Pushdown Automata (Introduction)

A Pushdown Automata (PDA) is a way to implement a Context Free Grammar in a similar way we design Finite Automata for Regular Grammar

-> It is more powerful than FSM
-> FSM has a very limited memory but PDA has more memory
-> PDA = Finite State Machine + A Stack

A stack is a way we arrange elements one on top of another

A stack does two basic operations:

PUSH: A new element is added at the Top of the stack

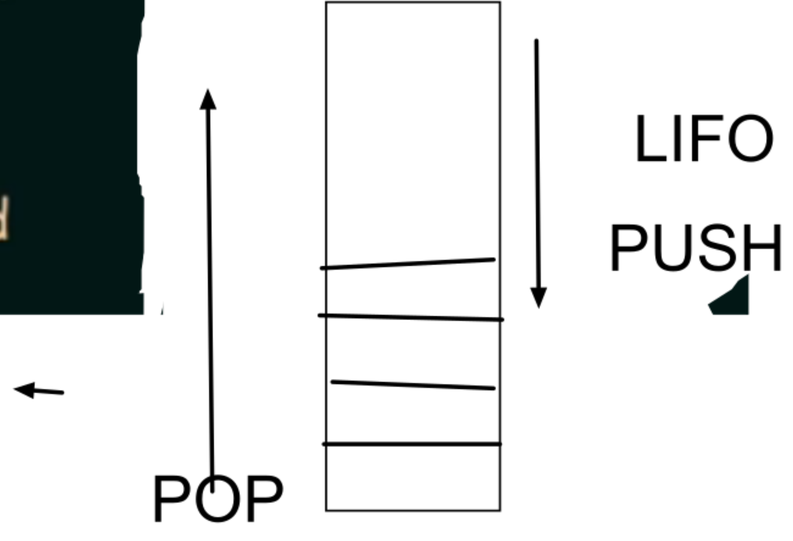
POP: The Top element of the stack is read and removed

MEELY

MOORE

p IN --->P OP

pIN PAST OP---> p OP



UNIT------3

Pushdown Automata: Definitions Moves
Instantaneous descriptions
Deterministic pushdown automata-Problems related to DPDA
Non - Deterministic pushdown automata-Problems related to NDPDA
Pushdown automata to CFL Equivalence-Problems of PDA to CFG
CFL to Pushdown automata Equivalence
Problems related to Equivalence of CFG

Finite control: The finite control has some pointer which points the current symbol which is to be read.

Stack: The stack is a structure in which we can push and remove the items from one end only.
It has an infinite size. In PDA, the stack is used to store the items temporarily.

