Descriptional complexity of push down automata

Bc. Lukas Kiss

Supervisor: prof. RNDr. Branislav Rovan, PhD.

17 June 2020

Dept. of Computer Science
FMFI

Contents

- How to define a good descriptional complexity on PDA.
    - Number of states? (one state is enough)

Contents

- How to define a good descriptional complexity on PDA.
  - Number of states? (one state is enough)
  - Number of stack symbols? (two stack symbols is enough)

Contents

- How to define a good descriptional complexity on PDA.
  - Number of states? (one state is enough)
  - Number of stack symbols? (two stack symbols is enough)
  - Combination? (not possible)

Contents

- How to define a good descriptional complexity on PDA.
  - Number of states? (one state is enough)
  - Number of stack symbols? (two stack symbols is enough)
  - Combination? (not possible)
- Two subclasses D(1,p) and D(n,2).

Contents

- How to define a good descriptional complexity on PDA.
  - Number of states? (one state is enough)
  - Number of stack symbols? (two stack symbols is enough)
  - Combination? (not possible)
- Two subclasses D(1,p) and D(n,2).
- PDA on Regular Languages.

Contents

- How to define a good descriptional complexity on PDA.
  - Number of states? (one state is enough)
  - Number of stack symbols? (two stack symbols is enough)
  - Combination? (not possible)
- Two subclasses D(1,p) and D(n,2).
- PDA on Regular Languages.
  - Upper bounds for a given number of stack symbols.

Contents

- How to define a good descriptional complexity on PDA.
  - Number of states? (one state is enough)
  - Number of stack symbols? (two stack symbols is enough)
  - Combination? (not possible)
- Two subclasses D(1,p) and D(n,2).
- PDA on Regular Languages.
  - Upper bounds for a given number of stack symbols.
  - Tight lower bounds and acceptance mode influence on complexity.

Contents

- How to define a good descriptional complexity on PDA.
  - Number of states? (one state is enough)
  - Number of stack symbols? (two stack symbols is enough)
  - Combination? (not possible)
- Two subclasses D(1,p) and D(n,2).
- PDA on Regular Languages.
  - Upper bounds for a given number of stack symbols.
  - Tight lower bounds and acceptance mode influence on complexity.
- PDA on Context Free Languages.

Contents

- How to define a good descriptional complexity on PDA.
  - Number of states? (one state is enough)
  - Number of stack symbols? (two stack symbols is enough)
  - Combination? (not possible)
- Two subclasses D(1,p) and D(n,2).
- PDA on Regular Languages.
  - Upper bounds for a given number of stack symbols.
  - Tight lower bounds and acceptance mode influence on complexity.
- PDA on Context Free Languages.
  - Lower bound for D(1,p) subclass.

Contents

- How to define a good descriptional complexity on PDA.
  - Number of states? (one state is enough)
  - Number of stack symbols? (two stack symbols is enough)
  - Combination? (not possible)
- Two subclasses D(1,p) and D(n,2).
- PDA on Regular Languages.
  - Upper bounds for a given number of stack symbols.
  - Tight lower bounds and acceptance mode influence on complexity.
- PDA on Context Free Languages.
  - Lower bound for D(1,p) subclass.
  - Upper bound for D(n,2) subclass.

Contents

- How to define a good descriptional complexity on PDA.
  - Number of states? (one state is enough)
  - Number of stack symbols? (two stack symbols is enough)
  - Combination? (not possible)
- Two subclasses D(1,p) and D(n,2).
- PDA on Regular Languages.
  - Upper bounds for a given number of stack symbols.
  - Tight lower bounds and acceptance mode influence on complexity.
- PDA on Context Free Languages.
  - Lower bound for D(1,p) subclass.
  - Upper bound for D(n,2) subclass.
- Upper bounds on operations $\cup, *, .$ in D(1,p) and D(n,2).

Combining Measures

- Similar approach as Labath and Rovan did on deterministic PDA.

Combining Measures

- Similar approach as Labath and Rovan did on deterministic PDA.

Theorem

There is no function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ meeting the following conditions:

1. For every two PDA A and $\hat{A}$ recognizing the language L:

$$\text{if } (n, p) \prec (\hat{n}, \hat{p}) \text{ then } f(n, p) < f(\hat{n}, \hat{p}).$$

2. If A and $\hat{A}$ are two minimal PDA recognizing L then:

$$f(n, p) = f(\hat{n}, \hat{p}).$$

Our Approach

Notation

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

- $D(1, p)$. (one state)

Our Approach

Notation

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

- $D(1, p)$. (one state)
  - $\mathbf{\Gamma c(L)} = x$, one state PDA needs at least x stack symbols to accept L.

