Information Processing Letters 111 (2011) 338–341

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

P-hardness of the emptiness problem for visibly pushdown languages ☆

Martin Lange

Dept. of Elect. Eng. and Computer Science, University of Kassel, Germany

ARTICLE INFO

Article history:
Received 4 August 2010
Received in revised form 8 December 2010
Accepted 16 December 2010
Available online 22 December 2010
Communicated by J. Torán

Keywords:
Formal languages
Complexity

ABSTRACT

Visibly pushdown languages form a subclass of the context-free languages which is appealing because of its nice algorithmic and closure properties. Here we show that the emptiness problem for this class is not any easier than the emptiness problem for context-free languages, namely hard for deterministic polynomial time. The proof consists of a reduction from the alternating graph reachability problem.

1. Introduction

Context-free languages (CFL) are an important formalism for the specification of the syntax of programming or natural languages. However, the class of context-free languages lacks some important closure properties, namely determinisability and closure under complement and intersection. The class of deterministic context-free languages (DCFL) trivially admits determinisability and is therefore closed under the complement operation but is not closed under intersections either. Many applications — for instance the specification of runs of recursive programs — require more expressive power then just the regular language on the one hand, and such closure properties on the other.

An important subclass of CFL which does possess all of these three properties is the class of *visibly pushdown languages* (VPL) [1] which was originally introduced under the name *input driven languages* [2]. These are recognised by visibly pushdown automata (VPA) which are nondeterministic pushdown automata (PDA) in which the input letter determines the type of stack action that the automaton carries out when reading this letter.

Since every VPA is also a PDA, upper bounds on decision problems for PDA trivially carry over to VPA. For instance, the emptiness problem or the membership problem are solvable in deterministic polynomial time [3].[1] In case of CFL, this is also known to be optimal: the emptiness problem for CFL is P-hard. This result seems to be folklore and is indeed easy to prove by a reduction from the alternating graph reachability problem which is an abstracted version of the membership problem for alternating, logarithmic space bounded Turing Machines. This is one of the standard P-hard decision problems [5].

A natural question to ask is whether or not the emptiness problem for VPA is P-hard. Here we show that the answer is “yes”. We give a reduction from the alternating graph reachability problem to the emptiness problem for VPA which follows along the same lines as the one for PDA but is slightly more involved due to the technical restrictions VPA have in relation to PDA.

Since there is no single natural class of visibly pushdown languages, but several classes — each parameterised by a visibly pushdown alphabet — we try to obtain lower bounds on the sizes of these alphabets which still cause P-hardness. Finally, we conclude with some remarks on the complexity of the membership problem for VPA, in partic-

☆ Research supported by the European Research Council under the European Union's Seventh Framewrok Programme (FP7/2007-2013)/ERC Grant Agreement No. 259267.

E-mail address: lange.martin@gmail.com.

[1] Note that many standard textbooks only present polynomial time algorithms for grammars in Chomsky normal form together with an exponential translation into this form. See [4] for further details.

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2010.12.013

ular why this does not simply follow from the emptiness problem as it does in the case of PDA.

2. Visibly pushdown automata and languages

A *visibly pushdown alphabet* is an alphabet Σ which is partitioned into three parts $(\Sigma_{\mathsf{push}}, \Sigma_{\mathsf{int}}, \Sigma_{\mathsf{pop}})$ consisting of those letters which, respectively, force a VPA to push a stack symbol, leave the stack untouched, and pop a stack symbol. The *size* of the alphabet Σ is a triple $(|\Sigma_{\mathsf{push}}|, |\Sigma_{\mathsf{int}}|, |\Sigma_{\mathsf{pop}}|)$. When comparing alphabet sizes we use the pointwise order.

