TAC
SEM IV

PUSHDOWN AUTOMATA

FINITE AUTOMATA

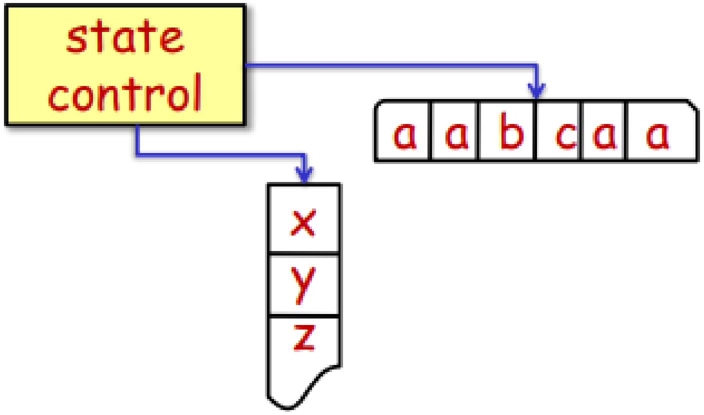
- A finite automaton has a huge limitation.
- It can count only by changing states.
- So an FA can distinguish between at most |Q| states.
- This limitation bounds the class of languages that an NFA can recognize to a rather small category - that of regular languages.

PDA

A pushdown automaton is a way to implement a context-free grammar in a similar way we design DFA for a regular grammar. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information.

Basically a pushdown automaton is −

"Finite state machine" + "a stack"



PDA

CFG

↓

CFL (DCFL & CFL)

↓

PDA

↙ ↘

DPDA NDPDA

WHY PDA ?

- DFAs accept regular languages.
- We want to design machines similar to DFAs that will accept context-free languages and is regular.
- A finite automation cannot accept string of the form (a^n,b^n) as it has to remember the no. of a's and so requires infinite no. of states.

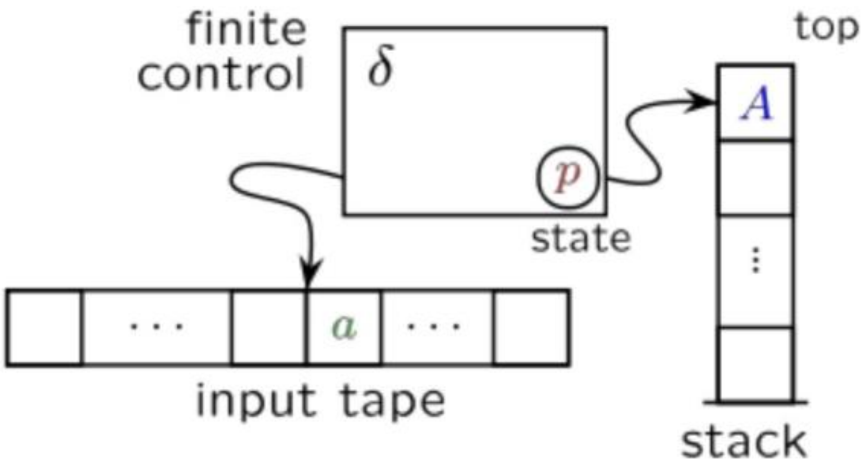
POWERS OF PDA

- This difficulty is avoided by adding a auxiliary memory in form of stack.
- It has a read only input tape and input alphabet.
- Final state control
- Set of final states
- Initial state (as in FA)
- Read write push down store.

COMPONENTS OF PDA

A pushdown automaton has three components −

- an input tape,
- a control unit, and
- a stack with infinite size.



The stack head scans the top symbol of the stack.

A stack does two operations −

Push − a new symbol is added at the top.

Pop − the top symbol is read and removed.

