Information and Computation 209 (2011) 951–965

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Leaf languages and string compression ☆

Markus Lohrey

Universität Leipzig, Institut für Informatik, Germany

ARTICLE INFO

Article history:
Received 24 November 2009
Available online 15 February 2011

Keywords:
Leaf languages
Straight-line programs
Compressed string
Complexity theory

ABSTRACT

Tight connections between leaf languages and strings compressed by straight-line programs (SLPs) are established. It is shown that the compressed membership problem for a language L is complete for the leaf language class defined by L via logspace machines. A more difficult variant of the compressed membership problem for L is shown to be complete for the leaf language class defined by L via polynomial time machines. As a corollary, it is shown that there exists a fixed linear visibly pushdown language for which the compressed membership problem is PSPACE-complete. For XML languages, it is shown that the compressed membership problem is coNP-complete. Furthermore it is shown that the embedding problem for SLP-compressed strings is hard for PP (probabilistic polynomial time).

1. Introduction

Leaf languages were introduced in [9,37] and have become an important concept in complexity theory. Let us consider a nondeterministic Turing machine M. For a given input x, one considers the yield string of the ordered computation tree (i.e., the string obtained by listing all leaves from left to right), where accepting (resp. rejecting) leaf configurations yield the letter 1 (resp. 0). This string is called the *leaf string* corresponding to the input x. For a given language $K \subseteq \{0,1\}^*$ let $\mathsf{LEAF}(M,K)$ denote the set of all inputs for M such that the corresponding leaf string belongs to K. By fixing K and taking for M all nondeterministic polynomial time machines, one obtains the polynomial time leaf language class $\mathsf{LEAF}_a^{\mathsf{P}}(K)$. The index a indicates that we allow Turing machines with arbitrary (non-balanced) computation trees. If we restrict to machines with balanced computation trees, we obtain the class $\mathsf{LEAF}_b^{\mathsf{P}}(K)$. See [17,19,21] for a discussion of the different shapes for computation trees.

Many complexity classes can be defined in a uniform way with this construction. For instance, $\mathsf{NP} = \mathsf{LEAF}_x^p(0^*1\{0,1\}^*)$ and $\mathsf{coNP} = \mathsf{LEAF}_x^p(1^*)$ for both $x = a$ and $x = b$. In [18], it was shown that $\mathsf{PSPACE} = \mathsf{LEAF}_b^{\mathsf{P}}(K)$ for a fixed regular language K. In [21], logspace leaf language classes $\mathsf{LEAF}_a^{\mathsf{L}}(K)$ and $\mathsf{LEAF}_b^L(K)$, where M varies over all (resp. all balanced) nondeterministic logspace machines, were investigated. Among other results, a fixed deterministic context-free language K with $\mathsf{PSPACE} = \mathsf{LEAF}_a^{\mathsf{L}}(K)$ was presented. In [10], it was shown that in fact a fixed deterministic *one-counter* language K as well as a fixed *linear* deterministic context-free language [20] suffices in order to obtain PSPACE. Here "linear" means that the pushdown automaton makes only one turn.

In [8,36], a tight connection between leaf languages and computational problems for succinct input representations was established. More precisely, it was shown that the membership problem for a language $K \subseteq \{0,1\}^*$ is complete (w.r.t. polynomial time reductions in [8] and projection reductions in [36]) for the leaf language class $\mathsf{LEAF}_b^{\mathsf{P}}(K)$, if the input string x is represented by a Boolean circuit. A Boolean circuit $C(x_1,\ldots,x_n)$ with n inputs represents a string x of length 2^n in the natural way: the ith position in x carries a 1 if and only if $C(a_1,\ldots,a_n) = 1$, where $a_1 \cdots a_n$ is the n-bit binary representation of i. In this paper we consider another compressed representation for strings, namely *straight-line programs* (SLPs) [33], which

☆ This work is supported by the German Research Foundation (DFG) via the research project ALKODA.
E-mail address: lohrey@informatik.uni-leipzig.de

0890-5401/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2011.01.009

is compared to Boolean circuits more amenable to efficient algorithms. A straight-line program is a context-free grammar $\mathbb{A}$ that generates exactly one string $\mathrm{val}(\mathbb{A})$. In an SLP, repeated subpatterns in a string have to be represented only once by introducing a nonterminal for the pattern. An SLP with n productions can generate a string of length 2^n by repeated doubling. Hence, an SLP can be seen indeed as a compressed representation of the string it generates. Several other dictionary-based compressed representations, like for instance Lempel–Ziv (LZ) factorizations [40], can be converted in polynomial time into SLPs and vice versa [33]. This implies that complexity results can be transferee from SLP-encoded input strings to LZ-encoded input strings.

