eval}(G_1) = \mathrm{eval}(G_2)$ can be verified in polynomial time, which generalizes the corresponding result for SLPs from [18]. The proof of Thm. 1 is based on:

[2] In [9], only productions of the form $A \to B[j, |\mathrm{eval}_G(B)|]C[1, i]$ are allowed. But this definition is easily seen to be equivalent to our formalism.

Lemma 2. *Assume that (Γ, R) is 2-homogeneous, confluent, and N-free. Then the following problem belongs to P:*

INPUT: Composition systems G_1 and G_2 with $\mathrm{eval}(G_1), \mathrm{eval}(G_2) \in \mathrm{IRR}(R)$.
QUESTION: Does $\mathrm{eval}(G_1)\mathrm{eval}(G_2) \xrightarrow{*}_R \varepsilon$ *hold?*

Proof of Thm. 1. Let (Γ, R) be 2-homogeneous, confluent, and N-free. Given SLPs G_1 and G_2 over the terminal alphabet Γ, we have to verify in polynomial time, whether $\mathrm{NF}_R(\mathrm{eval}(G_1)) = \mathrm{NF}_R(\mathrm{eval}(G_2))$. Using the result of [9] mentioned before, it suffices to prove that given a SLP G in Chomsky normal form over the terminal alphabet Γ, we can construct in polynomial time a composition system H such that $\mathrm{eval}(H) = \mathrm{NF}_R(\mathrm{eval}(G))$. We construct H inductively by adding more and more rules. Initially, $P(H)$ contains all rules from $P(G)$ of the form $A \to a$ with $a \in \Gamma$. Now assume that $A \to BC$ belongs to $P(G)$ and that H already contains enough rules such that $\mathrm{eval}_H(B) = \mathrm{NF}_R(\mathrm{eval}_G(B))$ and $\mathrm{eval}_H(C) = \mathrm{NF}_R(\mathrm{eval}_G(C))$. If i is the largest number such that

$$\mathrm{eval}_H(B) = u_1u_2, \quad \mathrm{eval}_H(C) = v_1v_2, \quad |u_2| = |v_1| = i, \quad u_2v_1 \xrightarrow{*}_R \varepsilon, \qquad (1)$$

then clearly $\mathrm{NF}_R(\mathrm{eval}_G(A)) = u_1v_2$. For a given $i \in \mathbb{N}$, we can check (1) in polynomial time by Lemma 2. Since i is bounded exponentially in the input size, the largest i satisfying (1) can be easily calculated in polynomial time by doing a binary search. For this largest i we add to the current H the production $A \to B[1, |\mathrm{eval}_H(B)| - i]C[i+1, |\mathrm{eval}_H(C)|\,]$. □

For Γ an alphabet, the monoid $F(\Gamma) = \mathcal{M}(\Gamma \cup \overline{\Gamma}, \{(c\overline{c}, \varepsilon) \mid c \in \Gamma \cup \overline{\Gamma}\})$ is a group, namely the *free group* generated by Γ. In case $|\Gamma| = n$ we also write F_n for $F(\Gamma)$. It is known that the (uncompressed) word problem for a free group is in logspace [14]. Moreover, the word problem for F_2 is hard for uniform NC^1 [20]. By Thm. 1, the compressed word problem for every free group is in P. By a reduction from the monotone circuit value problem we can prove:

Theorem 2. *The compressed word problem for F_2 is P-complete.*

4 Between P and PSPACE

P^{NP} is the class of all languages that can be accepted by a deterministic polynomial time machine that has additional access to an NP-oracle; it is contained in PSPACE. Several complete problems for P^{NP} can be found in [11].

