Compressed Membership in Automata with Compressed Labels

Markus Lohrey and Christian Mathissen

Institut für Informatik, Universität Leipzig, Germany
{lohrey,mathissen}@informatik.uni-leipzig.de

Abstract. The algorithmic problem of whether a compressed string is accepted by a (nondeterministic) finite state automaton with compressed transition labels is investigated. For string compression, straight-line programs (SLPs), i.e., context-free grammars that generate exactly one string, are used. Two algorithms for this problem are presented. The first one works in polynomial time, if all transition labels are nonperiodic strings (or more generally, the word length divided by the period is bounded polynomially in the input size). This answers a question of Plandowski and Rytter. The second (nondeterministic) algorithm is an NP-algorithm under the assumption that for each transition label the period is bounded polynomially in the input size. This generalizes the NP upper bound for the case of a unary alphabet, shown by Plandowski and Rytter.

1 Introduction

The topic of this paper is algorithms on compressed strings. The goal of such algorithms is to check properties of compressed strings and thereby beat a straightforward "decompress-and-check" strategy. Potential applications for such algorithms can be found for instance in genome databases, where massive volumes of string data are stored and analyzed. When talking about algorithms on compressed strings, one has to make precise the used compression scheme. Here, as in previous papers, we choose *straight-line programs* (SLPs); these are context-free grammars that generate exactly one string. Straight-line programs turned out to be a very flexible and mathematically clean compressed representation of strings. Several other dictionary-based compressed representations, like for instance Lempel-Ziv (LZ) factorizations [15], can be converted in polynomial time into SLPs and vice versa [13]. This implies that complexity results can be transferred from SLP-encoded input strings to LZ-encoded input strings.

Several algorithmic problems for SLP-compressed input strings were considered in the past, e.g. equivalence and pattern matching [5, 12], word problems for certain groups and monoids [8, 9, 11, 14], and membership problems for various language classes [3, 7, 9, 13]. In this paper, we study the membership problem for compressed words in automata with compressed labels. In this problem, the input consists of an SLP $\mathbb{A}$ and a nondeterministic automaton, where each transition is labeled with an SLP. Such an automaton generates a language in the obvious way, and it is asked whether the string generated by the SLP $\mathbb{A}$ belongs to that language. This problem was first studied in [13]; it is easily seen to be in PSPACE. Moreover, it was shown to be NP-complete for

a unary alphabet in [13]. In fact, NP-hardness in the unary case follows directly from NP-hardness of the SUBSETSUM problem. To the knowledge of the authors no better lower bound than NP-hardness is known for the non-unary case. This paper contains two algorithms for the membership problem for compressed words in automata with compressed labels (let $\mathcal{A}$ be the input automaton with compressed labels).

The first algorithm is deterministic and works in polynomial time if every SLP appearing in $\mathcal{A}$ generates a string with a small order (polynomial in the total input size). Here the order of a string is its length divided by its smallest period. Hence, having a small order means that the string looks quite aperiodic. In fact, as a corollary we obtain a polynomial time algorithm for the case that every SLP in $\mathcal{A}$ generates a nonperiodic word. This solves open problem 3 from [13]. The second algorithm is nondeterministic and works in polynomial time if every SLP in $\mathcal{A}$ generates a string with a small period (polynomial in the total input size). This generalizes the NP bound for the unary case from [13] (a unary word has period 1).

Hence, these two algorithms cover two extreme cases (almost nonperiodic versus highly periodic). But they do not cover the general case. An SLP $\mathbb{A}$ may generate a string, for which both the order and the period are exponential in the size of $\mathbb{A}$. Nevertheless, following [13], we conjecture that the general membership problem for compressed words in automata with compressed labels belongs to NP. We conclude this paper with another conjecture that implies the former one.

2 Preliminaries

We use $\mathrm{poly}(n_1, \ldots, n_k)$ as an abbreviation for $(n_1 + \cdots + n_k)^{O(1)}$. For $n \leq m$, we denote with $[n, m]$ the interval $\{n, n+1, \ldots, m\}$. An *arithmetic progression* is a set of natural numbers of the form $\{b + i \cdot p \mid i \in [0, \ell]\}$. This set can be represented succinctly by the triple (b, p, ℓ), where all three numbers are binary coded.

Let us fix a finite alphabet Σ. For a string $w \in \Sigma^*$ and $1 \leq i \leq |w|$ let $w[i]$ denote the i-th symbol in w. Moreover, for $1 \leq i, j \leq |w|$ let $w[i : j] = w[i] \cdots w[j]$ if $i \leq j$ and let $w[i : j] = \varepsilon$ if $i > j$. A number $1 \leq p \leq |w| - 1$ is a *period* of w if $w[i] = w[i+p]$ for all $1 \leq i \leq |w| - p$. With $\mathrm{per}(w)$ we denote the smallest period of w, where we set $\mathrm{per}(w) = |w|$ if w has no period. Let $\mathrm{ord}(w) = \lfloor \frac{|w|}{\mathrm{per}(w)} \rfloor$ be the *order* of w. Then $w = u^{\mathrm{ord}(w)} v$, where u is *primitive* (i.e., it is not of the form x^n for a string x and $n \geq 2$) and v is a proper prefix of u (possibly empty). A string w is called *nonperiodic*, if $\mathrm{ord}(w) = 1$. Two words $u, v \in \Sigma^*$ are *conjugated*, if there exist $x, y \in \Sigma^*$ with $u = xy$ and $v = yx$. An occurrence of a word p in another word t is a number $i \in [0, |t| - |p|]$ such that $w[i+1 : i+|p|] = p$. We say that this occurrence i *covers* all positions from the interval $[i+1, i+|p|]$, and that it *touches* all positions from the interval $[i, i+|p|]$.

Lemma 1 (cf. [5, Lemma 1]). *Let $t, p \in \Sigma^*$ and $j \in [0, |t|]$. The set of all occurrences of p in t that touch position j is an arithmetic progression of size at most $\mathrm{ord}(p)$.*

In Section 4.2 we need some basic concepts concerning *string rewriting systems*, see [1] for more details. A string rewriting system R over Σ is a finite subset of $\Sigma^* \times \Sigma^*$. A pair

$(\ell, r) \in R$ is called a rule of R and is often written as $\ell \to r$. R defines a rewrite relation $\to_R$ as follows: $u \to_R v$ for $u, v \in \Sigma^*$ if there exist $x, y \in \Sigma^*$ and a rule $(\ell, r) \in R$ such that $u = x\ell y$ and $v = xry$. The system R is *terminating* if there does not exist an infinite chain $u_0 \to_R u_1 \to_R u_2 \to_R \cdots$. Moreover, R is called *confluent* (resp. *locally confluent*) if for all $u, v, w \in \Sigma^*$ such that $u \to_R^* v$ and $u \to_R^* w$ (resp. $u \to_R v$ and $u \to_R w$) there exists $x \in \Sigma^*$ with $v \to_R^* x$ and $w \to_R^* x$. By Newman's lemma, a terminating system is confluent if and only if it is locally confluent. Moreover, for terminating systems, local confluence is decidable. For this one has to consider *critical pairs* that result from overlapping left hand sides, see [1] for more details. Let us set $\mathsf{IRR}(R) = \Sigma^* \setminus \{u \mid \exists v : u \to_R v\}$. If R is terminating and confluent, then for every $u \in \Sigma^*$ there exists a unique $v \in \mathsf{IRR}(R)$ such that $u \to_R^* v$; it is called the *irreducible normal form* of u and is denoted by $\mathsf{NF}_R(u)$.

