Theory of Computation (CT-502)

Course Instructor
ANUJ GHIMIRE

DISCLAIMER

- *This document does not claim any originality and cannot be used as a substitute for prescribed textbooks.*
- *The information presented here is merely a collection from various sources as well as freely available material from internet and textbooks.*
- *The ownership of the information lies with the respective authors or institutions.*

Pushdown Automata (PDA)

- Certain languages for example $L_1 = \{ wcw^R \mid w \in \Sigma^* \}$ or $L_2 = \{ a^n b^n : n \geq 1 \}$ can be generated by CFG but can't be accepted by FA.
- So all CFG's are not accepted by FA.
- Here in the given example L1 and L2, to recognize the string we need to remember the first part of the strings before it goes to next part of the string and compare it with first part.
- But FA can't do this because of the limited memory and finite states, FA cannot remember anything.

Pushdown Automata (PDA)

- So, we need a powerful automation than FA that can remember the content of the input tape which are already read or scanned.
- This automation is Pushdown Automata (PDA) which is essentially a FA with control of both an input tape and a ***stack*** to store what it has read.
- A stack is a LIFO (Last In First Out) data structure which is used to remember infinite amount of information.
- We can only access information in stack as Last In First Out.

Pushdown Automata (PDA)

- PDA is an automata that are used to recognize the string generated by the context free grammar.
- PDA is an abstract machine which can be thought as a ***ϵ-NFA*** with addition of stack and are determined by following three things.
  - Input Tape
  - Finite Control
  - Stack

Pushdown Automata (PDA)

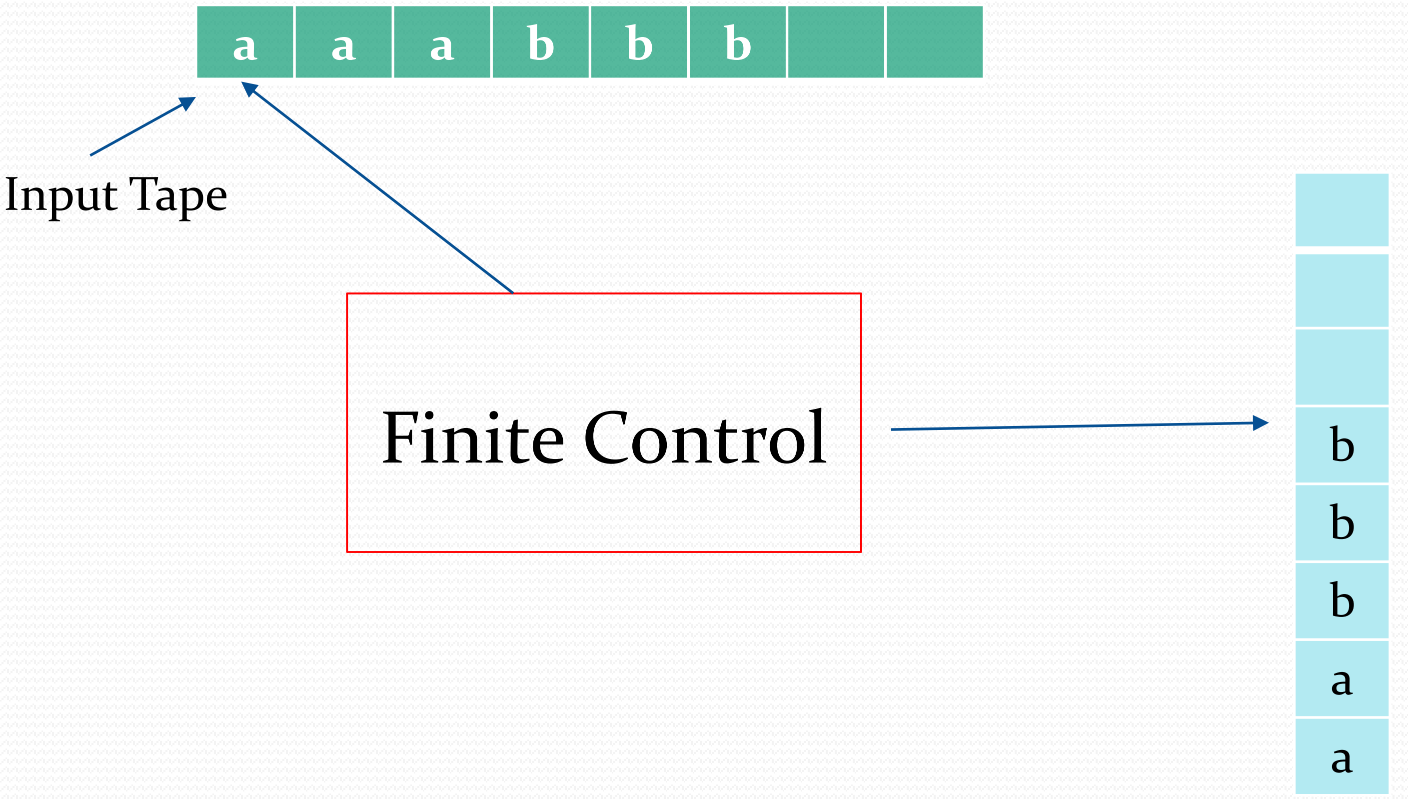


Fig: Block Diagram of Pushdown Automata

Pushdown Automata (PDA)

- Each move of machine is determined by three things:
  - Current state
  - Next input symbol
  - Symbol on top of stack
- The move consists of :
  - Changing state or stay on same state
  - Replace top of stack by string of zero or more symbols.
- Move of machine contains only one stack operation either push or pop.

Pushdown Automata (PDA)

- Popping the top symbol of the stack means replacing the top symbol of stack by ϵ.
- Pushing s on the stack means replacing stack top, say x by sx.
- A PDA can write symbols on the stack and read them back later.

