Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Theory of Computation
TY BTech

CONTEXT FREE GRAMMAR(CFG)
&
PUSH DOWN AUTOMATA(PDA)

Course Objectives & Course Outcomes

Course Objectives:

- To understand the basics of automata theory and its operations.
- To understand problem classification and problem solving by machines.
- To study computing machines by describing, classifying and comparing different types of computational models.
- To understand the fundamentals of decidability and computational complexity.

Course Outcomes:

- After completion of this course students will be able:
- To construct finite state machines to solve problems in computing.
- To write mathematical expressions and syntax verification for the formal languages.
- To construct and analyze Push Down Automata and Turing Machine for formal languages.
- To express the understanding of decidability and complexity.

Text Books & Reference Books

- **Text Books**

- John C. Martin, Introduction to Language and Theory of Computation, TMH, 3rdEdition, ISBN: 978-0-07-066048-9.
- Vivek Kulkarni, Theory of Computation, Oxford University Press, ISBN-13: 978-0-19-808458-7.

- **Reference Books**

- K.L.P Mishra,N. Chandrasekaran,Theory of Computer Science (Automata, Languages and Computation), Prentice Hall India, 2nd Edition.
- Michael Sipser, Introduction to the Theory of Computation,CENGAGE Learning, 3rd Edition, ISBN:13:978-81-315-2529-6.
- Daniel Cohen, Introduction to Computer Theory, Wiley India, 2nd Edition, ISBN: 9788126513345.
- Kavi Mahesh, Theory of Computation: A Problem Solving Approach, 1st Edition, Wiley-India, ISBN: 978-81-265-3311-4.

Unit III – CFG & PDA

CFG

Formal definition of Grammar, Chomsky Hierarchy, **CFG :** Formal definition of CFG, Derivations, Parse Tree, Ambiguity in grammars and languages, Language Specification using CFG, Normal Forms: Chomsky Normal Form and Greibach Normal Form. Closure properties of CFL.

Pushdown automata : Description and Definition, language of of PDA , Acceptance of PDA by final State and Empty Stack, Designing PDA, Equivalence of Pushdown automata and CFG, Deterministic Pushdown Automata, Nondeterministic Pushdown Automata. Intersection of CFLs and Regular Language , Introduction to context sensitive languages and context sensitive grammar (CSG)

Planner - Lectures

Lecture No	Topics Covered
1	Chomsky Hierarchy, CFG-Formal Definition, CFL
2	CFG <-> CFL, Derivation, Parse Tree
3	Ambiguity, CNF
4	Conversion of CFG to CNF, Closure Properties
5	GNF, PDA, Formal Definition, ID
6	Acceptance of a string, Designing PDA
7	PDA and NPDA Examples
8	Converting PDA to CFG

Lecture No	Topics Covered
9	Intersection of CFLs and Regular Language , Introduction to context sensitive languages and context sensitive grammar (CSG)

Introduction

- Finite Automata **accept** all regular languages and only regular languages

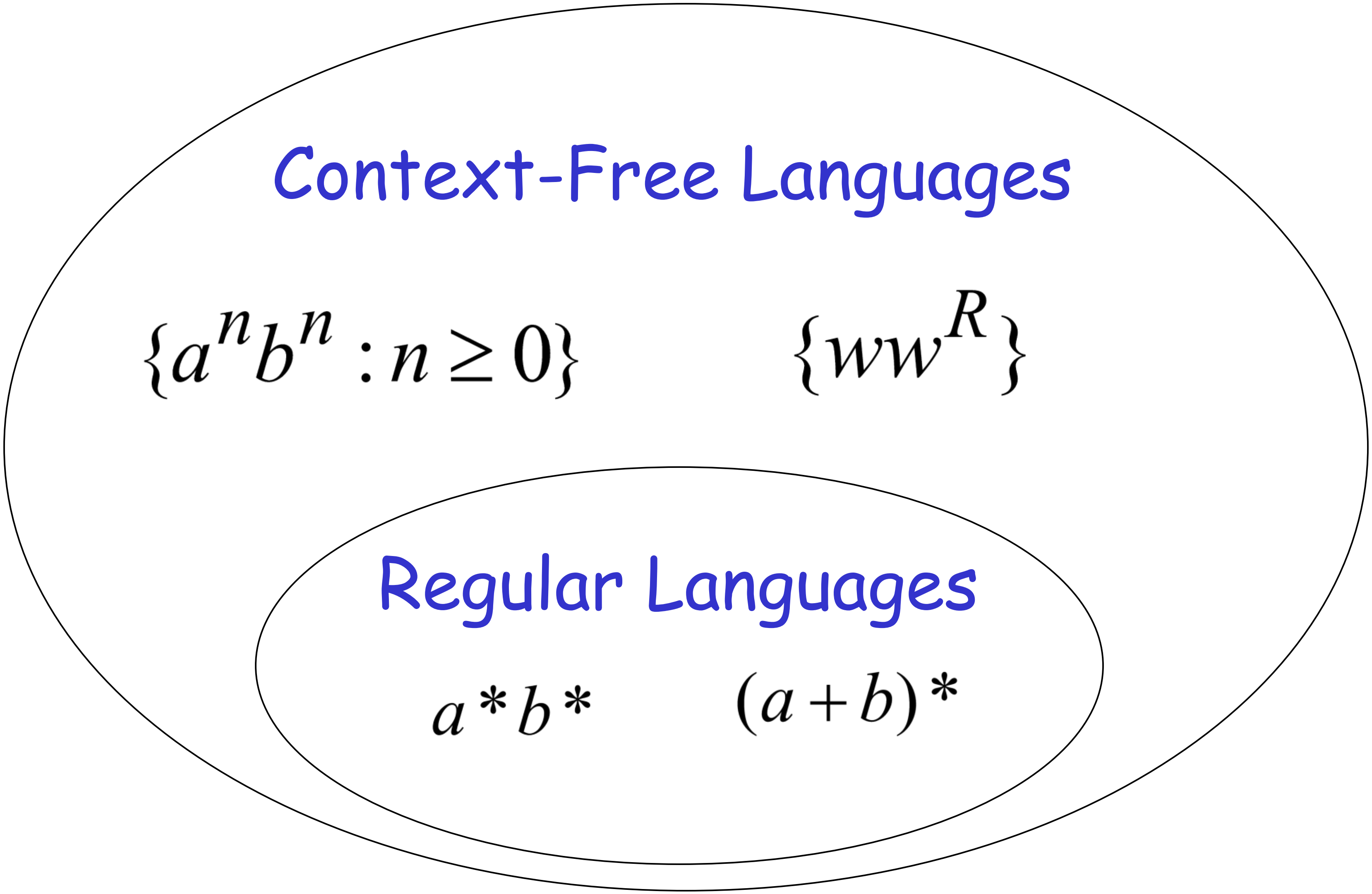
- Many simple languages are non regular:

  - $\{a^n b^n : n = 0, 1, 2, \ldots\}$
  - $\{w : w$ a is palindrome$\}$

and there is no finite automata that accepts them.

