CSC 218

COMPUTER ORGANIZATION AND FORMAL LANGUAGES

Push down automata (PDA).

Nnamene C.C

Overview

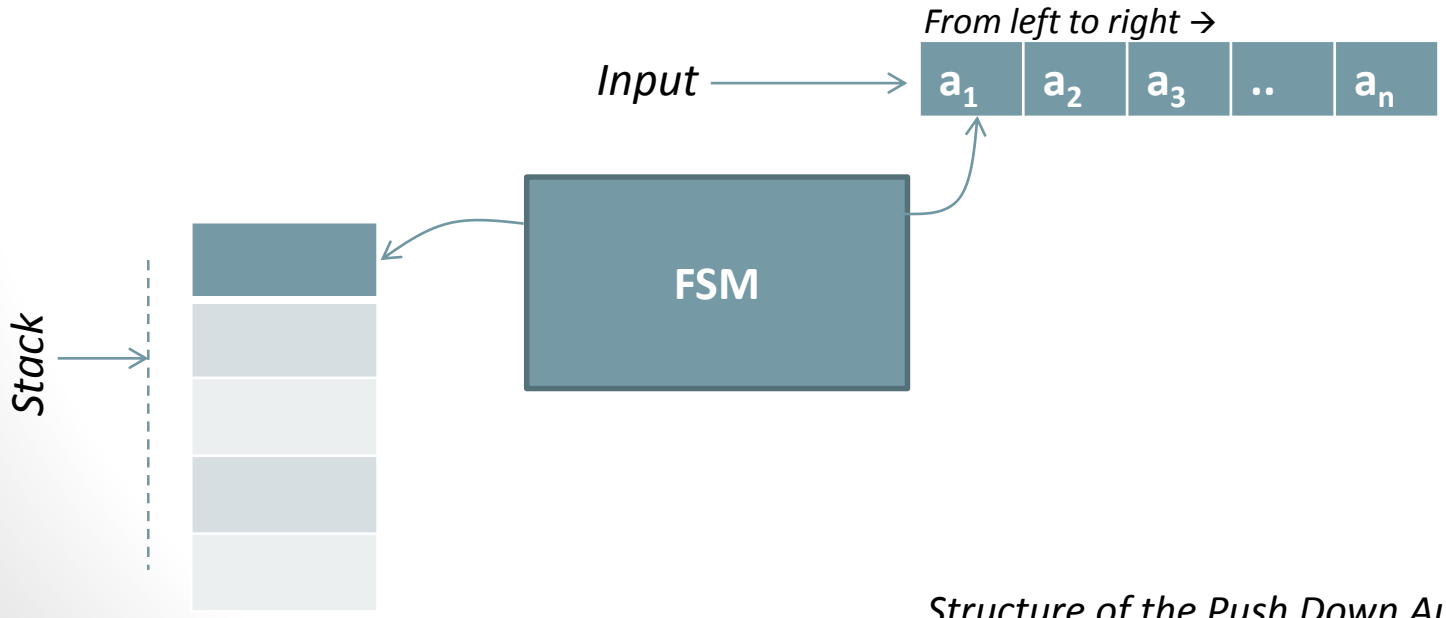
- Introduction to Push Down Automata (PDA)
- Implementation
- State Operations

Push down Automata (PDA)

- This is similar to the FSM, but has additional memory
- This memory uses a stack, which has two operations:
  - **Push** (add a new symbol at the top)
  - **Pop** (read and remove the top symbol)

Used to manipulate the stacks from the states

PDA = Finite Automata + Memory (Stack)



Structure of the Push Down Automaton

PDA cntd.