A VPA is a tuple $\mathcal{A} = (Q, \Sigma, \Gamma, q_0, \delta, F)$ where Q is a finite set of states, Σ as above, Γ is a finite stack alphabet containing the special bottom-of-stack symbol $\bot$, $q_0 \in Q$ is the designated initial state, and $F \subseteq Q$ is a set of final states. The transition relation δ is partitioned into three parts $\delta = \delta_{\mathsf{push}} \cup \delta_{\mathsf{int}} \cup \delta_{\mathsf{pop}}$ where

$$\delta_{\mathsf{push}} \subseteq Q \times \Sigma_{\mathsf{push}} \times Q \times \big(\Gamma \setminus \{\bot\}\big)$$

$$\delta_{\mathsf{int}} \subseteq Q \times \Sigma_{\mathsf{int}} \times Q$$

$$\delta_{\mathsf{pop}} \subseteq Q \times \Sigma_{\mathsf{pop}} \times \Gamma \times Q$$

A *configuration* is a $(q, \gamma, w) \in Q \times \Gamma^* \times \Sigma^*$. The computational behaviour of the VPA $\mathcal{A}$ is explained by a relation $\vdash$ on configurations. It is defined as follows. Let $q \in Q$, $\gamma \in \Gamma^*$, $a \in \Sigma$, $w \in \Sigma^*$, $B \in \Gamma$ with $B \neq \bot$.

$$(q, \gamma, aw) \vdash \big(q', B\gamma, w\big) \quad \text{if } \big(q, a, q', B\big) \in \delta_{\mathsf{push}}$$
$$(q, \gamma, aw) \vdash \big(q', \gamma, w\big) \quad \text{if } \big(q, a, q'\big) \in \delta_{\mathsf{int}}$$
$$(q, B\gamma, aw) \vdash \big(q', \gamma, w\big) \quad \text{if } \big(q, a, B, q'\big) \in \delta_{\mathsf{pop}}$$
$$(q, \bot, aw) \vdash \big(q', \bot, w\big) \quad \text{if } \big(q, a, \bot, q'\big) \in \delta_{\mathsf{pop}}$$

Let $\vdash^*$ be the reflexive–transitive closure of $\vdash$. Also, we write $\vdash^n$ to denote the n-fold iteration of this relation. The language accepted by $\mathcal{A}$ is then

$$L(\mathcal{A}) := \big\{ w \in \Sigma^* \;\big|\; \exists q \in F.\, \exists \gamma \in \Gamma^*.\, (q_0, \bot, w) \vdash^* (q, \gamma, \epsilon) \big\}$$

It is very easy to see that the language $\{a^n b^n \mid n \in \mathbb{N}\}$ for instance is a VPL over the visibly pushdown alphabet $(\{a\}, \emptyset, \{b\})$. Moreover, every Dyck language is a VPL over a suitable partitioning of the alphabet. On the other hand, $\{a^n b a^n \mid n \in \mathbb{N}\}$ is not a VPL over any alphabet since a PDA would have to perform push actions when reading the first a's, and pop actions when reading the latter a's.

Finally, consider the alphabet $(\{a\}, \{b\}, \{c\})$ and the language L_0 over this alphabet given by the following context-free grammar.

$$S \to \epsilon | bS | aScS$$

Again, it is not difficult to see that this is a VPL.

The proofs of soundness and completeness of the aforementioned reduction use the following technical lemma. We omit a detailed proof since it is very easy to prove by induction on n.

Lemma 1. *For all $n \in \mathbb{N}$, all $q, q' \in Q$, all $\gamma, \gamma', \delta \in \Gamma^*$, all $w, w', v \in \Sigma^*$ we have*:

(a) $(q, \gamma, w) \vdash^n (q', \gamma', w')$ *iff* $(q, \gamma, wv) \vdash^n (q', \gamma', w'v)$,
(b) *if* $(q, \gamma, w) \vdash^n (q', \gamma', w')$ *then* $(q, \gamma\delta, w) \vdash^n (q', \gamma'\delta, w')$,
(c) *if* $(q, \gamma\delta, w) \vdash^n (q', \gamma'\delta, w')$ *such that all intermediate configurations have a stack of the form* $\delta'\delta$ *then* $(q, \gamma, w) \vdash^n (q', \gamma', w')$.