Pushdown Automata (PDA)

- Formally, Pushdown Automata (PDA) is defined by 7 (seven) tuples:

$$P=(Q, \Sigma, \Gamma, \delta, q_o, z_o, F)$$

where,

Q=Finite set of states

Σ=Finite set of input symbols

Upper case Gamma

Γ=Finite set of stack alphabets

q_o=Start state of PDA, $q_o \in Q$

z_o=Initial stack symbol, $z_o \in \Gamma$

F=Set of final states

Pushdown Automata (PDA)

Here, $\boldsymbol{\delta}$ takes as argument a triplet $\boldsymbol{\delta(q, a, x)}$ where:

- ***q*** is a state in Q.
- ***a*** is either an Input Symbol ∑ or a = ε.
- ***x*** is a Stack Symbol, that is a member of Γ.

The output of $\boldsymbol{\delta}$ is finite set of pairs $\boldsymbol{(p, \gamma)}$ where:

p is a new state.

$\boldsymbol{\gamma}$ is a string of stack symbols that replaces ***x*** at the top of the stack.

Transition function δ maps:

$$Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$$

Pushdown Automata (PDA)

Example:

- If γ = ε, then the stack is popped.
- If γ =x then the stack is unchanged.
- If γ =yz then ***x*** ıs replaced by ***z*** and ***y*** is pushed onto the stack.

- ***Note:*** PDA are also defined by 6-tuple.It depends on whether the PDA accepts by final state or by empty stack. ***A final-state PDA is defined by a 7-tuple;*** an ***empty-stack PDA can be defined as a 6-tuple,*** because it doesn't need to specify final states.

Pushdown Automata (PDA)

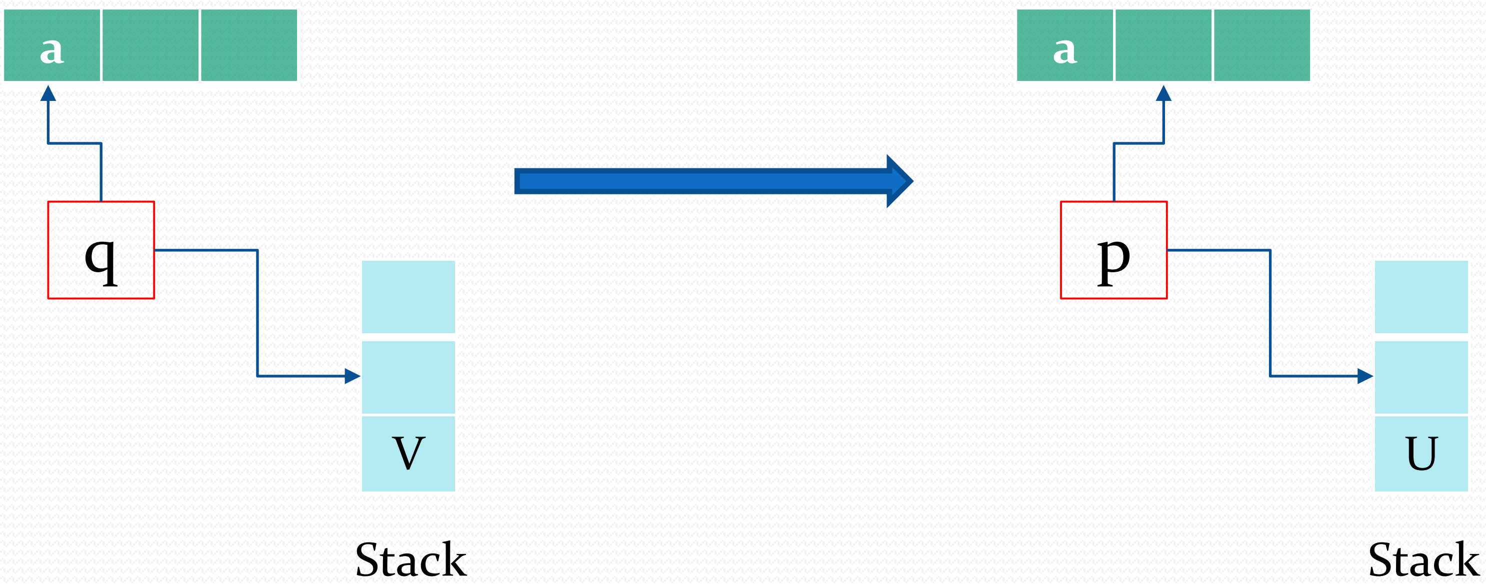
- ***Instantaneous Description (ID)***
  - Description of any PDA by a triplet ***(q, w, γ)*** where ***q*** is the state, ***w*** is the remaining input and ***γ*** is the stack contents is called Instantaneous description of PDA.
  - Instantaneous Description (ID) is an informal notation of how a PDA "computes" a input string and make a decision that string is accepted or rejected.
  - Notation of ID for PDA is used as to describe changes in state, input and stack.

Pushdown Automata (PDA)

- Let, $P=(Q, \sum, \Gamma, \delta, q_o, z_o, F)$ be a PDA.
  - Define a relation ⊢
  - Here ⊢ sign is called a "turnstile notation" and represent one move and ⊢* sign represents a sequence of moves.
  - Suppose $\boldsymbol{\delta(q, a, x)}$ contains $\boldsymbol{(p,\alpha)}$, then for all strings $\boldsymbol{w}$ in $\sum^*$ and β in Γ^* $\boldsymbol{(q, aw, x\beta) \vdash (p, w, \alpha\beta)}$
- This reflects the idea that by consuming $\boldsymbol{a}$ from input symbol and replacing $\boldsymbol{x}$ by $\boldsymbol{\alpha}$ on top of stack, we can go from state $\boldsymbol{q}$ to $\boldsymbol{p}$.