- A PDA is a septuple (7-tuple) system $(Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ where:
  - Q is a finite set of states $\{q_i \ldots., q_n)$
  - Σ is an input alphabet (alphabet of the language);
  - Γ is a stack alphabet (stack-symbols *denoted by Gamma*);
  - δ is the transition mapping from $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ to finite subsets of $Q \times \Gamma^*$
  - q_0 in Q is the initial state (start state);
  - Z_0 in Γ is a particular stack-symbol called the start symbol;
  - *F: the set of final states* ($F \subseteq Q$)

Conventions used

- Lower cases are used for input symbols eg. a, b, c,…
  - ε is also allowed as a possible value
- Upper cases are used for stack symbols
  - Eg. X, Y, Z, …
- Lower case letters at the end of the alphabets are used for strings of input symbols eg. x, y, z, …
- The beginning Greek alphabets are used for strings of stack symbols
  - Eg. α, β, …

Implementing PDA

- This take into account a transition function with the following arguments:
  1. A state, in Q
  2. An input symbol in *Σ or* ε
  3. A stack symbol *Γ*

For every δ(q, a, z), this will contain (p, α) such that:
p = a state
α = a string of stack symbols

In addition, a PDA having a state '**q**', and '**a**' at the front of the input, and **Z** at the top of the stack implies:

change the state to p
Remove '**a**' from the front of input
Replace **Z** on the top of the stack by '**α**'

Implementing PDA cntd.

Example.

To design a PDA which will accept the Language $\{0^n1^n \mid n \geq 1\}$

- The states in here shows:
  - q= the start state (permitting zeros at the beginning)
  - p = At least one 1 can proceed if the inputs are 1's
  - f = Final state (we will accept if the number of 1's matches the number of 0's)
  - Z_0 = Start symbol (this marks the bottom of the stark
  - X = a marker that counts the number of 0's in the input

Implementing PDA

Given the following input string

0	0	0	1	1	1

Does this satisfy the automaton?

String: from left to right
Stack: LIFO

The following transitions apply:

$\delta(q, 0, Z_0) = \{(q, \underline{XZ_0})\}$ — *Mapping (Replacement string)*

$\delta(q, 0, X) = \{(q, XX)\}$ — *Number of X on stack = the number of 0's read from input*

*Stay in state **q** since the input = 0's*

:

$\delta(q, 1, X) = \{(p, \varepsilon)\}$ — *When 1 is encountered, then we pop one X from the stack*