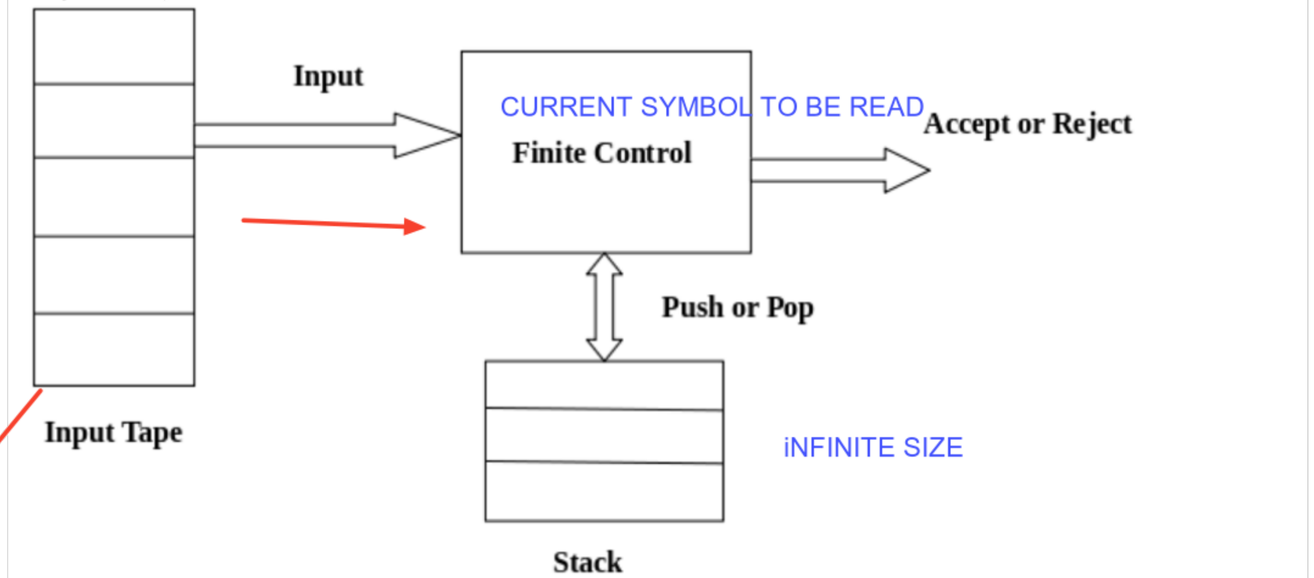


Fig: Pushdown Automata

PDA Components:

Input tape: The input tape is divided in many cells or symbols.
The input head is read-only and may only move from left to right, one symbol at a time.

Pushdown Automata (Formal Definition)

A Pushdown Automata is formally defined by 7 Tuples as shown below:

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

where,

- Q = A finite set of States
- Σ = A finite set of Input Symbols
- Γ = A finite Stack Alphabet
- δ = The Transition Function
- q_0 = The Start State
- z_0 = The Start Stack Symbol
- F = The set of Final / Accepting States

δ takes as argument a triple $\delta(q, a, X)$ where:

(i) q is a State in Q

(ii) a is either an Input Symbol in Σ or $a = \in$

(iii) X is a Stack Symbol, that is a member of Γ

DELTA

GAMMA

UPPER CASE GAMMA

$\bar{\delta}(q, a, X)$

Representation of State Transition

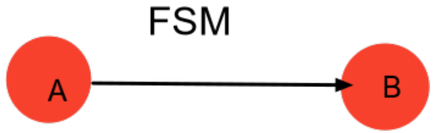
Delta Function (δ) is the transition function, the use of which will become more clear by taking a closer look at the Three Major operations done on Stack :-

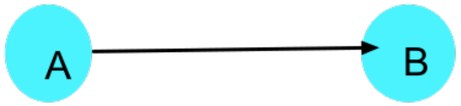
1. Push
2. Pop
3. Skip /No operation

LOWER CASE gAMMA

The output of δ is finite set of pairs (p, γ) where:
p is a new state
γ is a string of stack symbols that replaces X at the top of the stack

Eg. If γ = ∈ then the stack is popped (REMOVED)
If γ = X then the stack is unchanged
If γ = YZ then X is replaced by Z and Y is pushed onto the stack