Note that it is not possible to extend the model of VPA to allow for ϵ-transitions. Closure properties and determinisability for VPA depend on the fact that with every letter of the input that is consumed, exactly one of the three types of stack actions occur. Additional ϵ-transitions in between could therefore at most be allowed when the stack remains untouched.

3. The complexity of the emptiness problem

Recall that the following problem, known as the alternating graph reachability problem, is P-hard [5]. Let $G = (V, V_0, V_1, E, s, T)$ be a directed graph such that V is the set of nodes which is partitioned into $V_0 \cup V_1 \cup T$ and $E \subseteq V \times V$ is the set of edges. We will write xE for the set of all E-successors of a node $x \in V$. Furthermore, $s \in V$ is a designated starting node. This gives rise to an inductively defined set of nodes $R_G(T)$ from which T is *alternating-reachable* in G.

$$R_G^0(T) := T$$
$$R_G^{i+1}(T) := R_G^i(T) \cup \big\{ x \in V_0 \;\big|\; \exists y \in R_G^i(T).\, (x, y) \in E \big\} \cup \big\{ x \in V_1 \;\big|\; \forall y \in V.\, \text{if } (x, y) \in E \text{ then } y \in R_G^i(T) \big\}$$
$$R_G(T) := \bigcup_{i \in \mathbb{N}} R^i(T)$$

The problem then consists of deciding whether or not $s \in R_G(T)$.

Note that the configuration graph of a logarithmic space bounded Turing Machine is of at most polynomial size. Hence, the alternating graph reachability problem in fact asks whether or not an alternating logspace machine accepts a given input word.

We can furthermore assume that every node in V_1 in G has out-degree equal to 2. It is always possible to add two sink nodes to V_0 without outgoing edges, and add edges from every node in V_1 with outdegree < 2 to either or these nodes. Furthermore, for every node in $x \in V_1$ with successors $y_1, \ldots, y_k$ for some $k > 2$ we can add a new node z to V_1 with successors y_{k-1} and y_k, remove the edges (x, y_{k-1}) and (x, y_k) and add an edge (x, z). This reduces the outdegree of a node by 1 and can be iterated until all nodes in V_1 have outdegree 2. Suppose G' is the graph resulting from G by means of this transformation. It should be clear that $s \in R_{G'}(T)$ iff $s \in R_G(T)$. Furthermore the number of nodes in G' is linear in the number of edges in G.

We will use this specialised problem for the reduction to VPA emptiness. Let $G = (V, V_0, V_1, E, s, T)$ be such a graph with outdegree 2 for all nodes in V_1. We will

construct a VPA $\mathcal{A}_G = (Q, \Sigma, \Gamma, q_0, \delta, F)$ over the visibly pushdown alphabet Σ with $\Sigma_{\mathsf{push}} = \{\mathsf{a}\}$, $\Sigma_{\mathsf{int}} = \{\mathsf{b}\}$, and $\Sigma_{\mathsf{pop}} = \{\mathsf{c}\}$. Its components are as follows.

- $Q := V \cup \{\mathsf{fin}\}$ where $\mathsf{fin} \notin V$,
- $\Gamma := V \cup \{\bot\}$ where $\bot \notin V$,
- $q_0 = s$,
- $F = \{\mathsf{fin}\}$, and
- δ given by

$$\delta_{\mathsf{push}} := \big\{(x, \mathsf{a}, y_1, y_2) \;\big|\; x \in V_1 \text{ and } xE = \{y_1, y_2\}\big\}$$
$$\delta_{\mathsf{int}} := \big\{(x, \mathsf{b}, y) \;\big|\; x \in V_0 \text{ and } (x, y) \in E\big\}$$
$$\delta_{\mathsf{pop}} := \big\{(x, \mathsf{c}, y, y) \;\big|\; x \in T, y \in V\big\} \cup \big\{(x, \mathsf{c}, \bot, \mathsf{fin}) \;\big|\; x \in T\big\}$$