- **Context-Free Languages** are a larger class of languages that encompasses all regular languages and many others, including the two above.

Context-Free Languages and Regular Languages



Formal Definition of a Grammar

Grammar: $G = (V, T, S, P)$

Set of variables

Set of terminal symbols

Start variable

Set of productions

Language for the Grammar

Grammar:

$$S \rightarrow aSb$$
$$S \rightarrow \epsilon$$

Language of the grammar:

$$L = \{a^n b^n : n \geq 0\}$$

A Convenient Notation

We write: $$S \overset{*}{\Rightarrow} aaabbb$$

for zero or more derivation steps

Instead of:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

Generalizing:

$$w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow \cdots \Rightarrow w_n$$

Trivially:

$$w \overset{*}{\Rightarrow} w$$

Formal Definition-CFG

Definition: A context-free grammar is a 4-tuple (V, T, P, S), where:

- V is a set (each element in V is called **nonterminal**)
- T is an alphabet (each character in T is called **terminal**)
- P, the set of rules, is a subset of $V \times (T \cup V)^*$

  If $(\alpha,\beta) \in P$, we write production $\alpha \longrightarrow \beta$

  β is called a **sentential form**

- S, the **start symbol**, is one of the symbols in V

Example of Context-Free Grammar

productions

P={S→aSb,
S→ϵ}

$$G = (V, T, S, P)$$

$$V = \{S\}$$

variables

$$T = \{a, b\}$$

terminals

start variable

Context-Free Language

A language L is context-free

if there is a context-free grammar G

with $L = L(G)$

Context-Free Language-Example 1

$$L = \{a^n b^n : \quad n \geq 0\}$$

is a context-free language

since context-free grammar : G

$$\{S \longrightarrow aSb \mid \epsilon\}$$

generates $L(G) = L$

Context-Free Language – Example 2

Context-free grammar : G

$$\{S \longrightarrow aSa \mid bSb \mid \epsilon\}$$

Example derivations:

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abba$$

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abaSaba \Rightarrow abaaba$$

$$L(G) = \{ww^R : \; w \in \{a,b\}^*\}$$

Palindromes of even length

Derivation and Parse Trees

Consider the following example grammar with 5 productions:

$$S \rightarrow AB \qquad A \rightarrow aaA \mid \epsilon \qquad B \rightarrow Bb \mid \epsilon$$

Leftmost and Rightmost Derivation

Consider the following example grammar with 5 productions:

$S \longrightarrow AB \qquad A \longrightarrow aaA|\epsilon \qquad B \longrightarrow Bb|\epsilon$

Leftmost derivation: for the string aab

$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaB \Rightarrow aaBb \Rightarrow aab$$

Rightmost derivation: for the string aab

$$S \Rightarrow AB \Rightarrow ABb \Rightarrow Ab \Rightarrow aaAb \Rightarrow aab$$

Leftmost and Rightmost Derivation

$S \longrightarrow AB \qquad A \longrightarrow aaA|\epsilon \qquad B \longrightarrow Bb|\epsilon$

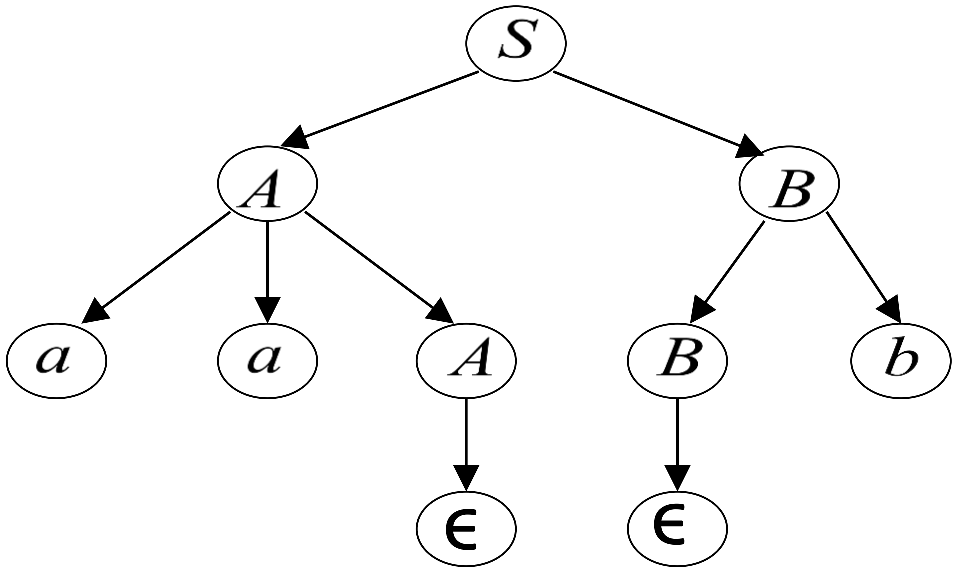
Leftmost derivation:

$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaB \Rightarrow aaBb \Rightarrow aab$$

Rightmost derivation:

$$S \Rightarrow AB \Rightarrow ABb \Rightarrow Ab \Rightarrow aaAb \Rightarrow aab$$

Give same derivation tree



Context Free Language for a CFG

Find the CFL generated for the given CFG.

