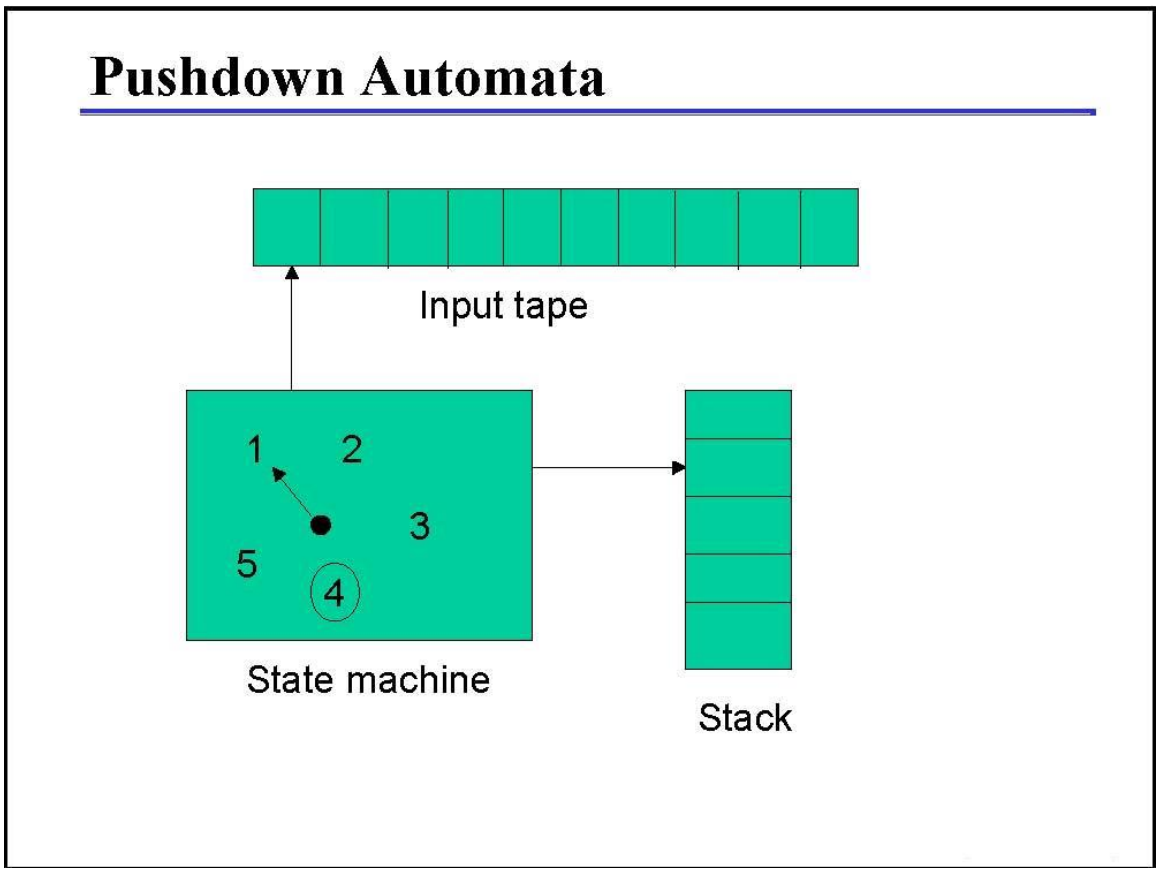
Pushdown Automata

Pushdown automata (PDA) recognizes CFL it is essentially like NDFA but have an extra component called stack. This extra component allows the automaton to have memory (in principle, infinite amount of memory), and to recognize some no regular languages.

PDA can write (push) a symbol on the top of stack or remove (pop) a symbol from the top the stack. The stack is unlimited and works as LIFO storage device.



A "move" of a PDA will depend on

1-Current state of machine

2- Current symbol in input tape

3- Current symbol on the top of stack

With each "move", the machine can

-Move into a new state

-Push symbol to the stack, remove or no any thing.