A *straight-line program (SLP)* over the terminal alphabet Σ is a context-free grammar $\mathbb{A} = (N, \Sigma, S, P)$ (N is the set of nonterminals, Σ is the set of terminals, $S \in N$ is the initial nonterminal, and $P \subseteq N \times (N \cup \Sigma)^*$ is the set of productions) such that: (i) for every $A \in N$ there exists exactly one production of the form $(A, \alpha) \in P$ for $\alpha \in (N \cup \Sigma)^*$, and (ii) the relation $\{(A, B) \in N \times N \mid (A, \alpha) \in P, B \text{ occurs in } \alpha\}$ is acyclic. The transitive closure of this relation is also called the *hierarchical order* of $\mathbb{A}$; it is a partial order. A production (A, α) is also written as $A \to \alpha$. Clearly, the language generated by the SLP $\mathbb{A}$ consists of exactly one word that is denoted by $\mathsf{val}(\mathbb{A})$. More generally, from every nonterminal $A \in N$ we can generate exactly one word that is denoted by $\mathsf{val}_{\mathbb{A}}(A)$ (thus $\mathsf{val}(\mathbb{A}) = \mathsf{val}_{\mathbb{A}}(S)$). We omit the index $\mathbb{A}$ if the underlying SLP is clear from the context. The size of $\mathbb{A}$ is $|\mathbb{A}| = \sum_{(A, \alpha) \in P} |\alpha|$. Every SLP $\mathbb{A}$ with $\mathsf{val}(\mathbb{A}) \neq \varepsilon$ can be transformed in polynomial time into an equivalent SLP in *Chomsky normal form*, i.e., all productions have the form (A, a) with $a \in \Sigma$ or (A, BC) with $B, C \in N$. In the sequel we assume that all SLPs are in Chomsky normal form.

Let us state some algorithmic problems that can be easily solved in polynomial time:

– Given an SLP $\mathbb{A}$, calculate $|\mathsf{val}(\mathbb{A})|$.
– Given an SLP $\mathbb{A}$ and a number $i \in \{1, \dots, |\mathsf{val}(\mathbb{A})|\}$, calculate $\mathsf{val}(\mathbb{A})[i]$; this problem is in fact P-complete [6].

Let $\mathbb{A}$ be an SLP with a production (A, BC). An occurrence $i \in [0, |\mathsf{val}(A)| - |p|]$ of the word p in $\mathsf{val}(A)$ *touches the cut* of A, if this occurrence touches position $|\mathsf{val}(B)|$. The following result by Lifshits implies in particular, that for given SLPs $\mathbb{A}$ and $\mathbb{B}$ one can check in time $O(|\mathbb{A}| \cdot |\mathbb{B}|^2)$, whether $\mathsf{val}(\mathbb{A})$ occurs as a pattern in $\mathsf{val}(\mathbb{B})$.

Theorem 2 ([5]). *For two given SLPs $\mathbb{A}$ and $\mathbb{B}$ we can compute in time $O(|\mathbb{A}| \cdot |\mathbb{B}|^2)$ a table that contains for every nonterminal B of $\mathbb{B}$ an arithmetic progression (stored by three binary encoded numbers) for the set of all occurrences of $\mathsf{val}(\mathbb{A})$ in $\mathsf{val}_{\mathbb{B}}(B)$ that are touching the cut of B.*

An *automaton with compressed labels* is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states, and δ is a finite set of transitions of the form $(p, \mathbb{A}, q)$, where p and q are states and $\mathbb{A}$ is an SLP over Σ, for which we assume $\mathsf{val}(\mathbb{A}) \neq \varepsilon$ (ε-transitions can be eliminated). A transition $(p, \mathbb{A}, q)$ with $|\mathsf{val}(\mathbb{A})| = 1$ is called *atomic*. The size of $\mathcal{A}$ is $|\mathcal{A}| =$

$|Q| + \sum_{(p,\mathbb{A},q)\in\delta} |\mathbb{A}|$. We say that a word w labels a path from state p to state q in $\mathcal{A}$ if there exists a sequence of transitions $(p_0, \mathbb{A}_0, p_1), (p_1, \mathbb{A}_1, p_2), \ldots, (p_{n-1}, \mathbb{A}_{n-1}, p_n) \in \delta$ $(n \geq 0)$ such that $p_0 = p$, $p_n = q$, and $w = \text{val}(\mathbb{A}_0) \cdots \text{val}(\mathbb{A}_{n-1})$. We say the transition starting at position $\sum_{i=0}^{\ell-1} |\text{val}(\mathbb{A}_i)|$ and ending at position $\sum_{i=0}^{\ell} |\text{val}(\mathbb{A}_i)|$ is $(p_\ell, \mathbb{A}_\ell, p_{\ell+1})$. The language $L(\mathcal{A}) \subseteq \Sigma^*$ is the set of all words that label a path from the initial state q_0 to some final state $q_f \in F$. We set $\text{ord}(\mathcal{A}) = \max\{\text{ord}(\text{val}(\mathbb{A})) \mid (p, \mathbb{A}, q) \in \delta\}$ and $\text{per}(\mathcal{A}) = \max\{\text{per}(\text{val}(\mathbb{A})) \mid (p, \mathbb{A}, q) \in \delta\}$. Note that in general, both $\text{ord}(\mathcal{A})$ and $\text{per}(\mathcal{A})$ are exponential in $|\mathcal{A}|$.

3 A deterministic algorithm

The goal of this section is to prove the following theorem.

Theorem 3. *Given an automaton with compressed labels $\mathcal{A}$ and an SLP $\mathbb{B}$, we can check $\text{val}(\mathbb{B}) \in L(\mathcal{A})$ in time $\text{poly}(|\mathbb{B}|, |\mathcal{A}|, \text{ord}(\mathcal{A}))$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, $\mathbb{B} = (N, \Sigma, S, P)$, and let $\mathbb{A}_1, \ldots, \mathbb{A}_n$ be a list of all SLPs that occur as labels in $\mathcal{A}$. By Theorem 2, we can compute in time $O(\sum_{i=1}^{n}(|\mathbb{A}_i| \cdot |\mathbb{B}|^2)) \leq O(|\mathcal{A}| \cdot |\mathbb{B}|^2)$ a table that contains for every $i \in [1, n]$ and every nonterminal $B \in N$ an arithmetic progression $\text{AP}(i, B)$ for the set of all occurrences of $\mathbb{A}_i$ in $\text{val}_\mathbb{B}(B)$ that are touching the cut of B. Moreover, by Lemma 1, this arithmetic progression contains at most $\text{ord}(\text{val}(\mathbb{A}_i))$ many numbers. In total, we have at most $|\mathbb{B}| \cdot \sum_{i=1}^{n} \text{ord}(\text{val}(\mathbb{A}_i)) \leq |\mathbb{B}| \cdot |\mathcal{A}| \cdot \text{ord}(\mathcal{A})$ many numbers.

We now define a context-free grammar G with empty terminal alphabet. The two major facts about G are:

- G can be computed in time $\text{poly}(|\mathbb{B}|, |\mathcal{A}|, \text{ord}(\mathcal{A}))$.
- $\varepsilon \in L(G)$ if and only if $\text{val}(\mathbb{B}) \in L(\mathcal{A})$.

