Descriptional complexity of push down automata[*]

Lukáš Kiss and Branislav Rovan

Comenius University in Bratislava,
Faculty of Mathematics, Physics and Informatics

Abstract: We study descriptional complexity of push down automata (PDA) accepting regular languages and context free languages. We have shown that the number of states of a PDA accepting a regular language can be smaller than that of a corresponding finite state automaton by utilizing stack symbols. No complexity function having desirable properties and combining the number of states and the number of stack symbols exists. We therefore study the number of stack symbols complexity in the family of one state PDA and the state complexity in the family of PDA using at most two stack symbols. We exhibit two infinite sequences of regular languages and prove tight bounds on their complexity using the above families of PDA. We also prove upper and lower bounds on the complexity of sequences of context-free languages and analyze the impact of the mode of acceptance.

Introduction

Although measuring complexity of problem solution by time and space requirements is most frequently used, the descriptional complexity is achieving more interest in recent decades. The complexity of a problem (given by a language L) solution is measured by a complexity of a device (e.g., an automaton) defining L. Measuring complexity of finite automata by the number of states dates back to the early days of automata theory. Majority of descriptional complexity of automata research concerns finite automata (see, e.g., survey papers [1], [2], [3]). The descriptional complexity of PDA was addressed by Goldstine at al.[4] and [5] where relation of the number of states and the number of stack symbols was studied. Later particular types of PDA were considered (see, e.g., surveys [6], [7]) In connection to the research on usefulness of information (see, e.g., Rovan and Sadovsky [8]) measuring complexity of PDA as a more powerful model became important (see the deterministic PDA case in Labáth and Rovan [9]).

The number of states was a natural descriptional complexity measure for finite state automata (FSA). In case of PDA there are two measures at hand – the number of states measure and the number of stack symbols measure. Both states and stack symbols are depended on each other. Goldstine, Price and Wotschke showed that there exists a transformation for any PDA using n states and p stack symbols to an equivalent PDA reducing the number of states to any desired number $n_0 \geq 1$ [4] by increasing the number of stack symbols. No complexity measure for PDA having desirable properties and combining the number of states and the number of stack symbols exists. Therefore, we consider two subfamilies of PDA, *one state PDA* and *max two stack symbols PDA*. Note that each of them defines the whole family of context-free languages. In the *one state PDA* subfamily, we fix the number of states to one and use the number of stack symbols as the measure. In the second subfamily, we fix the number of stack symbols to at most two and measure the number of states.

In this paper, we study descriptional complexity of PDA accepting regular and context-free languages. We exhibit some infinite sequences of languages allowing us to prove some lower and upper bounds on their descriptional complexity using the above subfamilies of PDA. We also show that acceptance mode can have impact on the descriptional complexity. The results presented are based on the Master Thesis of Lukáš Kiss [10].

1 Preliminaries and Notation

We shall assume basic knowledge and notation in formal languages theory. We mention some of the notation we shall use. The length of a word w is denoted by $|w|$. We use ε to denote the empty word. The cardinality of a set S is denoted by $|S|$. A nondeterministic push down automaton (PDA) is a 7-tuple $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ with the usual meaning of its components. Note that a PDA can in one computation step replace the top of the stack symbol by a word. Moreover, the PDA cannot move with the empty stack. The language accepted by a PDA A accepting by empty stack is denoted by $N(A)$ and the language accepted by a PDA A accepting by final state is denoted by $L(A)$. The family of context-free languages is denoted by $\mathscr{L}_{CF}$. We shall also use the operation shuffle. Given an alphabet Σ, the *Shuffle* of two languages $L_1, L_2 \subseteq \Sigma^*$ is $Shuf(L_1, L_2) = \{w | \exists n \in N, u_1, \ldots, u_n, v_1, \ldots, v_n \in \Sigma^*$; such that $w = u_1v_1u_2v_2 \ldots u_nv_n \wedge u_1 \ldots u_n \in L_1 \wedge v_1 \ldots v_n \in L_2\}$.

[*]This research has been supported in part by the grant 1/0601/20 of the Slovak Scientific Grant Agency VEGA.

2 Descriptional Complexity of PDA

In case of finite automata, the number of states is the most natural and mostly used descriptional complexity measure. In case of push-down automata (PDA) the size of the stack alphabet is another important parameter. It is known ([4], [5]) that it is possible to reduce the number of stack symbols by increasing the number of states and vice versa. It is also known (see, e.g., [11]) that one state suffices to define any context-free language and similarly two stack symbols suffice to define any context-free language. Thus it is natural to look for a descriptional complexity measure for PDA that would "combine" the two parameters – the number of states n and the number of stack symbols (the size of the stack alphabet) p. Just like in the case of deterministic PDA [9] it can be shown that a function assigning natural numbers to pairs (n,p) and having some desirable properties [1] does not exist.