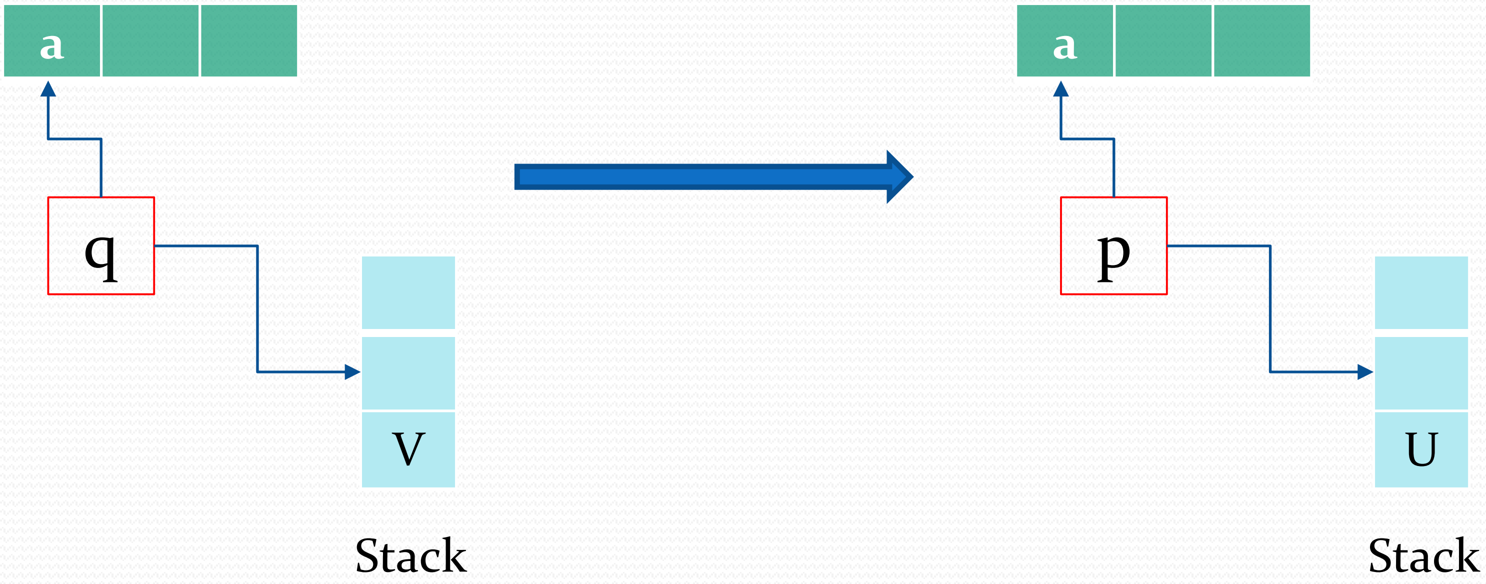
Pushdown Automata (PDA)

a) $\delta(q, a, V) \rightarrow (p, U)$,



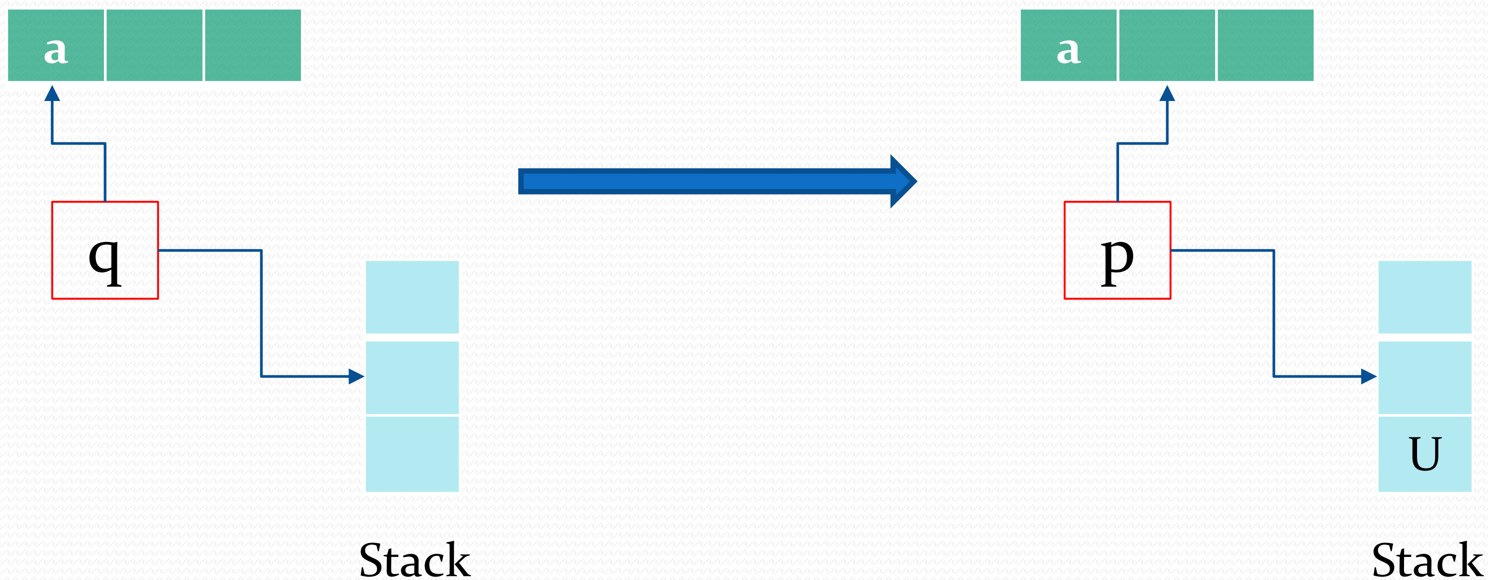
Pushdown Automata (PDA)