Our Approach

Notation

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

- $D(1, p)$. (one state)
  - $\mathbf{\Gamma c(L)} = x$, one state PDA needs at least x stack symbols to accept L.
- $D(n, 2)$. (two stack symbols)

Our Approach

Notation

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

- $D(1, p)$. (one state)
  - $\mathbf{\Gamma c(L)} = x$, one state PDA needs at least x stack symbols to accept L.
- $D(n, 2)$. (two stack symbols)
  - $\mathbf{Qc(L)} = y$, two stack symbols PDA needs at least y states to accept L.

PDA on Regular Languages

- How the state complexity can be effected by stack?
- Limit the number of stack symbols to some constant p.

PDA on Regular Languages

- How the state complexity can be effected by stack?
- Limit the number of stack symbols to some constant p.

Theorem

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 $N(A_2) = L(A_1)$.

PDA on Regular Languages

- How the state complexity can be effected by stack?
- Limit the number of stack symbols to some constant p.

Theorem

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 $N(A_2) = L(A_1)$.

- **Idea:** The PDA uses combination of stack symbol and state as representation of FSA state.

PDA on Regular Languages

Notation

Let $a_1, \ldots a_n$ be distinct symbols for any $n \geq 1$. Let

Notation

Let $a_1, \ldots a_n$ be distinct symbols for any $n \geq 1$. Let

- $L_1[n] = a_1^* a_2^* \ldots a_n^*$

PDA on Regular Languages

Notation

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

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

PDA on Regular Languages

Notation

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

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

Theorem

$\Gamma c(L_1[p]) = p, \forall p \in N$.

PDA on Regular Languages

Notation

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

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

Theorem

$\Gamma c(L_1[p]) = p, \forall p \in N$.

Theorem

$\Gamma c(L_2[n]) = 2$, for any $n \geq 2$ and $Qc(L_2[n]) = 1$, for any $n \geq 1$.

PDA on Regular Languages

- Allow one stack symbol.
- How does the descriptional complexity change for $L_2[n]$?

PDA on Regular Languages

- Allow one stack symbol.
- How does the descriptional complexity change for $L_2[n]$?
- Accepting by stack:

Theorem

The smallest number of states for any counter automaton accepting the language $L_2[n]$ by empty stack is **two**, for any $n \geq 2$.

PDA on Regular Languages

- Allow one stack symbol.
- How does the descriptional complexity change for $L_2[n]$?
- Accepting by stack:

Theorem

The smallest number of states for any counter automaton accepting the language $L_2[n]$ by empty stack is **two**, for any $n \geq 2$.

- Accepting by final state:

Theorem

The smallest number of states for any push down automaton using one stack symbol accepting language $L_2[n]$ by final state is **n**, for any $n \geq 2$.

D(1,p) on Context Free Languages

- The one state PDA using final state acceptance mode do not define all context free languages.

D(1,p) on Context Free Languages

- The one state PDA using final state acceptance mode do not define all context free languages.

Notation

Let $a_1, \ldots, a_p, b_1, \ldots, b_p$ be distinct symbols for any $p \geq 1$. Let $\Sigma_p = \{a_1, \ldots, a_p, b_1, \ldots, b_p\}$

$$L_p = \{w(h(w))^R | w \in \{a_1, a_2, \ldots, a_p\}^*\}$$

where h is the homomorphism defined by $h(a_i) = b_i$, for each $a_i \in \{a_1, a_2, \ldots, a_p\}$.

D(1,p) on Context Free Languages

Theorem

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

- We have proved that on each b_i the automaton has to pop a stack symbol.

D(1,p) on Context Free Languages

Theorem

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

- We have proved that on each b_i the automaton has to pop a stack symbol.
- On each b_i the automaton has to pop different stack symbol.

D(n,2) on Context Free Languages

- We have proved that both acceptance modes define all context free languages.

D(n,2) on Context Free Languages

- We have proved that both acceptance modes define all context free languages.
- We cannot use directly standard constructions.

D(n,2) on Context Free Languages

- We have proved that both acceptance modes define all context free languages.
- We cannot use directly standard constructions.
  - Additional stack symbol $\implies$ D(n,3).

D(n,2) on Context Free Languages

- We have proved that both acceptance modes define all context free languages.
- We cannot use directly standard constructions.
  - Additional stack symbol $\implies$ D(n,3).
- Reduction from three stack symbols to two.
  - Encoding function h.

D(n,2) on Context Free Languages

- We have proved that both acceptance modes define all context free languages.
- We cannot use directly standard constructions.
  - Additional stack symbol $\implies$ D(n,3).
- Reduction from three stack symbols to two.
  - Encoding function h.

