Pushdown Automata (PDA)

Miguel Ballesteros

Algorithms for NLP Course.
7-11

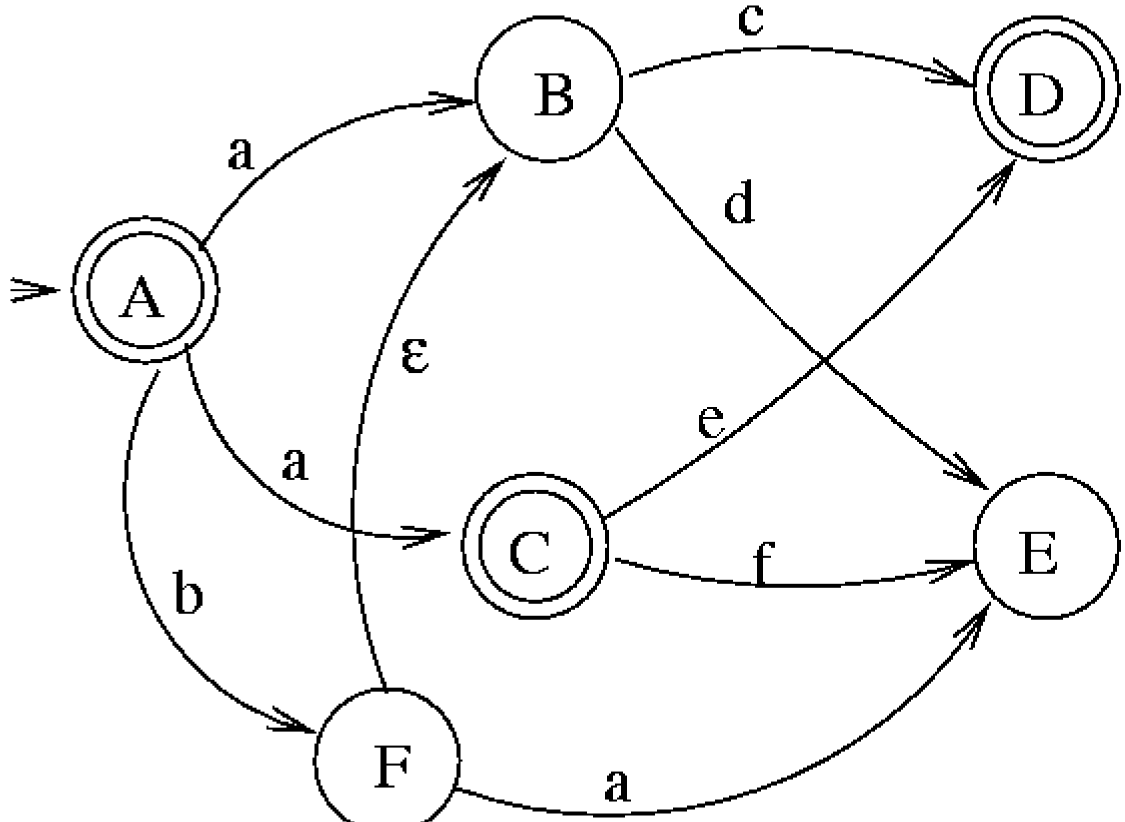
Carnegie Mellon

Review from last day

- CFGs and CFLs
  - CFGs. Context-free grammars when its production rules can be applied regardless of the context of a nonterminal.
  - CFLs. Context-free languages. Languages recognized by CFGs.

Review from last month

- FSAs: finite state automata
  - A finite-state automaton is a device that can be in one of a finite number of states



Pushdown Automata (PDA)

- PDAs: a more powerful computation device that can <u>recognize</u> CFLs.
- PDA: a ϵ-NFA with a "**stack**" which serves as "memory", and store a string of stack symbols.

- LR ↔ FSA
- CFL ↔ PDA

- The general non-deterministic version accepts ALL CFLs.
- The deterministic version accepts only a subsets of the CFLs.