b) $\boldsymbol{\delta(q, \varepsilon, V) \to (p, U)}$,



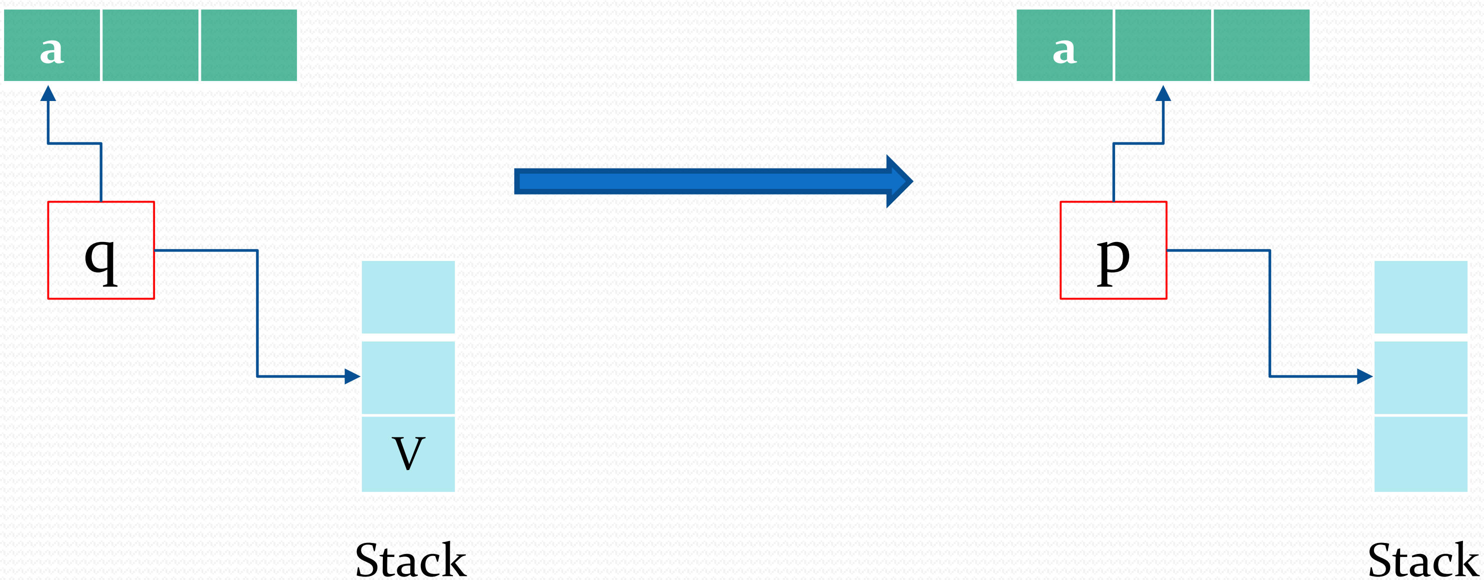
Pushdown Automata (PDA)

c) $\delta(q, a, \varepsilon) \rightarrow (p, U)$,

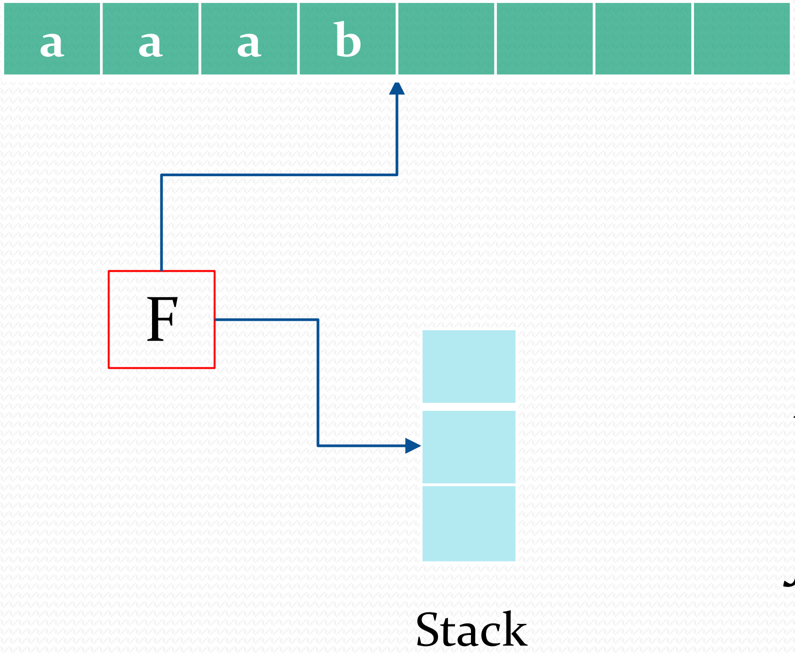


Pushdown Automata (PDA)

d) $\delta(q, a, V) \rightarrow (p, \varepsilon)$,



Pushdown Automata (PDA)



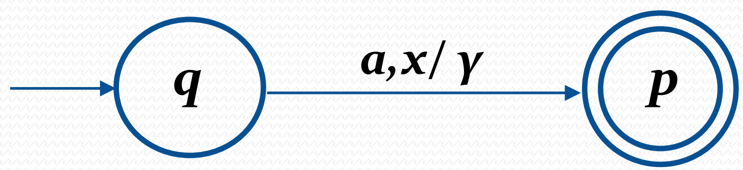
When tape head gets off the tape, PDA stops. An input string ***w*** *is accepted by PDA if PDA stop at final state and stack is empty otherwise input string is rejected.*

Pushdown Automata (PDA)

Graphical Notation of PDA

- We can use transition diagram to represent a PDA where
  - Any state is represented by a node in a diagram.
  - Any arc labeled with start indicate to start state and doubly circled states are accepting or final states.
- The arc correspond to transition of PDA labeled as:

$\boldsymbol{a, x/\ \gamma}$ means the transition: $\boldsymbol{\delta(q,\ a,\ x) \rightarrow (p,\ \gamma)}$.



