

Pushdown automata

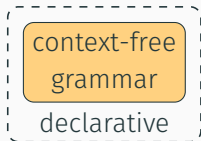
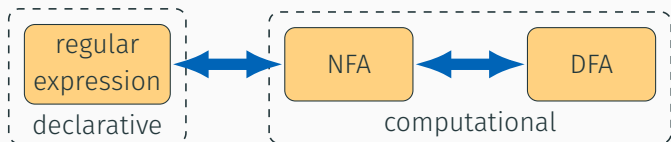
CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

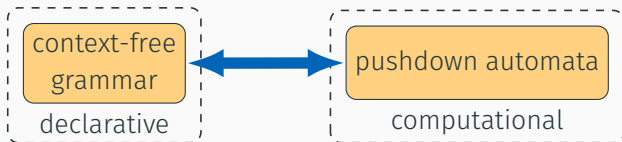
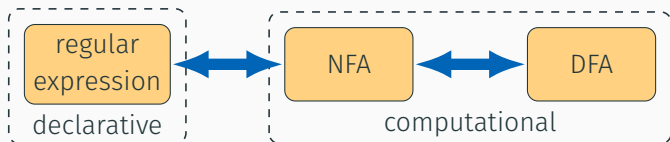
Fall 2021

Chinese University of Hong Kong

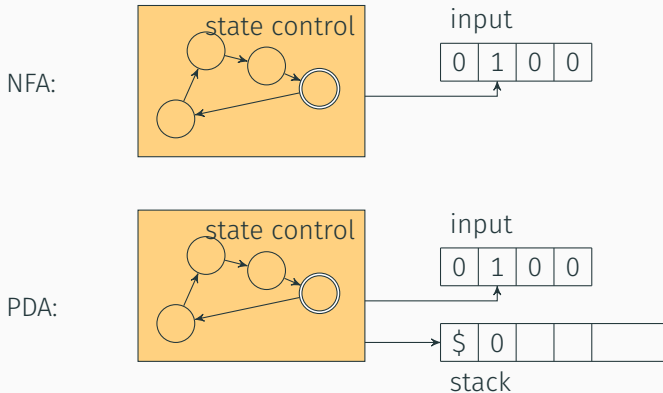
Declarative vs imperative/computational



Declarative vs imperative/computational

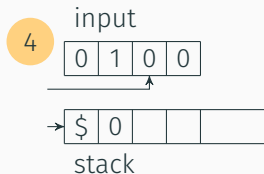
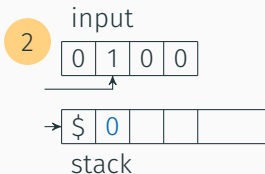
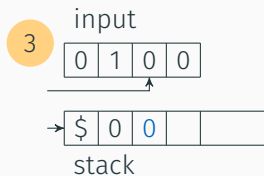
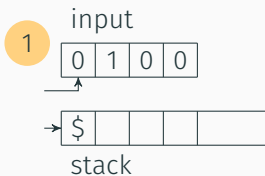


NFA vs pushdown automaton



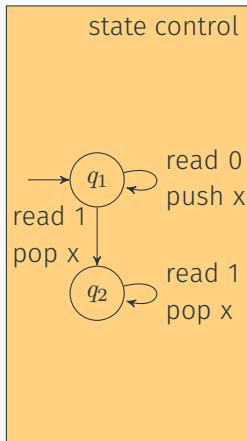
A pushdown automaton (PDA) is like an NFA but with an infinite [stack](#)

Pushdown automata



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

Building a PDA



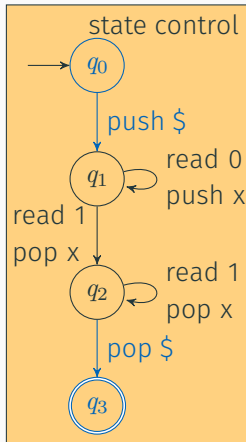
$$L = \{0^n 1^n \mid n \geq 1\}$$

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



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

Remember each 0 by **pushing** x onto the stack

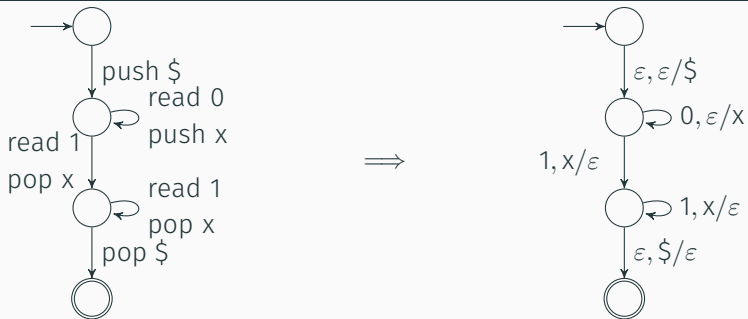
Upon reading a 1, **pop** x from the stack