Let us consider some subfamilies of push-down automata.

Notation 1. *$PDA(n,p)$ is the family of push down automata using at most n states and at most p stack symbols.*

Since there is no function on pairs (n,p) assigning the same complexity to two minimal automata for the same language, we shall concentrate on the following two 'extreme' subfamilies of PDA defining the whole family of context-free languages:

- $PDA(1,p)$ - the subfamily of one state PDA using the number of stack symbols to measure the complexity.
- $PDA(n,2)$ - the subfamily of all PDA using at most two stack symbols, measuring complexity by the number of states.

For each of these subfamilies of PDA, we define a minimal complexity of a given context-free language L as follows:

Definition 1. *Let L be a context-free language. The stack symbol complexity of L, denoted by $\Gamma c(L)$, is the smallest number of stack symbols of any A in $PDA(1,p)$ that accepts L.*

Definition 2. *Let L be a context-free language. The state complexity of L, denoted by $Qc(L)$, is the smallest number of states of any A in $PDA(n,2)$ that accepts L.*

In what follows, we shall use constructions and results on reducing the number of stack symbols by increasing the number of states in [5] and on reducing the number of states by increasing the number of stack symbols in [4].

[1]Preserving the natural partial order on pairs (n,p) based on component wise comparison and assigning the same value to pairs (n_1,p_1) and (n_2,p_2) corresponding to two minimal PDA's for the same language.

3 Push Down Automata on Regular Languages

We shall first consider PDA on regular languages. We shall consider the question whether and to what extent the number of states of a finite state automaton (FSA) can be reduced by using stack with stack alphabet of certain size. We shall consider this question for both acceptance modes of PDA. Next, we shall prove some lower bounds for PDA in the two particular subfamilies of PDA mentioned above.

3.1 Saving States by Adding Stack

Given an arbitrary finite state automaton, we shall construct an equivalent push down automaton with fewer states. It is easy to see that by allowing any number of stack symbols it is possible to have a one state PDA. Suppose, we allow a particular number of stack symbols. How much can the number of states of an equivalent PDA be reduced compared to the number of states of a given finite state automaton?

Theorem 1. *For any n state FSA A_1 there exists a PDA A_2 with $\lceil \frac{n}{p} \rceil$ states and p stack symbols such that $N(A_2) = L(A_1)$*

Proof. The push down automaton represents each state of the finite state automaton A_1 by a coding consisting of a state and a stack symbol on the top of the stack. The stack of A_2 shall contain at most one symbol. Each transition of the automaton A_1 from a state q to s is represented by a corresponding change of the state and the top stack symbol in the automaton A_2. Each accepting state of A_1 has also a corresponding coding s and Z. For each such coding corresponding to some accepting state of A_1 can A_2, in addition to the simulation of a step of A_1, nondeterministically decide to pop this stack symbol and thus empty its stack. If the stack is emptied before the automaton A_2 has finished processing the input word, A_2 shall halt, otherwise A_2 accepts the input word if and only if the automaton A_1 accepts. $\square$

The construction in the proof of Theorem 1 can be easily modified for acceptance by accepting state. It suffices to replace the possibility of popping the stack by a transition to a new accepting state. We thus have the following lemma.

Lemma 1. *For each n state FSA A_1 there exists a PDA A_2 with $\lceil \frac{n}{p} \rceil + 1$ states and p stack symbols such that $L(A_2) = L(A_1)$*

This upper bound construction introduces one more state, the accepting state, compared to the previous upper bound construction, but not in all cases the additional accepting state is necessary [2]. The

[2]Consider, e.g., the languages Σ^* or $\emptyset$.

question of necessity of the new state for a language is by itself an interesting independent problem.

It may seem that there is no significant impact of the choice of the accepting mode. By considering some lower bounds in the next subsection we shall see that this is not necessarily a case.

3.2 Lower Bounds for PDA on Regular Languages

In this subsection, we shall consider lower bounds on the complexity of particular regular languages considering PDA in the two above mentioned subfamilies of PDA, namely $PDA(1,p)$ and $PDA(n,2)$. We shall show that for the empty stack acceptance the upper bound construction in the proof of Theorem 1 is optimal for one state PDA.

We shall now introduce two particular sequences of regular languages to be used in our lower bound proofs.