The stack

- The stack has its own alphabet (Γ)
- Included in this alphabet is a special symbol used to indicate an empty stack (Z_0)
- This special symbol should not be removed from the stack

To formalize:

– pushdown automata (PDA) is a 7-tuple:

M = (Q, ∑, Γ, q_0, z_0, A, δ) where

– Q = finite set of states

– ∑= tape alphabet

– Γ = stack alphabet

–$q_0 \in Q$ = start state

– $Z_0 \in \Gamma$= initial stack symbol

–$A \subseteq Q$ = set of accepting states

Transition function

- The transition function δ:
  - During a move of a PDA:
- At most one character is read from the input tape
  Transitions are okay (don't move the reading head)
- The topmost character is popped from the stack
  Unless it is Z_0
- The machine will move to a new state based on:
  - The character read from the tape
  - The character on the top of stack

- The current state of the machine

• Zero or more symbols from the stack alphabet are pushed onto the stack.

Transition function continued:

– δ: Q x ($\Sigma \cup \{\Lambda\}$) x $\Gamma \longrightarrow$ (finite subsets of Q x Γ^*)

Domain:

• Q = state
• $\sum \cup \{\Lambda\}$ = current symbol read from the tape
• Γ = current symbol on the top of stack

Range

• Q = new state
• Γ^* = symbols pushed on the stack

When the character Λ is read from the TAPE, it means that we are out of input letters. The Λ-edge will lead to ACCEPT if the state we have stopped in is a final state and to REJECT if the Process stops in a state that is not a final state.

Example

δ (q,a,a)=(p,aa)
. Meaning:
- when in state q,
- reading in an 'a' from the tape
- with an 'a' on the top of stack

The machine will
- go into state p
- push an 'a' on the top of stack

Configuration of a PDA

– (p, x, α) where

- p is the current state
- x is a string indicating what remains to be read on the tape
- α is the current contents of the stack.

The language accepted by a PDA:

There are two natural ways to define this language:

1- Define the language accepted to be the set of all inputs for which some of sequence of moves cause the PDA to empty stack. This language called "language accepted by empty stack"

2- Define the language accepted to be the set of all inputs for which some choice of moves causes the PDA to enter a final state.

First example

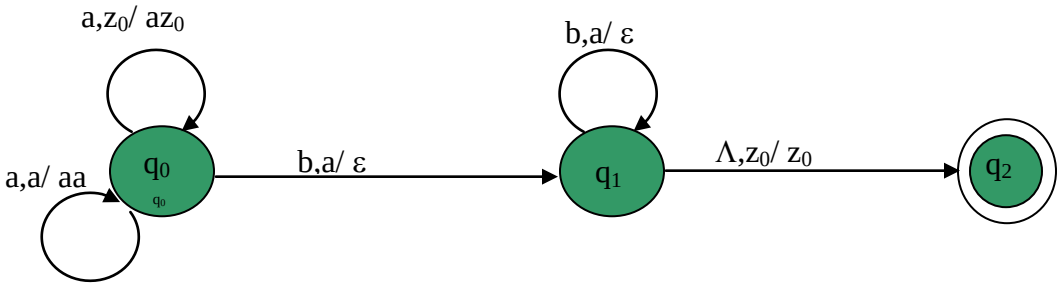
$L=\{a^ib^i, i>=1\}$

Basic idea for building a PDA:

- read chars "a" from the tape until you reach the "b".
- as you read chars "a" push them on the stack.
- when you read "b", check the top stack if "a" then pop it and move to the new state
- read chars "b" from the tape and check the top stack, if "a" then pop and stay at the same state, until the tape is empty and on the stack "z0" goto final state

Continuing first Example

$L=\{a^i b^i, i>=1\}$



Describe transition function:

$\delta(q_0, a, \delta) \rightarrow (q_0, az_0)$

$\delta(q_0, a, a) \rightarrow (q_0, a_0)$

$\delta(q_0, b, a) \rightarrow (q_1, \in)$

$\delta(q_1, b, a) \rightarrow (q_1, \in)$

$\delta(q_1, \in, z_0) \rightarrow (q_2, z_0)$

Transition for aabb

$(q_0, aabb, z_0) \rightarrow (q_0, abb, az_0)$ push a

$\rightarrow (q_0, bb, aa)$ push a

$\rightarrow (q_1, b, a)$ pop a

$\rightarrow (q_1, \in, z_0)$ pop a

$\rightarrow (q_2, \in, z_0)$ accept

***Transition for** abb*

$(q_0, aab, z_0) \rightarrow (q_0, bb, az_0)$ push a

$\rightarrow (q_1, b, z_0)$ pop a

$\rightarrow (q_1, b, z_0)$ Reject

Second example:

$L = \{ xcx^r \mid x \in \{ a,b \}^* \}$

-Basic idea for building a PDA:

- read chars from the tape until you reah the "c" (marker).
- as you read chars push them on the stack.
- after reading the "c", match the chars read with the chars on the stack, if match then pop char from stack, until all chars read.

-if at any point the char read does not match the char popped, the machine "fails".

Continuing our example:

– $L = \{ xcx^r \mid x \in \{ a,b \}^* \}$

– The PDA will have 4 states

• State 0 (initial) : reading before the 'c'

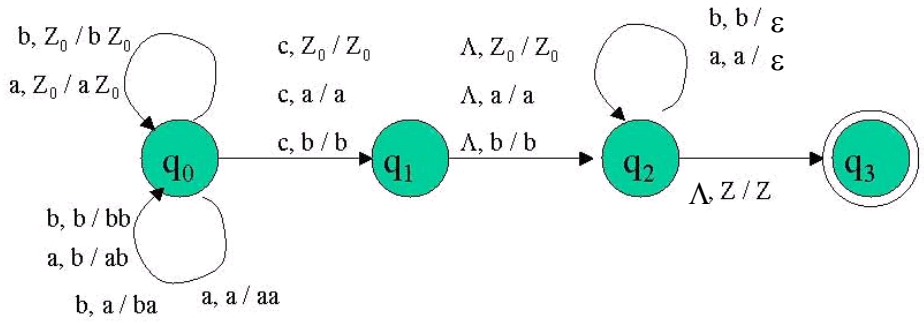
• State 1: read the 'c'

• State 2 :read after 'c', comparing chars

• State 3: (accepting): move only after all chars read and stack empty

Pushdown Automata

- Continuing our example:
  - $L = \{ xcx^r \mid x \in \{ a,b \}^* \}$



Describe transition function:

$\delta(q_0, a, z_0) \rightarrow (q_0, az_0)$

$\delta(q_0, b, z_0) \rightarrow (q_0, bz_0)$

$\delta(q_0, a, a) \rightarrow (q_0, aa)$

$\delta(q_0, b, b) \rightarrow (q_0, bb)$

$\delta(q_0, a, b) \rightarrow (q_0, ab)$

$\delta(q_0, b, a) \rightarrow (q_0, ba)$

$\delta(q_0, c, z_0) \rightarrow (q_1, z_0)$

$\delta(q_1, c, a) \rightarrow (q_1, a)$

$\delta(q_0, c, b) \rightarrow (q_1, b)$

$\delta(q_1, \wedge, z_0) \rightarrow (q_2, z_0)$

$\delta(q_1, \wedge, a) \rightarrow (q_2, a)$

$\delta(q_1, \wedge, b) \rightarrow (q_2, b)$

$\delta(q_2, b, b) \rightarrow (q_2, \in)$

$\delta(q_2, a, a) \rightarrow (q_2, \in)$

$\delta(q_2, \in, z_0) \rightarrow (q_3, z_0)$

***Transition for** abcba*

(q0,abcba,z0) $\rightarrow$ (q_0, bcba,az0) push a
$\rightarrow$ (q_0, cba, ba) push b
$\rightarrow$ (q_1, ba, ba) goto 2
$\rightarrow$ (q_2, ba, a)
$\rightarrow$ (q_2, a, z0) pop b
$\rightarrow$ (q_2, ε, z0) pop a
Accept

***Transition for** abcb*

(q0, abcba,z0) $\rightarrow$ □ (q_0,bcb,az0) push a
$\rightarrow$ □ (q_0, cb,ba) push b
$\rightarrow$ □ (q_1, b,ba) goto 2
$\rightarrow$ □□(q_2, b,ba)
$\rightarrow$ (q_2, ε,a) pop b
reject!