Intuitively, $\mathcal{A}_G$ starts in s to "search" for T. If its current state is a node in V_0 then it requests a b symbol and continues with a successor node. If it is in V_1 then it requests an a symbol, pushes one of the successor nodes onto the stack and continues with the other one. If it has reached T then it requests to read a c symbol and continues with the top-most node on the stack. If the stack is empty, it moves to a final state. Thus, it simply uses a depth-first search strategy in order to find a tree which witnesses that T is alternating-reachable from s. Such trees are encoded in words of the grammar L_0 presented in Section 2.

Theorem 2. *The emptiness problem for VPA over an alphabet of size* $(1, 1, 1)$ *is P-hard under* AC^0*-reductions.*

Proof. Suppose $G = (V, V_0, V_1, E, s, T)$ is a graph serving as an input to the alternating graph reachability problem. $\mathcal{A}_G$ can be constructed from G by an AC^0-reduction. It remains to be seen that $L(\mathcal{A}_G) \neq \emptyset$ iff $s \in R_G(T)$.

"$\Longleftarrow$" In order to use induction we need to strengthen the statement slightly and will show that for all $i \in \mathbb{N}$ and all $x \in R_G^i(T)$ there is a $w_x \in \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}^*$ such that $(x, \epsilon, w_x) \vdash^* (z, \epsilon, \epsilon)$ for some $z \in T$. Note that, if this is the case then $(x, \bot, w_x\mathsf{c}) \vdash^* (z, \bot, \mathsf{c}) \vdash (\mathsf{fin}, \bot, \epsilon)$ using Lemma 1 (a) and (b) and the construction of δ_{pop}. It should be clear that this implies the statement because of $q_0 = s$.

In the base case we have $i = 0$, i.e. $x \in R_G^0(T)$ which means $x \in T$. Clearly, $w_x := \epsilon$ witnesses the claim because $\vdash^*$ is reflexive.

In the inductive step let $i > 0$ and assume $x \in R_G^i(T)$. There are three cases. If $x \in R_G^{i-1}(T)$ then the claim follows from the hypothesis immediately. If $x \in V_0$ then there must be a y with $(x, y) \in E$. By hypothesis, there is a w_y such that $(y, \epsilon, w_y) \vdash^* (z, \epsilon, \epsilon)$ for some $z \in T$. Let $w_x := \mathsf{b}w_y$. Then we have $(x, \epsilon, \mathsf{b}w_y) \vdash (y, \epsilon, w_y) \vdash^* (z, \epsilon, \epsilon)$ which finishes this case.

In the last case we have $x \in V_1$. Let y_1, y_2 be its two successors. By hypothesis there is a w_{y_1} such that $(y_1, \epsilon, w_{y_1}) \vdash^* (z_1, \epsilon, \epsilon)$ for some $z_1 \in T$. According to Lemma 1 (b) we also have $(y_1, y_2, w_{y_1}) \vdash^* (z_1, y_2, \epsilon)$. Using the hypothesis a second time, we obtain a w_{y_2} such that $(y_2, \epsilon, w_{y_2}) \vdash_\gamma^* (z_2, \epsilon, \epsilon)$ for some $z_2 \in T$. Now let $w_x := \mathsf{a}w_{y_1}\mathsf{c}w_{y_2}$ and consider the following derivation.

$$(x, \epsilon, \mathsf{a}w_{y_1}\mathsf{c}w_{y_2}) \vdash (y_1, y_2, w_{y_1}\mathsf{c}w_{y_2}) \vdash^* (z_1, y_2, \mathsf{c}w_{y_2}) \vdash (y_2, \epsilon, w_{y_2}) \vdash^* (z_2, \epsilon, \epsilon)$$

This finishes the completeness part of the proof.