1. $S \longrightarrow aSb \mid ab$

$L(G)=\{a^n b^n \mid n>=1\}$

2. $S \longrightarrow aB|bA$

   $A \longrightarrow a|aS|bAA$

   $B \longrightarrow b|bS|aBB$

L(G)={x|x containing equal no of a’s and b’s}

CFG for a CFL

1. Write the grammar for generating strings over $\Sigma=\{a\}$, containing any(zero or more) number of a's.

   $S \longrightarrow a \mid aS \mid \epsilon$

2. Find the CFG represented by the RE (a+b)*

   $S \longrightarrow aS \mid bS \mid \epsilon$

3. Write the grammar for all strings consisting of a's and b's with atleast 2 a's.

   $S \longrightarrow AaAaA$

   $A \longrightarrow aA \mid bA \mid \epsilon$

Ambiguous Grammar

A context-free grammar G is ambiguous if there is a string $w \in L(G)$ which has:

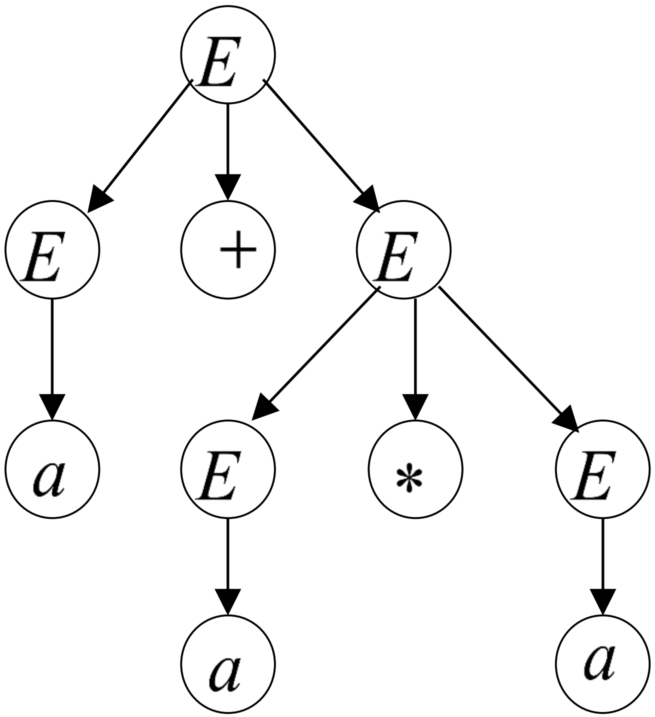
two different derivation trees
or
two leftmost derivations

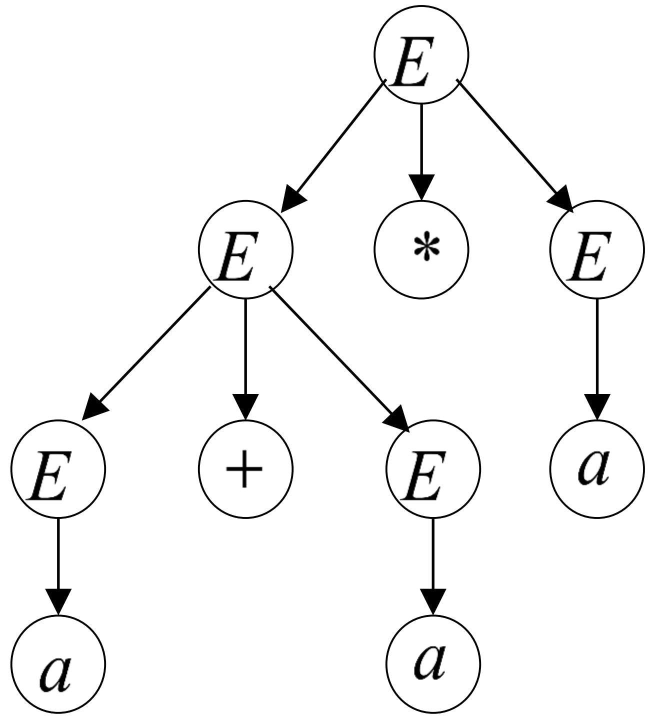
(Two different derivation trees give two different leftmost derivations and vice-versa)

Ambiguous Grammar – An Example

$$E \rightarrow E + E \quad | \; E * E \; | \; (E) \; | \; a$$

string $a + a * a$ has two derivation trees





Ambiguous Grammar – An Example

$$\boxed{E \to E + E \;\mid\; E * E \;\mid\; (E) \;\mid\; a}$$

string $a + a * a$ has two leftmost derivations

$$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E$$
$$\Rightarrow a + a * E \Rightarrow a + a * a$$

$$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow a + E * E$$
$$\Rightarrow a + a * E \Rightarrow a + a * a$$

