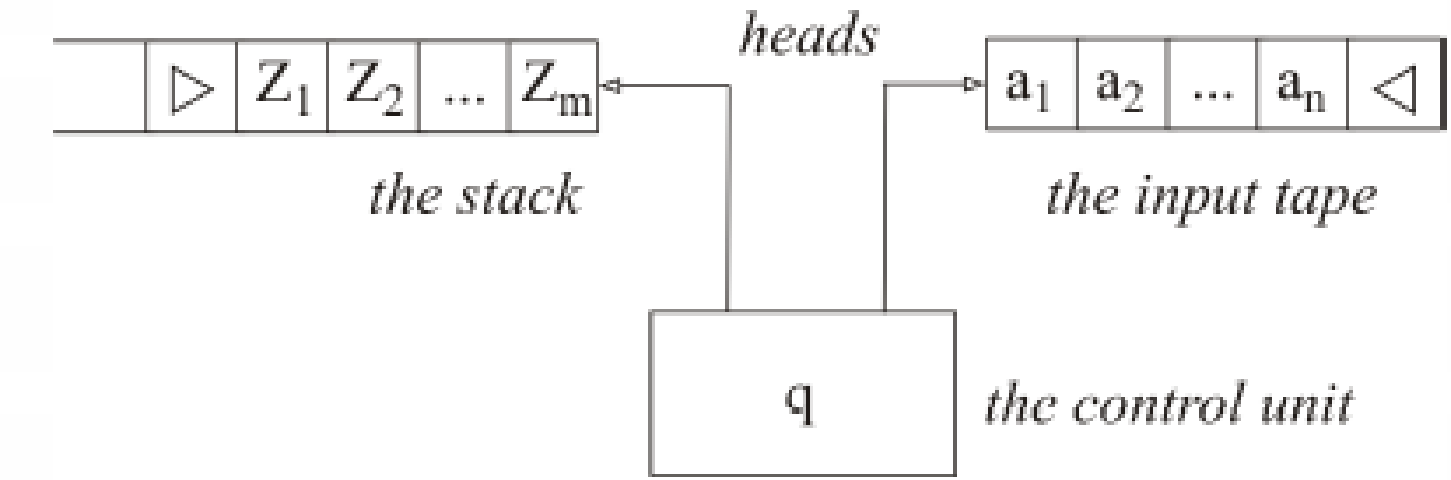
Automata Theory and Formal Languages
Class 11

Marcin Luckner, PhD
mluckner@mini.pw.edu.pl

Version 1.0
16 stycznia 2022

Pushdown automaton



Rysunek 1: Pushdown automaton

Definition

$$A = (Q, \Sigma, \Gamma, \delta, q0, \triangleright, \triangleleft, F)$$

The components have the following means

- Q A finite set of states
- Σ the set of input symbols,
- Γ the finite stack alphabet, the set of symbols that we are allowed to push onto the stack
- δ the transition function, a mapping from $Q \times \Sigma \times \Gamma$ to $Q \times \Gamma$
- q_0 the start state
- $\triangleright$ the start symbol, initially the stack consists of one instance this symbol and nothing else
- $\triangleleft$ the input end symbol
- F the set of final states $F \subseteq Q$

Operations

command	argument	stack before	stack after
stay	A	$\triangleright A$	$\triangleright A$
pop	ϵ	$\triangleright A$	$\triangleright$
push	AB	$\triangleright A$	$\triangleright AB$

Transition table

- Unlike a Turing machine, a pushdown automaton transition function has three arguments.
- Therefore, it cannot be represented by a two-dimensional table.
- We create a single table for each state, with rows labelled by the stack symbols and columns labelled by the input symbols.
- Both rows and columns include $\triangleright$ and $\triangleleft$ respectively.

Acceptance

- An input word is accepted when the final state is reached.
- However, another option is acceptance with an empty stack.
  - An input word is accepted, when the input end symbol is reached and the stack is empty.
- The acceptances by an empty stack and an acceptance by a final state are equivalent.