Since the word problem for context-free grammars can be decided in polynomial time, these two facts imply the theorem. The grammar G is constructed by a fixpoint process, where we add more and more nonterminals. The set of nonterminals contains the start nonterminal S_G; all other nonterminals are 5-tuples of the form (p, k, B, ℓ, q), where $p, q \in Q$, $B \in N$, and $k, \ell \in [0, |\text{val}(B)|]$ with $k + \ell \leq |\text{val}(B)|$. The intuition is that this 5-tuple should be viewed as the following assertion: Let w be the word that results from $\text{val}(B)$ by cutting off the prefix of length k and the suffix of length ℓ, i.e., $w = \text{val}(B)[k + 1 : |\text{val}(B)| - \ell]$. Then in the automaton $\mathcal{A}$ there exists a path from state p to state q labeled with the word w.
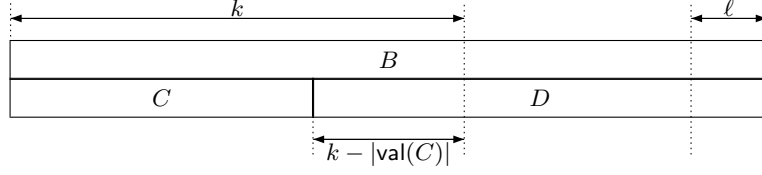
For G's start nonterminal S_G we introduce all productions of the form
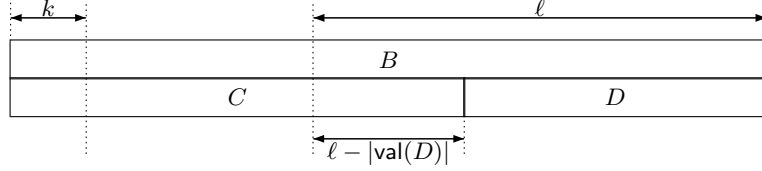
$$S_G \rightarrow (0, q_0, S, q_f, 0), \tag{1}$$

where $q_f \in F$ is a final state of $\mathcal{A}$. Now assume that at some point we have introduced a new nonterminal (k, p, B, q, ℓ). We distinguish 5 cases:

Case 1. $(B \rightarrow CD) \in P$ and $|\text{val}(C)| \leq k < |\text{val}(B)| - \ell$, see Figure 1. We introduce the production

$$(k, p, B, q, \ell) \rightarrow (k - |\text{val}(C)|, p, D, q, \ell).$$

Fig. 1. Case 1

Fig. 2. Case 2

Case 2. $(B \to CD) \in P$ and $|\mathsf{val}(D)| \leq \ell < |\mathsf{val}(B)| - k$, see Figure 2. We introduce the production

$$(k, p, B, q, \ell) \to (k, p, C, q, \ell - |\mathsf{val}(D)|).$$

Case 3. $(B \to CD) \in P$ and $k < |\mathsf{val}(C)|, \ell < |\mathsf{val}(D)|$, see Figure 3. We introduce a production for every transition $(r, \mathbb{A}_i, s)$ of $\mathcal{A}$ and every $j \in \mathsf{AP}(i, B)$ such that $j \geq k$ and $|\mathsf{val}(B)| - |\mathsf{val}(\mathbb{A}_i)| - j \geq \ell$. For such a choice, we introduce the production

$$(k, p, B, q, \ell) \to (k, p, C, r, |\mathsf{val}(C)| - j)\,(|\mathsf{val}(\mathbb{A}_i)| + j - |\mathsf{val}(C)|, s, D, q, \ell). \quad (2)$$

Case 4. $k + \ell = |\mathsf{val}(B)|$. If $p = q$, then we introduce the production

$$(k, p, B, q, \ell) \to \varepsilon.$$

Case 5. $(B \to a) \in P$ and $k = \ell = 0$. If in $\mathcal{A}$ there is a path from state p to state q labeled with the letter a, then we introduce the production

$$(0, p, B, q, 0) \to \varepsilon.$$

This concludes the description of G. It is straightforward to show that $\varepsilon \in L(G)$ if and only if $\mathsf{val}(\mathbb{B}) \in L(\mathcal{A})$.

We claim that G contains at most $O\big(|\mathbb{B}|^7 \cdot |\mathcal{A}|^4 \cdot \mathsf{ord}(\mathcal{A})^2\big)$ many nonterminals. This clearly implies that G can be constructed in time $\mathsf{poly}\big(|\mathbb{B}|, |\mathcal{A}|, \mathsf{ord}(\mathcal{A})\big)$. For the second, third and fourth component of a G-nonterminal (except of S_G) there are in total $|Q|^2 \cdot |N| \leq |\mathcal{A}|^2 \cdot |\mathbb{B}|$ possibilities. Let us bound the number of positions that may appear as a first component of a G-nonterminal (an analogous argument will apply to the fifth component). Let M_1 be the set of all positions that appear as a first component of a nonterminal of G. Moreover, let us define the set

$$J = \{|\mathsf{val}(\mathbb{A}_i)| + j - |\mathsf{val}(C)| \mid 1 \leq i \leq n, \exists B, D \in N : (B, CD) \in P, j \in \mathsf{AP}(i, B)\}.$$

5
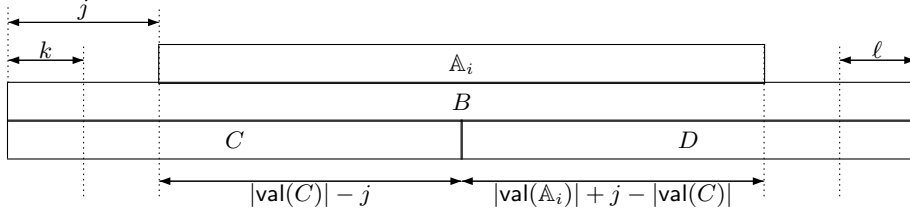
Fig. 3. Case 3

Every first component of the second G-nonterminal in the right-hand side of the G-production (2) is from J. Note that

$$|J| \leq |\mathbb{B}| \cdot \sum_{i=1}^{n} \mathsf{ord}(\mathsf{val}(\mathbb{A}_i)) \leq |\mathbb{B}| \cdot |\mathcal{A}| \cdot \mathsf{ord}(\mathcal{A}).$$

Let us now define a mapping f on $[0, |\mathsf{val}(\mathbb{B})|] \times N$ as follows:

$$f(k, B) = \begin{cases} (k - |\mathsf{val}(C)|, D) & \text{if } (B, CD) \in P, |\mathsf{val}(C)| \leq k \\ (k, C) & \text{if } (B, CD) \in P, |\mathsf{val}(C)| > k \\ \text{undefined} & \text{otherwise} \end{cases}$$

This mapping f describes the way the first and third component of a G-nonterminal evolve when applying the productions from Case 1, 2, and 3 (for Case 3, we only consider the first G-nonterminal in the right-hand side of (2)). Note that for every $(k, B) \in [0, |\mathsf{val}(\mathbb{B})|] \times N$, there is $\alpha \leq |N| - 1$ such that $f^{\alpha}(k, B) = $ undefined. Moreover, if $i \in M_1$, then there exists $(k, B) \in \{(0, S)\} \cup (J \times N)$ and $0 \leq \alpha \leq |N| - 1$ such that $f^{\alpha}(k, B) \in \{i\} \times N$. Hence, the size of M_1 is bounded by

$$(|J| \cdot |N| + 1) \cdot |N| \leq (|\mathbb{B}|^2 \cdot |\mathcal{A}| \cdot \mathsf{ord}(\mathcal{A}) + 1) \cdot |\mathbb{B}| \in O(|\mathbb{B}|^3 \cdot |\mathcal{A}| \cdot \mathsf{ord}(\mathcal{A})).$$

Hence, the number of nonterminals of G can be bounded by $O\big(|\mathbb{B}|^7 |\mathcal{A}|^4 \mathsf{ord}(\mathcal{A})^2\big)$. This concludes the proof of the theorem. $\qquad\square$

4 A nondeterministic algorithm

The goal of this section is to prove the following theorem:

Theorem 4. *Given an automaton with compressed labels $\mathcal{A}$ and an SLP $\mathbb{B}$, we can check* $\mathsf{val}(\mathbb{B}) \in L(\mathcal{A})$ *nondeterministically in time* $\mathsf{poly}\big(|\mathbb{B}|, |\mathcal{A}|, \mathsf{per}(\mathcal{A})\big)$.

