Pushdown automata

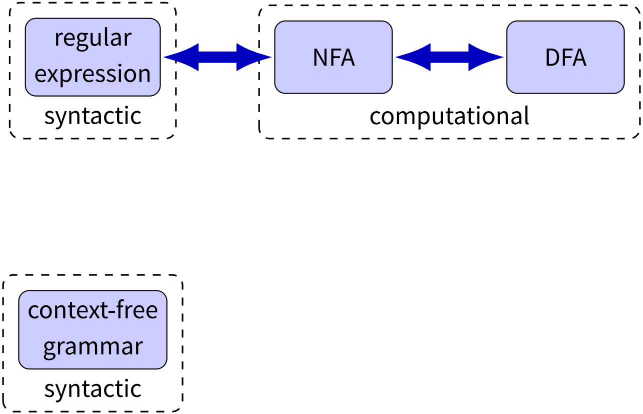
CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

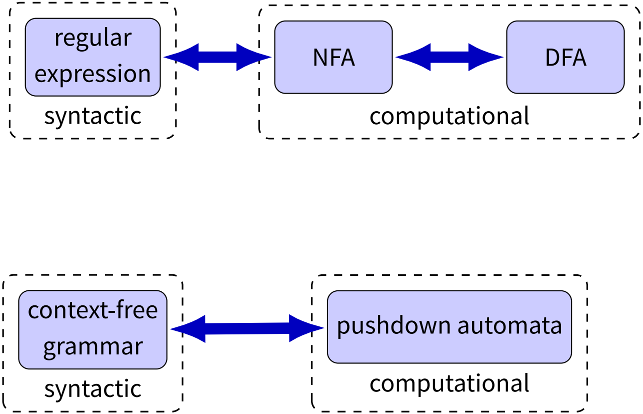
Chinese University of Hong Kong

Fall 2015

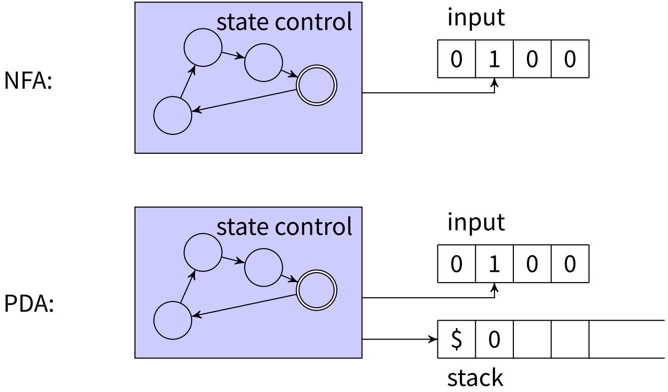
Syntax vs computation



Syntax vs computation

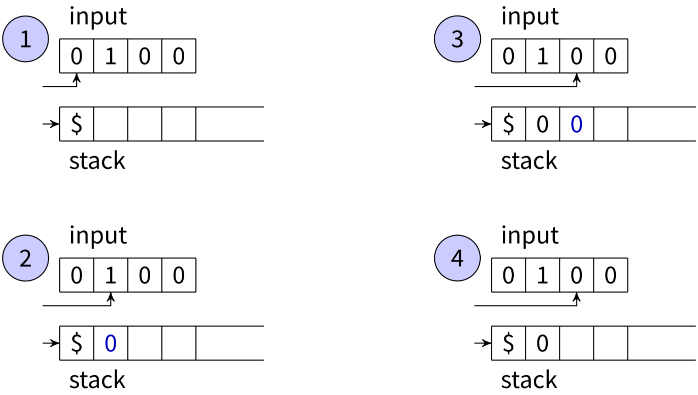


NFA vs pushdown automaton



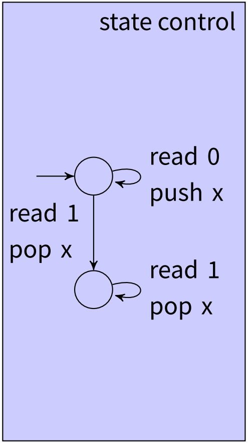
A pushdown automaton (PDA) is like an NFA but with an infinite stack

