Pushdown automata

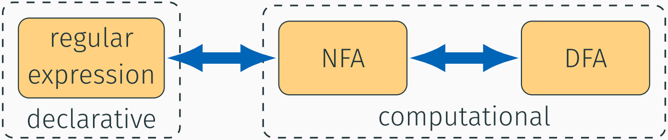
CSCI 3130 Formal Languages and Automata Theory

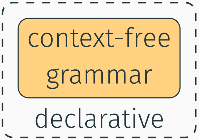
Siu On CHAN

Fall 2020

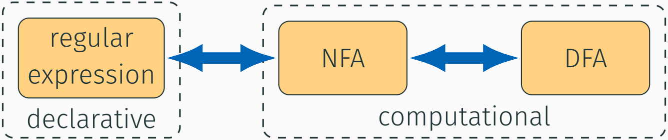
Chinese University of Hong Kong

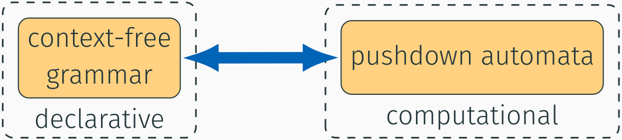
Declarative vs imperative/computational



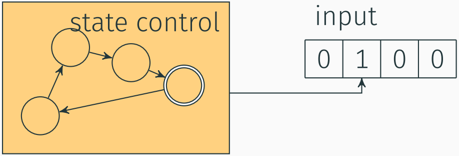


Declarative vs imperative/computational

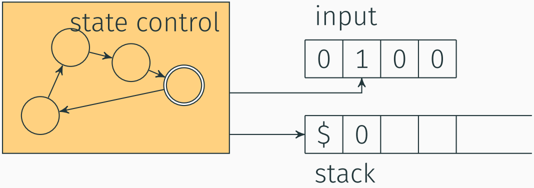




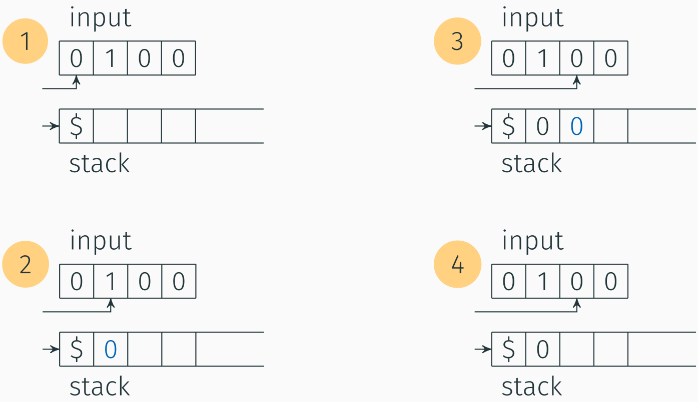
NFA:



PDA:

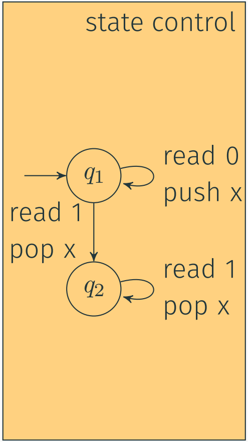


A pushdown automaton (PDA) is like an NFA but with an infinite stack



As the PDA reads the input, it can push/pop symbols from the top of the stack

Building a PDA



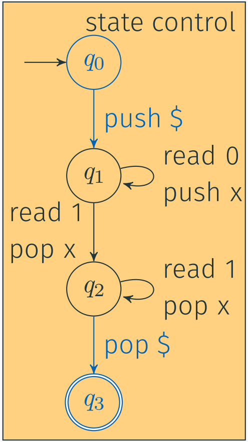
$$L = \{0^n 1^n \mid n \geqslant 1\}$$

Remember each 0 by pushing x onto the stack

Upon reading a 1, pop x from the stack

We want to accept when the PDA hit the stack bottom

Building a PDA



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

Remember each 0 by pushing x onto the stack

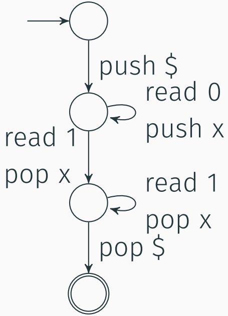
Upon reading a 1, pop x from the stack

We want to accept when the PDA hit the stack bottom

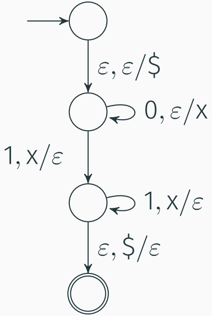
Use \$ to mark the stack bottom

Example input: 000111

Notation for PDAs



$\Longrightarrow$



read a, pop b / push c

If next symbol is a and top of stack is b
then read a, pop b and push c

If $a = \varepsilon$, don't read the next symbol
If $b = \varepsilon$, don't pop the next symbol

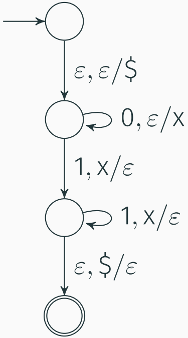
Definition of PDA

A pushdown automaton is $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where:

- Q is a finite set of states
- Σ is a finite set of input alphabet
- Γ is a finite set of stack alphabet
- $q_0 \in Q$ is the initial state
- $F \subset Q$ is the set of accepting states
- δ is the transition function

$$\delta : \underbrace{Q}_{\text{state}} \times \underbrace{(\Sigma \cup \{\varepsilon\})}_{\text{input symbol}} \times \underbrace{(\Gamma \cup \{\varepsilon\})}_{\text{pop symbol}} \to \text{subsets of} \left\{ \underbrace{Q}_{\text{state}} \times \underbrace{(\Gamma \cup \{\varepsilon\})}_{\text{push symbol}} \right\}$$

Example



$$\Sigma = \{0, 1\}$$

$$\Gamma = \{\$, \mathsf{x}\}$$

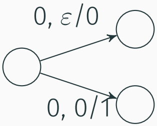
$$\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \$)\}$$
$$\delta(q_0, \varepsilon, \$) = \emptyset$$
$$\delta(q_0, \varepsilon, \mathsf{x}) = \emptyset$$
$$\delta(q_0, 0, \varepsilon) = \emptyset$$
$$\vdots$$

$$\delta : \underbrace{Q}_{\text{state}} \times \underbrace{(\Sigma \cup \{\varepsilon\})}_{\text{input symbol}} \times \underbrace{(\Gamma \cup \{\varepsilon\})}_{\text{pop symbol}} \to \text{subsets of} \left\{ \underbrace{Q}_{\text{state}} \times \underbrace{(\Gamma \cup \{\varepsilon\})}_{\text{push symbol}} \right\}$$

The language of PDA

A PDA is nondeterministic

multiple possible transitions on same input/pop symbol allowed



Transitions may but do not have to push or pop

A PDA accepts input x if, for some computational path, the PDA finishes reading all input symbols and stop at an accepting state