In a first step, we will deal with the special case that $\mathsf{per}(\mathcal{A}) = 1$, which means that every transition is labeled with a compressed unary word (Theorem 5 below). Note that the NP bound in Theorem 5 already generalizes [13, Theorem 4], since the unary alphabet for each transition is allowed to vary.

4.1 Compressed unary labels

Theorem 5. *Given an automaton $\mathcal{A}$ with compressed labels over unary alphabets and an SLP $\mathbb{B}$, we can check whether $\mathsf{val}(\mathbb{B}) \in L(\mathcal{A})$ in* NP.

Proof. We give an algorithm for the case $\Sigma = \{0, 1\}$. The general case is similar. W.l.o.g. we may assume that $\mathsf{val}(\mathbb{B}) \in 0\{0, 1\}^*1$. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ and $\mathbb{B} = (N, \{0, 1\}, S, P)$.

Step 1. Let $m \geq 2$, $\beta_1, \ldots, \beta_m \geq 1$ such that $\mathsf{val}(\mathbb{B}) = 0^{\beta_1} 1^{\beta_2} 0^{\beta_3} \cdots 1^{\beta_m}$. Note that m might be exponentially big, however the size of the set $I = \{\beta_1, \ldots, \beta_m\}$ is bounded by the number of nonterminals of $\mathbb{B}$. The binary codings of the numbers β_i can be computed bottom-up. For each production $(A, BC) \in P$ we get a new number β_i in case $\mathsf{val}_{\mathbb{B}}(B)$ ends with the same symbol as $\mathsf{val}_{\mathbb{B}}(C)$ starts. We transform $\mathbb{B}$ into an SLP $\mathbb{C}$ over the alphabet $\Theta = \{X_i, Y_i \mid i \in I\}$ such that $\mathsf{val}(\mathbb{C}) = X_{\beta_1} Y_{\beta_2} X_{\beta_3} \cdots Y_{\beta_m}$. This can be done in deterministic polynomial time similar to the construction from [8, proof of Theorem 2]. The SLP $\mathbb{C}$ contains all nonterminals from V plus some auxiliary nonterminals. The right-hand side in $\mathbb{C}$ of an old variable $A \in V$ will be of the form Z or $Z_1 A' Z_2$, where $Z, Z_1, Z_2 \in \Theta$ and A' is an auxiliary nonterminal. Consider a production $(A, BC) \in P$, and assume that the $\mathbb{C}$-productions $(B, Z_1 B' Z_2)$, $(C, Z_3 C' Z_4)$ are already computed (the case that the right-hand side of B or C is a single symbol from Θ is similar). In case Z_2 is of the form X_i and Z_3 is of the form Y_j $(i, j \in I)$ or vice versa, we introduce the $\mathbb{C}$-productions $(A, Z_1 A' Z_4)$ and $(A', B' Z_2 Z_3 C')$. On the other hand, if, e.g., $Z_2 = X_i$ and $Z_3 = X_j$ $(i, j \in I)$, then we introduce the $\mathbb{C}$-productions $(A, Z_1 A' Z_4)$ and $(A', B' X_{i+j} C')$.

Step 2. We build nondeterministically a new automaton $\mathcal{B} = (Q, \Theta, q_0, \delta', F)$ (with noncompressed labels). For all $q, p \in Q$ and for each $i \in I$ we guess whether there is a path in $\mathcal{A}$ from q to p labeled with 0^i (resp. 1^i). If this is true, then we add a transition (q, X_i, p) (resp. (q, Y_i, p)) to δ'.

Step 3. For each pair (q, p) it can be checked nondeterministically in time $\mathsf{poly}(|\mathbb{B}|, |\mathcal{A}|)$ whether there is a path from q to p in $\mathcal{A}$ with label 0^{β_i} (resp. 1^{β_i}) (see [13, Theorem 4]). So for each transition (q, X_i, p) and (q, Y_i, p) in $\mathcal{B}$ we can check whether there is in fact a corresponding path in $\mathcal{A}$.

Step 4. We can check deterministically in time $\mathsf{poly}(|\mathbb{C}|, |\mathcal{B}|)$ whether $\mathsf{val}(\mathbb{C}) \in L(\mathcal{B})$ (see [13, Theorem 2(a)]). Clearly, $\mathsf{val}(\mathbb{B}) \in L(\mathcal{A})$ if and only if there is an automaton $\mathcal{B}$, obtained as described above, such that $\mathsf{val}(\mathbb{C}) \in L(\mathcal{B})$. $\square$

In the rest of this paper, we will prove Theorem 4. First, we have to do some combinatorics on words.

4.2 Some combinatorics on words

The following lemma is well known.

Lemma 6 (e.g. [10]). *Let $u \in \Sigma^*$ be primitive and $u^2 = vuw$ for some $v, w \in \Sigma^*$. Then either $v = \varepsilon$ or $w = \varepsilon$.*

The next lemma is an easy consequence of the well-known periodicity theorem of Fine and Wilf.

Lemma 7 (cf. [2, Corollary 6.2]). *Let $u \neq v$ be two primitive words that are not conjugate, and let $n, m \in \mathbb{N}$. Then u^n and v^m do not have a common factor of length $|u| + |v|$.*

Let now $U = \{u_1, \ldots, u_n\} \subseteq \Sigma^+$ be a collection of primitive words that are pairwise not conjugated. Later U will consist of the primitive roots of labels occurring in an automaton with compressed labels. For $1 \leq i \leq n$ let $\alpha_i = 1 + \lceil |v|/|u_i| \rceil \geq 2$, where v is a longest word in U. Lemma 7 implies:

Lemma 8. *For $i \neq j$, $u_i^{\alpha_i}$ is not a factor of a word from u_j^*.*

Let $X_1, \ldots, X_n$ be fresh letters which are not in Σ. We now define a string rewriting system R_U over the alphabet $\Sigma \cup \{X_1, \ldots, X_n\}$. First, for $1 \leq i \leq n$ let R_i consist of the following 4 rules:

$$u_i^{2\alpha_i+1} \rightarrow u_i^{\alpha_i} X_i u_i^{\alpha_i} \tag{3}$$

$$u_i^{\alpha_i+1} X_i \rightarrow u_i^{\alpha_i} X_i^2 \tag{4}$$

$$X_i u_i^{\alpha_i+1} \rightarrow X_i^2 u_i^{\alpha_i} \tag{5}$$

$$X_i u_i^{\alpha_i} X_i \rightarrow X_i^{\alpha_i+2} \tag{6}$$

Finally, let $R_U = \bigcup_{i=1}^n R_i$. Let m_U be the maximal length of a left-hand side of R_U. The following obvious fact is useful in the further investigations:

Fact 9 *If $u \rightarrow_{R_U}^* v$, then u can be obtained from v by replacing some (but not necessarily all) occurrences of X_i by u_i ($1 \leq i \leq n$).*

Clearly, R_U is terminating. Moreover, we have:

Lemma 10. *R_U is confluent.*

Proof. No left-hand side of R_U is a factor of another left-hand side. Hence, we have to check critical pairs that result from overlappings between left-hand sides. Rule (3) replaces an occurrence of u_i by X_i within the context $(u_i^{\alpha_i}, u_i^{\alpha_i})$. Similarly, (4) (resp., (5)) replaces an occurrence of u_i by X_i within the context $(u_i^{\alpha_i}, X_i)$ (resp. $(X_i, u_i^{\alpha_i})$). Finally, (6) replaces an occurrence of $u_i^{\alpha_i}$ by $X_i^{\alpha_i}$ within the context (X_i, X_i). These observations imply that critical pairs that result from an overlapping between a left-hand side of R_i and a left-hand side of R_j with $i \neq j$ can be directly resolved: Lemma 8 implies that the replaced parts in the left-hand sides cannot overlap, i.e., the overlapping is restricted to the context. It remains to consider overlappings between left-hand sides of some R_i. Again, those overlappings that are restricted to the context can be directly resolved. Since u_i does not occur properly in u_i^2 (Lemma 6), the critical pairs from Figure 4 (shown together with the resolving derivations) remain (arrows are labeled with the applied rule and possibly a number indicating the number of rule applications). This concludes the confluence proof. □
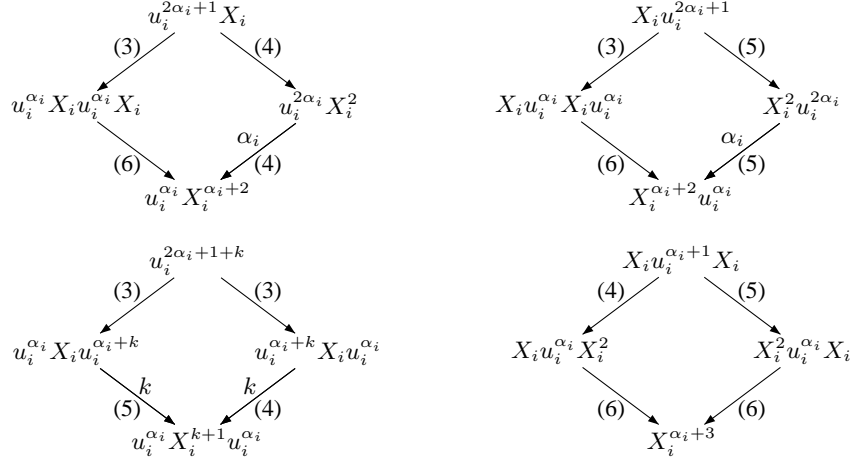
8

Fig. 4. Proving confluence of R_U

In the following, we write NF_U for NF_{R_U}. The next lemma is needed in order to prove the crucial Lemma 12 below.

Lemma 11. *Assume that* $x, y \in (\Sigma \cup \{X_1, \ldots, X_n\})^*$, $xu_i^{\alpha_i+1} \in \mathsf{IRR}(R_U)$, *and* $y \neq \varepsilon$ *neither starts with* u_i *nor* X_i. *If* $xu_i^{\alpha_i+1}y \to_{R_U}^* v$, *then* $v = xu_i^{\alpha_i+1}z$ *for some* $z \neq \varepsilon$ *that neither starts with* u_i *nor* X_i.

Proof. Using induction, it suffices to prove the lemma for the case that the derivation $xu_i^{\alpha_i+1}y \to_{R_U}^* v$ has length one, i.e., $xu_i^{\alpha_i+1}y \to_{R_U} v$. The case that v is obtained from $xu_i^{\alpha_i+1}y$ by applying a rule within the suffix y (i.e., $y \to_{R_U} y'$ and $v = xu_i^{\alpha_i+1}y'$) is clear; one can use Fact 9 to see that y' neither starts with u_i nor X_i. So, we have to consider an occurrence of a left-hand side ℓ of R_U that starts in $xu_i^{\alpha_i+1}$ and ends in y. Assume that ℓ is a left-hand side of $R_j \subseteq R_U$. If $i \neq j$, then Lemma 8 implies that the occurrence of ℓ has to start in the suffix $u_i^{\alpha_i+1}$ of $xu_i^{\alpha_i+1}$. So, only the rules $u_j^{2\alpha_j+1} \to u_j^{\alpha_j} X_j u_j^{\alpha_j}$ and $u_j^{\alpha_j+1} X_j \to u_j^{\alpha_j} X_j^2$ have to be considered. By Lemma 8, the length of an overlapping between $u_i^{\alpha_i+1}$ and $u_j^{2\alpha_j+1}$ (resp., $u_j^{\alpha_j+1}X_j$) is bounded by $\alpha_j \cdot |u_j|$. Hence, the prefix $xu_i^{\alpha_i+1}$ is not modified in the rewrite step. Moreover, the rewrite step either does not modify the first $|u_i|$ many positions of y or produces an occurrence of X_j within one of the first $|u_i|$ many positions of y. Hence, indeed, $v = xu_i^{\alpha_i+1}z$ for some $z \neq \varepsilon$ that neither starts with u_i nor X_i. Finally, consider the case $i = j$. By Lemma 6 the occurrence of the left-hand side ℓ of R_i has to start in one of the last $|u_i|$ many positions of $xu_i^{\alpha_i+1}$ (otherwise y would start with u_i or X_i). But then, the prefix $xu_i^{\alpha_i+1}$ as well as the first $|u_i|$ many positions of y are not modified in the rewrite step. $\qquad\square$

Lemma 12. *Let* $s, t \in \mathsf{IRR}(R_U)$, $s = s_1 s_2$, *and* $t = t_1 t_2$ *with* $(|s_2| = m_U$ *or* $s = s_2)$ *and* $(|t_1| = m_U$ *or* $t = t_1)$. *Then,* $\mathsf{NF}_U(st) = s_1 \mathsf{NF}_U(s_2 t_1) t_2$.

Proof. We only consider the case $|s_2| = |t_1| = m_U$. Since $st \rightarrow^*_{R_U} s_1\mathsf{NF}_U(s_2t_1)t_2$, it suffices to show that $s_1\mathsf{NF}_U(s_2t_1)t_2 \in \mathsf{IRR}(R_U)$. Assume for contradiction that $s_1\mathsf{NF}_U(s_2t_1)t_2$ is reducible. Since, $s_1, \mathsf{NF}_U(s_2t_1), t_2 \in \mathsf{IRR}(R_U)$, there has to be an occurrence of a left-hand side ℓ that starts in the prefix s_1 or that ends in the suffix t_2. By symmetry assume the former. Hence, $\ell = \ell_1\ell_2$ with $\ell_1 \neq \varepsilon \neq \ell_2$, ℓ_1 is a suffix of s_1, and ℓ_2 is a prefix of $\mathsf{NF}_U(s_2t_1)t_2$. We distinguish the following cases:

Case 1. $\ell = u_i^{2\alpha_i+1}$ for some $1 \leq i \leq n$. Then by Fact 9, $\ell_2 \in \Sigma^+$ must be a prefix of s_2t as well. Since $|s_2| \geq (2\alpha_i + 1) \cdot |u_i|$, it follows that ℓ_2 is in fact a prefix of s_2. Hence $s = s_1s_2$ is reducible, a contradiction.

