Pushdown Automata and Constant Height: Decidability and Bounds

Extended Abstract

Giovanni Pighizzini and Luca Prigioniero

Dipartimento di Informatica, Università degli Studi di Milano, Italy
{pighizzini,prigioniero}@di.unimi.it

Abstract. It cannot be decided whether a pushdown automaton accepts using constant pushdown height, with respect to the input length, or not. Furthermore, in the case of acceptance in constant height, the height cannot be bounded by any recursive function in the size of the description of the machine. In contrast, in the restricted case of pushdown automata over a one-letter input alphabet, i.e., unary pushdown automata, the above property becomes decidable. Moreover, if the height is bounded by a constant in the input length, then it is at most exponential with respect to the size of the description of the pushdown automaton. This bound cannot be reduced. Finally, if a unary pushdown automaton uses nonconstant height to accept, then the height should grow at least as the logarithm of the input length. This bound is optimal.

1 Introduction

The investigation of computational devices working with a limited amount of resources is a classical topic in automata theory. It is well known that by limiting the memory size of a device by some constant, the computational power of the resulting model cannot exceed that of finite automata. For instance, if we consider pushdown automata in which the maximum height of the pushdown is limited by some constant, the resulting devices, called *constant-height pushdown automata*, can recognize only regular languages. Despite their limited computational power, constant-height pushdown automata are interesting since they allow more succinct representations of regular languages than finite automata [4]. A natural generative counterpart of these devices are *non-self-embedding context-free grammars*, roughly context-free grammars without "true" recursion [3], which have been recently showed to be polynomially related in size to constant-height pushdown automata [6].

In this paper, we focus on standard pushdown automata, namely with an unrestricted pushdown store, that, however, are able to accept their inputs by making use only of a constant amount of the pushdown store. More precisely,

we say that a pushdown automaton $\mathcal{M}$ *accepts in constant height* h, for some given h, if for each word in the language accepted by $\mathcal{M}$ there exists one accepting computation in which the maximum height reached by the store is bounded by h. Notice that this does not prevent the existence of accepting or rejecting computations using an unbounded pushdown height.

It is a simple observation that a pushdown automaton $\mathcal{M}$ accepting in constant height h can be converted into an equivalent constant-height pushdown automaton: in any configuration it is enough to keep track of the current height in order to stop and reject when a computation tries to exceed the height limit. The description of the resulting constant-height pushdown automaton has size polynomial in h and in the size of the description of $\mathcal{M}$.

While studying these size relationships, we tried to understand *how large can h be with respect to the size of the description of $\mathcal{M}$*. We discovered that h can be arbitrarily large. Indeed, there is no recursive function bounding the maximal height reached by the pushdown store in a pushdown automaton accepting in constant height, with respect to the size of its description. Moreover, it cannot be decided if a pushdown automaton accepts in constant height.

In the second part of the paper we restrict the attention to the case of pushdown automata with a one-letter input alphabet, namely *unary pushdown automata*. By studying the structure of the computations of these devices, we were able to prove that, in contrast to the general case, it can be decided whether or not they accept in constant height. Furthermore, if a unary pushdown automaton $\mathcal{M}$ accepts in height h, constant with respect to the input length, then h can be bounded by an exponential function in the size of $\mathcal{M}$. This bound cannot be reduced.

In the final part of the paper, we consider pushdown automata that accept using height which is not constant in the input length. Our aim is to investigate how the pushdown height grows. In particular, we want to know if there exists a minimum growth of the pushdown height, with respect to the length of the input, when it is not constant. The answer to this question is already known and it derives from results on Turing machines: the height of the store should grow at least as a double logarithmic function [1]. This lower bound cannot be increased, because a matching upper bound which has been recently obtained in [2]. In the unary case this lower bound is logarithmic and it cannot be further increased.

This work is an extended abstract of the conference paper [10].