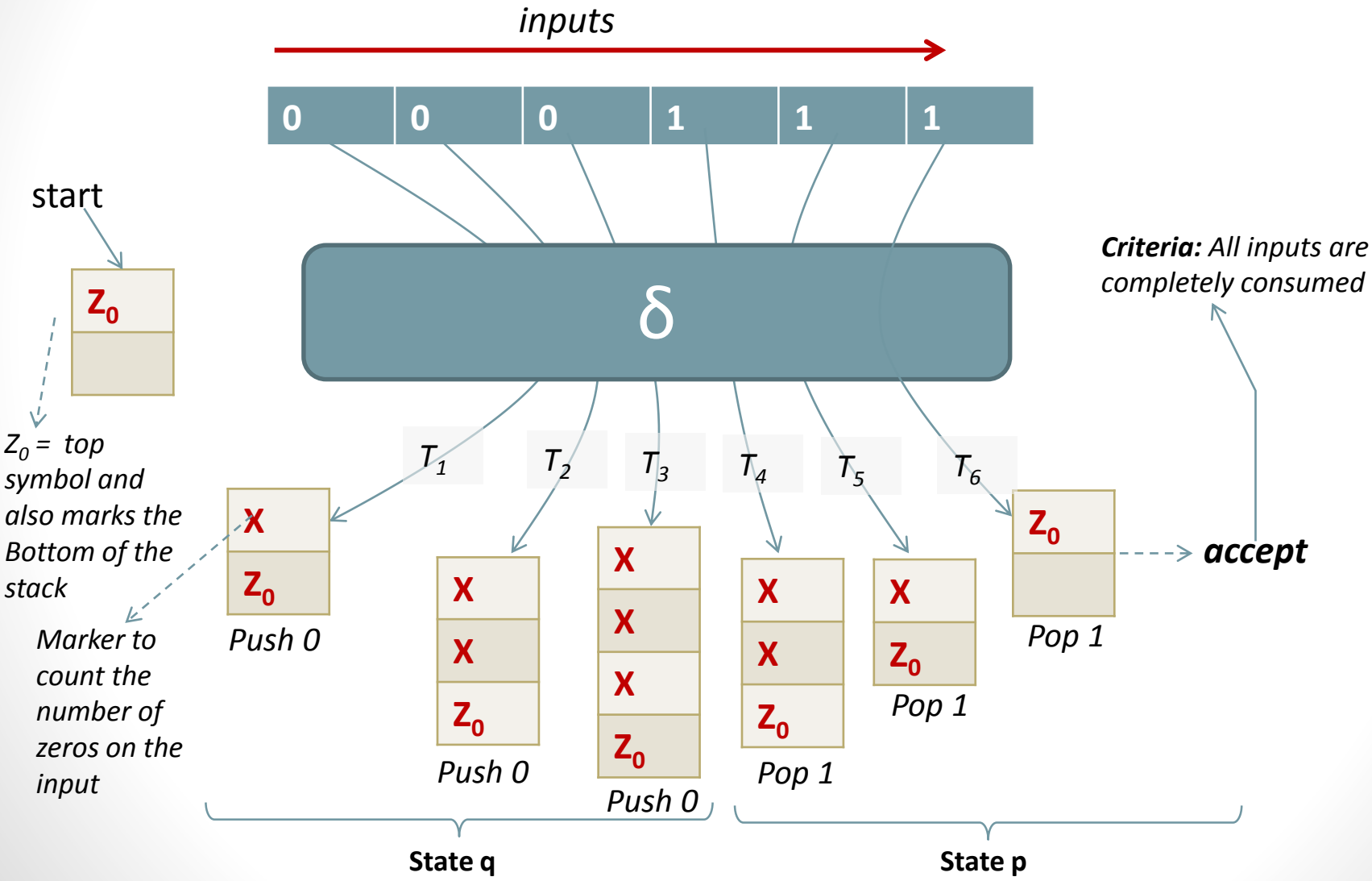
*Change to state **p** since the input =1's*

$\delta(p, 1, X) = \{(p, \varepsilon)\}$ — *Pop one X per occurrence of 1*

$\delta(p, \varepsilon, Z_0) = \{(f, Z_0)\}$ — *Z_0 at the top of stack*; *No input*

Transition function mapping each triple (q_i , a, X), where q_i = a state; a = input symbol or ∧; X = stack symbol