Notation 2. *Let $a_1,\dots,a_n$ be distinct symbols for any $n \geq 1$. Let*

- $L_1[n] = a_1^* a_2^* \dots a_n^*$
- $L_2[n] = \{a_1^{kn} | k \geq 0\}$

It is easy to see that the state complexity for both $L_1[n]$ and $L_2[n]$ on finite state automata is n.

The complexity of $L_1[n]$

We shall show that the stack symbols complexity of $L_1[n]$ on one state PDA is n.

Theorem 2. $\Gamma c(L_1[n]) = n, \forall n \in N$.

Proof. We shall show that $\Gamma c(L_1[n]) \leq n$ by constructing a push-down automaton A_n. The automaton A_n uses the top stack symbol Z_i as a "state". Its stack will contain at most one symbol. If Z_i is on the top of the stack, the automaton knows that it already has processed all input symbols $a_1,\dots,a_{i-1}$. The automaton can pop the top stack symbol at any time and empty its stack.

We shall prove $\Gamma c(L_1[n]) \geq n$ by contradiction.

Let A_n in $PDA(1,p)$ be an automaton accepting the language $L_1[n]$ for $p < n$. By the pigeon hole principle, there exist two input symbols a_i and a_j such that $i < j$, on which the automaton A_n does a computation step on the same stack symbol Z during an accepting computation on a word $w = a_1^{m_1} a_2^{m_2} \dots a_n^{m_n}$ for $m_1,\dots,m_n \geq 2p$.

Suppose the automaton A_n during an accepting computation on w pushes γ_i on the input symbol a_i and γ_j on the input symbol a_j, having the stack symbol Z on the top of the stack in each case. The automaton A_n accepts the word w by empty stack. Therefore, there exist some sequences of symbols u_i and u_j, on which A_n removes γ_i and γ_j from the stack[3].

Let $k, 1 \leq k \leq m_j$, be such that the automaton A_n has the stack symbol Z on the top of the stack after processing $a_1^{m_1} a_2^{m_2} \dots a_i^{m_i} \dots a_j^k$. Using the accepting computation of A_n on w we shall modify w and the accepting computation so that an accepting computation on a word not in $L_1[n]$ is obtained. Following $a_1^{m_1} a_2^{m_2} \dots a_i^{m_i} \dots a_j^k$ the automaton A_n can read the input symbol a_i and replace the stack symbol Z on the top by γ_i. Now, the word u_i can follow on the input. After processing this next part of the input word, u_i, the automaton can remove the γ_i and leave a word γ_{fin}[4] on the stack. Note that γ_{fin} also appears on the stack during the accepting computation of A_n on w. Thus there exists some word v, on which the automaton A_n removes γ_{fin} from the stack and accepts the word w by empty stack. Therefore A_n also accepts $\hat{w} = a_1^{m_1} a_2^{m_2} \dots a_i^{m_i} \dots a_j^k \mathbf{a_i} u_i v \notin L_1[n]$. □

The complexity of $L_2[n]$

We could use the upper bound construction in the proof of Theorem 1 on the minimal finite state automaton for the language[5] $L_2[n]$. However, this does not result in a minimal push down automaton in our complexity measures. Instead of using the combination of a state and a top stack symbol to represent the state of the minimal FSA, the minimal push down automaton uses its stack as a counter.

Before we present the construction and the proof, we shall prove that no push down automaton with one state and one stack symbol can accept $L_2[n]$ for [6] $n \geq 2$.

Lemma 2. *There does not exist a PDA using one state and one stack symbol accepting $L_2[n], n \geq 2$.*

Proof. By contradiction, let there exist a push down automaton A using one state q and one stack symbol Z_0 accepting $L_2[n], n \geq 2$. We know that A must use the empty stack acceptance mode. Therefore, A has to pop Z_0 from the stack on the input symbol a_1 or ε.

If A pops Z_0 on the input symbol a_1 then A accepts the word $a_1 \notin L_2[n]$ for any $n \geq 2$.

If A pops Z_0 on ε and pushes on a_1 a word $\gamma \in Z_0^*$ on the stack then A accepts the word $a_1 \notin L_2[n]$ for any $n \geq 2$.

□

Now, we are ready to prove matching upper and lower bounds for the language $L_2[n]$.

[3] If $\gamma_i = \varepsilon$ then $u_i = \varepsilon$

[4] Where $\gamma_{fin} Z$ was the content of the stack after processing $a_1^{m_1} a_2^{m_2} \dots a_i^{m_i} \dots a_j^k$.

[5] The minimal finite state automaton requires exactly n states.

[6] A finite state automaton using one state can accept $L = \{a_1\}^* = L_2[n]$, for $n = 1$.