Formal definition of a PDA

- $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

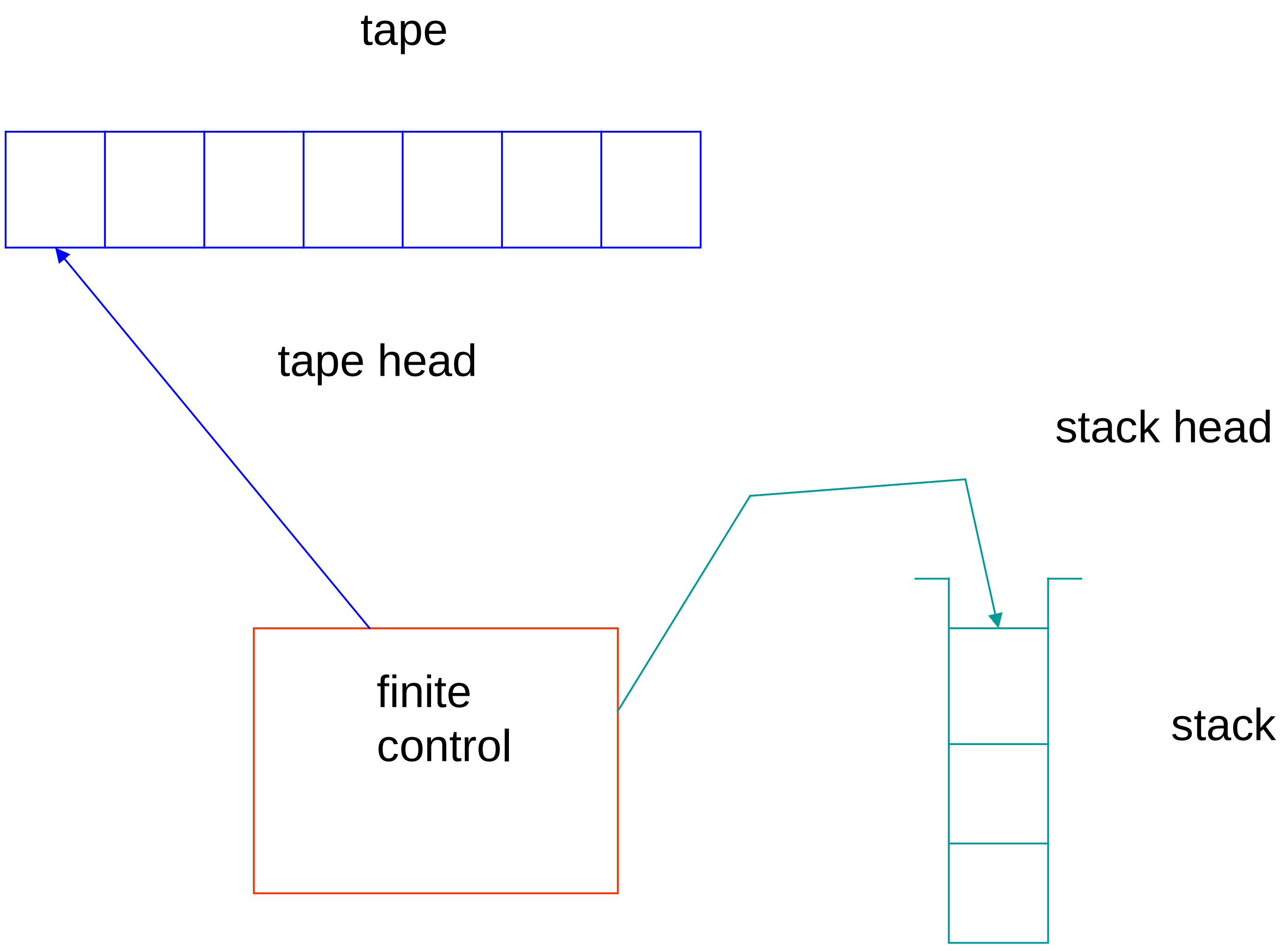
Q, Σ, q_0, F are the same as in FSAs. Which is?

Γ: set of stack symbols (finite)
(denoted using X, Y, Z)

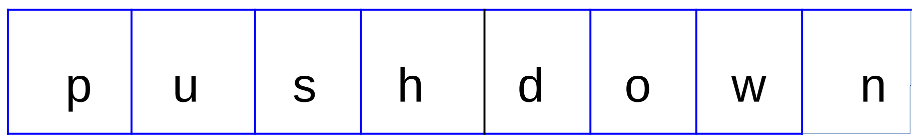
z_0: start stack symbol.

$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$ transitions

Pushdown Automata (PDA)

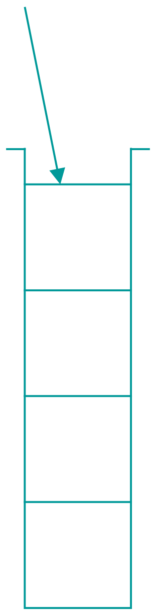


Pushdown Automata (PDA)



The **tape** is divided into finitely many cells. Each cell contains a symbol in an alphabet Σ.

Pushdown Automata (PDA)



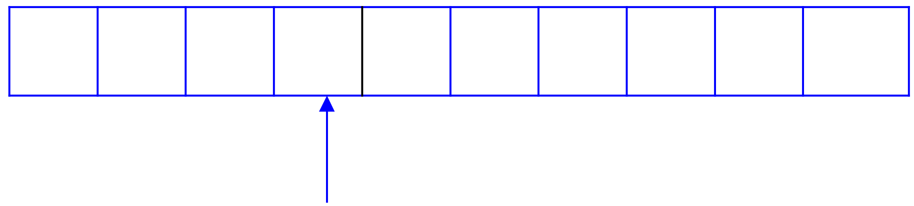
The **stack** head always scans the top symbol of the stack. It performs two basic operations:

Push: add a new symbol at the top.

Pop: read and remove the top symbol.