Theorem 3. *If (Γ, R) is 2-homogeneous and confluent (but not necessarily N-free), then the compressed word problem for $\mathcal{M}(\Gamma, R)$ is in P^{NP}.*

Proof. The key observation is that for a 2-homogeneous and confluent (but not necessarily N-free) presentation (Γ, R), the problem from Lemma 2 is in coNP: If $u_i = \mathrm{eval}(G_i)$ $(i = 1, 2)$ with $u_1, u_2 \in \mathrm{IRR}(R)$, then $u_1u_2 \xrightarrow{*}_R \varepsilon$ if and only if $|u_1| = |u_2| = n$ and $u_1[i]u_2[n - i + 1] \in \mathrm{dom}(R)$ for every $1 \le i \le n$. For a single i, the latter condition can be easily checked in polynomial time. Now the decision procedure from the proof of Thm. 1 in the previous section gives us a $\mathrm{P}^{\mathrm{coNP}}$-, i.e., P^{NP}-algorithm in the present situation. □

By a reduction from the complementary problem of SUBSETSUM, we can show:

Theorem 4. *Let $\Gamma = \{a, b, c, d\}$ and $R = \{(ac, \varepsilon), (ad, \varepsilon), (bc, \varepsilon)\}$ The compressed word problem for $\mathcal{M}(\Gamma, R)$ is coNP-hard.*

The precise complexity of the compressed word problem for 2-homogeneous, confluent, but non-N-free presentations remains open; it is located somewhere between coNP and P^{NP}.

5 Polynomial space and above

Our PSPACE upper bounds rely all on the following simple fact:

Proposition 2. *If the membership problem for the language L (the word problem for a finitely presented monoid $\mathcal{M}$) belongs to $\bigcup_{c>0} NSPACE(\log^c(n))$, then the compressed membership problem for L (the compressed word problem for $\mathcal{M}$) belongs to PSPACE.*

A presentation (Γ, R) is *weight-reducing* if there is a weight-function f on Γ^* with $f(s) > f(t)$ for all $(s, t) \in R$. A special case of weight-reducing presentations are *length-reducing presentations*, where $|s| > |t|$ for all $(s, t) \in R$. In [15] the author has shown that for every fixed weight-reducing and confluent presentation the (uncompressed) word problem is in LOGCFL [24]. Since LOGCFL $\subseteq$ NSPACE($\log^2(n)$) [13], Prop. 2 implies:

Proposition 3. *For every weight-reducing and confluent presentation (Γ, R), the compressed word problem for $\mathcal{M}(\Gamma, R)$ is in PSPACE.*

In the rest of this section, we show that PSPACE-hardness can be deduced already for a quite small subclass of weight-reducing and confluent presentations.

A presentation (Γ, R) is called *monadic* if for every $(\ell, r) \in R$: $|\ell| > |r|$ and $|r| \leq 1$. A *2-monadic* presentation is a monadic presentation (Γ, R) such that moreover $|\ell| = 2$ for every $\ell \in \mathrm{dom}(R)$. In the following, we present a construction that reduces the reachability problem for directed forests to the (uncompressed) word problem of a fixed 2-monadic and confluent presentation (Γ, R). Let $\Gamma = \{b_0, b_1, c_0, c_1, c_2, \#, \$, \triangleright, 0\}$ and let R be the 2-monadic semi-Thue system consisting of the following rules:

(1) $b_0 x \to \varepsilon$ for all $x \in \{\$, c_0, c_1, c_2\}$	(2) $b_1 c_0 \to \varepsilon$
(3) $b_1 \$ \to \triangleright$	(4) $\triangleright c_i \to \triangleright$ for all $i \in \{0, 1, 2\}$
(5) $\triangleright \$ \to \$$	(6) $\# \$ \to \varepsilon$
(7) $b_1 c_2 \to 0$	
(8) $0x \to 0$ for all $x \in \Gamma$	(9) $x0 \to 0$ for all $x \in \Gamma$

Only the rules involving the absorbing symbol 0 produce overlappings. In the resulting critical pairs, both words can be reduced to 0. Thus, R is confluent.

Assume now that (V, E) is a directed forest, where $V = \{v_1, \ldots, v_n\}$ and $i < j$ whenever $(v_i, v_j) \in E$. Let $v_\alpha \in V$ and $U \subseteq V$ be a set of nodes such that every

node in U has outdegree 0. For $i \leq j$ we define the interval $I_{i,j} = \{v_k \mid i \leq k \leq j\}$. Thus, $I_{1,n} = V$. If $i > j$ we set $I_{i,j} = \emptyset$. For every $i \in \{1, \ldots, n\}$ let:

$$\delta_i = \begin{cases} c_0^{n-j+i+1} & \text{if } (v_i, v_j) \text{ is the unique outgoing edge at node } v_i \\ c_1 & \text{if } v_i \in V \setminus U \text{ and } v_i \text{ has no outgoing edge} \\ c_2 & \text{if } v_i \in U \text{ (and thus has no outgoing edge)} \end{cases}$$

