Pushdown Automata

Lin Chen

Email: Lin.Chen@ttu.edu

Grader: zulfi.khan@ttu.edu

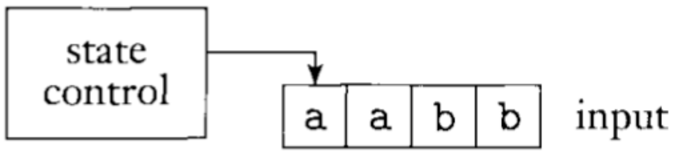
TEXAS TECH
UNIVERSITY

Pushdown automata

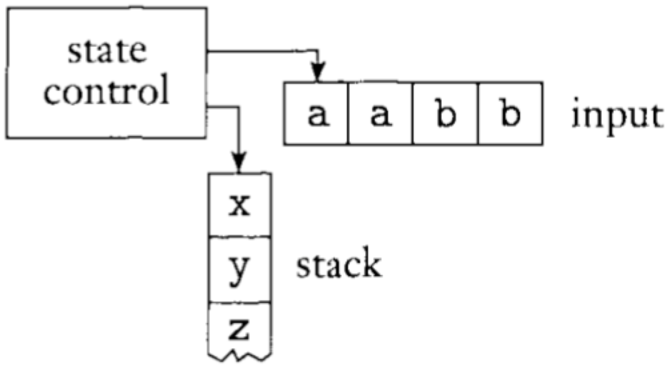
- Regular expressions are string generators
- Finite Automata (DFA, NFA) are string acceptors of REG
- CFGs are string generators
- What is the string acceptor of CFG?
  - Pushdown automata

Pushdown automata

Finite automata:

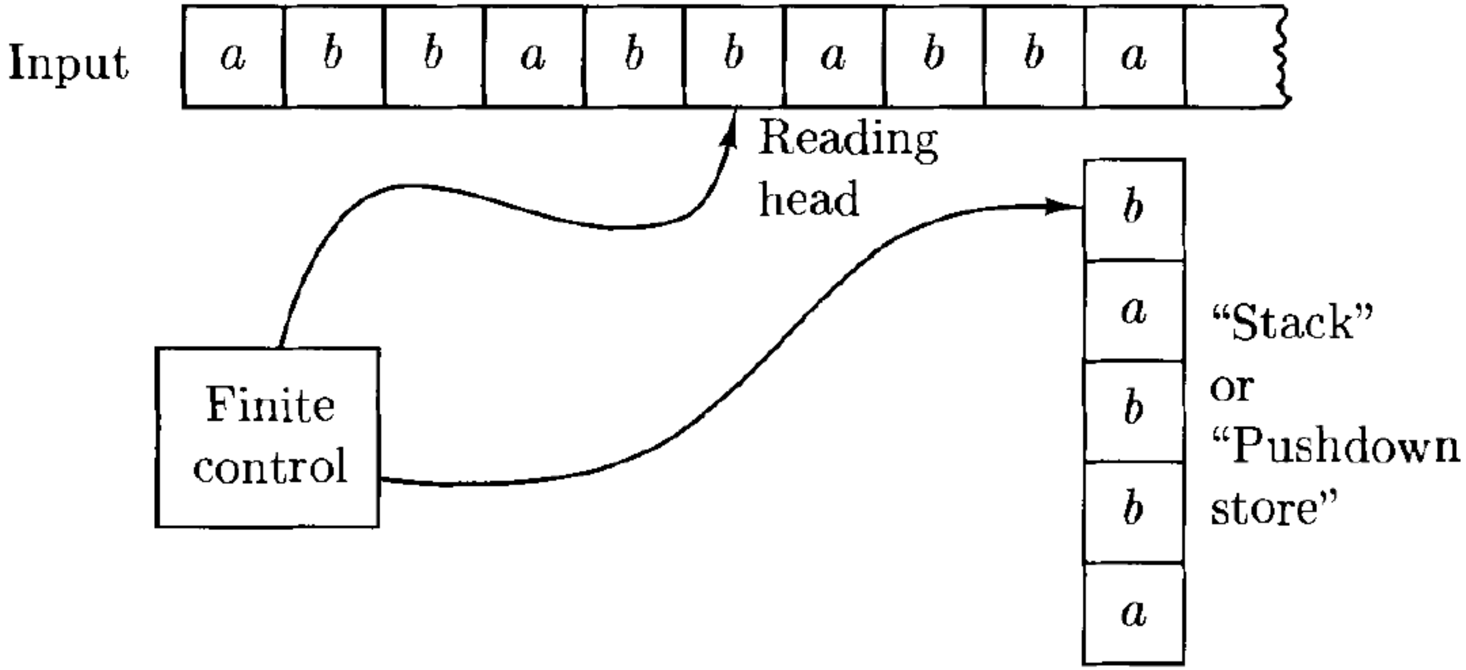


Pushdown automata:



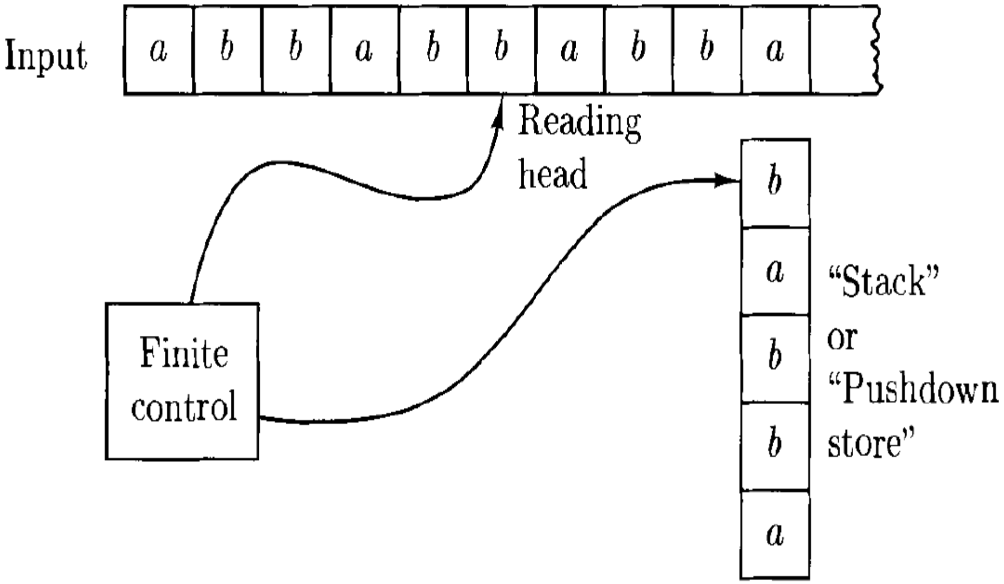
Pushdown automata

- Finite automata cannot accept $\{ww^R : w \in \{a,b\}^*\}$ because it requires some memory
- We can use a stack as memory



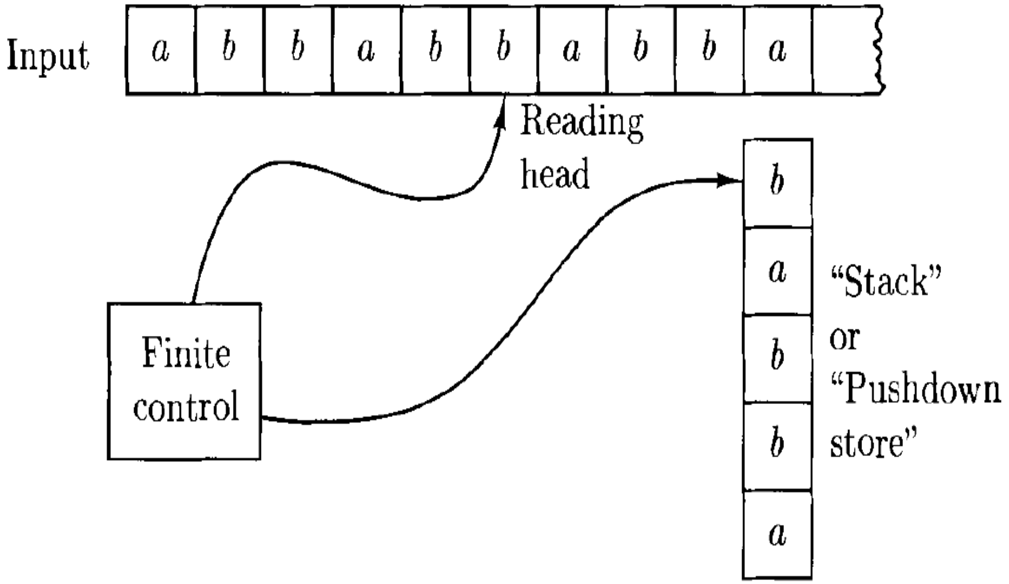
Pushdown automata

- Finite automata cannot accept $\{ww^R : w \in \{a,b\}^*\}$ because it requires some memory
- We can use a stack as memory