Another Example:

– $L = \{ xx^r \mid x \in \{a,b\}^* \}$

Basic idea for building a PDA

Much like last example, except

- this time we don't know when to start popping and comparing
- since PDAs are non-deterministic, this is not a problem

– The PDA will have 3 states

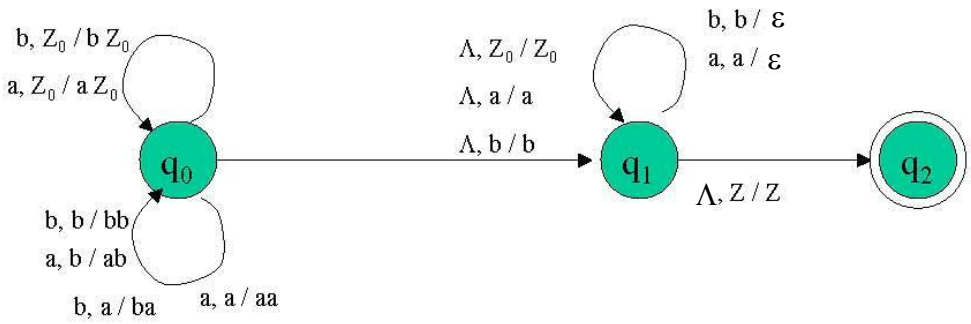
- State 0 (initial) : reading before the center of string
- State 1: read after center of string, comparing chars
- State 2 (accepting): after all chars read, stack should be empty

– The machine can choose to go from state 0 to state 1 at any time:

- Will result in many "wrong" set of moves
- All you need is one "right" set of moves for a string to be accepted.

Pushdown Automata

- Continuing another example:
  - $L = \{ xx^r \mid x \in \{ a,b \}^* \}$



Describe transition function:

$\delta(q_0, a, z_0) \rightarrow (q_0, az_0)$

$\delta(q_0, b, z_0) \rightarrow (q_0, bz_0)$

$\delta(q_0, a, a) \rightarrow (q_0, aa)$

$\delta(q_0, b, b) \rightarrow (q_0, bb)$

$\delta(q_0, a, b) \rightarrow (q_0, ab)$

$\delta(q_0, b, a) \rightarrow (q_0, ba)$

$\delta(q_0, \Lambda, z_0) \rightarrow (q_1, z_0)$

$\delta(q_1, \Lambda, a) \rightarrow (q_1, a)$

$\delta(q_0, \Lambda, b) \rightarrow (q_1, b)$

$\delta(q_1, b, b) \rightarrow (q_1, \in)$

$\delta(q_1, a, a) \rightarrow (q_1, \in)$

$\delta(q_1, \in, z_0) \rightarrow (q_2, z_0)$

Let's see a bad transition set for abba

– (q_0, abba, Z) ⊢ (q_0, bba, a) // push a
– ⊢ (q_0, ba, ba) // push b
– ⊢ (q_0, a, bba) // push b
– ⊢ (q_1, a, bba) // Λ transition
– Nowhere to go // Reject!

Let's see a good transition set for abba

– (q_0, abba, Z) ⊢ (q_0, bba, a) // push a
– ⊢ (q_0, ba, ba) // push b
– ⊢ (q_1, ba, ba) // Λtransition
⊢ (q_1, a, a) // pop b
– ⊢ (q_1, ε, Z) // pop a
– ⊢ (q_2, ε, Z) // Accept!

H.W

1-Build the PDAs that accept these languages:

a- L={ x ∈ {a, b}* | $n_a(x) > n_b(x)$}
b- L={$0^n 1^m$, n,m≥1,m=2n}

2- Formalize the PDA that accept the language

L={$a^i b^j$, i>j≥1}

My Best Wishes
Mohamed U. Mahdi