Design of PDA_Example(1)

Consider a CFL defined as $L = \{a^n b^n \mid n >= 0\}$

The set of string generated by the given language are

{ε, ab, aabb, aaabbb, aaaabbbb,..................}

*Here , the idea is that the number of **a's** and **b's** in the string must be equal and the occurrence of **b** will begin when the occurrence of **a** stops.*

*Furthermore, when the occurrence of **b** starts the occurrence of **a** is not possible.*

*PDA should push the input symbol in stack if it is **a** and if it is **b** then pop the symbol from stack.*

Design of PDA_Example(1)

Let us consider PDA to recognize the given language as:

$P=(Q, \Sigma, \Gamma, \delta, q_o, z_o, F)$

where,

Q=*Finite set of states* =**{s, q, p, f}**

Σ=*Finite set of input symbols* =**{a, b}**

Γ=*Finite set of stack alphabets* =**{a, b, z_o}**

q_o=*Start state of PDA*, $q_o \in Q$ =**{s}**

z_o=*Initial stack symbol*, $z_o \in \Gamma$

F=*Set of final states* =**{s,f}**

Design of PDA_Example(1)

Here, $\boldsymbol{\delta}$ *is the transition function of the form*

$$\boldsymbol{\delta(q, a, x) \rightarrow (p, \gamma)}$$

where:

- $\boldsymbol{q}$ *is a state in Q.*
- $\boldsymbol{a}$ *is either an Input Symbol ∑ or a = ε.*
- $\boldsymbol{x}$ *is a Stack Symbol, that is a member of* $\boldsymbol{\Gamma}$*.*
- $\boldsymbol{p}$ *is a new state.*
- $\boldsymbol{\gamma}$ *is a string of stack symbols that replaces* $\boldsymbol{x}$ *at the top of the stack.*

Design of PDA_Example(1)

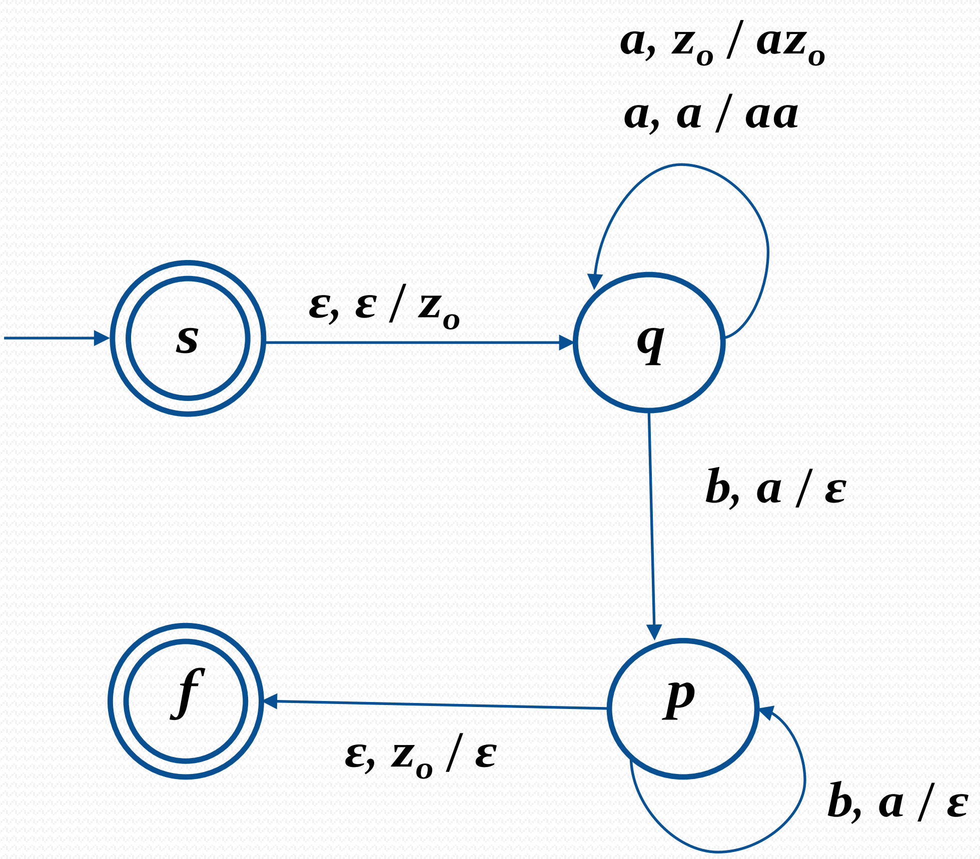
*Now we need to define the possible transition while processing the string **w**.*

*The strategy is whenever the symbol **a** is scanned we will perform the push operation without changing the state of the PDA and whenever symbol **b** is scanned the PDA transits to new state and we perform the pop operation.*

The Instantaneous Description (ID) for the PDA is then described to show changes in state, input and stack symbol as:

Design of PDA_Example(1)

1. $\delta(s, \varepsilon, \varepsilon) \rightarrow (q, z_0)$
2. $\delta(q, a, z_0) \rightarrow (q, az_0)$
3. $\delta(q, a, a) \rightarrow (q, aaz_0)$
4. $\delta(q, b, a) \rightarrow (p, \varepsilon)$
5. $\delta(p, b, a) \rightarrow (p, \varepsilon)$
6. $\delta(p, \varepsilon, z_0) \rightarrow (f, \varepsilon)$



Transition Diagram of PDA for $L = \{a^n b^n \mid n >= 0\}$

Design of PDA_Example(1)

*To process the string **w=aaabbb** we can create the transition table for the given transition relation:*

S.No.	State	Unread String	Stack Symbol	Transition
1	s	aaabbb	ε	-
2	q	aaabbb	z_o	1
3	q	aabbb	az_o	2
4	q	abbb	aaz_o	3
5	q	bbb	$aaaz_o$	3
6	p	bb	aaz_o	4
7	p	b	az_o	5
8	p	ε	z_o	5
9	f	ε	ε	6

Transition Table

*Here, after reading all symbols of string aaabbb, stack is empty and also reached in final state f. So, string **w = aaabbb** is accepted.*

Design of PDA_Example(1)

*To process the string **w=aabba** we can create the transition table for the given transition relation:*