Pushdown automata

- Finite automata cannot accept $\{ww^R : w \in \{a,b\}^*\}$ because it requires some memory
- We can use a stack as memory

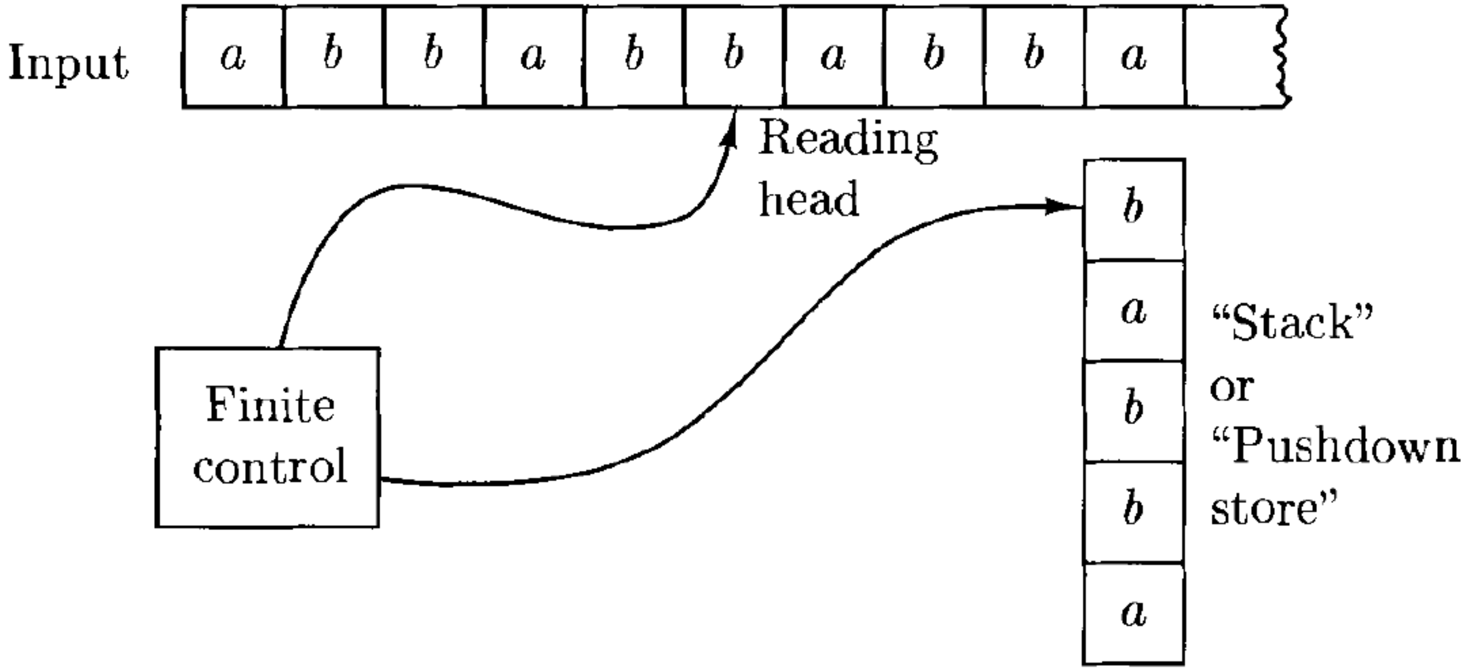


Writing a symbol on stack: Push
Removing a symbol from stack: pop

Pushdown automata

- How a stack is used?

  - We continue pushing symbols into stack, and then let it pop out at a suitable time.