a,b--->c

the symbol is pushed onto the stack

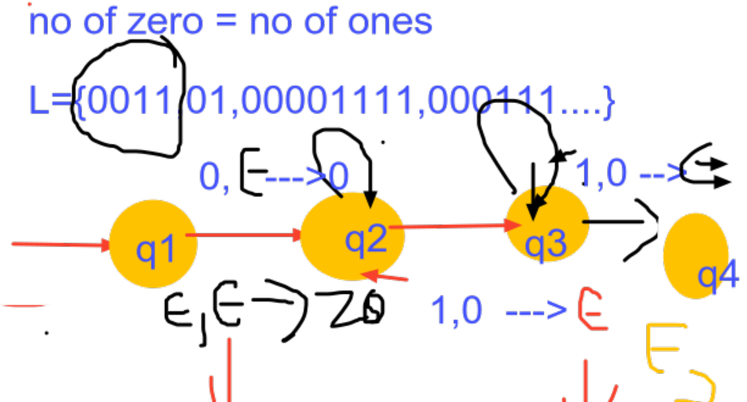
top of the stack

input symbol

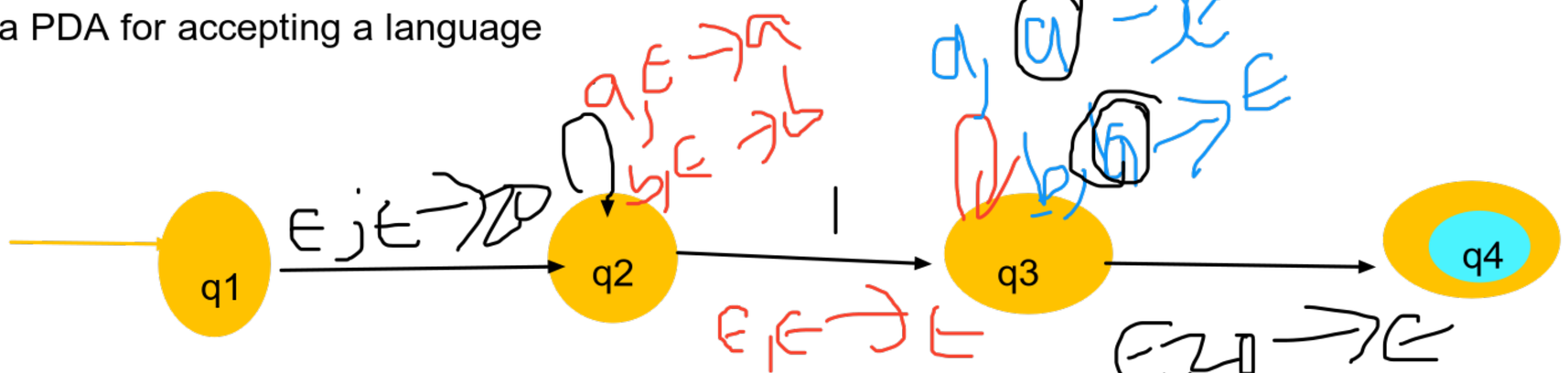
may be ∈

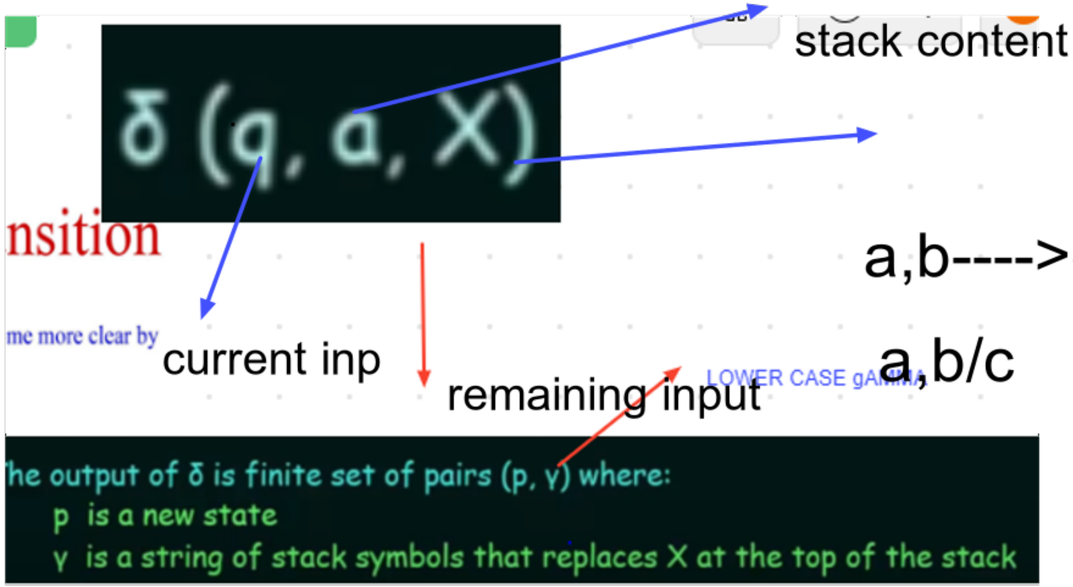
SYMBOL IS POPPED

Example 1:
Design a PDA for accepting a language $0^n 1^n$ | n>=0}.