Pushdown automata



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

Building a PDA



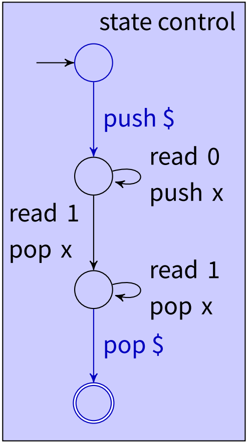
$$L = \{0^n 1^n \mid n \geqslant 1\}$$

Remember each 0 by pushing x onto the stack

Upon reading a 1, pop x from the stack

We want to accept when the 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

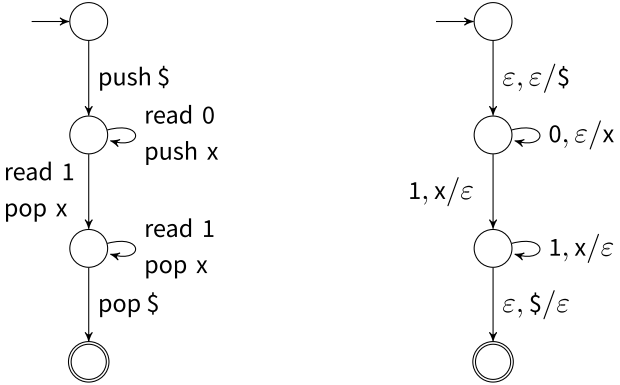
Upon reading a 1, pop x from the stack

We want to accept when the hit the stack bottom

Use $ to mark the stack bottom

Example input: 000111

Notation for PDAs



read, pop / push

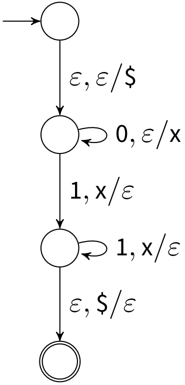
Definition of PDA

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

- Q is a finite set of states
- Σ is the input alphabet
- Γ is the stack alphabet
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ 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



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

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

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

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

The language of a PDA is the set of all strings in Σ^*
that can lead the PDA to an accepting state

Example 1

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

#, 0#0, 01#10 in L

ε, 01#1, 0##0 not in L

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

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

Example 1

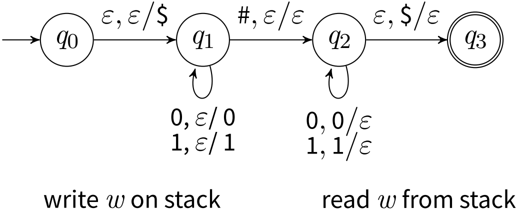
$$L = \{w\#w^R \mid w \in \{0,1\}^*\}$$

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

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

#, 0#0, 01#10 in L

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

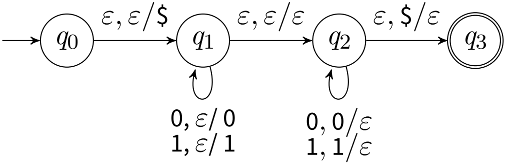
Example 2

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

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

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

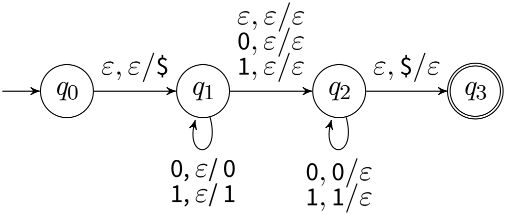
Example 3

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

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

ε, 00, 010, 0110 in L

011 not in L



middle symbol can be ε, 0, or 1

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

Example 4

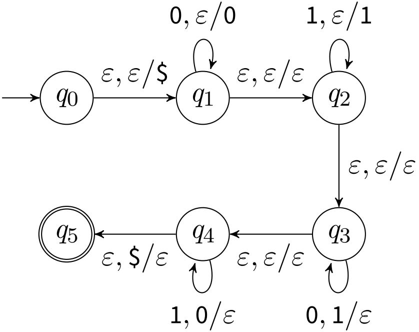
$$L = \{\mathtt{0}^n\mathtt{1}^m\mathtt{0}^m\mathtt{1}^n \mid n \geqslant 0, m \geqslant 0\}$$