Removing Ambiguity

Ambiguous Grammar

$$E \to E + E$$
$$E \to E * E$$
$$E \to (E)$$
$$E \to a$$

Equivalent Non-Ambiguous Grammar

$$E \to E + T \mid T$$
$$T \to T * F \mid F$$
$$F \to (E) \mid a$$

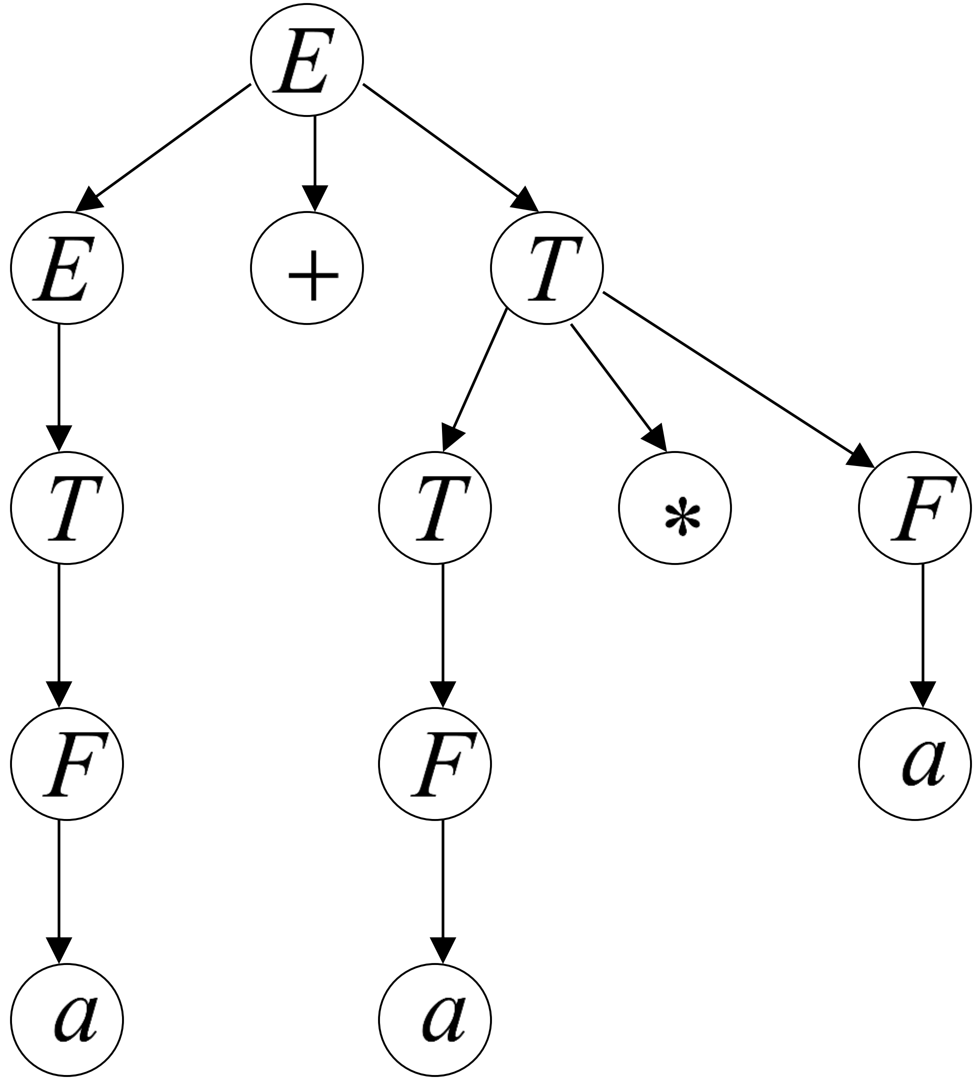
generates the same language

$$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow a+T \Rightarrow a+T*F$$
$$\Rightarrow a+F*F \Rightarrow a+a*F \Rightarrow a+a*a$$

$$E \rightarrow E+T \mid T$$

$$T \rightarrow T*F \mid F$$

$$F \rightarrow (E) \mid a$$



Unique derivation tree and leftmost derivation

For the string a+a*a

Simplification of Context Free Grammar

A CFG G can be simplified as:

1. Each variable and terminal of the CFG should appear in the derivation if at least one word in the L(G).
2. There should not be any production of the form A B where A and B are both non-terminals.

Simplification techniques are:

1.Removal of Useless Symbols

2.Removal of Unit Productions

3.Elimination of ∈ production

Removal of Useless symbol

Ex.1 Consider the following grammar:

G={ (S,A),(1,0),P,S}

Where P consists of the following productions:

S□ 1 0|0 S 1|1 S 0|A|S S

Simplify the grammar by removing the useless symbols if any.

Remove S□ A, as A is useless symbol.

S□ 1 0|0 S 1|1 S 0|S S

Removal of Useless symbol

Ex.2 Consider the following grammar:

G={ (S,A,B),(a),P,S}

Where P consists of the following productions:

S□ A B | a

A□ a

Simplify the grammar by removing the useless symbols if any.

B is useless, Remove B

S□ A | a

A□ a

Removal of Unit Production

- A production rule of the form A → B where A and B are both non-terminals is called a unit production.
- All the other productions (including ∈ productions) are non-unit productions.

Ex.1 Consider the following grammar:

G={ (A,B),(a,b),P,A}

Where P consists of :

A → B, B → a | b

Simplify the grammar by removing the unit productions if any.

On eliminating unit production A → B

A → a|b

Removal of Unit Production

Ex.2 Consider the following grammar:

S → a | X b | a Y a | b | a a

X → Y

Y → b | X

Simplify the grammar by removing the unit productions if any.

Simplified grammar is:

S → a | Yb | aYa | b | aa

Y → b

Removal of ∈ - Production

A production of the form A → ∈ where A is non-terminal is known as ∈ - Production.

Ex.1 Consider the following grammar,

S → a S a | b S b | ∈

Simplify the grammar by eliminating ∈ - Productions if any.

Simplified grammar is G':

S → aSa | bSb | aa | bb

G'=G-ϵ rules.

L(G')=L(G)-{ϵ}

Removal of ∈ - Production

Ex.2 Consider the following grammar,

S □ a | X b | a Y a

X□ Y | ∈

Y□ b | X

Simplify the grammar by eliminating ∈ - Productions if any.

Simplified grammar is:

S □ a | X b | a Y a | b | aa

X□ Y

Y□ b | X

Chomsky Hierarchy

1. Type 0 grammar (Unrestricted Grammar)
2. Type 1 grammar (Context Sensitive Grammar)
3. Type 2 grammar (Context Free Grammar)
4. Type 3 grammar (Regular Grammar)

Chomsky Hierarchy

- **Type 3 grammar:**

  It is also called as **regular grammar.**

  $A \rightarrow \alpha$

Recognized by **FSM.**

Two types of Regular Grammar are:

1. Left Linear Grammar   2. Right Linear Grammar

- **Type 2 grammar:**

  It is also called as **context free grammar.**

  $A \rightarrow \alpha$ where A is NT and α is sentential form.

Start symbol of the Grammar can also appear on RHS.

Recognized by **PDA(Pushdown Automata)**

Chomsky Hierarchy

- Type 1 grammar:

  It is **context sensitive grammar** or context dependent.

1. $\alpha \rightarrow \beta$ where length of β is at least as much as the length of α except $S \rightarrow \in$.
2. The rule $S \rightarrow \in$ is allowed only if start symbol S does not appear on RHS.
3. Productions are of the form

$\alpha_1 A \alpha_2 \rightarrow \alpha 1\ \beta\ \alpha_2\ (\beta \neq \in)$

Recognized by **TM(Turing Machine).**

Chomsky Hierarchy

- **Type 0 grammar:**

  It is **unrestricted grammar** that is no restriction on production.

  $\alpha \rightarrow \beta \qquad (\alpha \neq \in)$

Recognized by TM(Turing machines)

These languages are known as **Recursively Enumerable Languages.**

Normal Forms

There are certain standard ways of writing CFG.

They satisfy certain restrictions on the productions in the CFG.

Then the G is said to be in **Normal forms.**

1. Chomsky Normal Form (CNF)
2. Greibach Normal Form (GNF)

Chomsky Normal Form(CNF)

A CFG is in CNF if every production is of the form

A→ a, where A is NT and a is T

A → BC where A,B and C are NTs

S → $\in$ is in G if $\in$ belongs to L(G).

When $\in$ is in L(G),

we assume that S does not appear on the RHS of any production.

e.g. G is :

S→AB| $\in$

A→ a

B → b

Construction of G in CNF

Step 1. Elimination of null productions and unit productions using previous method.

Let the G is (V,T,P,S)

Step 2. Elimination of terminals on RHS.

Step 3. Restricting the number of variables on RHS.