We want to accept when the PDA hit the stack bottom

Use $\$$ to mark the stack bottom

Example input: 000111

Notation for PDAs



read a , pop b / push c

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

If $a = \epsilon$, don't read the next symbol

If $b = \epsilon$, don't pop the next symbol

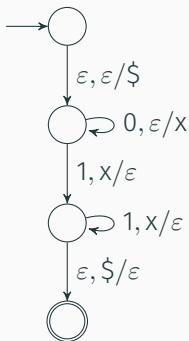
Definition of PDA

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**

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

Example



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

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

$$\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \$)\}$$

$$\delta(q_0, \varepsilon, \$) = \emptyset$$

$$\delta(q_0, \varepsilon, x) = \emptyset$$

$$\delta(q_0, 0, \varepsilon) = \emptyset$$

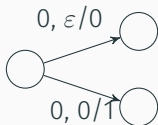
⋮

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

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

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

$\#, 0\#0, 01\#10$ in L

$\varepsilon, 01\#1, 0\#\#0$ not in L

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

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

Example 1

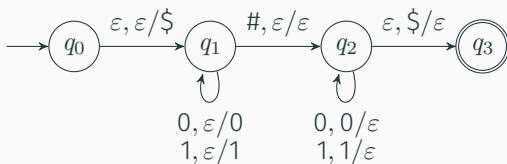
$$L = \{w\#w^R \mid w \in \{0,1\}^*\}$$

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

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

$\#, 0\#0, 01\#10$ in L

$\epsilon, 01\#1, 0\#\#0$ not in L



write w on stack

read w from stack

Example 2

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

$\varepsilon, 00, 0110$ in L

$011, 010$ not in L

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

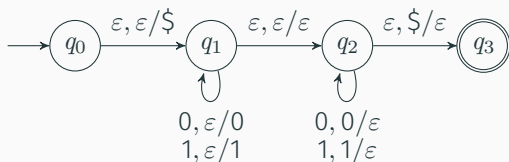
Example 2

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

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

ϵ , 00, 0110 in L

011, 010 not in L



guess middle of string

Example 3

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

$\varepsilon, 00, 010, 0110$ in L

011 not in L

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

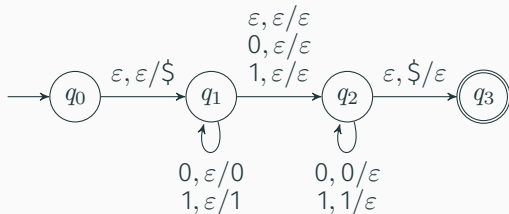
Example 3

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

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

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

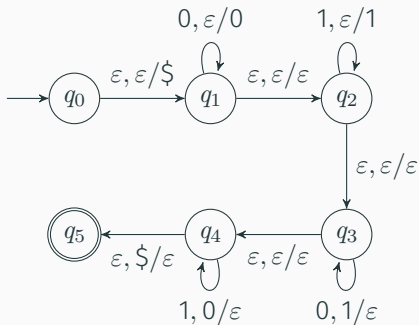
$$L = \{0^n 1^m 0^m 1^n \mid n \geq 0, m \geq 0\}$$

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

Example 4

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

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



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

stack: $0^n 1^m$

Example 5

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

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

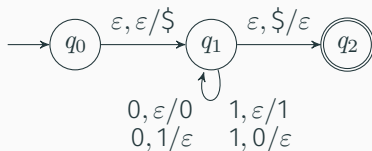
Example 5

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

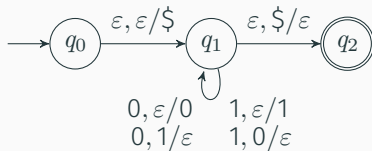
$\Sigma = \{0, 1\}$

Keep track of **the excess** of 0s or 1s

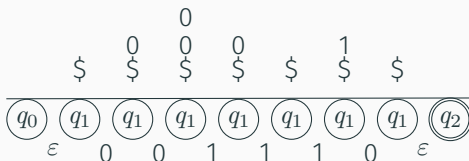
If at the end, the stack is empty, their numbers are equal



Example 5



Example input: 001110

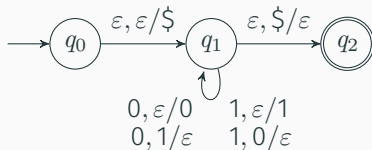


Why does the PDA work?

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