Case 2. $\ell = X_i u_i^{\alpha_i+1}$ for some $1 \leq i \leq n$. Since again $\ell_2 \in \Sigma^+$, we can argue as in Case 1.

Case 3. $\ell = u_i^{\alpha_i+1}X_i$. Let $\ell_1 = u_i^{m_1}u'$ (it is a suffix of s_1) and $\ell_2 = u''u_i^{m_2}X_i$ with $u_i = u'u''$ and $\alpha_i + 1 = m_1 + 1 + m_2$. Then $u''u_i^{m_2}$ is a prefix of s_2t. Note that $|s_2| \geq (2\alpha_i + 1) \cdot |u_i|$. Let $m_3 \geq m_2$ maximal such that $u''u_i^{m_3}$ is a prefix of s_2t. We must have $m_1 + 1 + m_3 < 2\alpha_i + 1$, because otherwise $s = s_1s_2$ would contain an occurrence of $u_i^{2\alpha_i+1}$ and therefore would be reducible. Since $|s_2| \geq (2\alpha_i + 1) \cdot |u_i|$, $u''u_i^{m_3}$ must be a prefix of s_2. Let $s_1 = xu_i^{m_1}u'$ and $s_2 = u''u_i^{m_3}y$. Since $m_3 + 1 < 2\alpha_i + 1$ and $|s_2| \geq (2\alpha_i + 1) \cdot |u_i|$, we have $|y| \geq |u_i|$ Now, consider the word $xu_i^{m_1+1+m_3}yt_1 = s_1s_2t_1$. We claim that yt_1 does not start with u_i or X_i. If it would do so, then, since $|y| \geq |u_i|$, y would start with u_i or X_i. This contradicts either the maximality of m_3 (if y starts with u_i) or implies that $s = s_1s_2$ contains an occurrence of $u_i^{\alpha_i+1}X_i$ (if y starts with X_i) and is therefore reducible. Hence yt_1 neither starts with u_i nor X_i. We can therefore apply Lemma 11 to $xu_i^{m_1+1+m_3}yt_1 = s_1s_2t_1 \rightarrow^*_{R_U} s_1\mathsf{NF}_U(s_2t_1)$. It follows that $s_1\mathsf{NF}_U(s_2t_1)$ has the form $xu_i^{m_1+1+m_3}z$, where $z \neq \varepsilon$ neither starts with u_i nor X_i. But by our assumption $s_1\mathsf{NF}_U(s_2t_1)t_2$ starts with $x\ell = xu_i^{m_1+1+m_2}X_i$. This leads to a contradiction, since $m_3 \geq m_2$.

Case 4. $\ell = X_i u_i^{\alpha_i}X_i$. Can be shown analogously to Case 3. □

Lemma 12 allows us to prove Lemma 13 below. For this, an extension of SLPs is useful. A *composition system* $\mathbb{B} = (N, \Sigma, S, P)$ is defined analogously to an SLP, but in addition to productions of the form $A \rightarrow \alpha$ ($A \in N, \alpha \in (N \cup \Sigma)^*$) it may also contain productions of the form $A \rightarrow B[i : j]$ for $N \in V$ and $i, j \in \mathbb{N}$. For such a production we define $\mathsf{val}_{\mathbb{B}}(A) = \mathsf{val}_{\mathbb{B}}(B)[i : j]$. The size of this production is $1 + \lceil \log_2(i) \rceil + \lceil \log_2(j) \rceil$. As for SLPs we define $\mathsf{val}(\mathbb{B}) = \mathsf{val}_{\mathbb{B}}(S)$. In [4], Hagenah presented a polynomial time algorithm, which transforms a given composition system $\mathbb{B}$ into an SLP $\mathbb{C}$ such that $\mathsf{val}(\mathbb{C}) = \mathsf{val}(\mathbb{B})$.[1] Below, we allow more general kinds of productions, where right-hand sides are arbitrary words, built up from terminals, nonterminals and symbols $B[i : j]$ for a nonterminal B and $i, j \in \mathbb{N}$. The semantics of such productions is the obvious one. Clearly, productions of this more general form can be transformed in polynomial time into the above standard form.

Lemma 13. *From a given SLP $\mathbb{A} = (N, \Sigma, S, P)$ and a set U as above, we can compute in time $\mathsf{poly}(\sum_{i=1}^n |u_i|, |\mathbb{A}|)$ an SLP $\mathbb{B}$ such that $\mathsf{val}(\mathbb{B}) = \mathsf{NF}_U(\mathsf{val}(\mathbb{A}))$.*

---

[1] The thesis [4] is written in German. An english presentation of Hagenah's algorithm can be found in [14].

Proof. Using Hagenah's algorithm, it suffices to construct in polynomial time a composition system $\mathbb{B} = (N, \Sigma \cup \{X_1, \ldots, X_n\}, S, R)$ such that $\mathsf{val}(\mathbb{B}) = \mathsf{NF}_U(\mathsf{val}(\mathbb{A}))$. To this aim we successively add productions to $\mathbb{B}$. W.l.o.g. assume that $\mathbb{A}$ is in Chomsky normal form. First, we put all productions $(A \to a) \in P$ with $a \in \Sigma$ into R. Now, consider a production $(A \to BC) \in P$, and assume that $\mathbb{B}$ contains already enough productions so that $\mathsf{val}_\mathbb{B}(B) = \mathsf{NF}_U(\mathsf{val}_\mathbb{A}(B))$ and $\mathsf{val}_\mathbb{B}(C) = \mathsf{NF}_U(\mathsf{val}_\mathbb{A}(C))$. Let $k_B = |\mathsf{val}_\mathbb{B}(B)|$ and $k_C = |\mathsf{val}_\mathbb{B}(C)|$, these numbers can be computed in time $\mathrm{poly}(|\mathbb{A}|)$. Moreover, in time $\mathrm{poly}(\sum_{i=1}^{n} |u_i|, |\mathbb{A}|)$, we can compute the words $x = \mathsf{val}_\mathbb{B}(B)[k_B - m_U + 1 : k_B]$, $y = \mathsf{val}_\mathbb{B}(C)[1 : m_U]$, and $z = \mathsf{NF}_U(xy)$. By Lemma 12, we have

$$\mathsf{NF}_U(\mathsf{val}_\mathbb{A}(A)) = \mathsf{val}_\mathbb{B}(B)[1 : k_B - m_U] \, z \, \mathsf{val}_\mathbb{B}(C)[m_U + 1 : k_C].$$

Hence, we introduce the production $A \to B[1 : k_B - m_U] \, z \, C[m_U + 1 : k_C]$. This concludes the construction of the composition system $\mathbb{B}$. □

4.3 Proof of Theorem 4

Assume that $\mathcal{A}$ is an automaton with compressed labels. First we will transform $\mathcal{A}$ in time $\mathrm{poly}(|\mathcal{A}|, \mathsf{per}(\mathcal{A}))$ into an equivalent automaton with compressed labels with some additional nice properties. For an SLP $\mathbb{A}$ let us write $\mathsf{ord}(\mathbb{A})$ and $\mathsf{per}(\mathbb{A})$ for $\mathsf{ord}(\mathsf{val}(\mathbb{A}))$ and $\mathsf{per}(\mathsf{val}(\mathbb{A}))$, respectively, in the following. For simplicity, we will denote the automaton resulting from each of Steps 1–3 below again with $\mathcal{A}$.