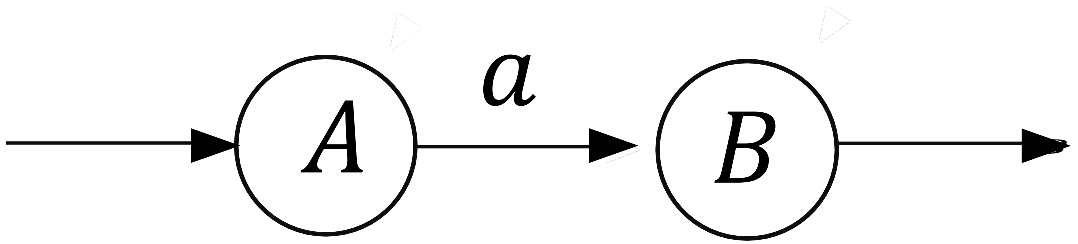


Pushdown automata

- How a stack is used?

  - We continue pushing symbols into stack, and then let it pop out at a suitable time.

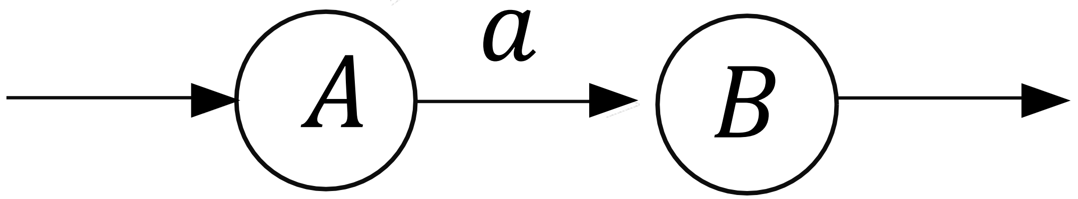
  - $A \to aB$ can be easily simulated by a DFA



Pushdown automata

- How a stack is used?

- $A \to aB$ can be easily simulated by a DFA



- What about $A \to aBb$
  - After we reach the final state from B, we need to "remember" to append an a
  - We push b to the stack, and eventually b will pop up

Pushdown automata

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

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

Equivalent definition through transition relation Δ

Pushdown automata

- How PDA (Pushdown automata) works?
- If $\big((p,a,\beta),(q,\gamma)\big) \in \Delta$, then the PDA M, once it is in state p with β at the top of the stack, may read a from he input tape, replace β by γ on top of the stack, and enter state q.

  - $\big((p,a,e),(q,\gamma)\big)$ reads a and pushes γ
  - $\big((p,a,\gamma),(q,e)\big)$ reads a and pops γ

Pushdown automata

- Configuration
  - $(q, w, \lambda) \in K \times \Sigma^* \times \Gamma^*$
  - current state q
  - the remainder of the string w
  - strings consisting of the stack symbol in the stack, top-down

Pushdown automata

- Yields (in one step)
  - $(p, x, \alpha) \vdash_M (q, y, \zeta)$ if exists transition $\big((p, a, \beta), (q, \gamma)\big) \in \Delta$
    - $x = ay$
    - $\alpha = \beta\eta$ and $\zeta = \gamma\eta$ for some $\eta \in \Gamma^*$
  - $\vdash_M^*$ indicates a sequence of $\vdash_M$ (reflexive and transitive closure)

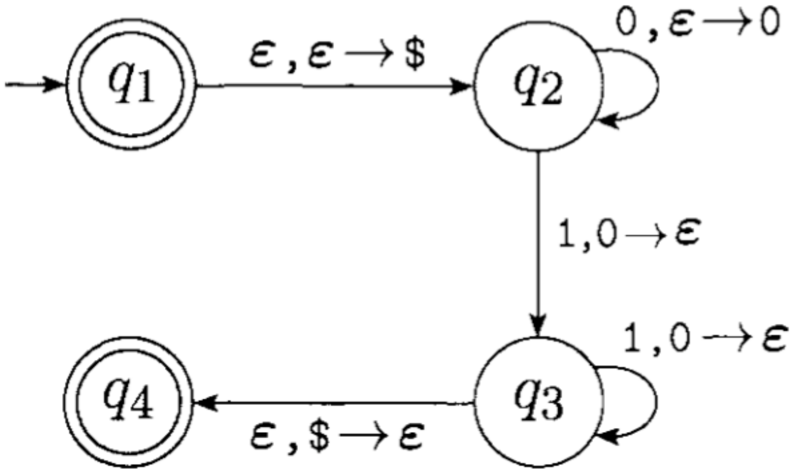
Pushdown automata

- Acceptance

  \- PDA M accepts a string $w \in \Sigma^*$ if and only if $(s, w, e) \vdash_M^* (f, e, e)$ for some final state $f \in F$