2 Preliminaries

We assume the reader familiar with the standard notions from formal language and automata theory as presented in classical textbooks, e.g., [8]. As usual, the cardinality of a set S is denoted by $\#S$, the length of a string x is denoted by $|x|$, and the empty string is denoted by ε.

A *pushdown automaton* (PDA, for short) is a tuple $\mathcal{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ where Q is the finite *set of states*, Σ is the *input alphabet*, Γ is the *pushdown*

alphabet, $q_0 \in Q$ is the *initial state*, $Z_0 \in \Gamma$ is the *start symbol*, $F \subseteq Q$ is the set of *final states*, and δ is the *transition function*.

Without loss of generality, we make the following assumptions about PDAs:

1. at the start of the computation the pushdown store contains only the start symbol Z_0, being at height 0; this symbol is never pushed on or popped off the stack;
2. the input is accepted if and only if the automaton reaches a final state, the pushdown store contains only Z_0 and all the input has been scanned;
3. every `push` adds exactly one symbol on the stack.

The height of a PDA $\mathcal{M}$ (i.e., *pushdown height*) in a given configuration is the number of symbols in the pushdown store *besides* the start symbol Z_0. Hence, in the initial and in the accepting configurations the height is 0. The height in a computation $\mathcal{C}$ is the maximum height reached in the configurations occurring in $\mathcal{C}$.

We say that $\mathcal{M}$ uses height $h(x)$ on an accepted input $x \in \Sigma^*$ if and only if $h(x)$ is the minimum pushdown height necessary to accept x, namely, there exists a computation accepting x using pushdown height $h(x)$, and no computations accepting x using height less than $h(x)$. Moreover, if x is rejected then $h(x) = 0$. To study pushdown height with respect to input lengths, we consider the worst case among all possible inputs of the same length. Hence, we define $h(n) = \max\{h(x) \mid x \in \Sigma^*, |x| = n\}$. When there is a constant H such that, for each n, $h(n)$ is bounded by H, we say that $\mathcal{M}$ *accepts in constant height*. Each PDA accepting in constant height can be easily transformed into an equivalent finite automaton. So the language accepted by it is regular.

In the following, by the *size of a* PDA we mean the length of its description. Notice that for each PDA in the above-defined form, over a fixed input alphabet Σ, the size is polynomial in the cardinalities of the set of states and of the pushdown alphabet. If we consider PDAs in different form, as for instance that in [8], to define the size we need to take into account also the number of symbols that can be pushed on the stack in one move. However, PDAs in that form can be turned in the form we consider here, with a polynomial increase in size and by preserving the property of being constant height.

3 Undecidability and Non-Recursive Bounds

We consider the problem of deciding if a PDA accepts in constant height. Using a technique introduced in [7], based on suitable encodings of single-tape Turing machine computations, we have proved the following result:

Theorem 1. *It is undecidable whether a* PDA *accepts in constant height.*

We also studied the problem of relating the maximal height h reached by a PDA $\mathcal{M}$ accepting in constant height to its size. Adapting an argument presented in [9] to prove non recursive trade-offs between the size of PDAs accepting regular languages and the number of states of equivalent automata, we proved the following "negative" result:

Theorem 2. *For any recursive function $f : \mathbb{N} \to \mathbb{N}$ and for infinitely many integers n there exists a* PDA *of size n accepting in constant height $H(n)$, where $H(n)$ cannot be bounded by $f(n)$.*[1]

With the same argument, we also obtained:

Theorem 3. *There is no recursive function bounding the size blowup from* PDA*s accepting in constant height to finite automata.*

4 Restricting to the Unary Case

In the restricted case of unary PDAs, the "negative" results in Section 3 do not hold.

Theorem 4. *It is decidable whether a unary* PDA *accepts in constant height.*

We will give an informal outline of the proof. To this aim, some technical notions are useful. Let $\mathcal{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ be a fixed PDA and let $s = 1 + (\#Q)^2 \cdot \#\Gamma$.