Algorithmic problems for SLP-compressed strings were studied e.g. in [6,25,26,28,29,32,33]. A central problem in this context is the *compressed membership problem* for a language K: it is asked whether $\mathrm{val}(\mathbb{A}) \in K$ for a given SLP $\mathbb{A}$. In [26] it was shown that there exists a fixed linear deterministic context-free language with a PSPACE-complete compressed membership problem. A straightforward argument shows that for every language K, the compressed membership problem for K is complete for the logspace leaf language class $\mathsf{LEAF}_a^{\mathsf{L}}(K)$ (Proposition 4). As a consequence, the existence of a linear deterministic context-free language with a PSPACE-complete compressed membership problem [26] can be deduced from the above mentioned $\mathsf{LEAF}_a^{\mathsf{L}}$-characterization of PSPACE from [10], and vice versa. For polynomial time leaf languages, we reveal a more subtle relationship to SLPs. Recall that the *convolution* $u \otimes v$ of two strings $u, v \in \Sigma^*$ is the string over the paired alphabet $\Sigma \times \Sigma$ that is obtained from gluing u and v in the natural way (we cut off the longer string to the length of the shorter one). We define a fixed projection homomorphism $\rho : \{0,1\} \times \{0,1\} \to \{0,1\}$ such that for every language K, the problem of checking $\rho(\mathrm{val}(\mathbb{A}) \otimes \mathrm{val}(\mathbb{B})) \in K$ for two given SLPs $\mathbb{A}, \mathbb{B}$ is complete for the class $\mathsf{LEAF}_b^{\mathsf{P}}(K)$ (Corollary 6). By combining Corollary 6 with the main result from [18] ($\mathsf{PSPACE} = \mathsf{LEAF}_b^{\mathsf{P}}(K)$ for a certain regular language K), we obtain a regular language L for which it is PSPACE-complete to check whether the convolution of two SLP-compressed strings belongs to L (Corollary 8). Recently, the convolution of SLP-compressed strings was also studied in [6], where for every $n \geq 0$, SLPs $\mathbb{A}_n$, $\mathbb{B}_n$ of size $n^{O(1)}$ were constructed such that every SLP for the convolution $\mathrm{val}(\mathbb{A}_n) \otimes \mathrm{val}(\mathbb{B}_n)$ has size $\Omega(2^{n/2})$.

From Corollary 8 we obtain a strengthening of one of the above mentioned results from [10] ($\mathsf{PSPACE} = \mathsf{LEAF}_a^{\mathsf{L}}(K)$ for a linear deterministic context-free language K as well as a deterministic one-counter language K) to *visibly pushdown languages* [1]. The latter constitute a subclass of the deterministic context-free languages which received a lot of attention in recent years due to its nice closure and decidability properties. Visibly pushdown languages can be recognized by *deterministic* pushdown automata, where it depends only on the input symbol whether the automaton pushes or pops. Visibly pushdown languages were already introduced in [39] as input-driven languages. In [12] it was shown that every visibly pushdown language can be recognized in NC^1; thus the complexity is the same as for regular languages [2]. In contrast to this, there exist linear deterministic context-free languages as well as deterministic one-counter languages with an L-complete membership problem [20]. We show that there exists a linear visibly pushdown language with a PSPACE-complete compressed membership problem (Theorem 9). Together with Proposition 4, it follows that $\mathsf{PSPACE} = \mathsf{LEAF}_a^{\mathsf{L}}(K)$ for a linear visibly pushdown language K (Corollary 10).

In [31], nondeterministic finite automata (instead of polynomial time (resp. logspace) Turing-machines) were used as a device for generating leaf strings. This leads to the definition of the leaf language class $\mathsf{LEAF}^{\mathsf{FA}}(K)$. It was shown that $\mathsf{CFL} \subsetneq \mathsf{LEAF}^{\mathsf{FA}}(\mathsf{CFL}) \subseteq \mathsf{DSPACE}(n^2) \cap \mathsf{DTIME}(2^{O(n)})$, and the question for sharper upper and lower bounds was posed. Here we give a partial answer to this question. For the linear visibly pushdown language mentioned in the previous paragraph, the class $\mathsf{LEAF}^{\mathsf{FA}}(K)$ contains a PSPACE-complete language (Theorem 11).

Another application of the connection between SLP-compression and leaf languages is presented in Section 4.2. The compressed embedding problem (briefly COMPRESSED-EMBEDDING) asks for two given SLPs $\mathbb{A}$ and $\mathbb{B}$ whether $\mathrm{val}(\mathbb{A})$ is a *subsequence* of $\mathrm{val}(\mathbb{B})$, i.e., whether $\mathrm{val}(\mathbb{A})$ can be embedded into $\mathrm{val}(\mathbb{B})$ where consecutive positions in $\mathrm{val}(\mathbb{A})$ can be mapped to non-consecutive positions in $\mathrm{val}(\mathbb{B})$. In [25], it was shown that COMPRESSED-EMBEDDING is hard for $\mathsf{P}_{\|}^{\mathsf{NP}}$, which is the class of all problems that can be solved on a deterministic Turing-machine with access to an NP-oracle, where all queries are sent in parallel to the oracle (non-adaptive oracle access). A simplified proof can be found in [28]. Here we will strengthen the lower bound of $\mathsf{P}_{\|}^{\mathsf{NP}}$ to PP (Theorem 13). A language L belongs to the class PP (probabilistic polynomial time) if there exists a polynomial time NTM M such that $w \in L$ if and only if on input w the number of accepting computations is larger than the number of rejecting computations. In other words, the acceptance probability has to be larger than $1/2$. It is known that $\mathsf{P}_{\|}^{\mathsf{NP}} \subseteq \mathsf{PP}$ [4]. Moreover, Toda's famous theorem [35] states that P^{PP} contains the polynomial time hierarchy. Hence, PP-hardness of COMPRESSED-EMBEDDING implies that COMPRESSED-EMBEDDING is not contained in the polynomial time hierarchy unless the latter collapses. The best known upper bound for COMPRESSED-EMBEDDING is still PSPACE.

Finally, in Section 5 we consider *XML-languages* [5], which constitute a subclass of the visibly pushdown languages. XML-languages are generated by a special kind of context-free grammars (XML-grammars), where every right-hand side of a production is enclosed by a matching pair of brackets. XML-grammars capture the syntactic features of XML document type definitions (DTDs), see [5]. We prove that, unlike for visibly pushdown languages, for every XML-language the compressed membership problem is in coNP and that there are coNP-complete instances.

A short version of this paper appeared in [27].