For an interval $I_{i,j}$ $(i \leq j)$ let $\sigma[I_{i,j}] = \delta_i \$ \delta_{i+1} \$ \cdots \delta_j \$$. We set $\sigma[\emptyset] = \varepsilon$. Using the rules in (4) and (5) we get $\triangleright\sigma[I_{i,j}] \xrightarrow{*}_R \$\sigma[I_{i+1,j}]$ if $i \leq j$. Finally, define

$$\beta = |\sigma[I_{1,\alpha-1}]| \quad \text{and} \quad w(v_\alpha, U) = (\#b_1^n)^n b_0^\beta \sigma[I_{1,n}].$$

Lemma 3. *We have $w(v_\alpha, U) \overset{*}{\leftrightarrow}_R 0$ if and only if $\exists v_i \in U : (v_\alpha, v_i) \in E^*$.*

The previous lemma yields the following result that is of independent interest. It sharpens a corresponding result of [4] for monadic systems.

Theorem 5. *There exists a fixed 2-monadic and confluent presentation (Γ, R) such that the word problem for $\mathcal{M}(\Gamma, R)$ is L-hard under NC^1-reductions.*

Theorem 6. *There exists a fixed 2-monadic and confluent presentation with a PSPACE-complete compressed word problem.*

Proof. We show that the compressed word problem for the 2-monadic presentation (Γ, R) from the previous discussion is PSPACE-complete. The upper bound follows from Prop. 3. For the lower bound we have to repeat a construction from [15]. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, q_f)$ be a deterministic linear bounded automaton (where Q is the set of states, Σ is the tape alphabet, q_0 (resp. q_f) is the initial (resp. final) state, and $\delta : Q \setminus \{q_f\} \times \Sigma \to Q \times \Sigma \times \{-1, +1\}$ is the transition function) that accepts a PSPACE-complete language. Such an automaton exists, see, e.g., [3]. Let $w \in \Sigma^*$ be an input for $\mathcal{A}$ with $|w| = N$. We may assume that $\mathcal{A}$ operates in phases, where a single phase consists of a sequence of $2 \cdot N$ transitions of the form $q_1\gamma_1 \overset{*}{\Rightarrow}_{\mathcal{A}} \gamma_2 q_2 \overset{*}{\Rightarrow}_{\mathcal{A}} q_3\gamma_3$, where $\gamma_1, \gamma_2, \gamma_3 \in \Sigma^N$ and $q_1, q_2, q_3 \in Q$. During the sequence $q_1\gamma_1 \overset{*}{\Rightarrow}_{\mathcal{A}} \gamma_2 q_2$ (resp. $\gamma_2 q_2 \overset{*}{\Rightarrow}_{\mathcal{A}} q_3\gamma_3$) only right-moves (resp. left-moves) are made. The automaton $\mathcal{A}$ accepts, if it reaches the final state q_f. Let $c > 0$ be a constant such that if w is accepted by $\mathcal{A}$, then $\mathcal{A}$, started on w, reaches the final state q_f after at most $2^{c \cdot N}$ phases. Let $\widehat{\Sigma}$ be a disjoint copy of Σ and similarly for $\widehat{Q}$. Let $\Delta = \Sigma \cup \widehat{\Sigma} \cup \{\triangleleft, 0, 1, £\}$ and $\Theta = Q \cup \widehat{Q} \cup \Delta$ and let S be the semi-Thue system over Θ that consists of the following rules, where x ranges over all symbols from Δ :

$0\widehat{q}x \to \widehat{q}£x$ for all $q \in Q \setminus \{q_f\}$	$xqa \to x\widehat{b}p$ if $\delta(q, a) = (p, b, +1)$
$1\widehat{q}x \to 0qx$ for all $q \in Q \setminus \{q_f\}$	$\widehat{a}\,\widehat{q}x \to \widehat{p}bx$ if $\delta(q, a) = (p, b, -1)$
$xq£ \to x1q$ for all $q \in Q \setminus \{q_f\}$	$xq\triangleleft \to x\widehat{q}\triangleleft$ for all $q \in Q \setminus \{q_f\}$