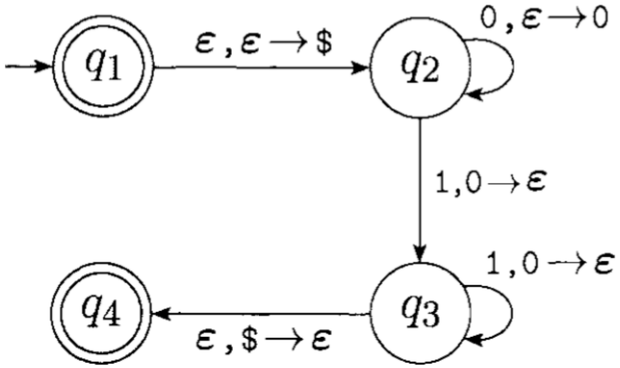
Pushdown automata-example

State diagram for the PDA M_1 that recognizes $\{0^n 1^n \mid n \geq 0\}$



Pushdown automata-example

State diagram for the PDA M_1 that recognizes $\{0^n 1^n \mid n \geq 0\}$



$Q = \{q_1, q_2, q_3, q_4\}$,

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

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

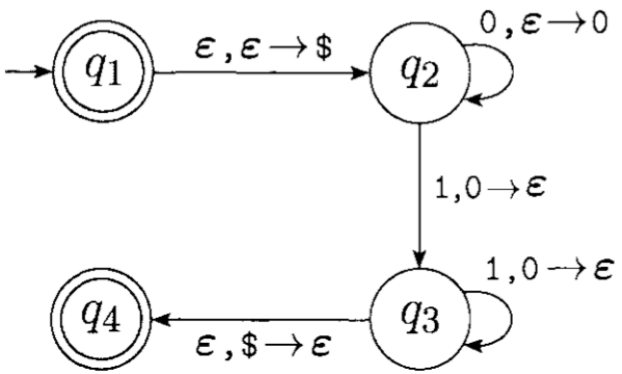
$F = \{q_1, q_4\}$, and

δ is given by the following table, wherein blank entries signify $\emptyset$.

Input:	0			1			ε		
Stack:	0	\$	ε	0	\$	ε	0	\$	ε
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \varepsilon)\}$					
q_3				$\{(q_3, \varepsilon)\}$				$\{(q_4, \varepsilon)\}$	
q_4									

Pushdown automata-example

State diagram for the PDA M_1 that recognizes $\{0^n1^n \mid n \geq 0\}$



$Q = \{q_1, q_2, q_3, q_4\}$,

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

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

$F = \{q_1, q_4\}$, and

δ is given by the following table, wherein blank entries signify $\emptyset$.

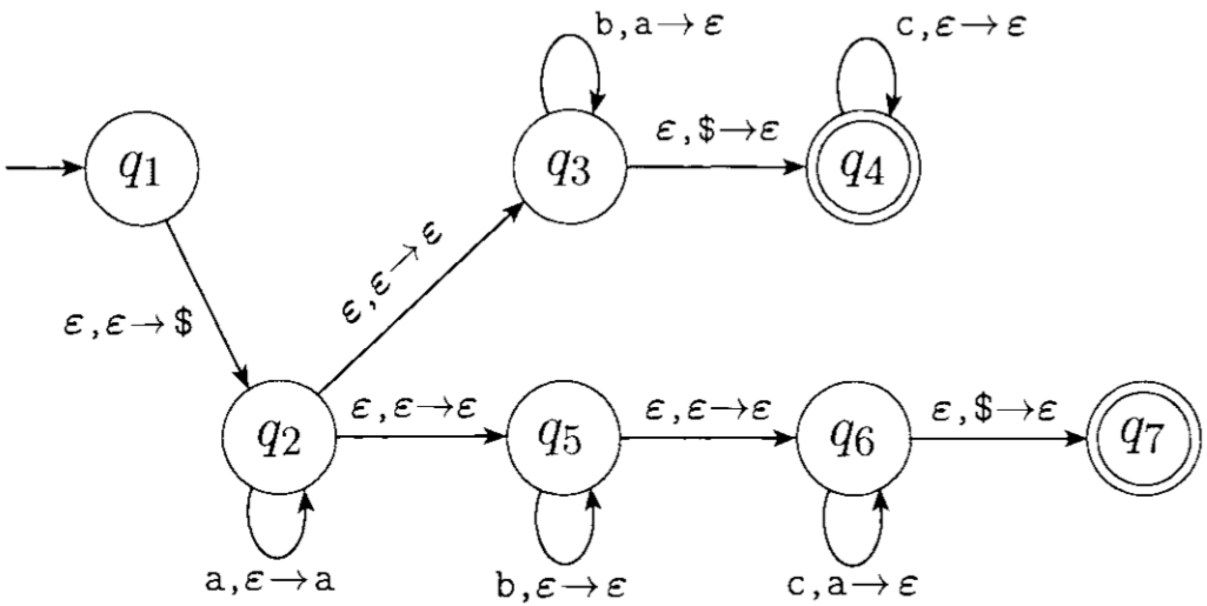
Input:	0			1			ε		
Stack:	0	\$	ε	0	\$	ε	0	\$	ε
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \varepsilon)\}$					
q_3				$\{(q_3, \varepsilon)\}$				$\{(q_4, \varepsilon)\}$	
q_4									