Nop – no operation performed

FORMAL DEFINITION OF PDA

A PDA can be formally described as a 7-tuple (Q, ∑, S, δ, q_0, Z, F) −

Q is the finite number of states

∑ is input alphabet

S is stack symbols

δ is the transition function: Q × (∑ ∪ {ε}) × S × Q × S*

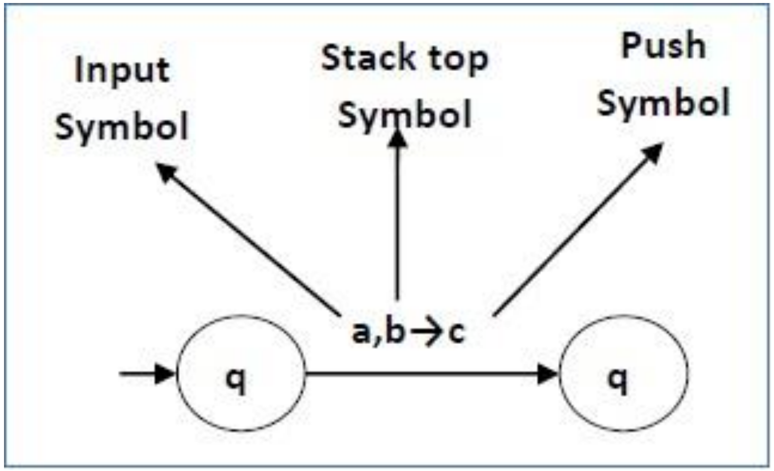
$\mathbf{q_0}$ is the initial state ($q_0 \in Q$)

$\mathbf{Z_0}$ is the initial stack top symbol ($Z_0 \in S$)

F is a set of accepting states ($F \in Q$)

FORMAL DEFINITION OF PDA

The following diagram shows a transition in a PDA from a state q_1 to state q_2, labeled as a,b → c −



This means at state $\mathbf{q_1}$, if we encounter an input string **'a'** and top symbol of the stack is **'b'**, then we pop **'b'**, push **'c'** on top of the stack and move to state $\mathbf{q_2}$.

Terminologies Related to PDA

Instantaneous Description: The instantaneous description (ID) of a PDA is represented by a triplet (q, w, s) where

- **q** is the state
- **w** is unconsumed input
- **s** is the stack contents

Turnstile Notation: The "turnstile" notation is used for connecting pairs of ID's that represent one or many moves of a PDA. The process of transition is denoted by the turnstile symbol "⊢".

Consider a PDA (Q, ∑, S, δ, q_0, Z_0, F). A transition can be mathematically represented by the following turnstile notation −

(p, aw, T) |-- (q, w, a)

This implies that while taking a transition from state **p** to state **q**, the input symbol **'a'** is consumed, and the top of the stack **'T'** is replaced by a new string **'a'**.

There are two different ways to define PDA acceptability:

- **Final State Acceptability**

In final state acceptability, a PDA accepts a string when, after reading the entire string, the PDA is in a final state. From the starting state, we can make moves that end up in a final state with any stack values. The stack values are irrelevant as long as we end up in a final state.

For a PDA (Q, ∑, S, δ, q_0, I, F), the language accepted by the set of final states F is −

$L(PDA) = \{w \mid (q_0, w, I) \vdash^* (q, \varepsilon, x), q \in F\}$

for any input stack string **x**.

- **Empty Stack Acceptability**

Here a PDA accepts a string when, after reading the entire string, the PDA has emptied its stack.

For a PDA (Q, ∑, S, δ, q_0, I, F), the language accepted by the empty stack is −

$L(PDA) = \{w \mid (q_0, w, I) \vdash^* (q, \varepsilon, \varepsilon), q \in Q\}$

EXAMPLE OF PDA

PUSH OPERATION :-

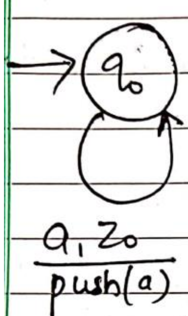
i) $\delta(q_0, a, Z_0) = (q_0, aZ_0)$

- q_0 → Current State
- a → i/p Symbol
- Z_0 → Stack top (Blank)
- q_0 → Next State
- a ↳ Stack top(a)