Alphabet of stack symbols: Γ

Pushdown Automata (PDA)



- The **head/pointer** scans at a cell on the **tape** and can *read* a symbol on the cell. In each move, the head can move to the right cell.

Pushdown Automaton (PDA)

- Transitions depends on
  - **Current state.**
  - **Input.**

    (these two, the same as FSAs)
  - **Top of the stack.**

    (which is the new thing)

Pushdown Automaton (PDA)

- After each transition:
  - **New state.**
  - **Pop the stack :** є
  - **And push the stack with new symbol.**

Pushdown Automaton (PDA)

- In one transition the PDA may do the following:
  - Consume the input symbol. If ϵ is the input symbol, then no input is consumed.
  - Go to a new state, which may be the same as the previous state.
  - Replace the symbol at the top of the stack by any string.
    - If this string is ϵ then this is a **pop** of the stack
    - The string might be the same as the current stack top (does **nothing**)
    - Replace with a new string (**pop** and **push**)
    - Replace with multiple symbols (multiple **pushes**)

PDA. Simple example

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

– Grammar for L:

$S \rightarrow 0\ S\ 1 \mid \epsilon$

- L is not regular. (see 9/8 by Chris)
  - If it is not regular a finite automaton is unable to recognize this language because it cannot store an arbitrarily large number of values in its finite memory.

- **Is there a PDA for L?**

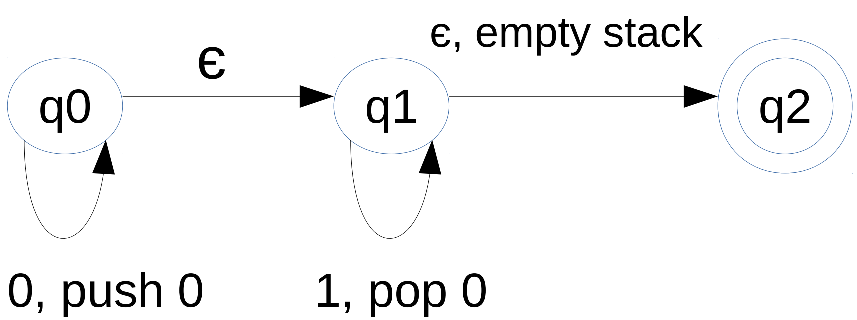
PDA. Simple example

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

- A PDA is able to recognize this language!
  - Can use its stack to store the number of 0's it has seen.
    - As each 0 is read, **push** it onto the stack
    - As soon as 1's are read, **pop** a 0 off the stack
    - If reading the input is finished exactly when the stack is empty, **accept** the input else **reject** the input

PDA. Simple example

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



- q0: push 0.
- q1: pop all 0s while reading 1s.
- q2: accept when we have an empty stack.

Some examples

PDA. Informal example

- $L = \{ w\, w^r \mid w \in \{a,b\}^* \}$

  This language is often referred as “w-w-reverse”, is the even-length *palindromes* over alphabet {a,b}

  CFG:

$$P \rightarrow \epsilon$$

$$P \rightarrow bPb$$

$$P \rightarrow aPa$$

PDA. Informal example

- $L = \{ w w^r \mid w \in \{a,b\}^* \}$

  PDA?

  - Start in a state **q0** that represents that we have not seen the end of w. While in **q0**, read symbols and store them on the stack.

PDA. Informal example

- $L = \{ w w^r \mid w \in \{a,b\}^* \}$

  PDA?

  – Start in a state **q0** that represents that we have not seen the end of w. While in **q0**, read symbols and store them on the stack.

  – At any time, we may go to a state **q1,** signifying that we are at the end of w. Since the automaton is not deterministic, we actually make both guesses: (1) stay in **q0** or (2) go to **q1.**

PDA. Informal example

- $L = \{ w w^r \mid w \in \{a,b\}^* \}$

  PDA?

  – Start in a state **q0** that represents that we have not seen the end of w. While in q0, read symbols and store them on the stack.

  – At any time, we may go to a state **q1,** signifying that we are at the end of w. Since the automaton is not deterministic, we actually make both guesses: (1) stay in **q0** or (2) go to **q1.**

  – Once in **q1,** we compare input symbols with the symbol at the top of the stack.

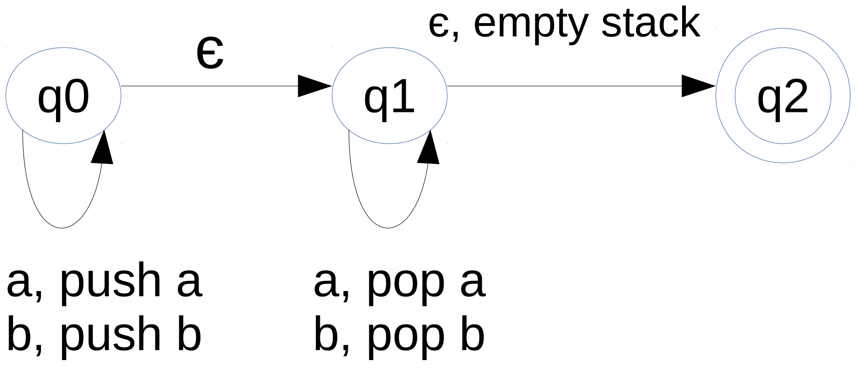
    - If they match, we consume the input symbol, pop the stack, and proceed.
    - If the do not match, this branch dies, although other branches of the nondeterministic automaton may reach the acceptance.

