COMPSCI 501: Formal Language Theory

Lecture 8: Pushdown Automata

Marius Minea
marius@cs.umass.edu

University of Massachusetts Amherst

8 February 2019

Generators and Recognizers

- *Generators*: produce all strings in a language
  regular expressions, grammars
  "declarative" (string/set operations)
- *Recognizer*: tells if string is in a language
  DFA, NFA, pushdown automata
  "mechanical" / algorithmic

Related, often simple, direct constructions
regular expression $\rightarrow$ NFA
regex derivative $\rightarrow$ DFA

Control and Data

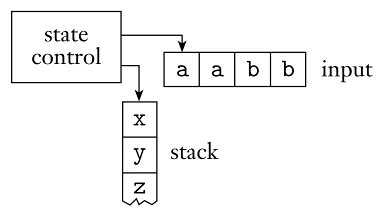
Programs have both control (program counter) and variables.

What programs are equivalent to DFAs?

finite (global) variables, no procedures

$Q = L_{pc} \times D_1 \times \ldots \times D_n$ where D_i = domain of variable v_i
L_{pc} = set of program counter locations,

Pushdown Automata: Add a Stack



One step moves to another *state* and *stack contents*.

Stack can store more than just input symbols
(different / richer alphabet)

In addition: **nondeterminism** (crucial in some cases)

Recognizing $0^n 1^n$

$L = \{0^n 1^n \mid n \geq 0\}$

- while zeroes, put on stack
- while ones, pop 0 from stack (reject on underflow)
- accept if end of input and stack empty

Can do: finite (global) variables + stack

How about global vars + function calls + finite local vars ?

Pushdown Automata: Definition

A pushdown automaton is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where all sets are finite, and

1. Q is the set of states
2. Σ is the input alphabet
3. Γ is the stack alphabet
4. $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function
5. $q_0 \in Q$ is the start state
6. $F \subseteq Q$ is the set of accept states

where $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$, $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$.

PDA can only look at *top* of stack
Full stack contents *not* part of transition function

Transitions and Stack

Transition: input $\times$ stack symbol $\rightarrow$ stack symbol

Written $a, b \rightarrow c$: on input a, replace stack top b with c

Can have any combination of:

- $a = \varepsilon$: read no input symbol
- $b = \varepsilon$: pop nothing from stack
- $c = \varepsilon$: push nothing on stack

In particular:

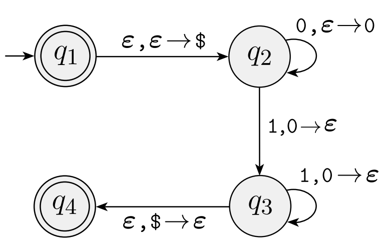
- $a, \varepsilon \rightarrow c$ means push(c) (on input a)
- $a, b \rightarrow \varepsilon$ means pop(b) (on input a)

Computations of PDA

PDA accepts input if we can write it (perhaps with ε's) as
$w = w_1 w_2 \dots w_m$, $w_i \in \Sigma_\varepsilon$, and we have
– a *state sequence* $r_0, r_1, \dots, r_m \in Q$ and
– a sequence of *stack strings* $s_0, s_1, \dots, s_m \in \Gamma^*$ such that

1. Initial: $r_0 = q_0, s_0 = \varepsilon$ (empty stack)
2. Step: $(r_i, b) \in \delta(r_i, w_{i+1}, a)$, where $s_i = at$ and $s_{i+1} = bt$
   (replace a with b, top of stack = start of stack string)
3. Accept: $r_m \in F$: in accept state at input end

Representing PDAs: $0^n 1^n$



Transition to accepting state happens on empty stack ($)

Useful Tests: empty stack, end of input

- Testing for empty stack

not in formal definition
transition $a, \varepsilon \rightarrow c$ does *not* mean stack empty
place special symbol at bottom and recognize

- Testing for end of input

PDA can't explicitly test for end of input
but checking for accept state only happens at end of input
(can effectively assume we're able to test for end of input)

Some PDA definitions require empty stack at the end – not here.
Versions are equivalent

Example: Three counters, two equal

$L = \{a^i b^j c^k \mid i, j, k \geq 0, i = j \text{ or } i = k\}$

How do we know whether to match b's or c's ? **Nondeterminism**

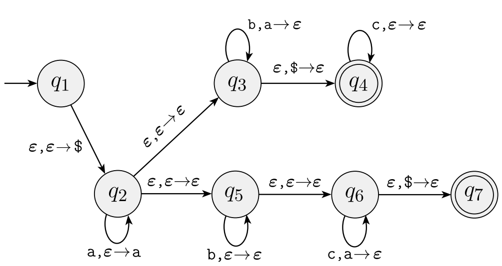
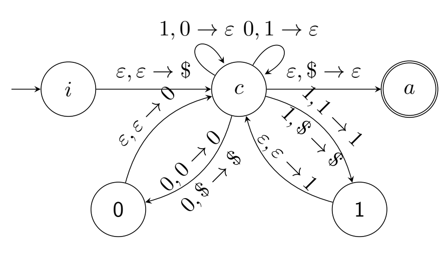


FIGURE 2.17
State diagram for PDA M_2 that recognizes
$\{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$

Other Examples

- Equal number of 0 and 1: stack keeps difference



- Even-length palindrome

$L = \{ww^R \mid w \in \{0,1\}^*\}$ must use nondeterminism