"$\Longrightarrow$" Suppose $L(\mathcal{A}_G) \neq \emptyset$, i.e. there is a w with $(q_0, \bot, w) \vdash^* (\mathsf{fin}, \bot, \epsilon)$. Note that $q_0 \neq \mathsf{fin}$ and fin is only reachable via a pop-transition on the stack $\bot$ from some $z \in T$. Hence, we must have $w = v\mathsf{c}$, and $(q_0, \bot, v) \vdash^* (z, \bot, \epsilon)$. Let $v = a_0 \dots a_{n-1}$. There must exist an $n \in \mathbb{N}$, $q_0, q_1, \dots, q_n \in Q$, $\gamma_0, \gamma_1, \dots, \gamma_n$ such that $(q_i, \gamma_i, a_i \dots a_{n-1}) \vdash (q_{i+1}, \gamma_{i+1}, a_{i+1} \dots a_{n-1})$ and $q_n = z$, $\gamma_0 = \bot$, $\gamma_n = \bot$. We will now show by induction on the length n of such a derivation that $q_0 \in R_G(T)$. It should be clear that this proves the claim because $q_0 = s$.

The base case of $n = 0$ is immediate because $q_n = z$, $z \in T$, and $T = R_G^0(T)$. For the step case we distinguish three cases. Remember that $Q = V$, i.e. automaton states are also nodes in the graph. If $q_0 \in T$ then we are finished as in the base case. If $q_0 \in V_0$ then we must have $(q_0, a_0, q_1) \in \delta_{\mathsf{int}}$ and therefore $(q_0, q_1) \in E$. Note that $(q_1, \bot, a_1 \dots a_{n-1}) \vdash^{n-1} (q_n, \bot, \epsilon)$ and therefore $q_1 \in R_G(T)$. But then $q_0 \in R_G(T)$ as well.

In the last case we have $q_0 \in V_1$. Note that q_0 must have two successors in G, and one of them must be q_1. Let y be the other. Then we have $\gamma_1 = y\bot$. Let i be the smallest index such that $\gamma_i = \bot$, i.e. take the moment in which y gets popped from the stack again. Then we must have $q_i = y$, and $(y, \bot, a_i \dots a_{n-1}) \vdash^{n-i} (q_n, \bot, \epsilon)$. Since $i \geqslant 1$ we have $y \in R_G(T)$ by hypothesis. Furthermore, we have $(q_1, y\bot, a_1 \dots a_{n-1}) \vdash^{i-1} (q_{i-1}, y\bot, a_{i-1} \dots a_{n-1}) \vdash (q_i, \bot, a_i \dots a_{n-1})$. The last step is only possible if $q_{i-1} \in T$. According to Lemma 1 (c) we also have $(q_1, \bot, a_1 \dots a_{i-1}) \vdash^{i-1} (q_{i-1}, \bot, a_{i-1})$ because y remains on the stack for the entire derivation by assumption. Using Lemma 1 (a) we get $(q_1, \bot, a_1 \dots a_{i-2}) \vdash^{i-1} (q_{i-1}, \bot, \epsilon)$. Since $i - 1 < n$, we have $q_1 \in R_G(T)$ by hypothesis and, together with $q_i \in R_G(T)$ we then have $q_0 \in R_G(T)$. □

Note that the constructed alphabet is small but not necessarily minimal. It is not too difficult to see that the result can be strengthened to an alphabet of size $(1, 0, 1)$. For this, one simply merges push-transitions with internal transitions that can follow them.

Proposition 3. *The emptiness problem for VPA over an alphabet of size* $(1, 0, 1)$ *is P-hard under* AC^0*-reductions.*

However, this introduces additional nondeterminism. It is also possible to show this lower bound for deterministic VPA, at a small increase in the alphabet.