S.No.	State	Unread String	Stack Symbol	Transition
1	s	aabba	ε	-
2	q	aabba	z_o	1
3	q	abba	az_o	2
4	q	bba	aaz_o	3
5	p	ba	az_o	4
6	p	a	z_o	4

Transition Table

*We have no transition relation such that **δ(p, a, z_o) so we cannot process the given string** . So, string **w** = **aaabbb** is rejected.*

Design of PDA_Example(2)

Design a PDA accepting a string over {a,b} such that number of a's and b's are equal. i.e. L = {w|wϵ{a,b}* and a's and b's are equal }

The set of string generated by the given language are

{ε, ab, ba, baab, aaabbb, ababaabb,..................}

*Here, the idea is that the number of **a's** and **b's** in the string must be equal and the occurrence of **a and b** can be in any spot.*

We can define the transition function of PDA as:

- *PDA should push the input symbol in stack if the top of stack symbol is same as the input symbol or* $\mathbf{z_0}$
- *otherwise pop the stack.*

Design of PDA_Example(2)

Let us consider PDA to recognize the given language as:

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

where,

Q=*Finite set of states* $=\{ \mathbf{q}, \mathbf{q_1}, \mathbf{q_2}\}$

Σ=*Finite set of input symbols* $=\{\mathbf{a}, \mathbf{b}\}$

Γ=*Finite set of stack alphabets* $=\{\mathbf{a}, \mathbf{b}, \mathbf{z_0}\}$

q_0=*Start state of PDA,* $q_0 \in Q =\{\mathbf{q}\}$

z_0=*Initial stack symbol,* $z_0 \in \Gamma$

F=*Set of final states* $=\{\mathbf{q},\mathbf{q_2}\}$

Design of PDA_Example(2)

Here, $\boldsymbol{\delta}$ *is the transition function of the form*

$$\boldsymbol{\delta(q, a, x) \rightarrow (p, \gamma)}$$

where:

- $\boldsymbol{q}$ *is a state in Q.*
- $\boldsymbol{a}$ *is either an Input Symbol* Σ *or a = ε.*
- $\boldsymbol{x}$ *is a Stack Symbol, that is a member of* $\boldsymbol{\Gamma}$*.*
- $\boldsymbol{p}$ *is a new state.*
- $\boldsymbol{\gamma}$ *is a string of stack symbols that replaces* $\boldsymbol{x}$ *at the top of the stack.*

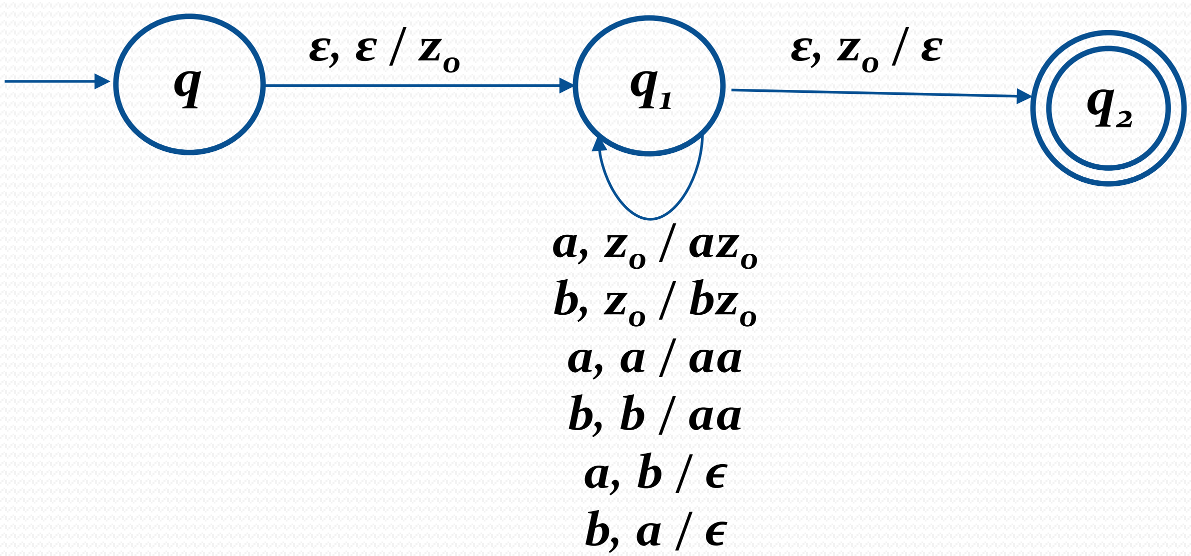
Design of PDA_Example(2)

*Now we need to define the possible transition while processing the string **w** for the PDA to show changes in state, input and stack symbol as:*

1. $\delta(q, \epsilon, \epsilon) \rightarrow (q, z_0)$
2. $\delta(q1, a, z_0) \rightarrow (q1, az_0)$
3. $\delta(q1, b, z_0) \rightarrow (q1, bz_0)$
4. $\delta(q1, a, a) \rightarrow (q1, aa)$
5. $\delta(q1, b, b) \rightarrow (q1, bb)$
6. $\delta(q1, a, b) \rightarrow (q1, \epsilon)$
7. $\delta(q1, b, a) \rightarrow (q1, \epsilon)$
8. $\delta(q1, \epsilon, z_0) \rightarrow (q2, \epsilon)$

Design of PDA_Example(2)

1. $\delta(q, \epsilon, \epsilon) \rightarrow (q_1, z_0)$
2. $\delta(q_1, a, z_0) \rightarrow (q_1, az_0)$
3. $\delta(q_1, b, z_0) \rightarrow (q_1, bz_0)$
4. $\delta(q_1, a, a) \rightarrow (q_1, aa)$
5. $\delta(q_1, b, b) \rightarrow (q_1, bb)$
6. $\delta(q_1, a, b) \rightarrow (q_1, \epsilon)$
7. $\delta(q_1, b, a) \rightarrow (q_1, \epsilon)$
8. $\delta(q_1, \epsilon, z_0) \rightarrow (q_2, \epsilon)$