Reduce to CNF

1. Convert to CNF, S□ aSa | bSb | a | b | aa | bb

Adding A□ a and B□ b we can rewrite the grammar as

S□ ASA|BSB|a|b|AA|BB

A□ a, B□ b

Only S□ ASA and S□ BSB are not in CNF

For S□ ASA we can write

S□ AR_1, R_1 □ SA

For S□ BSB we can write

S□ BR_2, R_2□ SB

Grammar in CNF is

S□ AR_1|BR_2|a|b|AA|BB

A□ a, B□ b

R_1□ SA, R_2□ SB

Reduce to CNF

2. Convert to CNF,

$S \rightarrow bA|aB$, $A \rightarrow bAA|aS|a$, $B \rightarrow aBB|bS|b$

Add $R_1 \rightarrow a$, $R_2 \rightarrow b$

Rewriting the grammar,

$S \rightarrow R_2 A|R_1 B$,

$A \rightarrow R_2 AA|R_1 S|a$,

$B \rightarrow R_1 BB|R_2 S|b$

$R_1 \rightarrow a$, $R_2 \rightarrow b$

$A \rightarrow R_2 AA$ and $B \rightarrow R_1 BB$ are not in CNF

Converted grammar in CNF is

$S \rightarrow R_2 A|R_1 B$,

$A \rightarrow R_2 R_3|R_1 S|a$, $R_3 \rightarrow AA$

$B \rightarrow R_1 R_4|R_2 S|b$, $R_4 \rightarrow BB$

$R_1 \rightarrow a$, $R_2 \rightarrow b$

Reduce to CNF

1. $S \rightarrow aAD$

   $A \rightarrow aB \mid bAB$

   $B \rightarrow b$

   $D \rightarrow d$

2. $S \rightarrow aAbB$

   $A \rightarrow aA \mid a$

   $B \rightarrow bB \mid b$

3. $S \rightarrow ABA$

   $A \rightarrow aA \mid \epsilon$

   $B \rightarrow bB \mid \epsilon$

Greibach Normal Form

A context free grammar is said to be in Greibach Normal Form if all productions are in the following form:

$$A \to \alpha X$$

- A is a non terminal symbols
- α is a terminal symbol
- X is a sequence of non terminal symbols.

  It may be empty.

Closure properties of CFL

CFLs are closed under:

- Union
- Concatenation
- Kleene closure operator
- Substitution
- Homomorphism, inverse homomorphism
- Reversal

CFLs are *not* closed under:

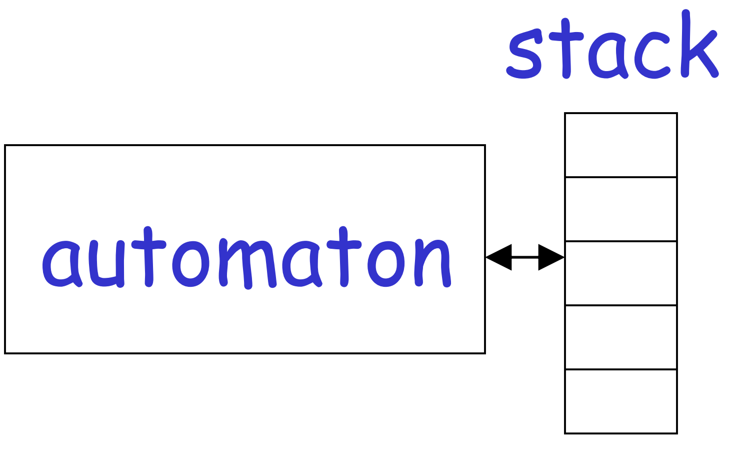
- Intersection
- Difference
- Complementation