PDA. Informal example

- $L = \{ w\, w^r \mid w \in \{a,b\}^* \}$

  PDA?

  - Start in a state **q0** that represents that we have not seen the end of w. While in q0, read symbols and store them on the stack.
  - At any time, we may go to a state **q1,** signifying that we are at the end of w. Since the automaton is not deterministic, we actually make both guesses: (1) stay in **q0** or (2) go to **q1.**
  - Once in **q1,** we compare input symbols with the symbol at the top of the stack.
    - If they match, we consume the input symbol, **pop the stack**, and proceed.
    - If the do not match, this branch dies, although other branches of the nondeterministic automaton may reach the acceptance.
  - If we empty the stack, then we have indeed seen $w\, w^r$.
    We accept the input



Formal definition of a PDA

- $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

Q, Σ, q_0, F are the same as in FSAs.

Γ: set of stack symbols (finite)
(denoted using X, Y, Z)

z_0: start stack symbol.

$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$ transitions

Formal definition of a PDA

- Q: a finite set of states (like in finite automaton)
- Σ: a finite set of input symbols (like in finite automaton)
- Γ: a finite stack alphabet. The components that we are allowed to push onto the stack.
- δ: the transition function δ(q,a,X) where:
  - q is a state in Q.
  - a is either an input symbol in Σ or a = ϵ
  - X ϵ Γ (X is a stack symbol)
- q_0: the start state.
- Z_0: the start symbol,
- F: the set of accepting states.

Example - Formal

- Let's describe te PDA of L_{wwr}
- L = { w wr | w є {a,b}* }

$P=(\{q_0,q_1,q_2\}, \{a,b\}, \{a,b,Z_0\}, \delta, q_0, Z_0, \{q_2\})$

and δ is as follows:

1

$\delta(q_0,a,Z_0) = \{(q_0,aZ_0)\}$

$\delta(q_0,b,Z_0) = \{(q_0,bZ_0)\}$

One of these rules applies initially when we are in q_0 and we see the start symbol at the top of the stack.

Example - Formal

- Let's describe te PDA of L_{wwr}

  $P=(\{q_0,q_1,q_2\}, \{a,b\}, \{a,b,Z_0\}, \delta, q_0, Z_0, \{q_2\})$

  and δ is as follows:

  1

  $\delta(q_0,a,Z_0) = \{(q_0,aZ_0)\}$

  $\delta(q_0,b,Z_0) = \{(q_0,bZ_0)\}$

  2

  $\delta(q_0,a,a) = \{(q_0,aa)\}$, $\delta(q_0,b,b) = \{(q_0,bb)\}$,
  $\delta(q_0,b,a) = \{(q_0,ba)\}$, $\delta(q_0,a,b) = \{(q_0,ab)\}$

  These four similar rules allow us to stay in q_0 and read inputs, pushing each onto the top of the stack, and leaving the previous stack symbol alone.

Example - Formal

- Let's describe te PDA of L_{wwr}

  $P=(\{q_0,q_1,q_2\}, \{a,b\}, \{a,b,Z_0\}, \delta, q_0, Z_0, \{q_2\})$

  and δ is as follows:

  1

  $\delta(q_0,0,Z_0) = \{(q_0,0Z_0)\}$

  $\delta(q_0,1,Z_0) = \{(q_0,1Z_0)\}$

  2

  $\delta(q_0,a,a) = \{(q_0,aa)\}$, $\delta(q_0,b,b) = \{(q_0,bb)\}$,
  $\delta(q_0,b,a) = \{(q_0,ba)\}$, $\delta(q_0,a,b) = \{(q_0,ab)\}$

  3

  $\delta(q_0,\epsilon,Z_0) = \{(q_1,Z_0)\}$, $\delta(q_0,\epsilon,a) = \{(q_1,a)\}$ and $\delta(q_0,\epsilon,b) = \{(q_1,b)\}$

  These three rules allow us to go from q_0 to state q_1 spontaneously (on ϵ input)

Example - Formal

- Let's describe te PDA of L_{wwr}

  $P=(\{q_0,q_1,q_2\}, \{a,b\}, \{a,b,Z_0\}, \delta, q_0, Z_0, \{q_2\})$

  and δ is as follows:

  1

  $\delta(q_0,a,Z_0) = \{(q_0,aZ_0)\}$

  $\delta(q_0,b,Z_0) = \{(q_0,bZ_0)\}$

  2

  $\delta(q_0,a,a) = \{(q_0,aa)\}$, $\delta(q_0,b,b) = \{(q_0,bb)\}$,
  $\delta(q_0,b,a) = \{(q_0,ba)\}$, $\delta(q_0,a,b) = \{(q_0,ab)\}$

  3

  $\delta(q_0,\epsilon,Z_0) = \{(q_1,Z_0)\}$, $\delta(q_0,\epsilon,0) = \{(q_1,a)\}$ and $\delta(q_0,\epsilon,b) = \{(q_1,b)\}$

  4

  $\delta(q_1,a,a) = \{(q_1,\epsilon)\}$, $\delta(q_1,b,b) = \{(q_1,\epsilon)\}$

  In state q_1 we can match input symbols against the top of the stack and **pop** when they match.