Theorem 3. $\Gamma c(L_2[n]) = 2$*, for any* $n \geq 2$.

Proof. We shall prove that $\Gamma c(L_2[n]) \leq 2$ by constructing A in $PDA(1,2)$ accepting the language $L_2[n]$.

The automaton uses the bottom of stack symbol Z_0 to indicate the starting point of a block and Z_1 as a representation of a_1 in the stack. On Z_0 it can nondeterministically decide to start processing a next block of n symbols a_1 on the input or empty the stack. The automaton pushes n sized block of Z_1 on the stack and for each a_1 it pops Z_1 from the stack until it reaches the symbol Z_0. Then the process repeats.

The fact that $\Gamma c(L_2[n]) \geq 2$ has been already shown in the Lemma 2. $\square$

Moreover, the construction in the proof of this theorem gives state complexity of each $L_2[n]$ when considering PDA in $PDA(n,2)$.

Corollary 1. $Qc(L_2[n]) = 1$*, for any* $n \geq 1$.

The previous construction results in a minimal push down automaton that accepts $L_2[n]$ by empty stack. The question arises, how many states are used by a minimal push down automaton using two stack symbols accepting the language $L_2[n]$ by final state? Clearly, it can not be one state. Therefore, the minimal PDA needs at least two states. The next theorem shows that two states are not only necessary, but also sufficient for two stack symbols PDA accepting $L_2[n]$ by accepting state.

Theorem 4. *Let A be a push down automaton using two stack symbols accepting the language $L_2[n], n \geq 2$. Then two states are necessary and sufficient to accept $L_2[n]$ by final state.*

Proof. We first show that two states are sufficient to accept the language $L_2[n]$ by final state by constructing a PDA A. The construction of the minimal PDA is similar to that in the proof of the Theorem 3. Instead of popping nondeterministically the symbol Z_0 from the stack and then accepting by empty stack, this automaton can move nondeterministically to the new accepting state.

We now show (by contradiction) that two states are necessary to accept the language $L_2[n]$ by final state using just 2 stack symbols. Let us assume that an automaton A using one state and two stack symbols exists. Hence, it has just one state, then this state has to be the accepting state. Since this automaton accepts the word a_1^n, it has some transitions on a_1. Therefore, the automaton also accepts the word $a_1 \notin L_2[n]$. $\square$

3.3 The complexity on one stack symbol PDA

In the previous part, we proved that any push down automaton in $PDA(1,p)$ needs at least two stack symbols to accept the language $L_2[n]$ for $n \geq 2$. Let us study one stack symbol PDA accepting the language $L_2[n]$, i.e., the PDA in $PDA(n,1)$. We shall show that there exists a minimal PDA using one stack symbol and two states accepting the language $L_2[n]$ by empty stack and a minimal PDA using one stack symbol and n states accepting the same language by final state.

We can conclude that the type of acceptance mode does have an effect on the complexity of a minimal PDA for a given language.

Accepting by empty stack

Let us consider push down automata using one stack symbol and accepting by empty stack. In this setup, we can construct a sequence of PDA $A_1, A_2, \ldots$ for the sequence of languages $L_2[1], L_2[2], \ldots$ such that each PDA shall use one stack symbol and two states. We shall prove this in the next Theorem.

Theorem 5. *Let $n \geq 2$. The minimal automaton $A_n \in PDA(n,1)$ accepting the language $L_2[n]$ by empty stack has two states.*

Proof. The automaton A_n guesses the number of blocks of length n in the initial state q_0 by pushing blocks of the stack symbol Z_0 on the stack in ε moves. It can nondeterministically move to the second state q_1 in which it compares the number of stack symbols in the stack to the number of symbols a_1 on the input. Note that, the automaton pushes only $n-1$ symbols on the top of the stack when it changes state.

By Lemma 2, the automaton accepting $L_2[n], n \geq 2$ using one state and one stack symbol does not exists. Therefore, the automaton A_n is minimal. $\square$

Accepting by state

Finally, we focus on the push down automata using one stack symbol accepting by final state. Even though, the previous automata had just one stack symbol, we were able to construct a sequence of automata accepting the languages $L_2[n]$ each using two states only. Let us show that this is not the case for automata using one stack symbol and accepting by final state.

Theorem 6. *Let $n \geq 2$. The minimal automaton $A_n \in PDA(n,1)$ accepting the language $L_2[n]$ by accepting state has n states.*

Proof. To show the upper bound, i.e., that n states are enough it suffices to consider the push down automaton which behaves in the same way as the finite state automaton accepting $L_2[n]$, ignoring the stack entirely.