Note that $\mathrm{dom}(R) \subseteq \Delta(Q \cup \widehat{Q})\Delta$. Moreover, (Θ, S) is length-preserving and for any linear order on Θ satisfying $Q \succ 1 \succ 0 \succ \widehat{\Sigma} \succ \widehat{Q}$ we have (for the

lexicographic extension of $\succ$) $s \succ t$ whenever $s \to_S t$. Let us choose such a linear order that moreover satisfies $Q \succ \Delta \succ \widehat{Q}$. In [15] the author argued that w is accepted by $\mathcal{A}$ if and only if $1q_0\pounds^{c\cdot N}w\triangleleft \xrightarrow{*}_S v$ for some word v with $\mathrm{alph}(v) \cap \{q_f, \widehat{q}_f\} \neq \emptyset$ (we have slightly modified the construction form [15] but the principal idea is the same). For $m = (c+1)N$ let $V = \bigcup_{i=0}^{m} \Delta^{i+1}(Q \cup \widehat{Q})\Delta^{m-i+1}$. Note that any S-derivation starting from $1q_0\pounds^{c\cdot N}w\triangleleft$ is completely contained in V. On the set V we construct a directed forest (V, E) by taking $E = (V \times V) \cap \to_S$. If we order V lexicographically by $\succ$ and write $V = \{v_1, \ldots, v_n\}$ with $v_1 \succ v_2 \succ \cdots \succ v_n$, then $(v_i, v_j) \in E$ implies $i < j$, i.e., (V, E) is an ordered directed forest. Note that $n = 2(m+1) \cdot |Q| \cdot |\Delta|^{m+2}$, which belongs to $2^{O(N)}$. Let $U = \{v \in V \mid \mathrm{alph}(v) \cap \{q_f, \widehat{q}_f\} \neq \emptyset\}$ and $v_\alpha = 1q_0\pounds^{c\cdot N}w\triangleleft$. Thus, $\alpha - 1$ is the number of words from V that are lexicographically larger than $1q_0\pounds^{c\cdot N}w\triangleleft$. The number α can be easily calculated in polynomial time from the input w.

The automaton $\mathcal{A}$ accepts w if and only if there is a path in (V, E) from v_α to a node in U. By Lemma 3 this is equivalent to $w(v_\alpha, U) \overset{*}{\leftrightarrow}_R 0$. Thus, it remains to show that $w(v_\alpha, U) \in \Gamma^*$ can be generated by a small SLP. Recall the definition of the words δ_i and $\sigma[I] \in \Gamma^*$, where $1 \leq i \leq n$ and I is an interval of $(V, \succ)$, from the discussion preceding Lemma 3. Note that if $v_i = u_1\ell u_2 \to_S u_1 r u_2 = v_j$ with $(\ell, r) \in S$, then the number $j - i$ (i.e., the number of words from V that are lexicographically between v_i and v_j) only depends on the rule (ℓ, r) (and thus ℓ) and $|u_2|$. We call this number $\lambda(\ell, |u_2|)$; it is of size $2^{O(N)}$. We now describe a small SLP that generates the word $\sigma[V] \in \Gamma^*$. Assume that $Q = \{p_1, \ldots, p_{n_1}\}$ and $\Delta = \{a_1, \ldots, a_{n_2}\}$ with $p_i \succ p_{i+1}$, $\widehat{p}_i \succ \widehat{p}_{i+1}$, and $a_i \succ a_{i+1}$. We introduce the following productions ($\prod_{i=1}^{k} u_i$ abbreviates $u_1 \cdots u_k$):

$$A_i \to \prod_{j=1}^{n_2} B_{i,j} A_{i+1} \widehat{B}_{i,j} \text{ for } 0 \leq i < m, \qquad A_m \to \prod_{j=1}^{n_2} B_{m,j}\widehat{B}_{m,j}$$

$$B_{i,j} \to \prod_{k=1}^{n_1}\prod_{\ell=1}^{n_2} (C_{i,j,k,\ell}\$)^{|\Delta|^{m-i}} \text{ for } 0 \leq i \leq m,\ 1 \leq j \leq n_2$$