$\delta(q, 0, X) = \{(q, XZ_0)\}$



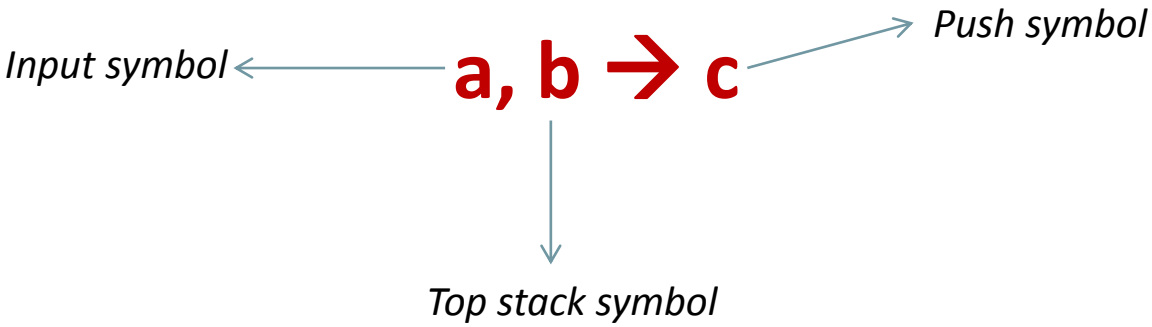
State Operations

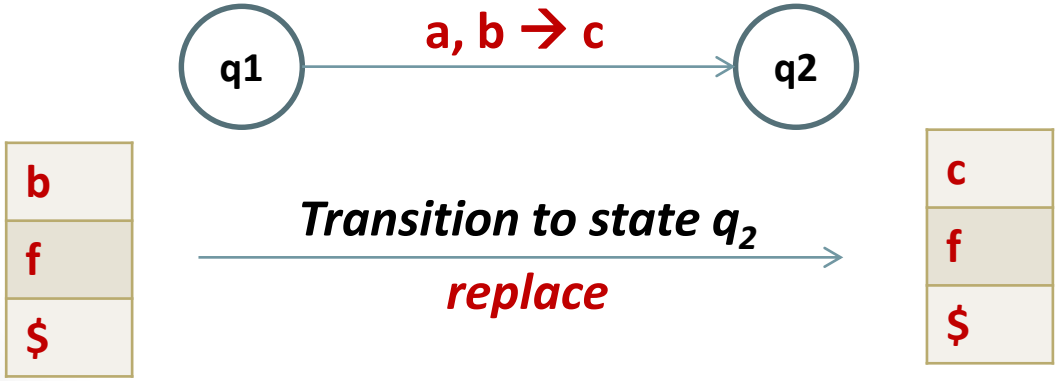
- Note that transitions of the PDA is the form:

This implies that if “a” is seen in the input string, and there is an “x” symbol on top of the stack,
“y” is then added and the “x” is removed

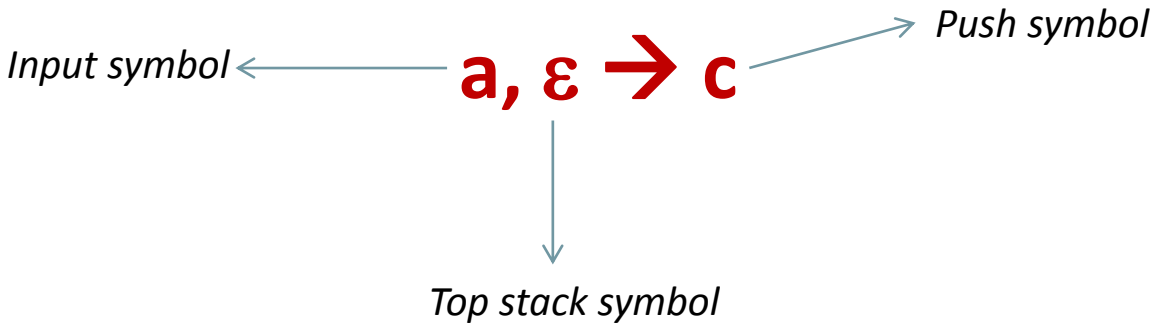
This applies to the operations (Replace, push, pop, No Change)

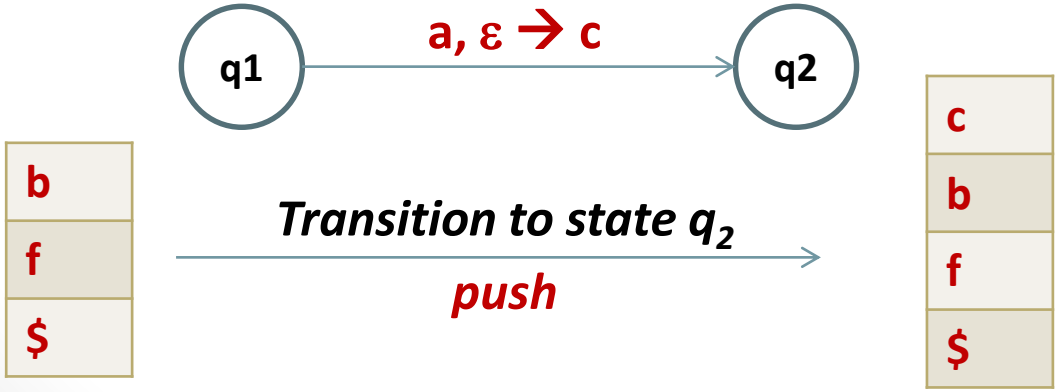
State Operations (Replace)