Equivalent description:

$\Delta = \{((q_1, \epsilon, \epsilon), (q_2, \$)) \ldots\}$

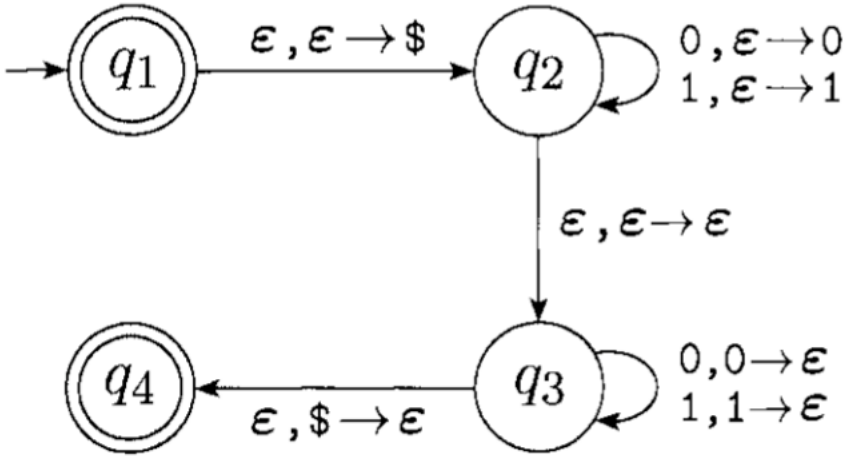
Pushdown automata-example

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



Pushdown automata-example

State diagram for the PDA M_3 that recognizes $\{ww^{\mathcal{R}} \mid w \in \{0,1\}^*\}$



PDA and CFG

- PDA serves as a checker for context free language

Theorem: A language is context free if and only if some pushdown automaton recognizes it.

PDA and CFG

- PDA serves as a checker for context free language

Theorem: A language is context free if and only if some pushdown automaton recognizes it.

Prove two directions:

If a language is context free, then some pushdown automaton recognizes it.

If a pushdown automaton recognizes some language, then it is context free.

PDA and CFG

If a language is context free, then some pushdown automaton recognizes it.

We need a more generalized, but essentially equivalent PDA definition.

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

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

PDA and CFG

If a language is context free, then some pushdown automaton recognizes it.

We need a more generalized, but essentially equivalent PDA definition.

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

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

Replacing Γ_ϵ with Γ^*

PDA and CFG

If a language is context free, then some pushdown automaton recognizes it.

We need a more generalized, but essentially equivalent PDA definition.

What is the difference between $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow P(Q \times \Gamma_\epsilon)$ and $\delta: Q \times \Sigma_\epsilon \times \Gamma^* \rightarrow P(Q \times \Gamma^*)$?

PDA and CFG

If a language is context free, then some pushdown automaton recognizes it.

We need a more generalized, but essentially equivalent PDA definition.

What is the difference between $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow P(Q \times \Gamma_\epsilon)$ and $\delta: Q \times \Sigma_\epsilon \times \Gamma^* \rightarrow P(Q \times \Gamma^*)$?