A *surface pair* is defined by a state $q \in Q$ and a symbol $A \in \Gamma$, and it is denoted by $[qA]$. The surface pair in a given configuration is defined by the current state and the topmost stack symbol, namely the only part of the stack which is relevant in order to decide the next move. A *surface triple* is defined by two states $q, p \in Q$ and a symbol $A \in \Gamma$, and it is denoted by $[qAp]$. Surface triples are used to study computation paths starting and ending at the same pushdown height and that do not go below that height in between. In particular, a computation $\mathcal{C}$ of such a form is called $[qAp]$-*computation* if q and p are the states at the beginning and at the end of $\mathcal{C}$, respectively, and A is the symbol at the top of the stack at the beginning (and at the end) of $\mathcal{C}$.

A *horizontal loop* on a surface pair $[qA]$ is any $[qAq]$-computation which consumes *at least one input symbol*.

If a $[qAp]$-computation $\mathcal{C}$ contains a proper $[qAp]$-computation $\mathcal{C}'$ (the surface triple is the *same*), which starts with stack *higher* than at the beginning of $\mathcal{C}$, then the pair $(\mathcal{X}, \mathcal{Y})$ where $\mathcal{X}$ is the prefix of $\mathcal{C}$ ending in the first configuration of $\mathcal{C}'$, and $\mathcal{Y}$ is the suffix of $\mathcal{C}$ starting from the last configuration of $\mathcal{C}'$, is called *vertical loop*. Notice that at the end of $\mathcal{X}$ a nonempty string $A\alpha$ is on the pushdown above the occurrence of A which was on the top at the beginning of $\mathcal{C}$, and such a string is popped off during $\mathcal{Y}$.

We now sketch the proof of Theorem 4. Informally, any accepting computation on a sufficiently long input should contain horizontal or vertical loops. The use of vertical loops could lead to computations using unbounded height. Hence, to study how long strings are accepted, it is useful to consider the following two languages L_h and L_v, where $L = L_h \cup L_v$ is the language accepted by $\mathcal{M}$:

- L_h is the set of strings accepted by the computations of $\mathcal{M}$ which visit at least one surface pair having a horizontal loop.

[1] Notice that here $H(n)$ is a function of the size of the PDA and *not* of the input.

– L_v is the set of strings accepted by the computations of $\mathcal{M}$ which visit only surface pairs that do not have horizontal loops.

According to the following lemma, for every string a^ℓ in L_h, namely for which there exists an accepting computation visiting a surface pair having a horizontal loop, there is another accepting computation for the same input in which almost all occurrences of the vertical loops are replaced by occurrences of such horizontal loop. The number of vertical loops which remain in the resulting computation is bounded by a constant. As a consequence, the amount of pushdown store sufficient to accept a^ℓ is also bounded by a constant. This allows to bound by a constant the amount of pushdown store sufficient to accept a^ℓ.

Lemma 1. *For any accepting computation on input a^ℓ which visits a surface pair having a horizontal loop there exists another accepting computation on a^ℓ which uses pushdown height at most $2^{O(s^2)}$.*

In contrast, when all accepting computations on a long string a^ℓ do not visit any surface pair having a horizontal loop, vertical loops and an increasing of the stack cannot be avoided. Hence, deciding if $\mathcal{M}$ works in constant height is equivalent to decide if the cardinality of the set $L_v \setminus L_h$ is finite. We notice that by modifying $\mathcal{M}$ we can obtain two PDAs $\mathcal{M}_v$ and $\mathcal{M}_h$ accepting languages L_v and L_h, respectively. So these two languages are context-free. Furthemore, since the alphabet is unary, they are also regular [5]. So the finiteness of their difference is decidable. This allows to decide if $\mathcal{M}$ accepts in constant height.