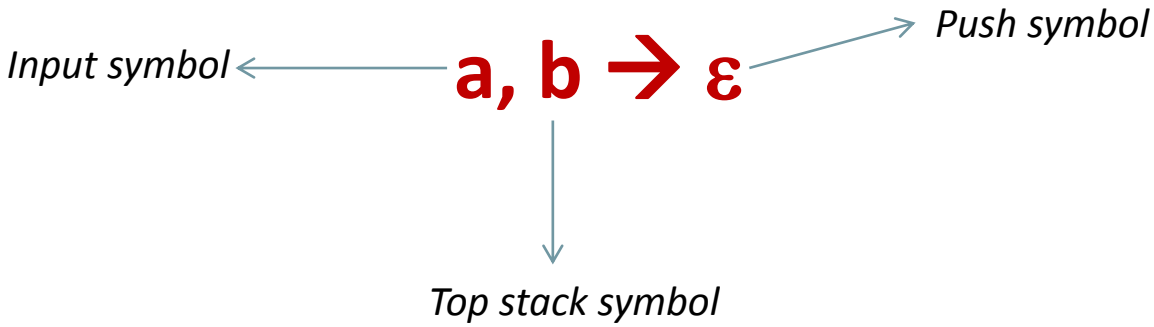


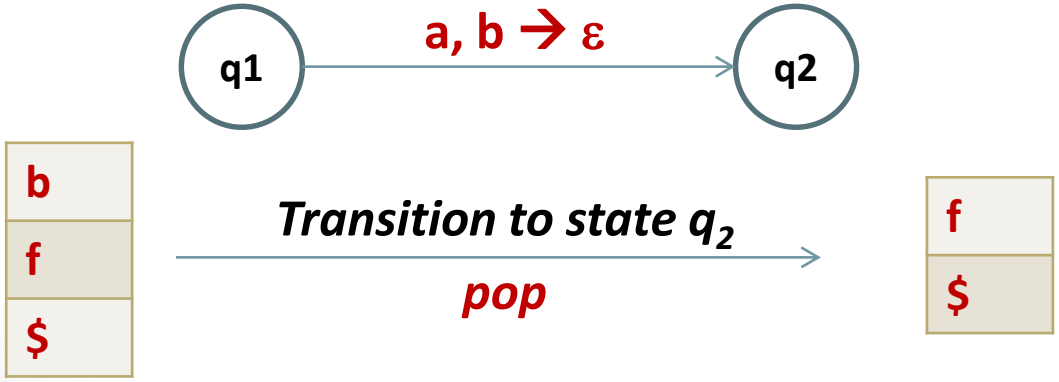
State Operation (Push)