We are now allowed to simultaneously replace a bunch of top symbols on the stack with another bunch.

For example, $\big((p, a, \alpha\beta), (q, \gamma\zeta)\big)$ is now allowed

PDA and CFG

$$M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$$

- Two states, $\{p, q\}$
- Stack alphabet = terminals + nonterminals
- Transitions:

(1) $((p, e, e), (q, S))$
(2) $((q, e, A), (q, x))$ for each rule $A \to x$ in R.
(3) $((q, a, a), (q, e))$ for each $a \in \Sigma$.

PDA and CFG

Example 3.4.1: Consider the grammar $G = (V, \Sigma, R, S)$ with $V = \{S, a, b, \mathrm{c}\}$, $\Sigma = \{a, b, c\}$, and $R = \{S \to aSa, S \to bSb, S \to c)$, which generates the language $\{wcw^R : w \in \{a,b\}^*\}$. The corresponding pushdown automaton, according to the construction above, is $M = (\{p,q\}, \Sigma, V, \Delta, p, \{q\})$, with

$$\begin{aligned}
\Delta = \{&((p,e,e),(q,S)), && (T1)\\
&((q,e,S),(q,aSa)), && (T2)\\
&((q,e,S),(q,bSb)), && (T3)\\
&((q,e,S),(q,c)), && (T4)\\
&((q,a,a),(q,e)), && (T5)\\
&((q,b,b),(q,e)), && (T6)\\
&((q,\mathrm{c},\mathrm{c}),(q,e))\} && (T7).
\end{aligned}$$

PDA and CFG

If a language is context free, then some pushdown automaton recognizes it.

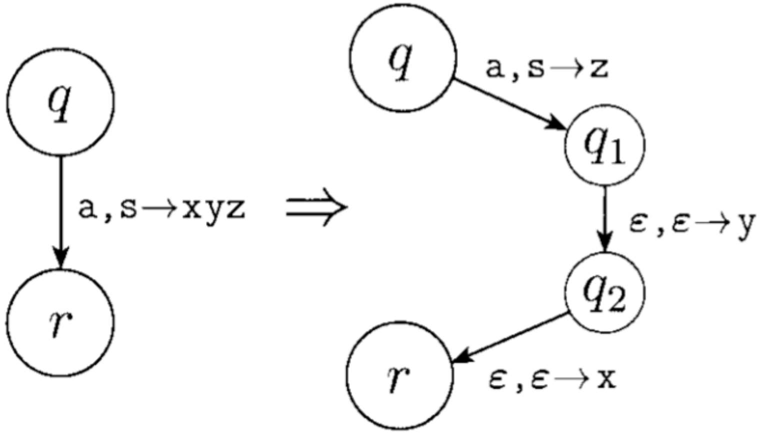
We need a more generalized, but essentially equivalent PDA definition.

What is the difference between $\delta: Q\times\Sigma_\epsilon\times\Gamma_\epsilon \rightarrow P(Q\times\Gamma_\epsilon)$ and $\delta: Q\times\Sigma_\epsilon\times\Gamma^* \rightarrow P(Q\times\Gamma^*)$?

We are now allowed to simultaneously replace a bunch of top symbols on the stack with another bunch.

For example, $\big((p, a, \alpha\beta), (q, \gamma\zeta)\big)$ is now allowed

We can simulate this with a normal PDA.