We shall show (by contradiction) that the number of states has to be at least n. Let A be a push down automaton using one stack symbol and $n-1$ states accepting the language $L_2[n]$. Thus A accepts $w = a_1^m$, n divides m, for some $m \geq 2n$. During the accepting computation of the automaton A on the input word w some state q will repeat.

Consider the part of the computation between the two occurrences of the state q where some number j, $1 \leq j \leq n-1$ of symbols a_1 was read. Suppose the size of the stack was not increased during this part of the computation. By leaving out this part of the computation we clearly obtain an accepting computation on the word $w' = a_1^{m-j} \notin L_2[n]$. Similarly, in case the size of the stack was increased during this part of the computation, by repeating this part of the computation we clearly obtain an accepting computation on the word $w" = a_1^{m+j} \notin L_2[n]$. (Note that by considering the two cases we ensured that the computation on the modified word does not block because of the empty stack.)

$\square$

4 PDA on Nonregular Context Free Languages

In this section, we shall analyze complexity of (nonregular) context free languages using both subfamilies of PDA considered. In $PDA(1,p)$, we shall show lower and upper bounds for a particular sequence of languages and in $PDA(n,2)$, we shall analyze the relation of complexities for the two modes of acceptance. Furthermore we shall show some upper bounds for a particular sequence of langauges.

4.1 The Number of Stack Symbols

Let us consider the following sequence of context free languages.

Notation 3. *Let $a_1,\dots,a_p,b_1,\dots,b_p$ be distinct symbols for any $p \geq 1$. Let*

$$\Sigma_p = \{a_1,\dots,a_p,b_1,\dots,b_p\}$$
$$L_p = \{w(h(w))^R | w \in \{a_1,a_2,\dots,a_p\}^*\}$$

where h is the homomorphism defined by $h(a_i) = b_i$, for each i, $1 \leq i \leq p$.

We can easily see that any PDA using one state has to use empty stack acceptance mode to accept the language L_p. This shows that the empty stack acceptance mode and the final state acceptance mode do not have the same power in $PDA(1,p)$.

Our intention is to prove that the number of stack symbols complexity of the language L_p for one state PDA is exactly $p+1$. To prove the lower bound, we shall first prove several lemmas. Each lemma exhibits a property, each push down automaton accepting L_p has to have.

Lemma 3. *Let A in $PDA(1,i)$ be an automaton accepting the language L_p, where $p,i \in N$. Let $x \in \Sigma_p$ be an input symbol. Then in each accepting computation on $w \in L_p$, $w = ux^jv, j \geq 1, u,v \in \Sigma_p^*$, A has to modify the stack while reading x.*

Proof. Let $x \in \Sigma_p$ be a symbol, which does not modify the stack in an accepting computation and let A accept $w = ux^jv \in L_p, j \geq 1, u,v \in \Sigma_p^*$. Then A also accepts $\hat{w} = ux^{j+1}v \notin L_p$. $\square$

Lemma 4. *Let A in $PDA(1,p)$ be a minimal automaton accepting the language L_p, where $p \in N$. Then $\forall i, 1 \leq i \leq p\ \exists Z \in \Gamma$ such that $(q_0,\varepsilon) \in \delta(q_0,b_i,Z)$.*

Proof. Let us consider words $w_1 = a_1^m b_1^m,\dots,w_p = a_1^m b_1^m$ in L_p for sufficiently large m. Note that the size of the stack after reading the a-part of the words cannot be bounded. (Otherwise, for sufficiently large m some stack content would repeat. Leaving out this part of the input and computation would lead to an accepting computation on a word with smaller number of a's and unchanged number of b's.) Thus, after reading the a-part of the words $w_1,\dots,w_p$ the stack will contain some (large) word in Γ^+ which the next part of the accepting computation has to remove while reading the b-part of the word.[7]

Now, suppose that to the contrary of the statement of lemma there exists some b_k such that A does not pop any stack symbol from the stack on b_k. Formally, $\forall Z \in \Gamma$ it holds that $(q_0,\varepsilon) \notin \delta(q_0,b_k,Z)$.