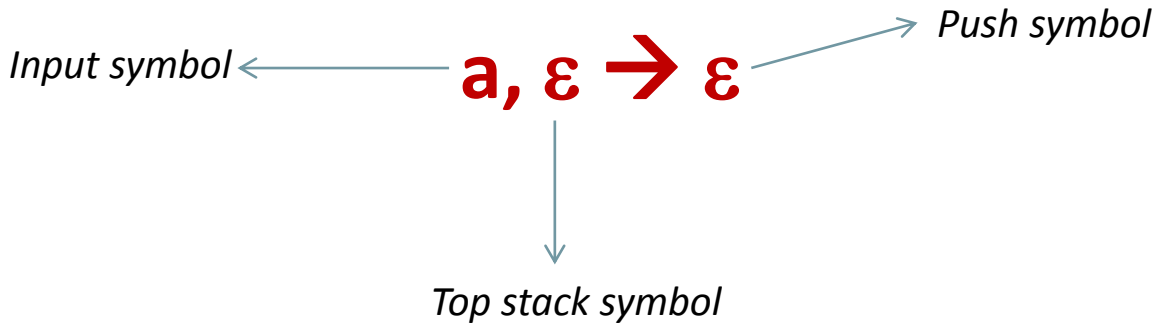


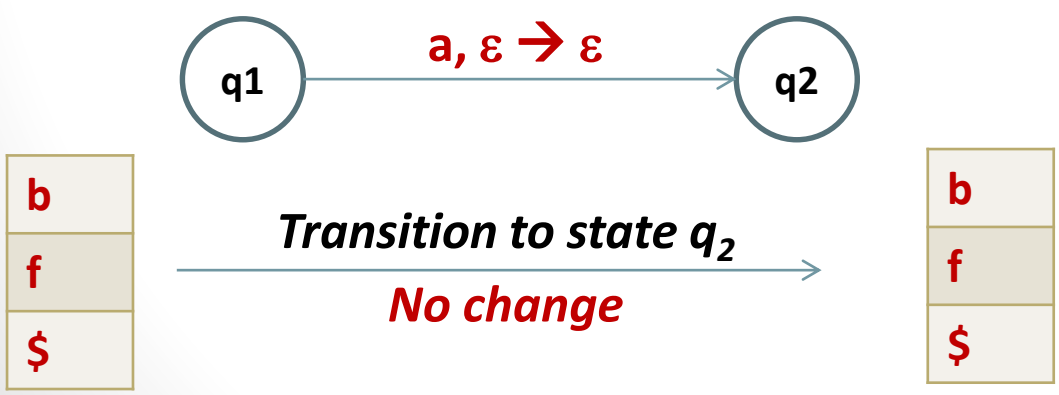
State Operation (Pop)





State Operation (No change)





Note: the Automaton will halts or rejects input if there is an attempt to perform a pop or replace operation on an empty stack.

Condition for PDA acceptance or rejection

- The PDA accepts if there is a computation such that:
  - the computation path ends in an accept state, and
  - all the inputs in the string is consumed.