Example - Formal

- Let's describe te PDA of L_{wwr}

  $P=(\{q_0,q_1,q_2\}, \{a,b\}, \{a,b,Z_0\}, \delta, q_0, Z_0, \{q_2\})$

  and δ is as follows:

  1

  $\delta(q_0,a,Z_0) = \{(q_0,aZ_0)\}$

  $\delta(q_0,b,Z_0) = \{(q_0,bZ_0)\}$

  2

  $\delta(q_0,a,a) = \{(q_0,aa)\}$, $\delta(q_0,b,b) = \{(q_0,bb)\}$,
  $\delta(q_0,b,a) = \{(q_0,ba)\}$, $\delta(q_0,a,b) = \{(q_0,ab)\}$

  3

  $\delta(q_0,\epsilon,Z_0) = \{(q_1,Z_0)\}$, $\delta(q_0,\epsilon,0) = \{(q_1,a)\}$ and $\delta(q_0,\epsilon,b) = \{(q_1,b)\}$

  4

  $\delta(q_1,a,a) = \{(q_1,\epsilon)\}$, $\delta(q_1,b,b) = \{(q_1,\epsilon)\}$

  5

  $\delta(q_1,\epsilon,Z_0) = \{(q_2,Z_0)\}$

  Finally, if we see the bottom of the stack marker Z_0 and we are in state q_1 we have found a palindrome wwr. We, thus go to q_2 and accept.

Example - Formal

- Let's describe te PDA of L_{wwr}

  $P=(\{q_0,q_1,q_2\}, \{a,b\}, \{a,b,Z_0\}, \delta, q_0, Z_0, \{q_2\})$

  and δ is as follows:

  1

  $\delta(q_0,a,Z_0) = \{(q_0,aZ_0)\}$

  $\delta(q_0,b,Z_0) = \{(q_0,bZ_0)\}$

  2

  $\delta(q_0,a,a) = \{(q_0,aa)\}$, $\delta(q_0,b,b) = \{(q_0,bb)\}$,
  $\delta(q_0,b,a) = \{(q_0,ba)\}$, $\delta(q_0,a,b) = \{(q_0,ab)\}$

  3

  $\delta(q_0,\epsilon,Z_0) = \{(q_1,Z_0)\}$, $\delta(q_0,\epsilon,0) = \{(q_1,a)\}$ and $\delta(q_0,\epsilon,b) = \{(q_1,b)\}$