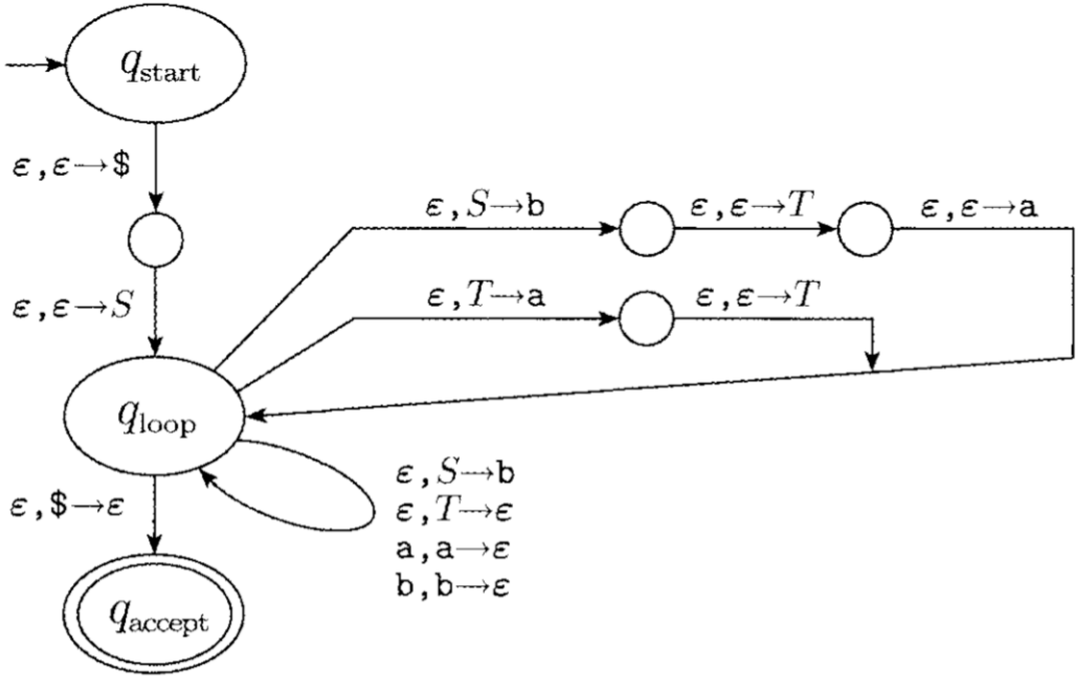


PDA and CFG

If a language is context free, then some pushdown automaton recognizes it.

Example:

$$S \rightarrow \mathtt{a}T\mathtt{b} \mid \mathtt{b}$$
$$T \rightarrow T\mathtt{a} \mid \varepsilon$$



PDA and CFG

If a pushdown automaton recognizes some language, then it is context free.

Given a PDA, modify it such that:

1. It has a single accept state, q_{accept}.
2. It empties its stack before accepting.
3. Each transition either pushes a symbol onto the stack (a *push* move) or pops one off the stack (a *pop* move), but it does not do both at the same time.

PDA and CFG

If a pushdown automaton recognizes some language, then it is context free.

Given a PDA, modify it such that:

It has a single accept state, q_{accept}.

Each transition either pushes a symbol onto the stack (a *push* move) or pops one off the stack (a *pop* move), but it does not do both at the same time.

If $\big((p, a, \beta), (q, \gamma)\big) \in \Delta$, replace it with $\big((p, a, \beta), (p', e)\big)$ and $\big((p', e, e), (q, \gamma)\big)$

PDA and CFG

If a pushdown automaton recognizes some language, then it is context free.

We design a CFG such that A_{pq} generates all strings that take M from state p to state q, starting and ending with empty stack.

PDA and CFG

If a pushdown automaton recognizes some language, then it is context free.

We design a CFG such that A_{pq} generates all strings that take M from state p to state q, starting and ending with empty stack.

- When M reads any string of A_{pq}, the first move is push, the last move is pop.

PDA and CFG

If a pushdown automaton recognizes some language, then it is context free.

We design a CFG such that A_{pq} generates all strings that take M from state p to state q, starting and ending with empty stack.

- When M reads any string of A_{pq}, the first move is push, the last move is pop.
- If the first push and last pop is the same symbol, add $A_{pq} \to aA_{rs}b$

PDA and CFG

If a pushdown automaton recognizes some language, then it is context free.

We design a CFG such that A_{pq} generates all strings that take M from state p to state q, starting and ending with empty stack.

- When M reads any string of A_{pq}, the first move is push, the last move is pop.
- If the first push and last pop is the same symbol, add $A_{pq} \rightarrow aA_{rs}b$
- If the first push and last pop is different, add $A_{pq} \rightarrow A_{pr}A_{rq}$

PDA and CFG

If a pushdown automaton recognizes some language, then it is context free.

Formal construction:

- If $\big((p,a,e),(r,\beta)\big)$, $((s,b,\beta),(q,e)) \in \Delta$, add rule $A_{pq} \to aA_{rs}b$
- For all states p,r,q, add $A_{pq} \to A_{pr}A_{rq}$
- For all state p, add $A_{pp} \to e$

CFG and REG

Corollary: Every regular language is context free.

Why it is a corollary?