Push Down Automata

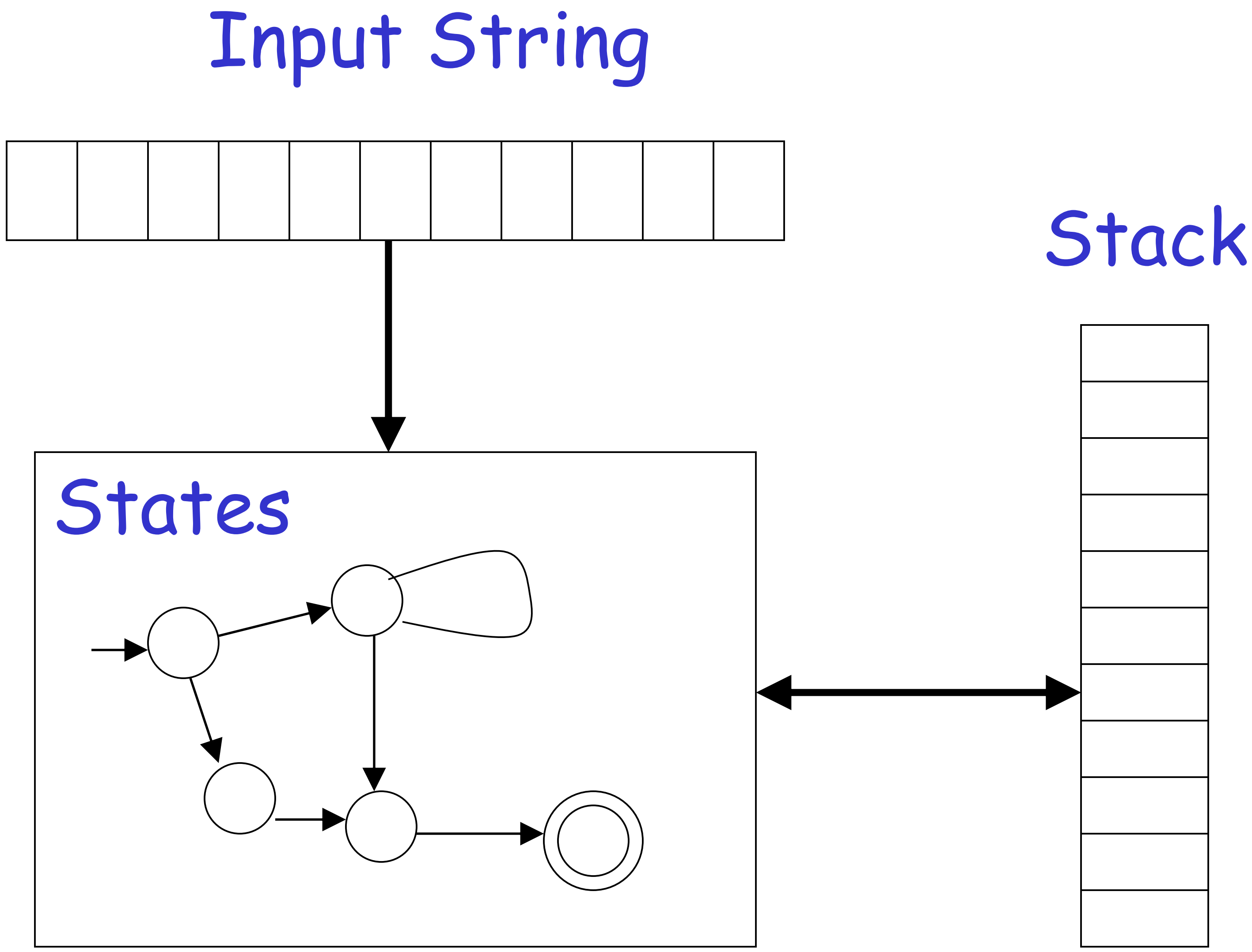
Context-Free Languages

Context-Free Grammars

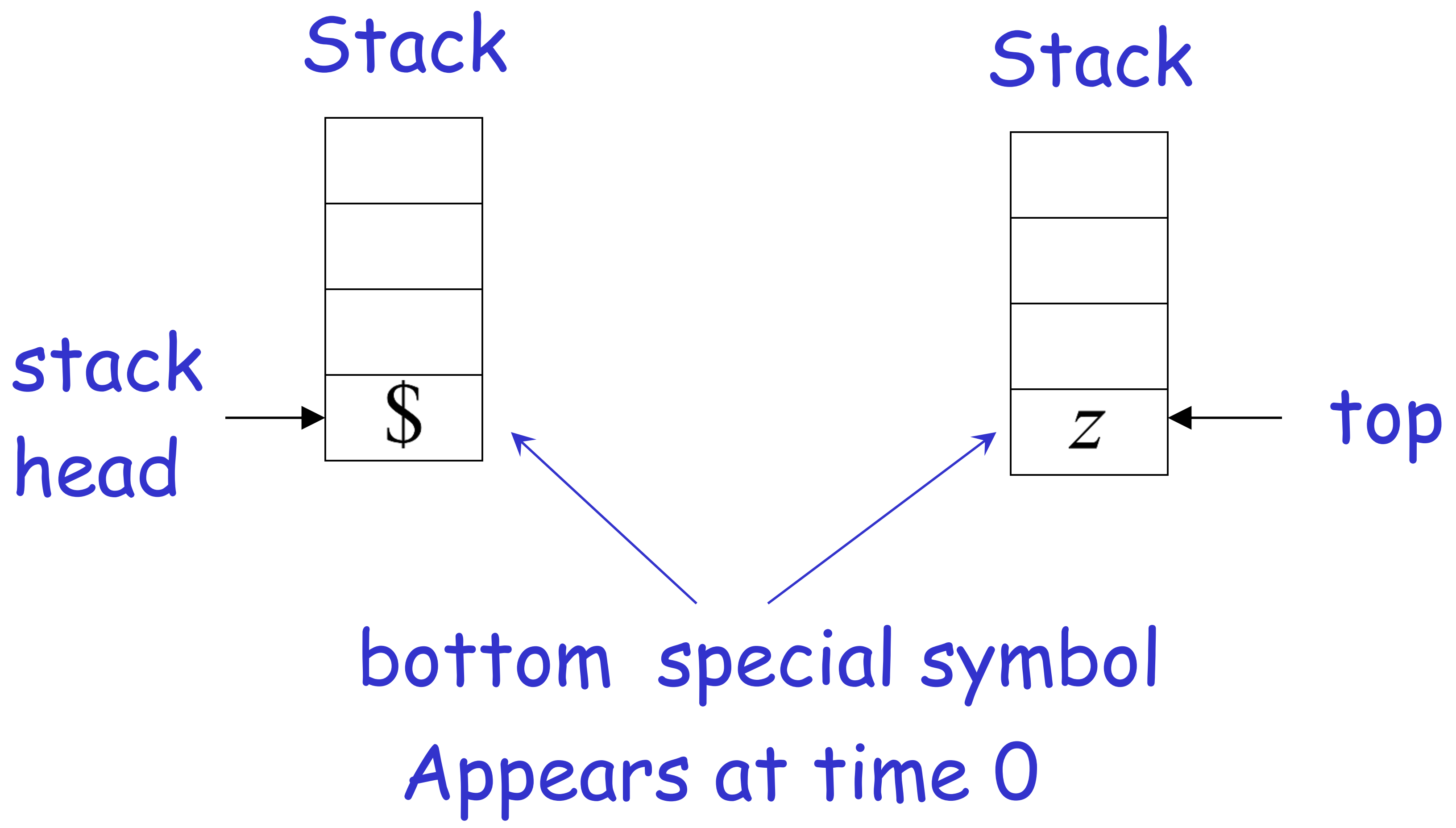
Pushdown Automata



Pushdown Automaton - PDA



Initial Stack Symbol



Formal definition of PDA

The PDA is as:

$A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

Where

Q : A finite set of states

Σ : A finite set of input symbols

Γ : A finite stack alphabet or pushdown symbols

δ : the transition function **$Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$ to the set of finite subsets of $Q \times \Gamma^*$**

q_0: the start state

Z_0 : the start symbol(pushdown symbol)

F : the set of accepting state or final states

Transition Function

δ :The transition function is a triple **δ(q,a,x)**

where

1. q is a state in Q
2. a is either an input symbol in Σ or a =ϵ, the empty string,
3. x is a stack symbol, that is a member of Γ.

The output of δ is a finite set of pairs **(p,ɣ)**

Where

p is the new state and

ɣ is the string of stack symbols that replaces x at the top of the stack.