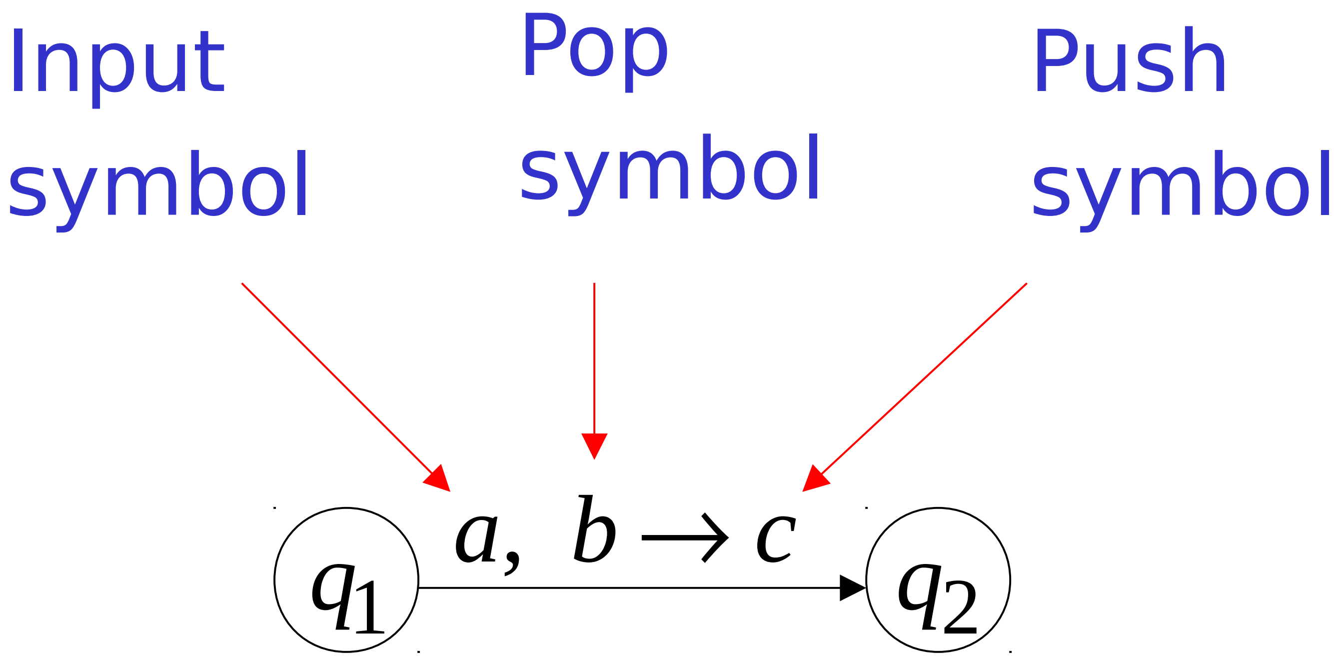
  4

  $\delta(q_1,a,a) = \{(q_1,\epsilon)\}$, $\delta(q_1,b,b) = \{(q_1,\epsilon)\}$

  5

  $\delta(q_1,\epsilon,Z_0) = \{(q_2,Z_0)\}$

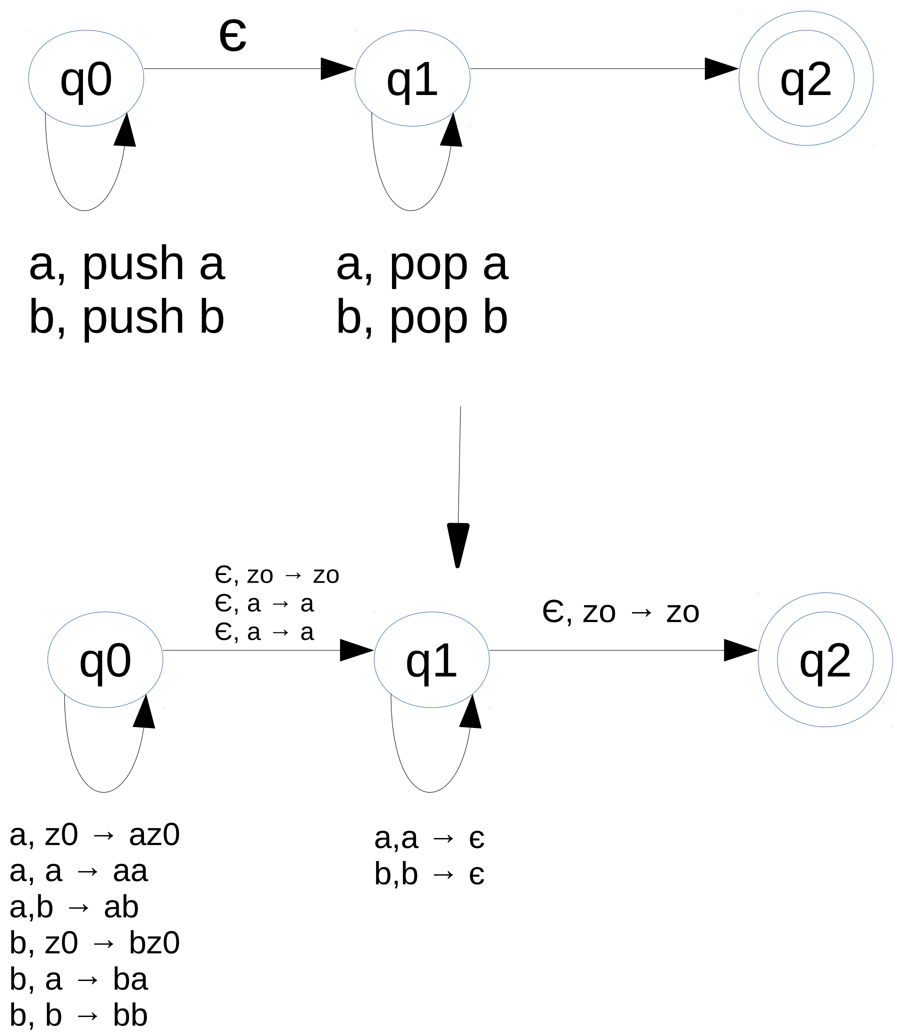
Pushdown Automata (PDA)



Transition

Pushdown Automata (PDA)

L = { w wr | w є {a,b}* }



More examples

See blackboard.

Two different kind of PDAs

- Empty stack PDA:
  - Sometimes, it is easier to design a PDA that accepts if and only if it is capable of having an empty stack.
- Final state PDA.

Two different kind of PDAs

- Empty stack PDA:
  - Sometimes, it is easier to design a PDA that accepts if and only if it is capable of having an empty stack.

- Final state PDA.