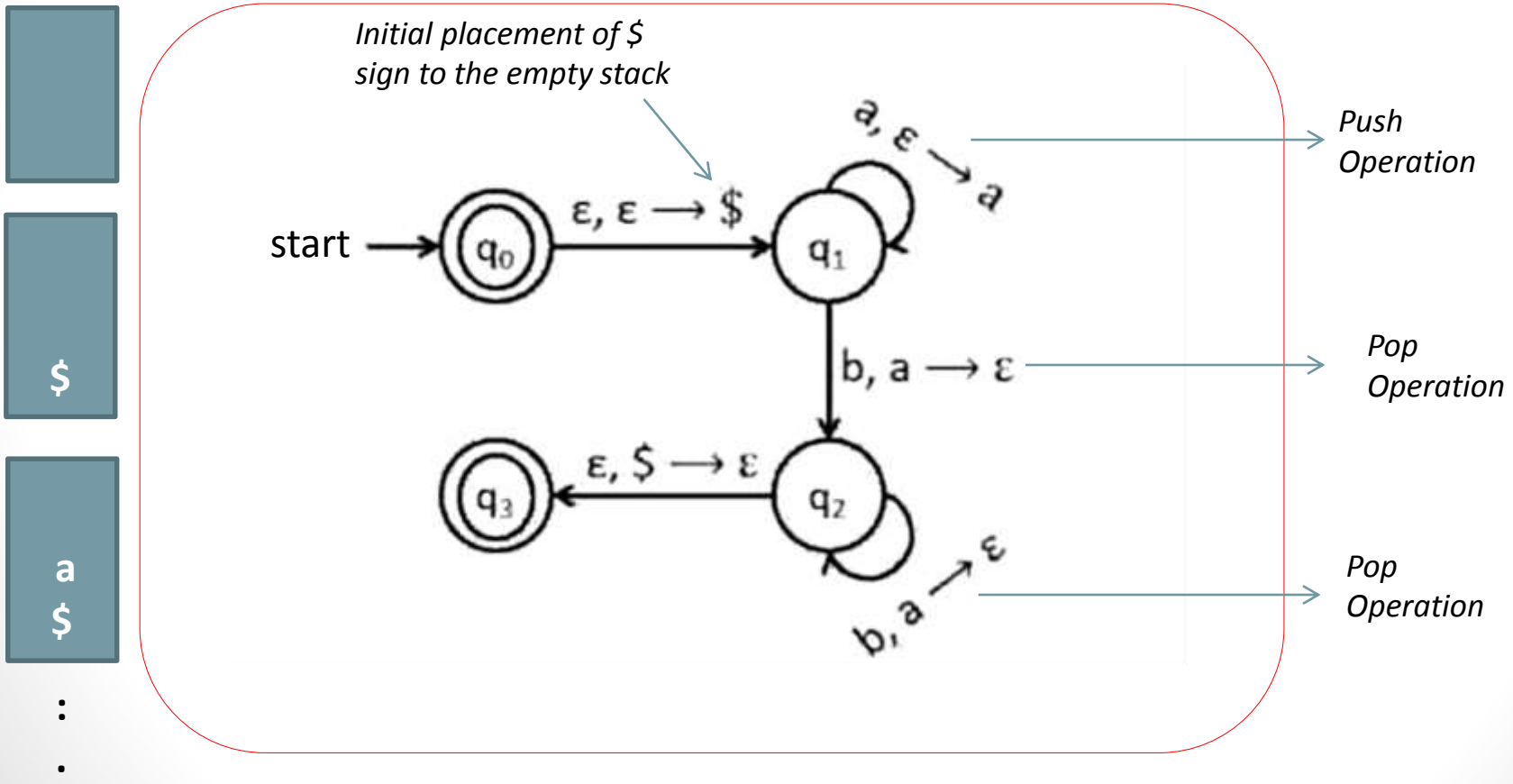
- On the other hand, the PDA rejects if in all paths:
  - the computation path ends in the non accepting state, and
  - there is an incomplete computation path such that there is no possible transition in the input and stack symbols.

Empty stack indicator ($)

- We place dollar sign ($) or Z_0 as the case may be to help indicate when the stack is empty.
- Discovering the sign again indicates the end of the stack

Familiar Example

- Consider the Machine to accept the PDA for the language **{$a^n b^n \mid n \geq 0$}**



Sequence of moves (Goes to "⊢")

- To describe the sequence of moves of the previous example, the following holds:
  - $(q, 000111, Z_0) \vdash (q, 00111, XZ_0) \vdash$
  - $(q,0111, XXZ_0) \vdash (q, 111, XXXZ_0) \vdash$
  - $(p, 11, XXZ_0) \vdash (p, 1, Z_0) \vdash$
  - $(p, \varepsilon, Z_0) \vdash (f, \varepsilon, Z_0)$

  - Therefore, $(q, 000111, Z_0) \vdash (f, \varepsilon, Z_0)$

Sequence of moves (Goes to "⊢")

- In a situation where there is an extra 1 eg. 0001111 on the input, this gives the following sequence:
  - $(q, 0001111, Z_0) \vdash (q, 001111, XZ_0) \vdash$
  - $(q, 01111, XXZ_0) \vdash (q, 1111, XXXZ_0) \vdash$
  - $(p, 111, XXZ_0) \vdash (p, 11, XZ_0) \vdash (p, 1, Z_0) \vdash$
  - $(f, 1, Z_0)$

Therefore, 0001111 is not accepted.

THANK YOU!