When accepting an input string, the stack need not be empty

The language of a PDA is the set of all strings in Σ^* it accepts

Example 1

$$L = \{w\#w^R \mid w \in \{0,1\}^*\}$$

$\Sigma = \{0, 1, \#\}$

$\Gamma = \{0, 1, \$\}$

#, 0#0, 01#10 in L

ε, 01#1, 0##0 not in L

Example 1

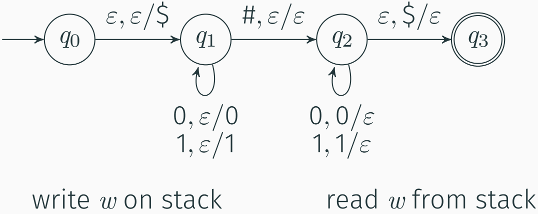
$$L = \{w\#w^R \mid w \in \{0,1\}^*\}$$

$\Sigma = \{0, 1, \#\}$

$\Gamma = \{0, 1, \$\}$

#, 0#0, 01#10 in L

ε, 01#1, 0##0 not in L



Example 2

$$L = \{ww^R \mid w \in \Sigma^*\}$$

$\Sigma = \{0, 1\}$

ε, 00, 0110 in L
011, 010 not in L

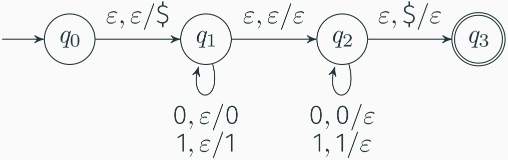
Example 2

$$L = \{ww^R \mid w \in \Sigma^*\}$$

$\Sigma = \{0, 1\}$

ε, 00, 0110 in L

011, 010 not in L



guess middle of string

Example 3

$$L = \{w \in \Sigma^* \mid w = w^R\}$$

$$\Sigma = \{0, 1\}$$

ε, 00, 010, 0110 in L

011 not in L

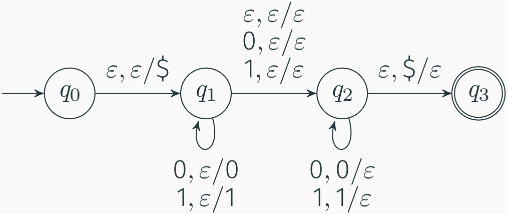
Example 3

$$L = \{w \in \Sigma^* \mid w = w^R\}$$

$\Sigma = \{0, 1\}$

ε, 00, 010, 0110 in L

011 not in L



middle symbol can be ε, 0, or 1

Example: $\underbrace{0010}_{x}\underbrace{0100}_{x^R}$ or $\underbrace{0010}_{x}1\underbrace{0100}_{x^R}$

Example 4

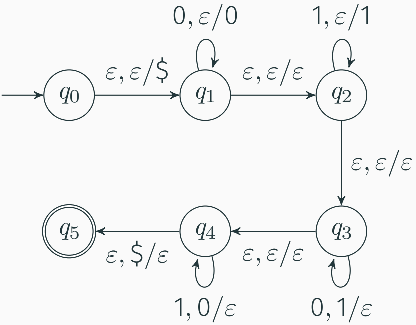
$$L = \{0^n 1^m 0^m 1^n \mid n \geqslant 0, m \geqslant 0\}$$

$$\Sigma = \{0, 1\}$$

Example 4

$L = \{0^n 1^m 0^m 1^n \mid n \geqslant 0, m \geqslant 0\}$

$\Sigma = \{0, 1\}$



input: $0^n 1^m 0^m 1^n$

stack: $0^n 1^m$

Example 5

$L =$ same number of 0s and 1s

$\Sigma = \{0, 1\}$

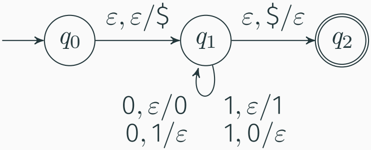
Example 5

$L =$ same number of 0s and 1s

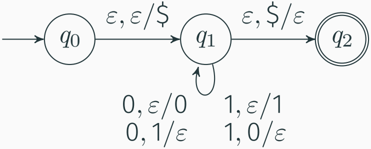
$\Sigma = \{0, 1\}$

Keep track of the excess of 0s or 1s

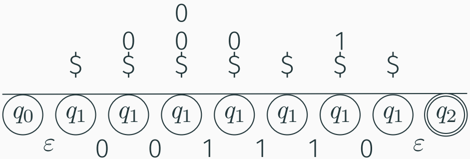
If at the end, the stack is empty, their numbers are equal



Example 5



Example input: 001110

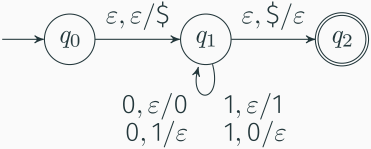


Why does the PDA work?

Example 5

$L =$ same number of 0s and 1s

$\Sigma = \{0, 1\}$



Invariant: In every execution path,
$\#1 - \#0$ on stack = actual $\#1 - \#0$ so far

If $w \notin L$, it must be rejected

Property: In some execution path,
stack consists only of 0s or 1s (or is empty)

If $w \in L$, some execution path will accept