If there is a PDA P that accepts a language by **final state,** then there is another PDA P' that accepts the same language by **empty stack.**

Normally P ≠ P'

Translation between Empty Stack PDA to Final State PDA.

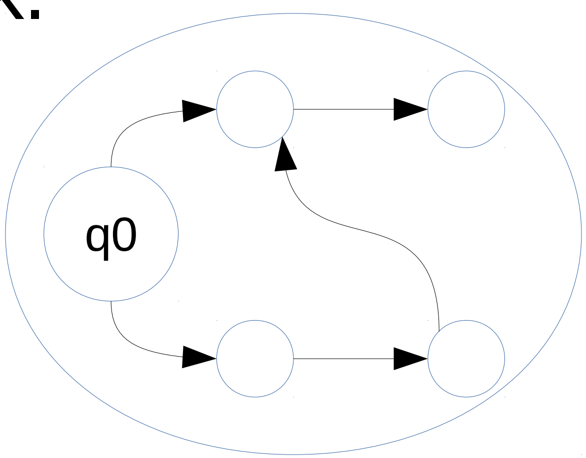
- Let Pv be a PDA that accepts by empty stack.

  $Pv = (Q, \Sigma, \Gamma, \delta, q_0, z_0)$

- Then, we define

  $Pf = (Q \cup \{q0F, qF\}, \Sigma, \Gamma \cup \{z0F\}, \delta F, q_0, z_0, \{qF\})$

  that accepts by final state the same strings as Pv by empty stack.



Translation between Empty Stack PDA to Final State PDA.

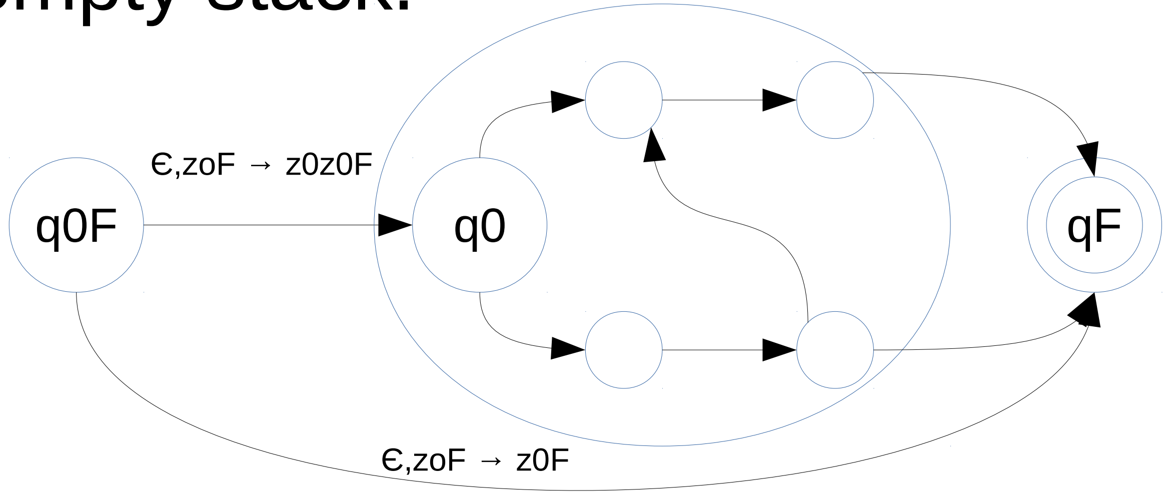
- Let Pv be a PDA that accepts by empty stack.

  $Pv = (Q, \Sigma, \Gamma, \delta, q_0, z_0)$

- Then, we define

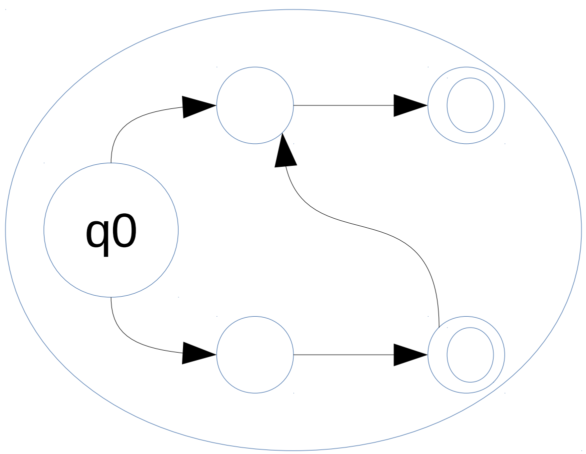
  $Pf = (Q \cup \{q0F, qF\}, \Sigma, \Gamma \cup \{z0F\}, \delta F, q_0, z_0, \{qF\})$

  that accepts by final state the same strings as Pv by empty stack.