Design a PDA for accepting a language





a,b---->c

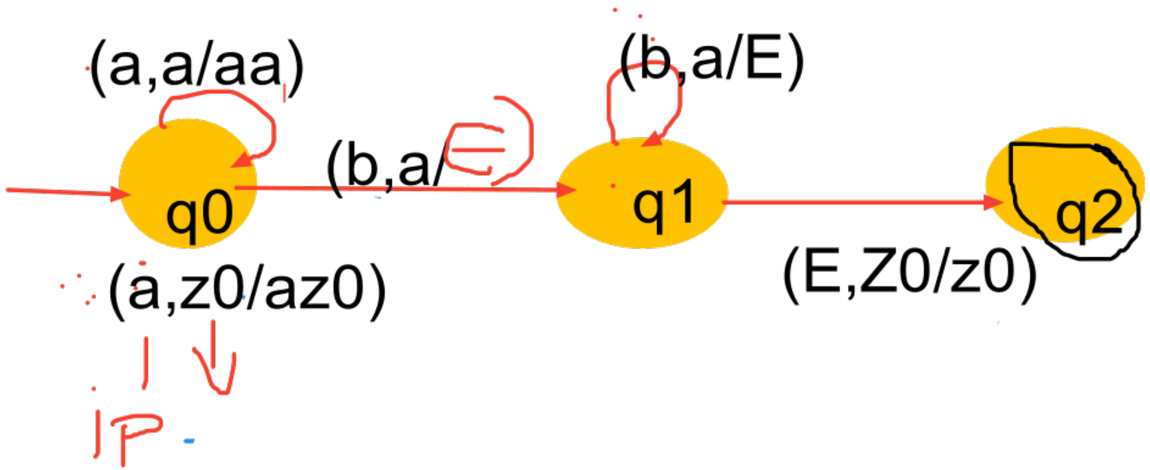
a,b/c

let us take a

aabb

upto a i need to push it when first b

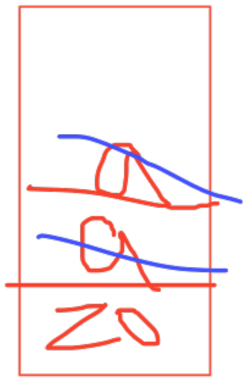
i need to pop it a one b



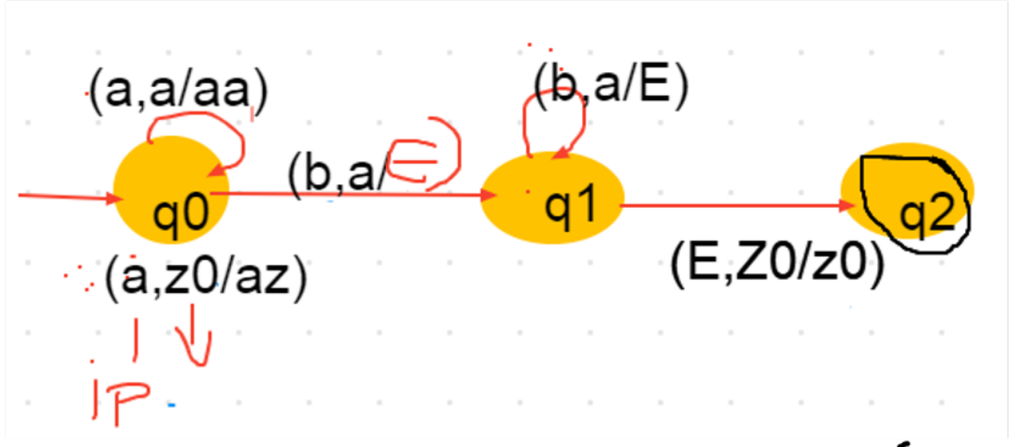
Example 1:
Design a PDA for accepting a

language $\{0^n 1^n \mid n>=0\}$.