Transition Diagram of PDA for L = {w|wϵ{a,b} and a's and b's are equal }*

Design of PDA_Example(2)

*To process the string **w=abaabb** we can create the transition table for the given transition relation:*

S.No.	State	Unread String	Stack Symbol	Transition
1	q	abaabb	ε	-
2	q_1	abaabb	z_0	1
3	q_1	baabb	az_0	2
4	q_1	aabb	z_0	7
5	q_1	abb	az_0	2
6	q_1	bb	aaz_0	2
7	q_1	b	az_0	7
8	q_1	ε	z_0	7
9	q_2	ε	ε	8

Transition Table

*Here, after reading all symbols of string aaabbb, stack is empty and also reached in final state q_2. So, string **w = abaabb** is accepted.*

Design of PDA_Example(2)

*For **w=babaabbab***

S.No.	State	Unread String	Stack Symbol	Transition
1	q	babaabbab	ε	-
2	q_1	babaabbab	z_0	1
3	q_1	abaabbab	bz_0	3
4	q_1	baabbab	z_0	6
5	q_1	aabbab	bz_0	3
6	q_1	abbab	z_0	6
7	q_1	bbab	az_0	2
8	q_1	bab	z_0	7
9	q_1	ab	bz_0	3
10	q_1	b	z_0	6
11	q_1	ε	bz_0	3

Transition Table

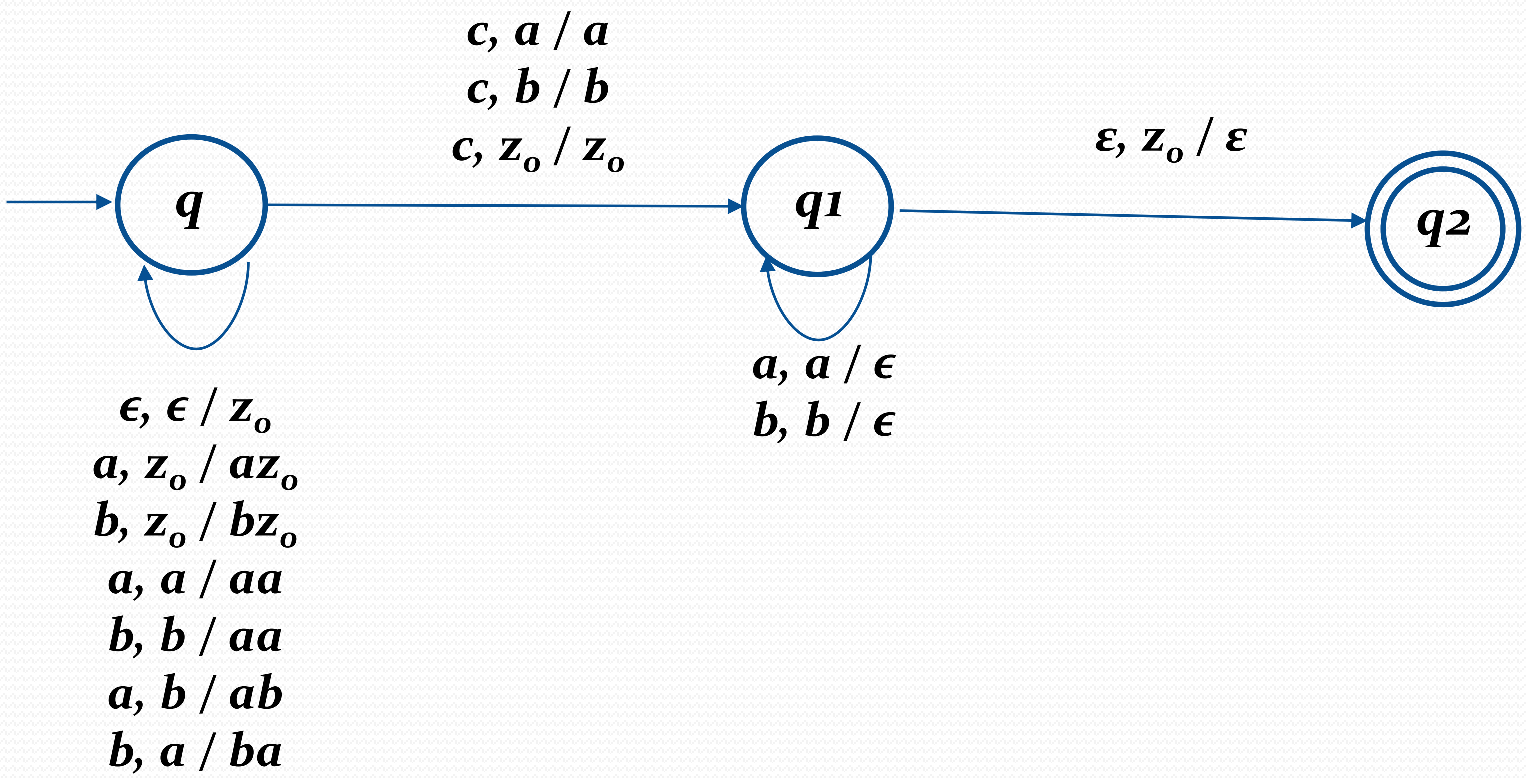
*Here, after reading all symbols of string **babaabbab**, stack is not empty as there is **b** still in the top of stack. So, string **w** = **babaabbab** is rejected.*

Design of PDA_Example(3)

Design a PDA for language L:{wcw^R|wε{a,b}*} and check your design for string w: abbcbba