$$C_{i,j,k,\ell} \to \begin{cases} c_0^{n-\lambda(a_j p_k a_\ell, m-i)+1} & \text{if } a_j p_k a_\ell \in \mathrm{dom}(R) \\ c_1 & \text{if } a_j p_k a_\ell \notin \mathrm{dom}(R) \text{ and } p_k \neq q_f \\ c_2 & \text{if } p_k = q_f \end{cases}$$

$$\widehat{B}_{i,j} \to \prod_{k=1}^{n_1}\prod_{\ell=1}^{n_2} (\widehat{C}_{i,j,k,\ell}\$)^{|\Delta|^{m-i}} \text{ for } 0 \leq i \leq m,\ 1 \leq j \leq n_2$$

$$\widehat{C}_{i,j,k,\ell} \to \begin{cases} c_0^{n-\lambda(a_j \widehat{p}_k a_\ell, m-i)+1} & \text{if } a_j \widehat{p}_k a_\ell \in \mathrm{dom}(R) \\ c_1 & \text{if } a_j \widehat{p}_k a_\ell \notin \mathrm{dom}(R) \text{ and } \widehat{p}_k \neq \widehat{q}_f \\ c_2 & \text{if } \widehat{p}_k = \widehat{q}_f \end{cases}$$

The integer exponents that appear in the right-hand sides of these productions are all of size $2^{O(N)}$ and can therefore be easily replaced by ordinary productions. Note that $\mathrm{eval}(C_{i,j,k,\ell}) = \delta_s$ for every $v_s \in \Delta^i a_j p_k a_\ell \Delta^{m-i}$ and $\mathrm{eval}(\widehat{C}_{i,j,k,\ell}) = \delta_s$ for every $v_s \in \Delta^i a_j \widehat{p}_k a_\ell \Delta^{m-i}$. It follows that for all $0 \leq i \leq m$, all $u \in \Delta^i$, and

all $1 \leq j \leq n_2$ we have (note that $ua_jQ\Delta^{m-i+1} \subseteq V$ is an interval of $(V, \succ)$)

$$\mathrm{eval}(B_{i,j}) = \sigma[ua_jQ\Delta^{m-i+1}] \quad \text{and} \quad \mathrm{eval}(\widehat{B}_{i,j}) = \sigma[ua_j\widehat{Q}\Delta^{m-i+1}].$$

By induction on $i \in \{0, \ldots, m\}$ (for $i = m$ down to 0), we can show that $\sigma[I] = \mathrm{eval}(A_i)$, where I is the interval $\bigcup_{j=1}^{m-i+1} u\Delta^j(Q \cup \widehat{Q})\Delta^{m-i-j+2}$ of the linear order $(V, \succ)$ and $u \in \Delta^i$ is arbitrary. For $i = 0$ we get $\mathrm{eval}(A_0) = \sigma[V]$. The number $\beta = |\sigma[I_{1,\alpha-1}]| \in 2^{O(N)}$ can be calculated from the input word w using simple arithmetic. Now it is easy to construct a SLP G of size polynomial in the input size N with $\mathrm{eval}(G) = (\#b_1^n)^n b_0^\beta \sigma[V] = w(v_\alpha, U)$. This concludes the proof. □

Since (Γ, R) is monadic and confluent, the language $\{w \in \Gamma^* \mid w \xrightarrow{*}_R 0\}$ is deterministic context-free [8, Thm. 4.2.7]. Thus, we obtain a fixed deterministic context-free language with a PSPACE-complete compressed membership problem. This solves an open problem from [9, 19]. We can even show a slightly stronger result: In [10] a language is called *deterministic linear* if it is accepted by a deterministic 1-turn pushdown automaton. It is easy to see that the language $\{w \in \Gamma^* \mid w \xrightarrow{*}_R 0\} \cap (\#b_1^+)^+ b_0^+ ((c_0^+ \cup c_1 \cup c_2)\$)^+$ is deterministic linear. Moreover, it contains all words of the form $w(v_\alpha, U)$. Thus, we obtain:

Corollary 1. *There exists a fixed deterministic linear language L such that the compressed membership problem for L is PSPACE-complete.*

Also a uniform variant of the compressed membership problem for context-free languages is PSPACE-complete:

Theorem 7. *The following problem is PSPACE-complete:*
INPUT: A context-free grammar G and a SLP H
QUESTION: $\mathrm{eval}(H) \in L(G)$*?*

Finally, we take a look at EXPSPACE-complete cases: A presentation (Γ, R) is *weight-lexicographic* if there are a linear order $\succ$ on Γ and a weight-function f on Γ^* with $f(\ell) > f(r)$ or $(f(\ell) = f(r) \wedge \ell \succ r)$ for all $(\ell, r) \in R$. If $|\ell| > |r|$ or $(|\ell| = |r| \wedge \ell \succ r)$ for all $(\ell, r) \in R$, then (Γ, R) is *length-lexicographic*. A slight variation of a construction from [15] yields the following two results:

Theorem 8. *For every weight-lexicographic and confluent presentation, the compressed word problem is in EXPSPACE. There is a fixed length-lexicographic and confluent presentation with an EXPSPACE-complete compressed word problem.*

Theorem 9. *There exists a fixed context-sensitive language L such that the compressed membership problem for L is EXPSPACE-complete.*

6 Circuit complexity and compression

In this section we study compressed membership problems for languages from very low complexity classes, which are usually defined by uniform families of

small depth Boolean circuits. An equivalent and for our purpose more suitable definition is based on *alternating Turing-machines* with logarithmic time bounds. See [17] for background on alternating Turing-machines. When dealing with logarithmic time bounds it is necessary to enrich the machine model with a random access mechanism in form of a special address tape that contains a binary coded number p. If the machine enters a special query state, then it has random access to the p-th input position. ALOGTIME is the class of all languages that can be recognized on an alternating Turing-machine in time $O(\log(n))$, it is equal to uniform NC^1. Within ALOGTIME, we can define the logtime hierarchy: For $k \geq 1$, $\Sigma_k^{\log}$ (resp. $\Pi_k^{\log}$) is the class of all languages that can be decided by an alternating Turing-machine in time $O(\log(n))$ within $k-1$ alternations, starting in an existential (resp. universal) state. In [2], $\Sigma_k^{\log} \cup \Pi_k^{\log}$ is proposed as a uniform version of the circuit complexity class AC_k^0. The union $\bigcup_{k\geq 1} \Sigma_k^{\log} \cup \Pi_k^{\log}$ is called the *logtime hierarchy* LH [22]. It turns out that in a compressed setting, the levels of LH and the polynomial time hierarchy $\mathrm{PH} = \bigcup_{k\geq 1} \Sigma_k^{\mathrm{poly}} \cup \Pi_k^{\mathrm{poly}}$ (see [17] for details on PH) are in a tight correspondence:

Theorem 10. *For every language in $\Sigma_k^{\log}$ ($\Pi_k^{\log}$) the compressed membership problem belongs to Σ_k^{poly} (Π_k^{poly}). There is a fixed language in $\Sigma_k^{\log}$ ($\Pi_k^{\log}$) with a Σ_k^{poly}-complete (Π_k^{poly}-complete) compressed membership problem.*

Every language in $\bigcup_{c>0} \mathrm{NSPACE}(\log^c(n))$ has a compressed membership problem within PSPACE (Prop. 2). Languages with a PSPACE-complete compressed membership problem can be already found in $\mathrm{ALOGTIME} \subseteq \mathrm{DSPACE}(\log(n))$:

Theorem 11. *There exists a fixed language L in ALOGTIME such that the compressed membership problem for L is PSPACE-complete.*

It is *not* the case that for every ALOGTIME-complete language the compressed membership problem is PSPACE-complete (unless P = PSPACE): The word problem for the finite group S_5 is ALOGTIME-complete [1] but its compressed word problem is in P. Thus, a general upgrading theorem analogously to [25] does not hold for straight-line programs. This is similar to the situation for hierarchical graphs [12], where the correlation between the complexity of a problem in its compressed and uncompressed variant, respectively, is quite loose.