We can also evaluate the height of the stack used to accept the strings in $L_v \setminus L_h$, in case $\mathcal{M}$ accepts in constant height. According to Corollary 2 in [11], the PDAs $\mathcal{M}_v$ and $\mathcal{M}_h$ can be converted into equivalent DFAs with $2^{O(s^2)}$ states. Hence $L_v \setminus L_h$, which is finite, is accepted by a DFA with less than $2^{O(s^2)}$ states. So, the longest string in $L_v \setminus L_h$ has length at most $2^{O(s^2)}$. This implies that each string in $L_v \setminus L_h$ is accepted by $\mathcal{M}$ using height $2^{O(s^2)}$. From this discussion, we obtain the following result which gives an exponential upper bound for the maximum stack height, with respect to the size of $\mathcal{M}$:

Theorem 5. *A unary* PDA *accepts in constant height if and only if it accepts in height bounded by $2^{O(s^2)}$.*

Such an exponential bound cannot be reduced:

Theorem 6. *For each integer $k > 0$ there exists a* PDA *having a size polynomial in k and accepting in height which is constant with respect to the input length but exponential in k.*

In conclusion, we turn our attention to PDAs that accept using height which is not constant in the input length. It is known that in this case the height of the pushdown store should grow at least as the function $\log \log n$, with respect to the input length n [1]. Furthermore, this lower bound is optimal [2]. In the unary case the optimal bound increases to a logarithmic function.

Theorem 7. *Let $\mathcal{M}$ be a unary* PDA *using height $h(n)$. Then either $h(n)$ is bounded by a constant or there exists $c > 0$ such that $h(n) \geq c \log n$ infinitely often. This bound is tight.*

References

1. Alberts, M.: Space complexity of alternating Turing machines. In: Budach, L. (ed.) Fundamentals of Computation Theory, FCT '85, Cottbus, GDR, September 9-13, 1985. Lecture Notes in Computer Science, vol. 199, pp. 1–7. Springer (1985). https://doi.org/10.1007/BFb0028785
2. Bednárová, Z., Geffert, V., Reinhardt, K., Yakaryilmaz, A.: New results on the minimum amount of useful space. Int. J. Found. Comput. Sci. **27**(2), 259–282 (2016). https://doi.org/10.1142/S0129054116400098
3. Chomsky, N.: A note on phrase structure grammars. Information and Control **2**(4), 393–395 (1959). https://doi.org/10.1016/S0019-9958(59)80017-6
4. Geffert, V., Mereghetti, C., Palano, B.: More concise representation of regular languages by automata and regular expressions. Inf. Comput. **208**(4), 385–394 (2010). https://doi.org/10.1016/j.ic.2010.01.002
5. Ginsburg, S., Rice, H.G.: Two families of languages related to ALGOL. J. ACM **9**(3), 350–371 (1962). https://doi.org/10.1145/321127.321132
6. Guillon, B., Pighizzini, G., Prigioniero, L.: Non-self-embedding grammars, constant-height pushdown automata, and limited automata. In: Câmpeanu, C. (ed.) CIAA 2018, Proceedings. Lecture Notes in Computer Science, vol. 10977, pp. 186–197. Springer (2018). https://doi.org/10.1007/978-3-319-94812-6_16
7. Hartmanis, J.: Context-free languages and Turing machine computations. In: Mathematical Aspects of Computer Science. Proceedings of Symposia in Applied Mathematics, vol. 19, pp. 42–51. American Mathematical Society (1967)
8. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
9. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: 12th Annual Symposium on Switching and Automata Theory, East Lansing, Michigan, USA, October 13-15, 1971. pp. 188–191. IEEE Computer Society (1971). https://doi.org/10.1109/SWAT.1971.11
10. Pighizzini, G., Prigioniero, L.: Pushdown automata and constant height: Decidability and bounds. In: Descriptional Complexity of Formal Systems, DCFS 2019, Proceedings. Lecture Notes in Computer Science, vol. 11612, pp. 260–271. Springer (2019). https://doi.org/10.1007/978-3-030-23247-4_20
11. Pighizzini, G., Shallit, J., Wang, M.: Unary context-free grammars and pushdown automata, descriptional complexity and auxiliary space lower bounds. J. Comput. Syst. Sci. **65**(2), 393–414 (2002). https://doi.org/10.1006/jcss.2002.1855