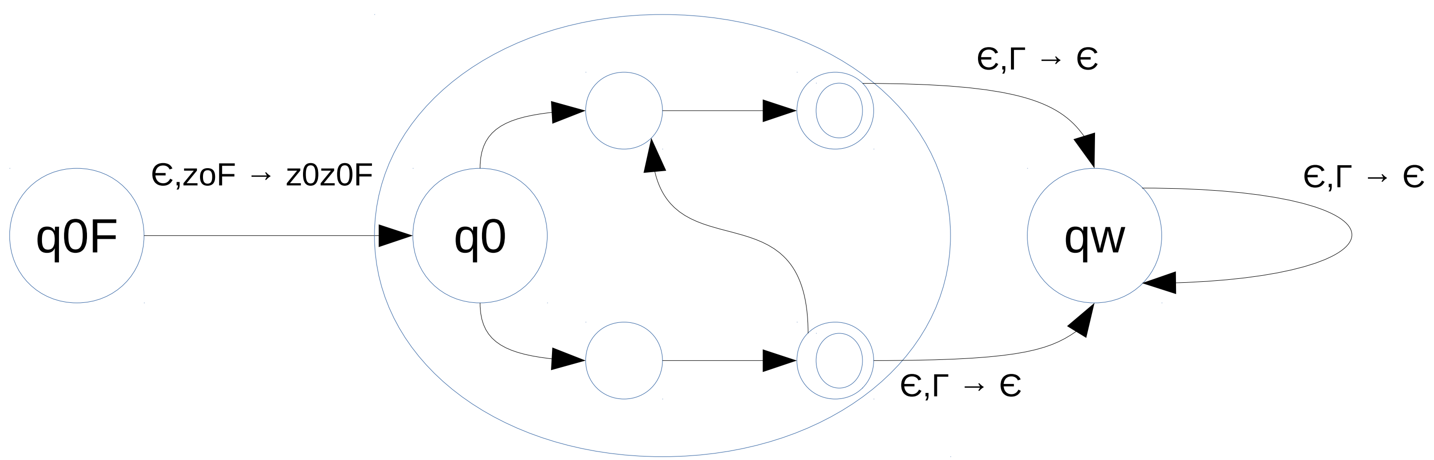
Translation between Final State PDA to Empty Stack PDA.

- We need a state that is in charge of making the stack empty.



Translation between Final State PDA to Empty Stack PDA.

- We need a state that is in charge of making the stack empty.



Equivalence of PDA's and CFG's

- Let's demonstrate that the languages defined by PDAs are exactly CFLs.
- Let's prove that these three are the same:

  1. CFLs (languages defined by CFGs).

  2. Languages accepted by final state by some PDA.

  3. Languages accepted by empty stack by some PDA.

Overview

- When we talked about closure properties of regular languages, it was useful to be able to jump between RE and DFA representations.
- Similarly, CFG's and PDA's are both useful to deal with properties of the CFL's.

Overview – (2)

- Also, PDA's, being "algorithmic," are often easier to use when arguing that a language is a CFL.
- Example: It is easy to see how a PDA can recognize balanced parentheses; not so easy as a grammar.
- But all depends on knowing that CFG's and PDA's both define the CFL's.

Equivalence of PDA's and CFG's

- Let's demonstrate that the languages defined by PDAs are exactly CFLs.
- Let's prove that these three are the same:
  1. CFLs (languages defined by CFGs).
  2. Languages accepted by final state by some PDA.
  3. Languages accepted by empty stack by some PDA.
- We have already shown that 2 and 3 are the same.

Converting a CFG to a PDA

- Let L = L(G).
- Construct PDA P such that N(P) = L.
- P has:
  - One state q.
  - Input symbols = terminals of G.
  - Stack symbols = all symbols of G.
  - Start symbol = start symbol of G.

Intuition About the PDA P

- Given input w, P will step through a leftmost derivation of w from the start symbol S.
- Since P can't know what this derivation is, or even what the end of w is, it uses nondeterminism to "guess" the production to use at each step.

Intuition About the PDA P – (2)

- At each step, P represents some *left-sentential form* (step of a leftmost derivation).
- If the stack of P is α, and P has so far consumed x from its input, then P represents left-sentential form $x\alpha$.
- At empty stack, the input consumed is a string in L(G).

From CFG to PDA

- Given a CFG G, we construct a PDA that simulates the leftmost derivations of G.
- Any left sentential form that is not a terminal string can be written as $xA\alpha$, where
  - A is the leftmost variable.
  - x is whatever terminals appear to its left.
  - α is the string of terminals and variables that appear to the right of A.

From CFG to PDA

- Let G=(V,T,Q,S) be a CFG. Construct the PDA P that accepts L(G) by empty stack.
  - P=({q}, T, V U T, δ, q S)
  - Where δ is defined by:
    - For each variable A,

      δ (q, є, A) = {(q, β) | A → β is a production of G}

    - For each terminal a,

      δ(q,a,a) = {(q,є)}

An example

- See blackboard.

Summary

- Pushdown automata:
  - Like FSAs with an auxiliary stack.
  - Notation.
  - Examples.
  - PDA by final state.
  - PDA by empty stack.
  - Transformations between final state and empty stack, and the other way around.
  - Equivalence of PDA's and CFG's.