References

1. D. A. M. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38:150–164, 1989.
2. D. A. M. Barrington, C.-J. Lu, P. B. Miltersen, and S. Skyum. Searching constant width mazes captures the AC^0 hierarchy. In *Proc. STACS 98*, LNCS 1373, pages 73–83. Springer, 1998.
3. G. Bauer and F. Otto. Finite complete rewriting systems and the complexity of the word problem. *Acta Inf.*, 21:521–540, 1984.
4. M. Beaudry, M. Holzer, G. Niemann, and F. Otto. McNaughton families of languages. *Theor. Comput. Sci.*, 290(3):1581–1628, 2003.

5. M. Beaudry, P. McKenzie, P. Péladeau, and D. Thérien. Finite monoids: From word to circuit evaluation. *SIAM J. Comput.*, 26(1):138–152, 1997.
6. R. V. Book. Homogeneous Thue systems and the Church–Rosser property. *Discrete Math.*, 48:137–145, 1984.
7. R. V. Book, M. Jantzen, B. Monien, C. P. Ó'Dúnlaing, and C. Wrathall. On the complexity of word problems in certain Thue systems. In *Proc. MFCS'81*, LNCS 118, pages 216–223. Springer, 1981.
8. R. V. Book and F. Otto. *String–Rewriting Systems.* Springer, 1993.
9. L. Gasieniec, M. Karpinski, W. Plandowski, and W. Rytter. Efficient algorithms for Lempel-Ziv encoding (extended abstract). In *Proc. SWAT 1996*, LNCS 1097, pages 392–403. Springer, 1996.
10. M. Holzer and K.-J. Lange. On the complexities of linear LL(1) and LR(1) grammars. In *Proc. FCT'93*, LNCS 710, pages 299–308. Springer, 1993.
11. M. W. Krentel. The complexity of optimization problems. *J. Comput. Syst. Sci.*, 36(3):490–509, 1988.
12. T. Lengauer and E. Wanke. The correlation between the complexities of the nonhierarchical and hierarchical versions of graph problems. *J. Comput. Syst. Sci.*, 44:63–93, 1992.
13. P. M. Lewis II, R. E. Stearns, and J. Hartmanis. Memory bounds for recognition of context-free and context-sensitive languages. In *Proc. Sixth Annual IEEE Symp. on Switching Circuit Theory and Logic Design*, pages 191–202, 1965.
14. R. J. Lipton and Y. Zalcstein. Word problems solvable in logspace. *J. Assoc. Comput. Mach.*, 24(3):522–526, 1977.
15. M. Lohrey. Word problems and confluence problems for restricted semi-Thue systems. In *Proc. RTA 2000*, LNCS 1833, pages 172–186. Springer, 2000.
16. M. Lohrey. Word problems for 2-homogeneous monoids and symmetric logspace. In *Proc. MFCS 2001*, LNCS 2136, pages 500–511. Springer, 2001.
17. C. H. Papadimitriou. *Computational Complexity.* Addison Wesley, 1994.
18. W. Plandowski. Testing equivalence of morphisms on context-free languages. In *Proc. ESA'94*, LNCS 855, pages 460–470. Springer, 1994.
19. W. Plandowski and W. Rytter. Complexity of language recognition problems for compressed words. In *Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa*, pages 262–272. Springer, 1999.
20. D. Robinson. *Parallel Algorithms for Group Word Problems.* PhD thesis, University of California, San Diego, 1993.
21. W. Rytter. Compressed and fully compressed pattern matching in one and two dimensions. *Proc. IEEE*, 88(11):1769–1778, 2000.
22. M. Sipser. Borel sets and circuit complexity. In *Proc. STOC 1983*, pages 61–69. ACM Press, 1983.
23. J. Stillwell. The word problem and the isomorphism problem for groups. *Bull. Am. Math. Soc., New Ser.*, 6(1):33–56, 1982.
24. I. H. Sudborough. On the tape complexity of deterministic context–free languages. *J. Assoc. Comput. Mach.*, 25(3):405–414, 1978.
25. H. Veith. Succinct representation, leaf languages, and projection reductions. *Inf. Control*, 142(2):207–236, 1998.
26. K. W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Inf.*, 23(3):325–356, 1986.
27. Y. Zhang and R. Gupta. Path matching in compressed control flow traces. In *Proc. DCC 2002*, pages 132–141. IEEE Computer Society Press, 2002.
28. J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. on Inf. Theory*, 23(3):337–343, 1977.