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

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

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

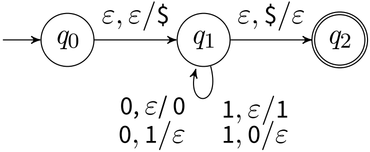
Example 5

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

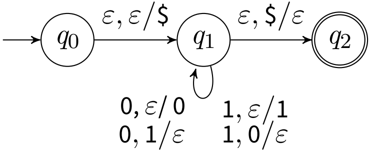
$\Sigma = \{0, 1\}$

Keep track of excess of 0s or 1s

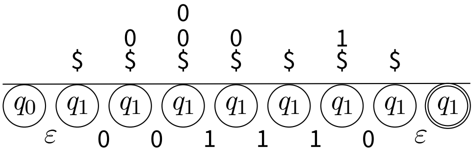
If at the end, the stack is empty, number is equal



Example 5



Example input: 001110

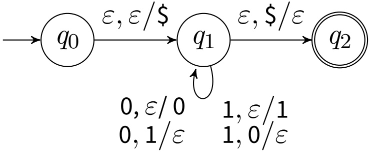


Why does the PDA work?

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,

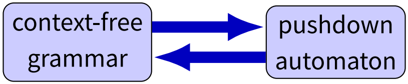
stack consists only of 0s or 1s (or is empty)

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

CFG $\leftrightarrow$ PDA conversions

CFGs and PDAs

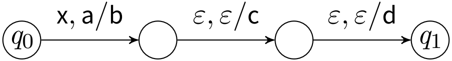
L has a context-free grammar if and only if it is accepted by some pushdown automaton.



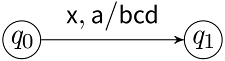
Will first convert CFG to PDA

Convention

A sequence of transitions like



will be abbreviated as



replace a by bcd on stack

Converting a CFG to a PDA

Idea: Use PDA to simulate derivations

$A \to 0A1$
$A \to B$
$B \to \#$

Example:
$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$

Rules:

1. Write the start symbol A onto the stack
2. Rewrite variable on top of stack (in reverse) according to production

PDA control		stack	input
write start variable	$\varepsilon, \varepsilon / A$	$\$A$	00#11
replace by production in reverse	$\varepsilon, A / 1A0$	$\$1A0$	00#11

Converting a CFG to a PDA

Idea: Use PDA to simulate derivations

$A \to 0A1$
$A \to B$
$B \to \#$

Example:
$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$

Rules:

1. Write the start symbol A onto the stack
2. Rewrite variable on top of stack (in reverse) according to production
3. Pop top terminal if it matches input

PDA control		stack	input
write start variable	$\varepsilon, \varepsilon / A$	$\$A$	00#11
replace by production in reverse	$\varepsilon, A / 1A0$	$\$1A0$	00#11
pop terminal and match	$0, 0/\varepsilon$	$\$1A$	0#11
replace by production in reverse	$\varepsilon, A / 1A0$	$\$11A0$	0#11

⋮

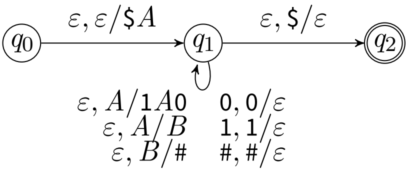
Converting a CFG to a PDA

CFG

$A \to 0A1$

$A \to B$

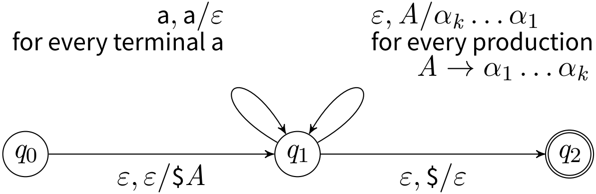
$B \to \#$



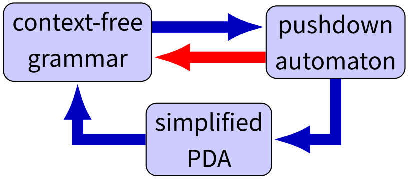
input	00#11	00#11	00#11	00#11	00#11
stack	$\$A$	$\$1A0$	$\$1A$	$\$11A0$	$\$11A$
input	00#11	00#11	00#11	00#11	00#11
stack	$\$11B$	$\$11\#$	$\$11$	$\$1$	$\$$