Step 1. For each $\mathcal{A}$-transition $(p, \mathbb{A}, q)$, we can compute in time $\mathrm{poly}(|\mathbb{A}|)$ SLPs $\mathbb{U}$ and $\mathbb{V}$ such that $|\mathsf{val}(\mathbb{V})| < |\mathsf{val}(\mathbb{U})| = \mathsf{per}(\mathbb{A})$ and $\mathsf{val}(\mathbb{A}) = \mathsf{val}(\mathbb{U})^{\mathsf{ord}(\mathbb{A})}\mathsf{val}(\mathbb{V})$ (see e.g. [3]). Moreover, in time $O(\mathsf{per}(\mathcal{A}))$, we can explicitly compute $u = \mathsf{val}(\mathbb{U})$ (it is primitive) and $v = \mathsf{val}(\mathbb{V})$. We now replace the transition $(p, \mathbb{A}, q)$ by a path of $|v| + 1$ many transitions: a transition labeled with an SLP for $u^{\mathsf{ord}(\mathbb{A})}$, followed by a sequence of $|v|$ atomic transitions, which give an v-labeled path ending in state q. Hence, we can assume that for every transition $(p, \mathbb{A}, q)$ of $\mathcal{A}$ we have $\mathsf{val}(\mathbb{A}) = u^n$ for a primitive word u. In the following, a transition $(p, \mathbb{A}, q)$ with $\mathsf{val}(\mathbb{A}) = u^n$ (u primitive) is just written as (p, u, n, q) (an atomic transition (p, a, q) can be viewed as $(p, a, 1, q)$). In fact, instead of an SLP for u^n, we can store the pair (u, n), where n is binary coded. All following steps are polynomial w.r.t. this new representation.

Step 2. Next, assume that there are two transitions (p, u, n, q) and (r, v, m, s) such that the primitive words u and v are conjugated. Hence, there are non-empty words $x, y \in \Sigma^+$ such that $u = xy$ and $v = yx$. We may assume that $m \geq 2$, as otherwise we replace the transition (r, v, m, s) by a path of atomic transitions. We now replace the transition (r, v, m, s) by a path of $|v| + 1$ many transitions: a path of $|y|$ many atomic transitions labeled y, followed by a transition labeled with the pair $(u, m - 1)$, followed by a path of $|x|$ many atomic transitions labeled with x.

Let $U = \{u_1, \ldots, u_n\}$ be the set of primitive words that occur in transitions of $\mathcal{A}$. W.l.o.g. we can assume that $\Sigma \subseteq U$. By Step 2, u_i and u_j are not conjugated for $i \neq j$. This allows us to construct the confluent and terminating system R_U from Section 4.2. Let v be a longest word in U. Recall that we defined $\alpha_i = 1 + \lceil |v|/|u_i| \rceil$ for $1 \leq i \leq n$.
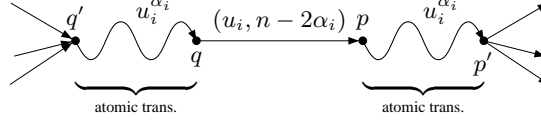
Fig. 5.

We can compute all these numbers (even in unary notation) within our preprocessing time bound $\mathsf{poly}(|\mathcal{A}|, \mathsf{per}(\mathcal{A}))$.

Step 3. The aim of this step is to ensure the following technical condition.

Each transition (q, u_i, n, p) of $\mathcal{A}$ is either atomic or there are states q', p' such that: In $\mathcal{A}$ there is a path π of atomic transitions from q' to q labeled with $u_i^{\alpha_i}$ and there is a path π' of atomic transitions from p to p' labeled with $u_i^{\alpha_i}$. Moreover, on the path π the only state with indegree > 1 could be q' and on the path π' the only state with outdegree > 1 could be p'.

To ensure this condition, we first split each transition (q', u_i, n, p') with $n \le 2\alpha_i$ into a path of atomic transitions. After that, each transition (q', u_i, n, p') with $n > 2\alpha_i$ is replaced by (see Figure 5):

- a path of atomic transitions from q' to some fresh state q labeled $u_i^{\alpha_i}$,
- a transition $(q, u_i, n - 2\alpha_i, p)$ for some fresh state p and
- a path of atomic transitions from p to p' labeled $u_i^{\alpha_i}$.

Observe that all our modifications preserve $L(\mathcal{A})$ and that they can be executed within the time bound $\mathsf{poly}(|\mathcal{A}|, \mathsf{per}(\mathcal{A}))$. This ends the preprocessing of the automaton $\mathcal{A}$.

Step 4. Let us introduce a new symbol X_i for every primitive word u_i (see also the definition of R_U). We now modify the automaton $\mathcal{A}$ as follows. For each primitive word $u_i \in U$ we replace every non-atomic transition (p, u_i, m, q) by (p, X_i, m, q). Moreover for any two states p, q of $\mathcal{A}$ we test whether there is a path of atomic transitions in $\mathcal{A}$ from p to q labeled u_i. If there is such a path we introduce a new transition (p, X_i, q). Let $\mathcal{B}$ denote our modified automaton. Now, consider an SLP $\mathbb{C}$ with $\mathsf{val}(\mathbb{C}) = \mathsf{NF}_U(\mathsf{val}(\mathbb{B}))$; such an SLP can be computed in polynomial time from $\mathbb{B}$ by Lemma 13. We claim that $\mathsf{val}(\mathbb{B}) \in L(\mathcal{A})$ if and only if $\mathsf{val}(\mathbb{C}) \in L(\mathcal{B})$. This concludes the proof of Theorem 4 as the latter question belongs to NP by Theorem 5. So it remains to prove that indeed $\mathsf{val}(\mathbb{B}) \in L(\mathcal{A})$ if and only if $\mathsf{val}(\mathbb{C}) \in L(\mathcal{B})$.

For the if-direction, consider a path from the initial state q_0 to some final state q_f in $\mathcal{B}$ labeled $\mathsf{val}(\mathbb{C})$. Replacing every transition (q, X_i, m, p) by (q, u_i, m, p) and replacing every atomic transition (q, X_i, p) by an appropriate u_i-labeled path of atomic transitions in $\mathcal{A}$ gives a path in $\mathcal{A}$ from q_0 to q_f labeled $\mathsf{val}(\mathbb{B})$.