L = {aabb,ab,aaabbb...}



aabbE



$\delta(q, a, X) \rightarrow (p, \gamma)$

δ (q0,a,z0)----> (q0,az0)

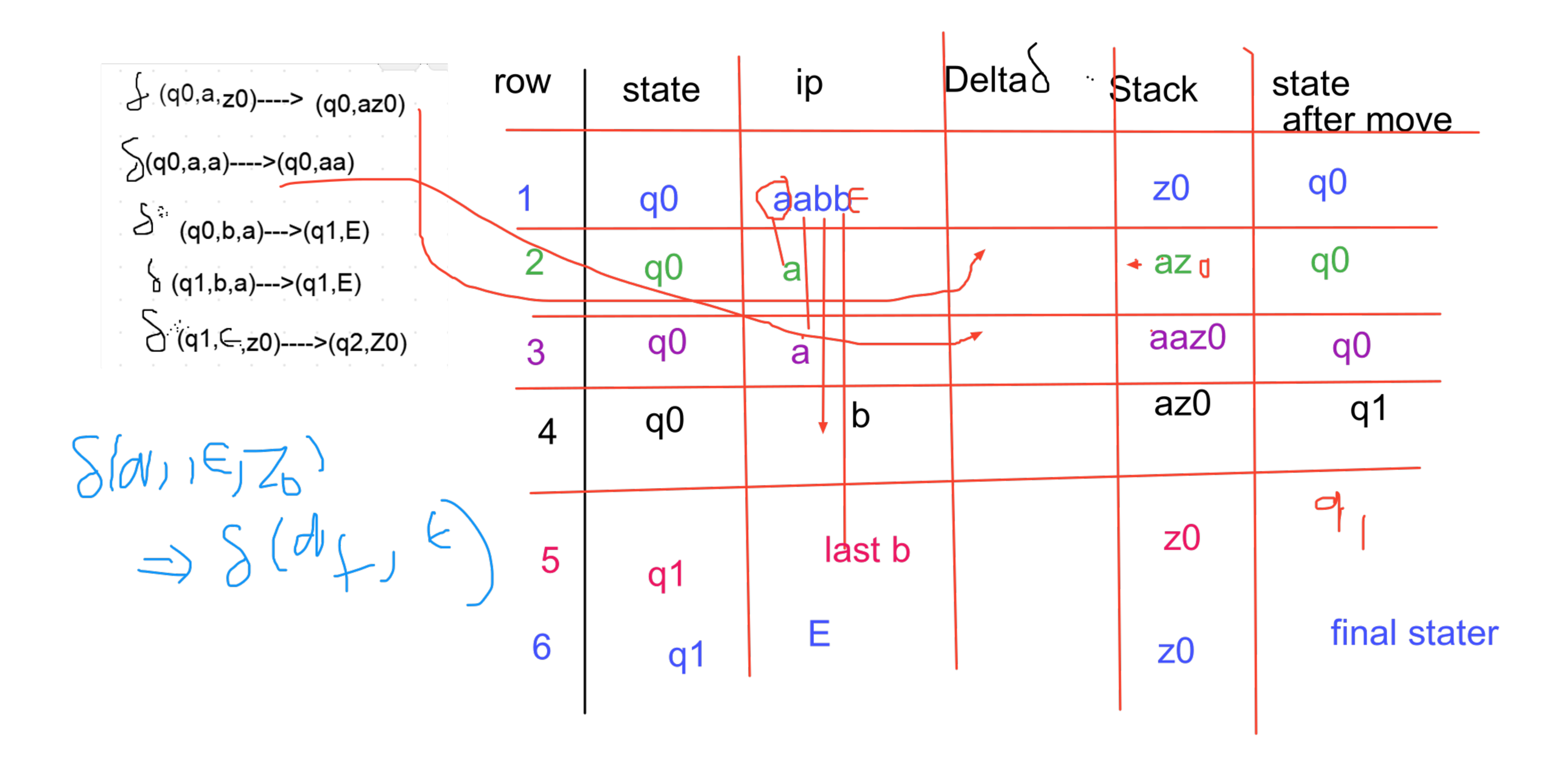
δ(q0,a,a)---->(q0,aa)

δ (q0,b,a)--->(q1,E)

δ (q1,b,a)--->(q1,E)

δ (q1,∈,z0)---->(q2,Z0)

state transition


δ (q0,a,z0)----> (q0,az0)
δ(q0,a,a)---->(q0,aa)
δ (q0,b,a)--->(q1,E)
δ (q1,b,a)--->(q1,E)
δ (q1,∈,z0)---->(q2,Z0)
δ(q0,∈,Z0)
⇒ δ(q_f, ∈)
row
state
ip
Delta δ
Stack
state after move
1
q0
aabb∈
z0
q0
2
q0
a
az0
q0
3
q0
a
aaz0
q0
4
q0
b
az0
q1
5
q1
last b
z0
q_f
6
q1
E
z0
final stater