1. $\delta(q, \epsilon, \epsilon) \rightarrow (q, z_o)$
2. $\delta(q, a, z_o) \rightarrow (q, az_o)$
3. $\delta(q, b, z_o) \rightarrow (q, bz_o)$
4. $\delta(q, a, a) \rightarrow (q, aa)$
5. $\delta(q, b, b) \rightarrow (q, bb)$
6. $\delta(q, a, b) \rightarrow (q1, ab)$
7. $\delta(q, b, a) \rightarrow (q1, ba)$
8. $\delta(q, c, z_o) \rightarrow (q1, z_o)$
9. $\delta(q, c, a) \rightarrow (q1, a)$
10. $\delta(q, c, a) \rightarrow (q1, b)$
11. $\delta(q1, a, a) \rightarrow (q1, \epsilon)$
12. $\delta(q1, b, b) \rightarrow (q1, \epsilon)$
13. $\delta(q1, \epsilon, z_o) \rightarrow (q2, \epsilon)$

Design of PDA_Example(3)



Transition Diagram of PDA for L:{wcw^R|wε{a,b}}*

Language of PDA

- We can define acceptance of any string by PDA in 2 ways
  - Acceptance by final state
  - Acceptance by empty stack
- ***Acceptance by Final State***

Given a PDA M, the language accepted by final state L(M) is

$\{w \mid (q_o, w, z_o) \vdash^* (q, \varepsilon, r)\}$, such that, $q \in F$ and $r \in \Gamma$

  - That is, starting in the initial ID with ***w*** waiting on the input.
  - M consumes ***w*** from the input and enters an accepting state.
  - The content of the stack at that time is irrelevant.

Language of PDA

- ***Acceptance by Empty Stack***

Given a PDA M, the language accepted by final state L(M) is

$$\{w \mid (q_o, w, z_o) \vdash^* (q, \varepsilon, \varepsilon)\}, \text{ such that, } q \in F \text{ and } r\ \Gamma = \varepsilon$$

- That is, starting in the initial ID with ***w*** waiting on the input.
- M consumes ***w*** from the input and at the same time empty its stack.
- The content of the stack at that time is relevant.

Equivalence of PDA and CFG

- The class of languages accepted by PDA is exactly the class of context free languages
  - Each CFL is accepted by some PDA.
  - If a language is accepted by a PDA , it is a CFG

Let $G=(V, \Sigma, R, S)$ *be a CFG*

Now, we need to construct PDA for this grammar such that $L(M) = L(G)$.

Let PDA be:

$$M=(Q, \Sigma, \Gamma, \delta, q_o, z_o, F)$$

Equivalence of PDA and CFG

- To convert a given CFG to it's equivalent PDA, it is needed to convert all the production rules of the given CFG into their equivalent transition functions.
- We can define push down automata for the CFG with two state ***p*** and ***q***, where ***p*** being start state and remains permanently in state ***q*** after its first move.
- Non-terminals of the given CFG are pushed to the stack and terminal symbols are popped from the stack. That is both non-terminals and terminals are used as stack symbol.

Equivalence of PDA and CFG

- Here, idea is that, the stack symbol initially is supposed to be **ε** and PDA starts with state ***p*** and on reading ***ε*** symbol, insert start symbol ***S*** of CFG into stack.

PDA can be defined as

$$M=(Q, \Sigma, \Gamma, \delta, q_o, z_o, F)$$

where,

$M= \{p,q\}$

$\Sigma= \Sigma$ *(non-terminals of CFG)*

$\Gamma= \{ V \cup \Sigma\}$

$q_o=p$

$z_o=\varepsilon$

$F=q$

Equivalence of PDA and CFG

The transition function δ is then defined as:

1) $(p, \varepsilon, \varepsilon) \rightarrow (q, S)$ *[as S is starting non-terminal CFG]*
2) $(q, \varepsilon, A) \rightarrow (q, x)$ *[for each rule $A \rightarrow x$ in CFG]*
3) $(q, a, a) \rightarrow (q, \varepsilon)$ *[for each $a \in \Sigma$]*

Equivalence of PDA and CFG_Example (1)

Consider the grammar $G = (V, \sum, R, S)$ with

$V = \{S, a, b ,c\}$

$\sum = \{a, b, c\}$,

R consists of

$S \rightarrow aSa$

$S \rightarrow bSb$

$S \rightarrow c$

which generates the language $\{wcw^R : w \in \{a, b\}^*\}$. Design a pushdown automata.

Equivalence of PDA and CFG_Example (1)

Let us consider PDA for the given grammar can be defined as

$$M=(Q, \Sigma, \Gamma, \delta, q_o, z_o, F)$$

where,

$M= \{p,q\}$

$\Sigma= \{a,b,c\}$

$\Gamma= \{S, a, b ,c\}$

$q_o=p$

$z_o=\varepsilon$

$F=q$

Equivalence of PDA and CFG_Example (1)

The transition function δ is then defined as:

1) $(p, \varepsilon, \varepsilon) \rightarrow (q, S)$
2) $(q, \varepsilon, S) \rightarrow (q, aSa)$
3) $(q, \varepsilon, S) \rightarrow (q, bSb)$
4) $(q, \varepsilon, S) \rightarrow (q, c)$
5) $(q, a, a) \rightarrow (q, \varepsilon)$
6) $(q, b, b) \rightarrow (q, \varepsilon)$
7) $(q, c, c) \rightarrow (q, \varepsilon)$

The string ***abbcbba*** is accepted by M through the following sequence of moves.

State	Unread Symbol	Stack Content	Transition Used
p	abbcbba	ε	
q	abbcbba	S	1
q	abbcbba	aSa	2
q	bbcbba	Sa	5
q	bbcbba	bSba	3
q	bcbba	Sba	6
q	bcbba	bSbba	3
q	cbba	Sbba	6
q	cbba	cbba	4
q	bba	bba	7
q	ba	ba	6
q	a	a	6
q	ε	ε	5

Equivalence of PDA and CFG_Example (2)

Consider the grammar $G = (V, \sum, R, S)$ with

$V = \{S, A, B, 0, 1\}$

$\sum = \{0, 1\}$

R consists of

$S \rightarrow 0S1 \mid 0AA \mid 1BB$

$A \rightarrow 1A \mid 0$

$B \rightarrow 0B \mid 1$

Design a pushdown automata and check for the string ***w=0010101***