For the other direction, consider a path π from q_0 to some final state q_f in $\mathcal{A}$ labeled $\mathsf{val}(\mathbb{B})$ and fix an occurrence of $u_i^{\alpha_i} X_i^{\beta} u_i^{\alpha_i}$ in $\mathsf{val}(\mathbb{C}) = \mathsf{NF}_U(\mathsf{val}(\mathbb{B}))$ for some $u_i \in U$

($\beta > 0$). Let $j > 0$ be the position of $\mathsf{val}(\mathbb{B})$ such that the factor u_i^β corresponding to the block X_i^β occurs at j, i.e.,

$$\mathsf{val}(\mathbb{B})[j - \alpha_i|u_i| + 1 : j + \beta|u_i|] = u_i^{\alpha_i + \beta}.$$

Let (p, u_s, m, q) be the unique transition in π that starts at $k < j$ and ends at $\ell \geq j$. Thus, $\mathsf{val}(\mathbb{B})[k + 1 : \ell] = u_s^m$, i.e., k is an occurrence of u_s^m in $\mathsf{val}(\mathbb{B})$. Assume for contradiction that $\ell > j$. Hence (p, u_s, m, q) is non-atomic and by the condition from Step 3 above, the occurrence k of u_s^m in $\mathsf{val}(\mathbb{B})$ is preceded by $u_s^{\alpha_s}$, i.e.,

$$\mathsf{val}(\mathbb{B})[k - \alpha_s|u_s| + 1 : \ell] = u_s^{\alpha_s + m}.$$

If $s = i$, then $k < j < \ell$ implies that the occurrence $j - \alpha_i|u_i|$ of $u_i^{\alpha_i}$ (which covers all positions from $[j - \alpha_i|u_i| + 1, j]$) is strictly contained in the occurrence $k - \alpha_i|u_i|$ of $u_i^{\alpha_i + m}$ (which covers all positions from $[k - \alpha_i|u_i| + 1, \ell]$). In particular, the occurrence $j - \alpha_i|u_i|$ of u_i is contained in an occurrence $< j - \alpha_i|u_i|$ of u_i^2. Lemma 6 implies that $j - \alpha_i|u_i| - |u_i| = j - (\alpha_i + 1)|u_i|$ is an occurrence of u_i as well. Hence, we have $\mathsf{val}(\mathbb{B})[j - (\alpha_i + 1)|u_i| + 1 : j] = u_i^{\alpha_i + 1}$. But then, in $\mathsf{val}(\mathbb{C}) = \mathsf{NF}_U(\mathsf{val}(\mathbb{B}))$ we would obtain the factor $u_i^{\alpha_i} X_i^\gamma u_i^{\alpha_i}$ for some $\gamma > \beta$ instead of $u_i^{\alpha_i} X_i^\beta u_i^{\alpha_i}$, which is a contradiction. Hence $s \neq i$. But then either $u_i^{\alpha_i}$ is contained in $u_s^{\alpha_s + m}$ (if $k - \alpha_s|u_s| \leq j - \alpha_i|u_i|$) or $u_s^{\alpha_s}$ is contained in $u_i^{\alpha_i}$ (if $j - \alpha_i|u_i| \leq k - \alpha_s|u_s|$). This contradicts Lemma 8.

Hence $\ell = j$, i.e., there is a transition in π starting at j. A symmetric argument shows that there is a transition ending at $j + \beta|u_i|$ and hence there is a subpath π' of π from p' to q' that corresponds exactly to the block X_i^β. In fact, our argument also shows that this subpath π' cannot contain a non-atomic transition (p'', u_s, m, q'') with $s \neq i$ (we would obtain again a contradiction to Lemma 8). Hence, by Lemma 6 (u_i is primitive), π' can be decomposed into atomic paths labeled u_i and non-atomic transitions of the form (p, u_i, m, q). Hence, π' has a corresponding X_i^β-labeled path from p' to q' in $\mathcal{B}$. By doing this argument for all factors $u_i^{\alpha_i} X_i^\beta u_i^{\alpha_i}$ in $\mathsf{val}(\mathbb{C})$, we obtain a path from q_0 to q_f in $\mathcal{B}$ labeled $\mathsf{val}(\mathbb{C})$. This concludes the proof of Theorem 4.

5 Conclusion

We have considered the membership problem for a compressed string and an automaton with compressed labels. Two algorithms for this problem were developed. The first algorithm is deterministic and works in polynomial time if all transition labels have a small order (polynomial in the input size). The second algorithm is nondeterministic and works in polynomial time if all transition labels have a small period (polynomial in the input size), i.e., are highly periodic. Hence, these two algorithms cover two extreme cases (almost nonperiodic versus highly periodic). But the complexity of the general case remains open. Following [13], we conjecture that the general membership problem for compressed strings and automata with compressed labels belongs to NP. This would follow from the truth of the following conjecture:

Conjecture 14. If $\mathsf{val}(\mathbb{A}) \in L(\mathcal{A})$ for an SLP $\mathbb{A}$ and an automaton $\mathcal{A}$ with compressed labels, then there exists an accepting run of $\mathcal{A}$ on $\mathsf{val}(\mathbb{A})$ (viewed as a word over the set of transition triples of $\mathcal{A}$), which can be generated by an SLP of size $\mathsf{poly}(|\mathbb{A}|, |\mathcal{A}|)$.

Indeed, if this conjecture is true, we simply can guess an SLP $\mathbb{R}$ of size $\mathsf{poly}(|\mathbb{A}|, |\mathcal{A}|)$ over the set of transition tuples of $\mathcal{A}$. In polynomial time, we can check, whether $\mathbb{R}$ indeed generates an accepting run of $\mathcal{A}$ for some word (this is a regular property). Moreover, an SLP $\mathbb{B}$ for that word can be computed easily from $\mathbb{R}$. It remains to check whether $\mathsf{val}(\mathbb{A}) = \mathsf{val}(\mathbb{B})$, which can be done in polynomial time [12]. One might first study Conjecture 14 for the case that $\mathcal{A}$ is *deterministic*. Here, an automaton with compressed labels is deterministic, if for each pair of transition triples $(p, \mathbb{A}, q), (p, \mathbb{B}, r)$, neither $\mathsf{val}(\mathbb{A})$ is a prefix of $\mathsf{val}(\mathbb{B})$ nor vice versa. In this case, if there is an accepting run of $\mathcal{A}$ on a word w, there is a unique such run. Even for deterministic automata with compressed labels we are not aware of a better upper bound than PSPACE.

References

1. R. V. Book and F. Otto. *String–Rewriting Systems*. Springer, 1993.
2. C. Choffrut and J. Karhumäki. Combinatorics on words. In G. Rozenberg and A. Salomaa, editors, *Word, Language, Grammar*, volume 1 of *Handbook of Formal Languages*, chapter 6, pages 329–438. Springer, 1997.
3. 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.
4. Ch. Hagenah. *Gleichungen mit regulären Randbedingungen über freien Gruppen*. PhD thesis, University of Stuttgart, Institut für Informatik, 2000.
5. Y. Lifshits. Processing compressed texts: A tractability border. In *Proc. CPM 2007*, LNCS 4580, pages 228–240. Springer, 2007.
6. Y. Lifshits and M. Lohrey. Querying and embedding compressed texts. In *Proc. MFCS 2006*, LNCS 4162, pages 681–692. Springer, 2006.
7. M. Lohrey. Compressed membership problems for regular expressions and hierarchical automata. *Internat. J. Found. Comput. Sci.*, 21(5):817–841, 2010.
8. M. Lohrey and S. Schleimer. Efficient computation in groups via compression. In *Proc. CSR 2007*, LNCS 4649, pages 249–258. Springer, 2007.
9. M. Lohrey. Word problems and membership problems on compressed words. *SIAM J. Comput.*, 35(5):1210 – 1240, 2006.
10. M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, Reading, MA, 1983.
11. J. Macdonald. Compressed words and automorphisms in fully residually free groups. *Internat. J. Algebra Comput.*, 20(3):343–355, 2010.
12. W. Plandowski. Testing equivalence of morphisms on context-free languages. In *Proc. ESA'94*, LNCS 855, pages 460–470. Springer, 1994.
13. 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.
14. S. Schleimer. Polynomial-time word problems. *Comment. Math. Helv.*, 83(4):741–765, 2008.
15. J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.