Let us analyze the accepting computation on the word w_k. Since A has to empty the stack while reading the b-part of w_k and cannot pop the stack symbols while reading b_k, the symbols on the stack have to be removed in ε-transitions. Thus there are at least two distinct symbols Z_1 used in the b_k transition and Z_2 used in the ε-transition in Γ used in the b-part of the accepting computation on w_k. (Having these symbols equal would enable accepting computation on a word with fewer b_k's.) Neither of these symbols can be popped from the stack by some b_i, $i \neq k$ since this would allow to modify accepting computations on the words $w_1,\dots,w_p$ to accepting computations on words not in L_p. (A detailed analysis of the accepting computations on the words the words $w_1,\dots,w_p$ shows that p symbols are not enough when b_k cannot pop any symbol in Γ.) $\square$

Lemma 5. *Let A in $PDA(1,p)$ be an automaton accepting the language L_p, where $p \in N$. Suppose $(q_0,\varepsilon) \in \delta(q_o,b_i,Z)$ and $(q_0,\varepsilon) \in \delta(q_o,b_j,\hat{Z})$ for $i \neq j$. Then $Z \neq \hat{Z}$.*

Proof. Suppose that one state push down automaton A accepting the language L_p pops on b_i and b_j the same stack symbol Z from the stack and let A accepts $w = a_i^m b_i^m$. Then A accepts $\hat{w} = a_i^m b_j^m \notin L_p$. $\square$

Combining the previous lemmas 4 and 5, we can infer that one state push down automaton needs at least p stack symbols to accept L_p. The next theorem shows that in fact $p+1$ stack symbols are required.

[7]The acceptance is by empty stack.

Before we show the proof of a lower bound and construct an upper bound, we shall infer some properties about the δ function from the previous lemmas.

Corollary 2. *Let $A = (\{q_0\}, \{a_1, \dots, a_p, b_1, \dots, b_p\}, \{Z_1, Z_2, \dots, Z_p\}, \delta, q_0, Z_i, \emptyset)$ in $PDA(1, p)$ be an automaton accepting by empty stack the language L_p, where $p \in N$. Then there exists some permutation s on $\{1, \dots, p\}$, such that $(q_0, \varepsilon) \in \delta(q_0, b_i, Z_{s(i)})$, where Z_i is a stack symbol used by the automaton A.*

We are now ready to present the complexity of the languages L_p.

Theorem 7. $\Gamma c(L_p) = p + 1$, $\forall p \in N$

Proof. We shall show that $\Gamma c(L_p) \leq p + 1$ by constructing an automaton A in $PDA(1, p+1)$ accepting L_p by empty stack.

The automaton pushes on each input symbol a_i the stack symbol Z_i onto the stack. The automaton pops the Z_i from the stack on the input symbol b_i. The initial stack symbol is Z_0 and is kept on the top of the stack while reading a_i symbols. On any input symbol a_i, it can nondeterministically change the top stack symbol to Z_i instead of pushing the new Z_i onto the stack. This moves the push down automaton to the next phase. In this phase it is comparing the reverse order of b symbols to a symbols by popping Z symbols from the stack. At the end of computation, the push down automaton A accepts by empty stack.

We shall prove $\Gamma c(L_p) \geq p + 1$ by contradiction.

Let A in $PDA(1, p)$ be an automaton accepting the language L_p and let Z_i be the initial stack symbol. By corollary (2): there exists some permutation s on $\{1, \dots, p\}$, such that $(q_0, \varepsilon) \in \delta(q_0, b_i, Z_{s(i)})$. Then A accepts $w = b_{s(i)} \notin L_p$. $\square$

note that this theorem shows that the number of stack symbols complexity hierarchy of one state PDA is not bounded.

4.2 The Number of States

We shall now discuss the influence of the acceptance mode on the complexity. The family of one state PDA was forced to use the empty stack acceptance mode since the final state acceptance mode could only be used for specific languages. The influence of the acceptance mode on the complexity thus was not an issue.

Comparison of Acceptance Modes

The family $PDA(n, 2)$ of automata allows both types of acceptance mode, empty stack or final state. Any one of them can be used to accept arbitrary context free language. It is known ([11]) that at least two stack symbols are necessary to accept all context-free languages by empty stack or final state.

Let us consider the influence of these two acceptance modes on the complexity. We can not use the standard constructions[11] to prove equivalence of the computational power of the two acceptance modes, because both of them introduce a new stack symbol. The family of automata we consider allows at most two stack symbols. The problem can be solved by prefix encoding used by Goldstine, Price and Wotschke in their paper *On reducing the number of stack symbols*[5]. The results they present show the upper and lower bounds for the number of states. These results cannot be directly applied in our case, due to their approximate form. That is why we present a lemma, which shows exact upper bound for three to two stack symbol transformation.