Corollary 4. *The emptiness problem for deterministic VPA over an alphabet of size* $(1, 2, 1)$ *is P-hard under* AC^0*-reductions.*

Proof. Note that the assumption about outdegree 2 of all nodes in V_1 can equally be made for all nodes in V_0. Now take two int-symbols b_1 and b_2, fix a total ordering $<$ on V and re-define the transitions in $\mathcal{A}_G$ with

$$\delta_{\mathsf{push}} := \big\{(x, \mathsf{a}, y_1, y_2) \bigm| x \in V_1,\ xE = \{y_1, y_2\} \text{ and } y_1 < y_2\big\}$$

$$\delta_{\mathsf{int}} := \big\{(x, \mathsf{b}_1, y_1), (x, \mathsf{b}_2, y_2) \bigm| x \in V_0,\ xE = \{y_1, y_2\} \text{ and } y_1 < y_2\big\}$$

The pop-transitions remain unchanged. Note that they were deterministic already, and so are the others as well now. The proof of Theorem 2 goes through with this definition as well. □

4. Remarks on the membership problem

The corresponding P-hardness proof for CFL immediately yields P-hardness of the (universal) membership problem, where the input consists of a word and a PDA or a CFG. The reason for this is the closure of CFL under homomorphisms. A homomorphism of particular interest is h_ϵ which maps every alphabet symbol to ϵ. Clearly, L is non-empty iff $\epsilon \in h_\epsilon(L)$. Since a PDA or CFG for $h_\epsilon(L)$ can easily be obtained from a PDA or CFG for L by replacing every alphabet symbol in the transition table or grammar rules by ϵ, we immediately get P-hardness of the question whether ϵ is contained in the language of a given PDA or CFG. This, however, is just a special case of the universal membership problem.

Now note that VPL is not closed under homomorphisms in general, in particular not under those that map certain alphabet symbols to ϵ. P-hardness of the universal membership problem for VPA is therefore not a consequence of the proof above. In fact, the problem is very unlikely to be P-hard, since P $\supseteq$ LOGCFL and it is not too difficult to see that the membership problem is included in LOGCFL: it is known that LOGCFL is characterised by nondeterministic auxiliary pushdown automata with a logarithmic worktape [6,7]. Such a machine can simulate a given visibly pushdown automaton on a given input word. It uses its stack to simulate the stack of the input automaton, and its auxiliary tape to search the transition table of the input automaton as well as remember the current position in the input word.

On the other hand, we do not believe that the universal membership problem for VPA is LOGCFL-hard. It remains to characterise the exact complexity of this problem.

Acknowledgement

We would like to thank Friedrich Otto for useful comments on this matter. In particular, the observation that the universal membership problem for VPA is in LOGCFL is his. We would also like to thank an anonymous referee for helpful comments, in particular for the observation that the reduction is in fact AC^0 and not just logspace.

References

[1] R. Alur, P. Madhusudan, Visibly pushdown languages, in: Proc. of the 36th Ann. ACM Symp. on Theory of Computing, STOC'04, ACM Press, New York, 2004, pp. 202–211.
[2] K. Mehlhorn, Pebbling mountain ranges and its application to DCFL-recognition, in: Proc. of the 7th Int. Coll. on Automata, Languages and Programming, ICALP'80, in: LNCS, vol. 85, Springer, 1980, pp. 422–435.
[3] S. Sippu, E. Soisalon-Soininen, Parsing Theory. Vol. I: Languages and Parsing, EATCS Monographs on Theoretical Computer Science, Springer, Berlin, 1988.
[4] M. Lange, H. Leiß, To CNF or not to CNF? An efficient yet presentable version of the CYK algorithm, Informatica Didactica 8 (2009).
[5] A.K. Chandra, D.C. Kozen, L.J. Stockmeyer, Alternation, Journal of the ACM 28 (1) (1981) 114–133.
[6] I.H. Sudborough, Time and tape bounded auxiliary pushdown automata, in: Proc. of the 6th Symp. on Mathematical Foundations of Computer Science, MFCS'77, in: LNCS, vol. 53, Springer, 1977, pp. 493–503.
[7] S. Cook, Characterizations of pushdown machines in terms of time-bounded computers, Journal of the ACM 18 (1) (1971) 4–18.