Instantaneous Description of a PDA

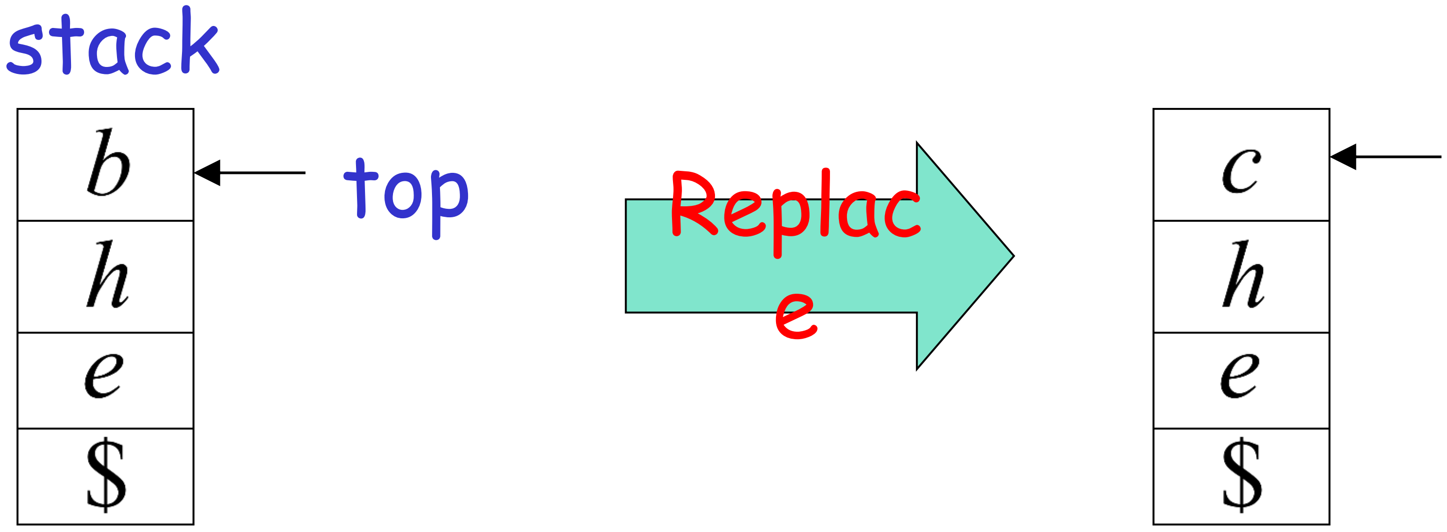
The configuration of a PDA by a **triple (q,w,ɣ)** where

1. **q** is the state
2. **w** is the remaining input
3. **ɣ** is the stack contents

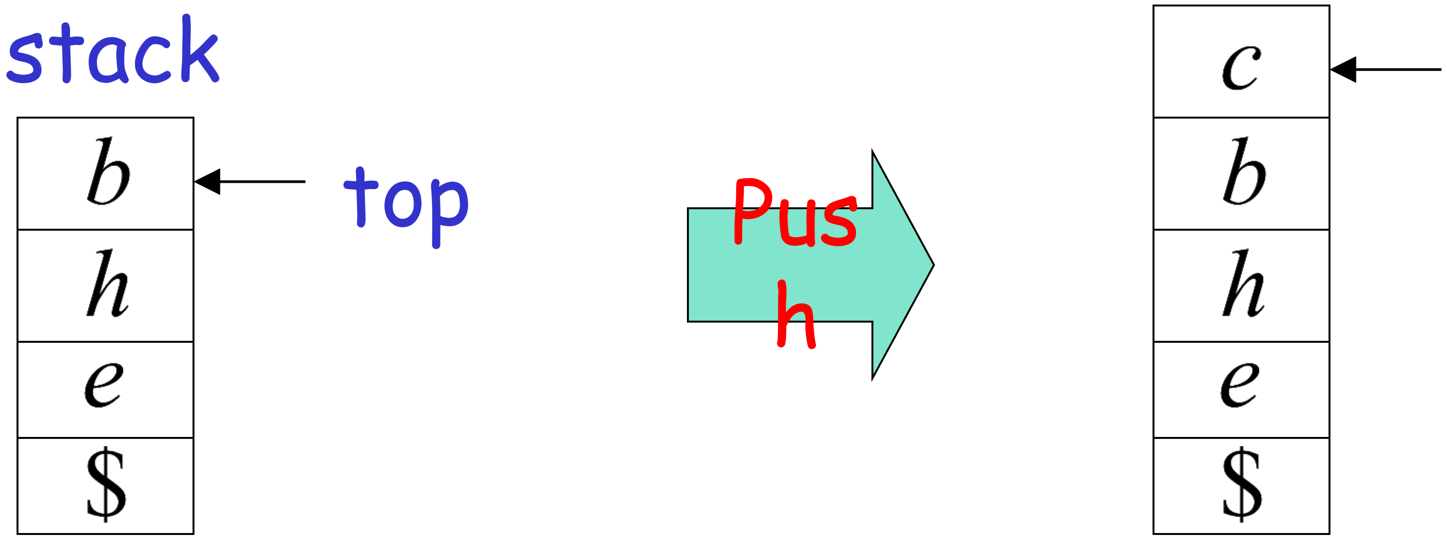
we show the top of the stack at the left end of ɣ and the bottom at the right end.

Such a triple is called an **Instantaneous Description** or **ID** of a PDA.