The new automaton has fewer stack symbols, each stack symbol Z is represented by a string $h(Z)$. Naturally this increases the number of states. In our construction, let A in $PDA(s, 3)$ be an automaton using stack symbols H_1, H_2, H_3. We construct an equivalent automaton B using two stack symbols Z_0, Z_1. Each stack symbol H_i is represented in B by a string $h(H_i)$ of symbols Z_0 and Z_1. The states of B are pairs of the form $[q, \gamma]$, where q is a state of A and γ is a proper prefix[8] of $h(H_i)$. The idea is that the automaton B reads the encoded stack symbol from the stack and saves it to the current state. Then the automaton B does the same computation as A. For the mapping h, we have chosen:

- $h(H_1) = Z_1$
- $h(H_2) = Z_0 Z_0$
- $h(H_3) = Z_1 Z_0$

PDA A		PDA B	
State	Stack	State	Stack
q	$\dots H_j$	$[q, \varepsilon]$	$\dots \underbrace{Z_{i_1} \dots Z_{i_s}}_{h(H_j)}$
		$[q, Z_{i_s} \dots Z_{i_2}]$	$\dots Z_{i_1}$
p	$\dots H_{j_1} \dots H_{j_k}$	$[p, \varepsilon]$	$\dots \underbrace{Z_{f_1} \dots Z_{f_n}}_{h(H_{j_1}) \dots h(H_{j_k})}$

Figure 1: Example of one computation step of PDA A simulated on PDA B using $(p, H_{j_1} \dots H_{j_k}) \in \delta(q, a, H_j)$, where a is some input symbol. The figure illustrates the general case encoding an arbitrary number of symbols H_i. In our case s can be at most two.

Finally, the accepting states of B shall be all states $[q, \varepsilon]$, where q is an accepting state of A. The last item

[8]The string x is a proper prefix of xy if $y \neq \varepsilon$.

to specify is the initial stack symbol of B. Without loss of generality, we can assume that H_1 is the initial stack symbol of A. Otherwise, we can rename the stack symbols of A.

We shall now describe our construction.

Lemma 6. *Let A in $PDA(s,3)$ be an automaton. Then there exists a push down automaton B using two stack symbols and $2s$ states such that $L(B) = L(A)$.*

Proof. The construction uses the h function as an encoding function for stack symbols of the automaton A to stack symbols of the automaton B. Then B simulates the automaton A by decoding the encoded stack symbols from the stack. If Z_1 is on the top of the stack and the automaton is in the state $[q,\varepsilon]$ then it can simulate the step of the automaton A in the state q and H_1 as the top stack symbol. On the other hand, if the top stack symbol is Z_0 and it is in the state $[q,\varepsilon]$ then the automaton B needs to read one more stack symbol from the stack. So it saves the symbol read in the state. Then it reads the next stack symbol in the state $[q,Z_0]$ and simulates a computation step of A. □

Let us now present the construction from the empty stack acceptance mode to the final state acceptance mode. We omit a formal description of the transformation since it is straightforward.

Theorem 8. *Given a push down automaton A using two stack symbols and s states, there exists a push down automaton B using two stack symbols and $2s+2$ states such that $L(B) = N(A)$.*

Proof. Let us use a general construction (see, e.g., [11]), which for a given PDA A constructs a PDA C such that $L(C) = N(A)$. This PDA C uses three stack symbols and $s+1$ states. By Lemma 6, there exists a push down automaton B using two stack symbols and $2s+2$ states such that $L(B) = L(C)$. □

In order to replace the final state acceptance by the empty stack acceptance we need a lemma similar to Lemma 6 dealing with the empty stack acceptance mode.

Lemma 7. *Let A in $PDA(s,3)$ be an automaton. Then there exists a push down automaton B using 2 stack symbols and $2s$ states such that $N(B) = N(A)$.*

Proof. The automaton B shall have the states: $Q_b = Q_a \times \{\varepsilon, Z_0\}$ and we shall use the same encoding[9] h. Using the reduction and mapping h, we construct the δ_B transition function of B similarly as in the proof of Lemma 6. The automaton B uses two stack symbols and $2s$ states and $N(B) = N(A)$. □

Now, we are ready to present the construction from the final state acceptance mode to the empty stack acceptance mode.

[9] $h(H_1) = Z_1, h(H_2) = Z_0Z_0, h(H_3) = Z_1Z_0$

Theorem 9. *Given a push down automaton A using two stack symbols and s states, there exists a push down automaton B using two stack symbols and $2s+2$ states such that $N(B) = L(A)$.*

Proof. Let us use a general construction (see, e.g., [11]), which constructs a new automaton C. The automaton C is using three stack symbols and $s+1$ states. By the previous Lemma 7, there exists a push down automaton B using two stack symbols and $2s+2$ states such that $N(B) = L(C)$. □

Upper bounds Let us consider the following sequence of context free languages.