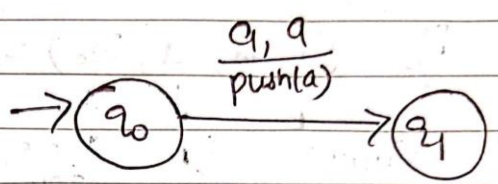
ii) $\delta(q_0, a, a) = (q_1, aa)$

- q_0 → Current State
- a → i/p symbol
- a → Stack top
- q_1 → Next State
- a ↳ stack top(a)

Transition Graph :-



(i)



(ii)

EXAMPLE OF PDA

POP OPERATION

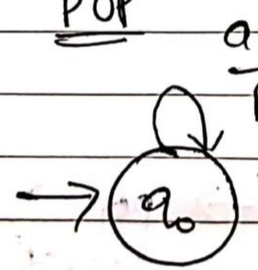
i) $\delta(q_0, a, a) = (q_0, \Lambda)$ → pop operation

- q_0 → Current State
- a → i/p symbol
- a → Stack top
- q_0 → Next State

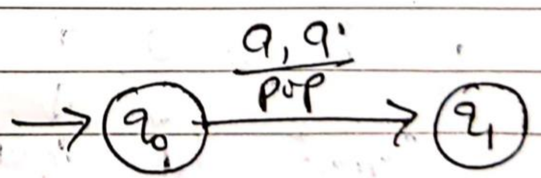
ii) $\delta(q_0, a, a) = (q_1, \Lambda)$ → Pop

- q_0 → Current State
- a → i/p symbol
- a → Stack top
- q_1 → Next State

POP



(i)

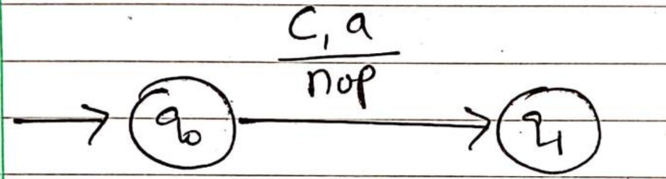


(ii)

NOP (No Operation) :-

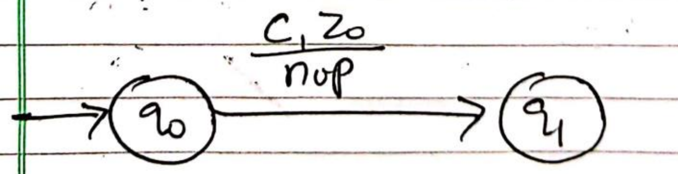
i) $\delta(q_0, c, a) = (q_1, a)$

- q_0 → Current State
- c → i/p Symbol
- a → Stack top
- q_1 → Next State
- a → Same stack top



ii) $\delta(q_0, c, Z_0) = (q_1, Z_0)$

- q_0 → Current State
- c → i/p Symbol
- Z_0 → Stack top
- q_1 → Next State
- Z_0 → Same stack top



Example:

$$A = (\{q_0, q_1, q_f\}, \{a, b, c\}, \{a, b, Z_0\}, \delta, q_0, Z_0, \{q_f\})$$

is a pda, where δ is defined as

$$\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}, \quad \delta(q_0, b, Z_0) = \{(q_0, bZ_0)\} \tag{7.5}$$

$$\delta(q_0, a, a) = \{(q_0, aa)\}, \quad \delta(q_0, b, a) = \{(q_0, ba)\} \tag{7.6}$$

$$\delta(q_0, a, b) = \{(q_0, ab)\}, \quad \delta(q_0, b, b) = \{(q_0, bb)\} \tag{7.7}$$

$$\delta(q_0, c, a) = \{(q_1, a)\}, \quad \delta(q_0, c, b) = \{(q_1, b)\}, \ \delta(q_0, c, Z_0) = \{(q_1, Z_0)\} \tag{7.8}$$

$$\delta(q_1, a, a) = \delta(q_1, b, b) = \{(q_1, \Lambda)\} \tag{7.9}$$

$$\delta(q_1, \Lambda, Z_0) = \{(q_f, Z_0)\} \tag{7.10}$$