Lemma

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

Lemma

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

D(n,2) on Context Free Languages

Notation

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

- $L = \{w = a^m b^m | m \geq 1\}$
- $L_1[r] = \{c^m | 0 \leq m \leq r\}$
- $L_2[r] = Shuf(L, L_1[r])$.

- Both stack symbols are used for keeping track of symbols a and b.

D(n,2) on Context Free Languages

Notation

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

- $L = \{w = a^m b^m | m \geq 1\}$
- $L_1[r] = \{c^m | 0 \leq m \leq r\}$
- $L_2[r] = Shuf(L, L_1[r])$.

- Both stack symbols are used for keeping track of symbols a and b.
- We modify L in order to **"force" the PDA** to check some additional property.

D(n,2) on Context Free Languages

Notation

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

- $L = \{w = a^m b^m | m \geq 1\}$
- $L_1[r] = \{c^m | 0 \leq m \leq r\}$
- $L_2[r] = Shuf(L, L_1[r])$.

- Both stack symbols are used for keeping track of symbols a and b.
- We modify L in order to **"force" the PDA** to check some additional property.

Theorem

There exists a PDA A_r using two stack symbols and $r + 1$ states such that $N(A_r) = L_2[r]$.

Complexity of Operations in D(1,p)

operation	number of stack symbols
$\cup$	$p_1 + p_2 + 1$
$\cdot$	$p_1 + p_2 + 1$
$*$	$p_1 + 1$

Table: Sufficient number of stack symbols.

Complexity of Operations in D(n,2)

operation	*empty stack*	*final state*
$\cup$	$r+s+1$	$r+s+1$
$.$	$2(r+s)+2$	$2(r+s)+2$
$*$	$2r+2$	$2r+2$

Table: Sufficient number of states

- The descriptional complexity does not depend on acceptance mode.

Thank you for your attention

Lemma 2.2.1

Notation

$L_1[n] = a_1^* a_2^* \dots a_n^*$

Lemma 2.2.1

Let A in $D(1, p)$ be an automaton accepting the language $L_1[n]$, where $p, n \in N$. Suppose $\delta(q_0, a_i, Z) \neq \emptyset$ and $\delta(q_0, a_j, \hat{Z}) \neq \emptyset$ for $i \neq j$. Then $Z \neq \hat{Z}$.

Notation

$L_1[n] = a_1^* a_2^* \dots a_n^*$

Lemma 2.2.1

Let A in $D(1, p)$ be an automaton accepting the language $L_1[n]$, where $p, n \in N$. Suppose $\delta(q_0, a_i, Z) \neq \emptyset$ and $\delta(q_0, a_j, \hat{Z}) \neq \emptyset$ for $i \neq j$. Then $Z \neq \hat{Z}$.

Lemma 2.2.1

Let A in $D(1, p)$ be an automaton accepting the language $L_1[n]$, where $p, n \in N$ and $p \leq n$. Suppose $\delta(q_0, a_i, Z) \neq \emptyset$ and $\delta(q_0, a_j, \hat{Z}) \neq \emptyset$ for $i \neq j$. Then $Z \neq \hat{Z}$.

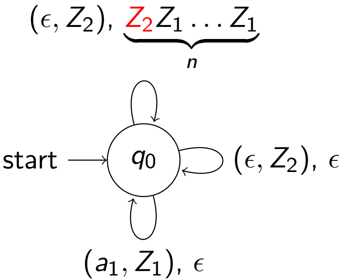
Lemma 2.2.1

- We wanted to show that there exists $\epsilon - cycle$, on which the automaton removes γ_i from the stack.
- **Corrected**: There exists a input word w_i, on which the automaton removes γ_i from the stack.

Question 1.

Notation

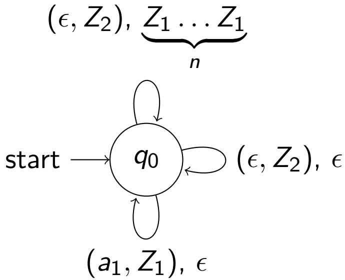
$L_2[n] = \{a_1^{kn} | k \geq 0\}$.



Question 1.

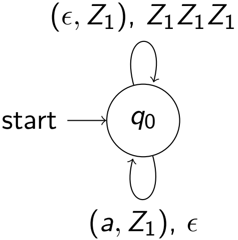
Notation

$L_{Q1} = \{\epsilon, \underbrace{a_1 \dots a_1}_{n}\}$.



Question 2.

- $L = \emptyset, L = \Sigma^*$
- $L_{odd} = \{a^k|$ k is odd $\} = \{a^{2m+1}|m \in N\}$



- **Generally**: $L_k = \{a^{km+1}|m \in N\}$