Notation 4. *Let a,b,c be distinct symbols. Let $\Sigma = \{a,b,c\}$. For each $r \geq 1$ let*

- $L = \{w = a^m b^m | m \geq 1\}$
- $L_3[n] = \{c^m | 0 \leq m \leq n\}$
- $L_4[n] = Shuf(L, L_3[n])$

At first, let us discuss the properties of the language L. Note that the language L coincides and thus has the same properties as the language L_p (defined in the Section 4.1), for $p = 1$. We proved that one state automaton needs at least two stack symbols to accept $L_p, p = 1$. Then using the general idea from Theorem 7, we can construct the minimal one state PDA accepting L.

We shall now modify the language L in order to "force" the PDA to check some additional property. Both stack symbols are used for keeping track of symbols a and b. Adding another property to this language should result in increasing the number of states. The property we shall use is the language $L_3[n]$, the number of c symbols should be equal or less then n. Mixing properties of L and $L_3[n]$ languages results in the language $L_4[n]$

Finally, we should decide, which acceptance mode we shall use. Let us use the same acceptance mode as in the previous Section 4.1, empty stack acceptance mode. This results in an easier upper bound construction.

Our automaton has $n+1$ states. Each state represents the number of c symbols read. If the automaton reads the $(n+1)$st symbol c it blocks.

Theorem 10. *There exists a PDA A_n using two stack symbols and $n+1$ states such that $N(A_n) = L_4[n]$.*

Proof. We shall construct a PDA $A_n = (\{q_0,\dots,q_n\}, \{a,b,c\}, \{Z_1,Z_2\}, \delta_r, q_0, Z_2, \emptyset)$
The stack symbol Z_1 is used as a counter. On input symbol a, the automaton pushes Z_1 on the stack and it keeps the stack symbol Z_2 on the top. The automaton keeps Z_2 on the top of the stack. This indicates A_n is still reading the part of the input containing the a symbols. The automaton A_n can

nondeterministically pop Z_2 on ε and start to pop Z_1 on each b. On symbol c, the automaton changes the state from q_i to q_{i+1}. In case A_n is in the state q_n and reads the input symbol c, it blocks. $\square$

5 Conclusion

Since no function combining the number of states and the number of stack symbols of PDA to a descriptional complexity measure having desirable properties exists, we studied complexity in two 'extreme' sub classes of PDA each able to define all context-free languages. It would be natural to study behaviour of complexity when operations on languages are performed (some upper bounds appear in [10]) or when some transformations are performed on PDA.

References

[1] M. Holzer and M. Kutrib, "Descriptional complexity — an introductory survey," in *Scientific Applications of Language Methods*, pp. 1–58, World Scientific, 2011.

[2] M. Holzer and M. Kutrib, "Descriptional and computational complexity of finite automata—a survey," *Information and Computation*, vol. 209, no. 3, pp. 456–470, 2011.

[3] J. Dassow, "Descriptional complexity and operations – two non-classical cases," in *Descriptional Complexity of Formal Systems* (G. Pighizzini and C. Câmpeanu, eds.), pp. 33–44, Springer International Publishing, 2017.

[4] J. Goldstine, J. K. Price, and D. Wotschke, "On reducing the number of states in a pda," *Mathematical systems theory*, vol. 15, no. 1, pp. 315–321, 1981.

[5] J. Goldstine, J. K. Price, and D. Wotschke, "On reducing the number of stack symbols in a pda," *Mathematical systems theory*, vol. 26, no. 4, pp. 313–326, 1993.

[6] A. Okhotin, X. Piao, and K. Salomaa, "Descriptional complexity of input-driven pushdown automata," in *Languages Alive*, pp. 186–206, Springer, 2012.

[7] J. Goldstine, M. Kappes, C. M. Kintala, H. Leung, A. Malcher, and D. Wotschke, "Descriptional complexity of machines with limited resources," *J. UCS*, vol. 8, no. 2, pp. 193–234, 2002.

[8] B. Rovan and S. Sadovsky, "On usefulness of information: Framework and nfa case," in *Adventures Between Lower Bounds and Higher Altitudes: Essays Dedicated to Juraj Hromkovič on the Occasion of His 60th Birthday*, pp. 85–99, Springer, 2018.

[9] P. Labath and B. Rovan, "Simplifying dpda using supplementary information," in *International Conference on Language and Automata Theory and Applications*, pp. 342–353, Springer, 2011.

[10] L. Kiss, "Descriptional complexity of push down automata." Master Thesis, Comenius University in Bratislava, 2020.

[11] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. USA: Addison-Wesley Longman Publishing Co., Inc., 2006.