$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$$

General PDA to CFG conversion



From PDAs to CFGs

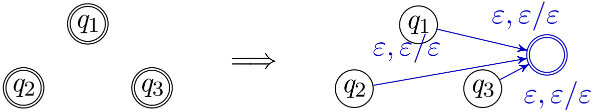


Simplified pushdown automaton:

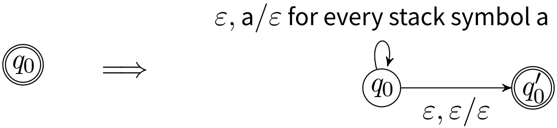
- Has a single accepting state
- Empties its stack before accepting
- Each transition is either a push, or a pop, but not both

Simplifying the PDA

Single accepting state

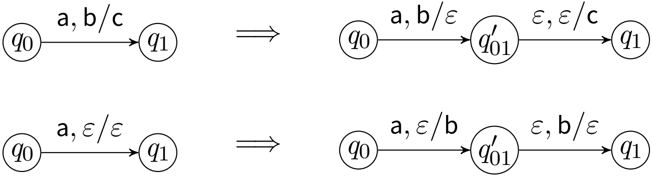


Empties its stack before accepting



Simplifying the PDA

Each transition either pushes or pops, but not both

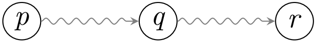
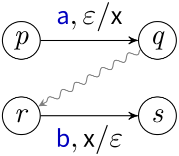


Simplified PDA to CFG

For every pair (q, r) of states in PDA, introduce variable A_{qr} in CFG

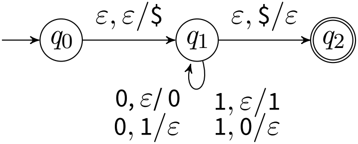
Intention: A_{qr} generates all strings that allow the PDA to go from q to r
(with empty stack both at q and at r)

Simplified PDA to CFG

PDA	CFG
	$A_{qq} \to \varepsilon$
	$A_{pr} \to A_{pq} A_{qr}$
	$A_{ps} \to$ a A_{qr} b a $=$ ε or b $=$ ε allowed

Start variable: A_{pq} (initial state p, accepting state q)

Example: Simplified PDA to CFG

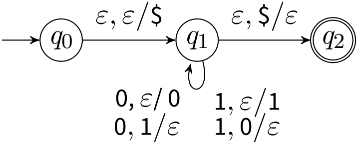


productions:

variables:

start variable:

Example: Simplified PDA to CFG



variables: $A_{00}, A_{11}, A_{22}, A_{01}, A_{02}, A_{12}$

start variable: A_{02}

productions:

$$A_{02} \to A_{01} A_{12}$$

$$A_{01} \to A_{01} A_{11}$$

$$A_{12} \to A_{11} A_{12}$$

$$A_{11} \to A_{11} A_{11}$$

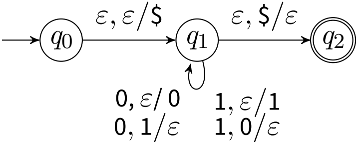
$$A_{11} \to 0 A_{11} 1$$

$$A_{11} \to 1 A_{11} 0$$

$$A_{02} \to A_{11}$$

$A_{00} \to \varepsilon$, $A_{11} \to \varepsilon$, $A_{22} \to \varepsilon$

Example: Simplified PDA to CFG



variables: $A_{00}, A_{11}, A_{22},$
A_{01}, A_{02}, A_{12}

start variable: A_{02}

productions:

$A_{02} \to A_{01} A_{12}$

$A_{01} \to A_{01} A_{11}$

$A_{12} \to A_{11} A_{12}$

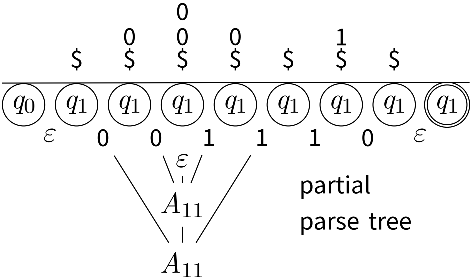
$A_{11} \to A_{11} A_{11}$

$A_{11} \to 0 A_{11} 1$

$A_{11} \to 1 A_{11} 0$

$A_{02} \to A_{11}$

$A_{00} \to \varepsilon,\ A_{11} \to \varepsilon,$
$A_{22} \to \varepsilon$



partial
parse tree