Case I

Design deterministic pushdown automaton for language over alphabet $\Sigma = \{0, 1\}$ of words with an equal number of 0's and 1's.

Case I - idea

- We can count the symbols by storage their representation on the stack.
- If the input symbol's representation is on the top of the stack, we add symbol (push).
- If an opposite symbol is on the top of the stack, we remove the symbol from the stack (pop).
- The automaton will accept with the empty stack.

Case I - transition table

q_0	0	1	$\lhd$
O	(q_0, OO)	(q_0, ϵ)	q_R
I	(q_0, ϵ)	(q_0, II)	q_R
$\rhd$	$(q_0, \rhd O)$	$(q_0, \rhd I)$	q_A

Case II

Design deterministic pushdown automaton for language over alphabet $\Sigma = \{0, 1\}$ of words with twice as many 0's as 1's.

Case II - idea

- The task looks similar to the previous one. However, there is one important difference.
- If we put symbols 0 on the stack, then two symbols should be removed at once with a single 1.
- It cannot be done with a single pop action. Therefore, the automaton should be nondeterministic or the double 0 must be simulated in another way.
- In the proposed solution we will use a state to simulate that.

Case II - transition tables

q_0	0	1	$\lhd$
O	(q_1, OO)	(q_0, ϵ)	q_R
I	(q_1, ϵ)	(q_0, II)	q_R
$\rhd$	$(q_1, \rhd O)$	$(q_0, \rhd I)$	q_A

q_1	0	1	$\lhd$
O	(q_0, O)	(q_0, ϵ)	q_R
I	(q_0, I)	(q_1, II)	q_R
$\rhd$	$(q_0, \rhd)$	$(q_1, \rhd I)$	q_R

Case II - computation example

q_0	0	1	$\lhd$
O	(q_1, OO)	(q_0, ϵ)	q_R
I	(q_1, ϵ)	(q_0, II)	q_R
$\rhd$	$(q_1, \rhd O)$	$(q_0, \rhd I)$	q_A

q_1	0	1	$\lhd$
O	(q_0, O)	(q_0, ϵ)	q_R
I	(q_0, I)	(q_1, II)	q_R
$\rhd$	$(q_0, \rhd)$	$(q_1, \rhd I)$	q_R

- Input word: 101000

1. $\mathbf{1}01000\lhd, q_0, \rhd \rightarrow 01000\lhd, q_0, \rhd I$
2. $\mathbf{0}1000\lhd, q_0, \rhd \mathbf{I} \rightarrow 1000\lhd, q_1, \rhd$
3. $\mathbf{1}000\lhd, q_1, \rhd \rightarrow 000\lhd, q_1, \rhd I$
4. $\mathbf{0}00\lhd, q_1, \rhd \mathbf{I} \rightarrow 00\lhd, q_0, \rhd I$
5. $\mathbf{0}0\lhd, q_0, \rhd \mathbf{I} \rightarrow 0\lhd, q_1, \rhd$
6. $\mathbf{0}\lhd, q_1, \rhd \rightarrow 0\lhd, q_0, \rhd$
7. $\lhd, q_0, \rhd \rightarrow q_A$

Assignments I

1. Can be the following task solved using one-state deterministic pushdown automata?
   - Design pushdown automata for the language over alphabet $\Sigma = \{0, 1\}$ of words with twice as many 0's as 1's.
2. Design a deterministic pushdown automaton for the following languages
   - The language over alphabet $\Sigma = \{0, 1\}$ of words with no more ones than zeros for each prefix.
   - The language over alphabet $\Sigma = \{0, 1\}$ of words with the odd difference between the numbers of zeros and ones.
   - The language over alphabet $\Sigma = \{a, b, c\}$ of words in form $a^i b^j c^k$ where $k < i + j$ and $k > i$.