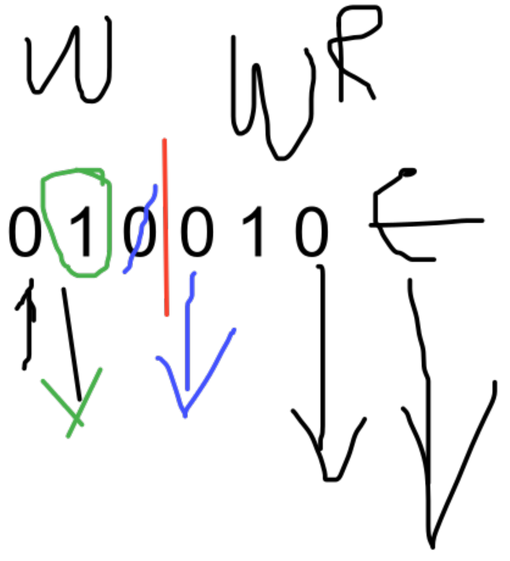
de.

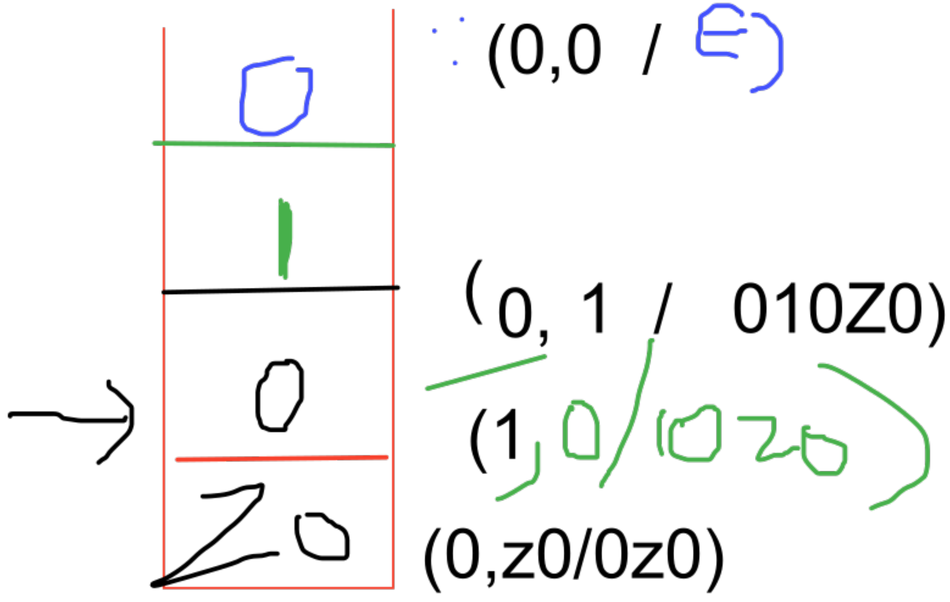
ww^R (0+1)*

0110

001100

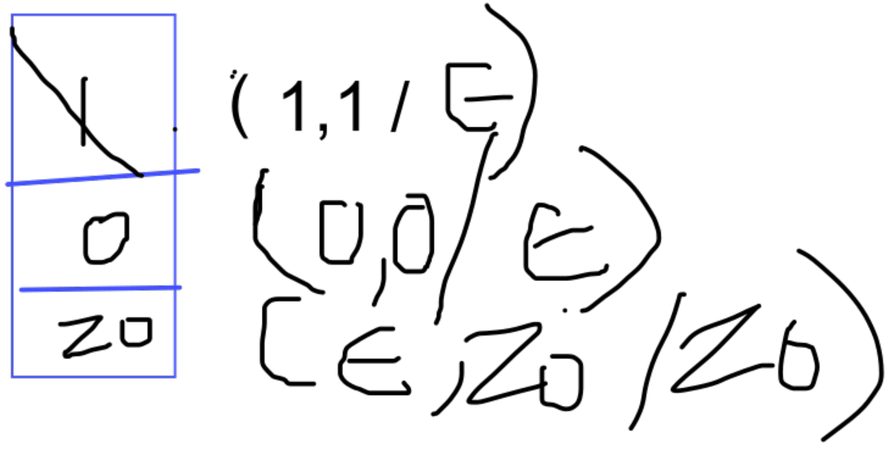
010010

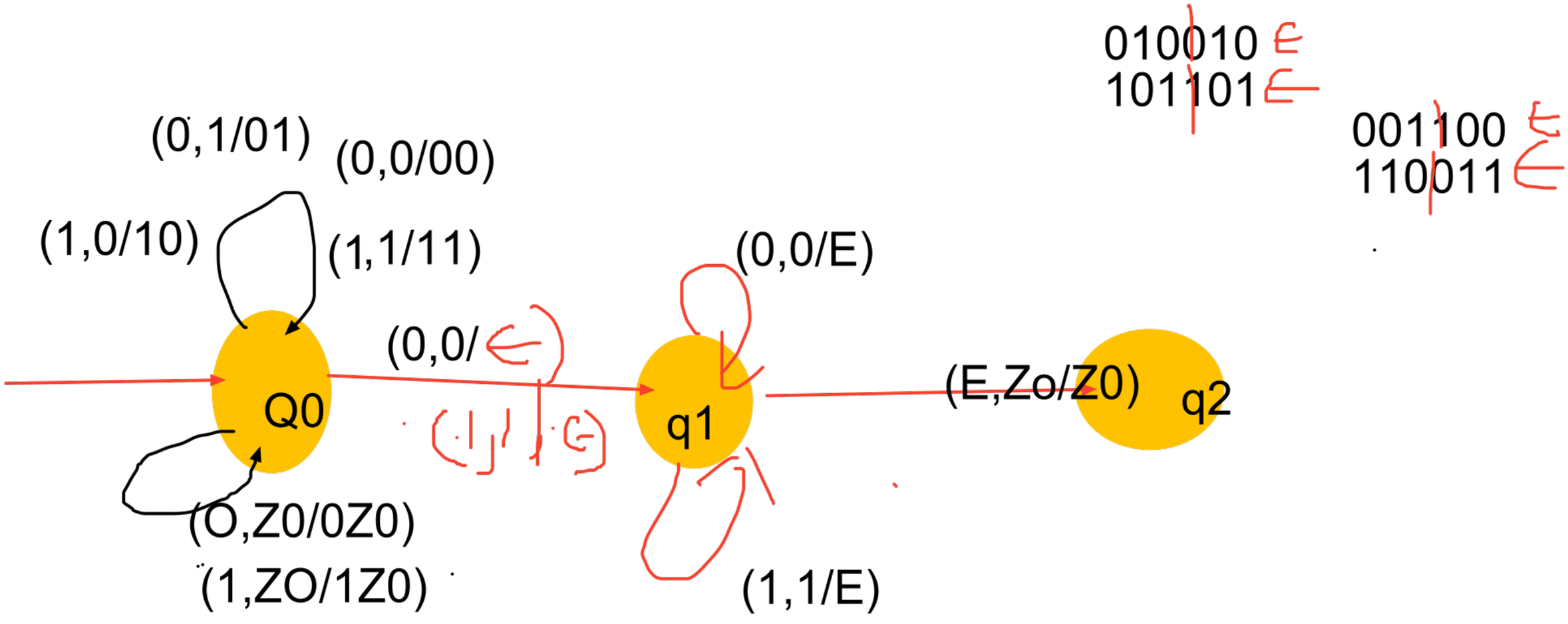




case 1 : if the input symbol and stack symbol is same

case 2 : if the input symbol and stack symbol is not same




010010 ∈
101101 ∈
001100 ∈
110011 ∈
(0,1/01)
(0,0/00)
(1,0/10)
(1,1/11)
(0,0/E)
(0,0/
Q0
q1
(E,Zo/Z0)
q2
(O,Z0/0Z0)
(1,ZO/1Z0)
(1,1/E)