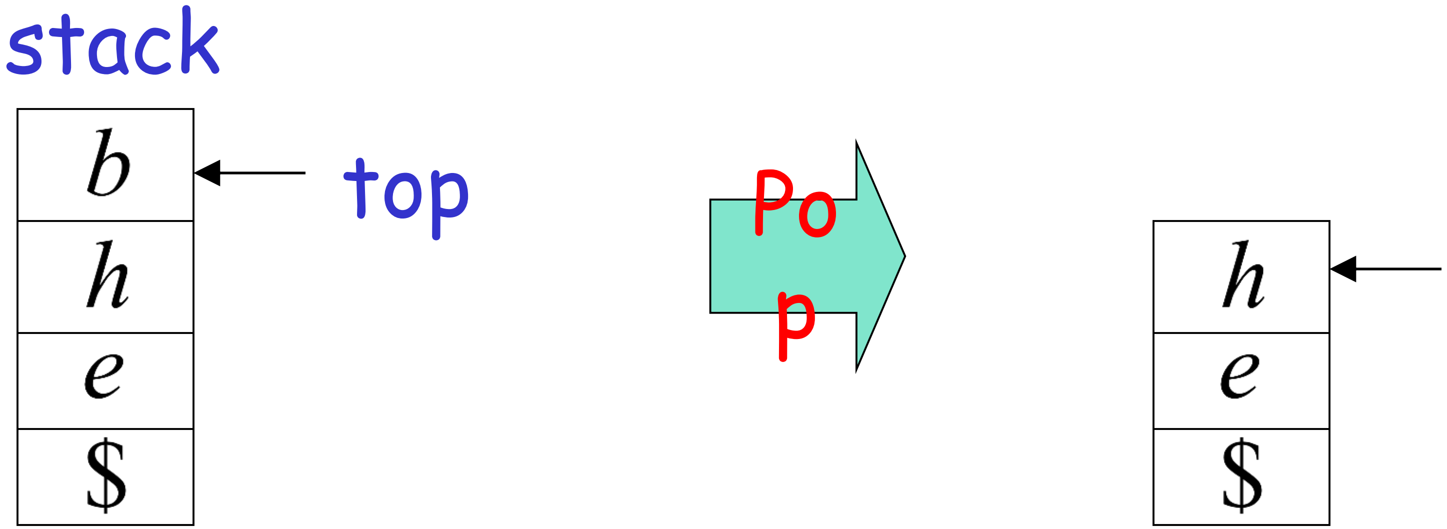
$$\delta(q, a, b) = (q, c)$$



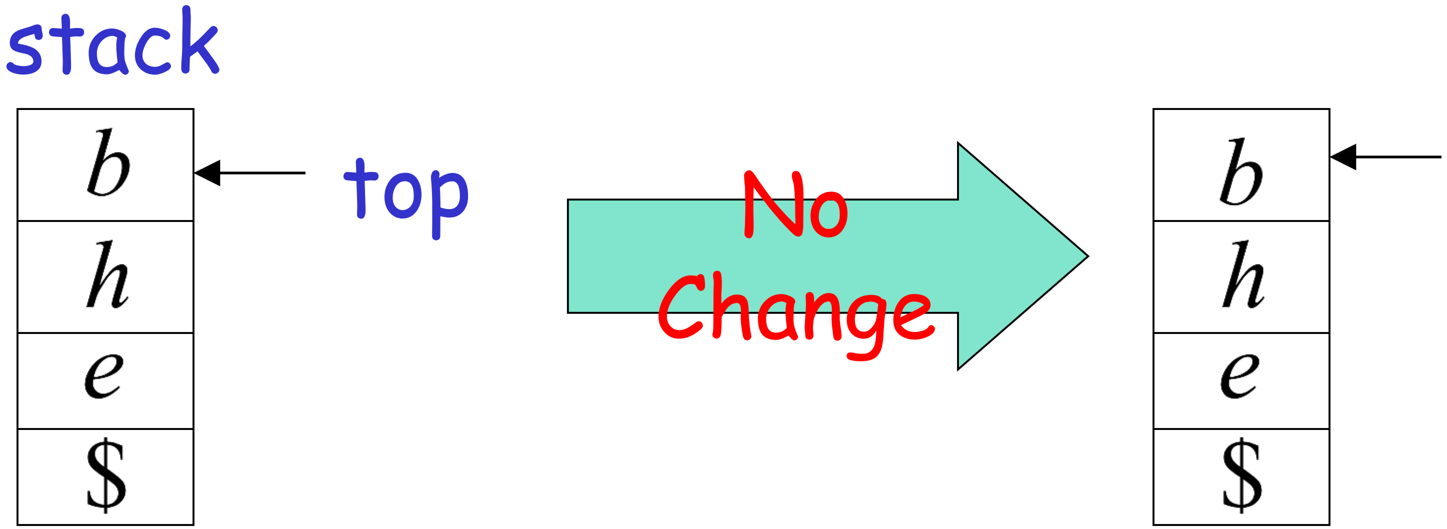
$$\delta(q, a, b) = (q, cb)$$



$$\delta(q, a, b) = (q, \epsilon)$$



$$\delta(q, a, b) = (q, b)$$



A PDA Example

$A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

$Q=\{q_0, q_1, q_2\}$, $\Sigma=\{a,b\}$, $\Gamma=\{Z_0, a\}$, $F=\{q_2\}$, δ as below

Move no	**State**	**input**	**stack symbol**	**Move**
1	q_0	a	Z_0	(q_0,aZ_0)
2	q_0	a	a	(q_0,aa)
3	q_0	b	a	$(q_1, \in)$
4	q_1	b	a	$(q_1,\in)$
5	q_1	$\in$	Z_0	$\mathbf{(q_2, Z_0)}$

Acceptance by PDA using final state

Move no	State	input	stack symbol	Move
1	q_0	a	Z_0	(q_0,aZ_0)
2	q_0	a	a	(q_0,aa)
3	q_0	b	a	$(q_1, \in)$
4	q_1	b	a	$(q_{1,}\in)$
5	q_1	$\in$	Z_0	$(\mathbf{q_2, Z_0})$

For the string aabb

$(q_{0,}$ aabb $_,$ $\mathbf{Z_0})$
$\vdash (q_{0,}$ abb $_,$ $a\mathbf{Z_0})$
$\vdash (q_{0,}$ bb $_,$ $aa\mathbf{Z_0})$
$\vdash (q_{1,}$ b $_,$ $a\mathbf{Z_0})$
$\vdash (q_{1,} \in _, \mathbf{Z_0})$
$\vdash (q_{2,}\in _, \mathbf{Z_0})$

String **aabb** is accepted, as final state q2 is reached on reading string **aabb** completely

Rejection by PDA

Move no	State	input	stack symbol	Move
1	q_0	a	Z_0	(q_0,aZ_0)
2	q_0	a	a	(q_0,aa)
3	q_0	b	a	$(q_1, \in)$
4	q_1	b	a	$(q_1,\in)$
5	q_1	$\in$	Z_0	**(q_2, Z_0)**

For the string aabbb

(q_0, aabbb, Z_0)
$\vdash (q_0, \text{abbb}, aZ_0)$
$\vdash (q_0, \text{bbb}, aaZ_0)$
$\vdash (q_1, \text{bb}, aZ_0)$
$\vdash (q_1, \text{b}, Z_0)$

String **aabbb** is rejected as q_1 is not final state and string **aabbb** is not read completely.

Acceptance by PDA using null store or empty store or empty stack

Move no	**State**	**input**	**stack symbol**	**Move**
1	q_0	a	Z_0	(q_0,aZ_0)
2	q_0	a	a	(q_0,aa)
3	q_0	b	a	$(q_1, \in)$
4	q_1	b	a	$(q_1,\in)$
5	q_1	$\in$	Z_0	$(q_1, \in)$

For the string aabb

$(q_0, aabb, \mathbf{Z_0})$
$\vdash (q_0, abb, a\mathbf{Z_0})$
$\vdash (q_0, bb, aa\mathbf{Z_0})$
$\vdash (q_1, b, a\mathbf{Z_0})$
$\vdash (q_1, \in, \mathbf{Z_0})$
$\vdash (q_1, \in, \in)$

String **aabb** is accepted, as stack is empty on reading string **aabb** completely

Rejection by PDA

Move no	State	input	stack symbol	Move
1	q_0	a	Z_0	(q_0,aZ_0)
2	q_0	a	a	(q_0,aa)
3	q_0	b	a	$(q_1, \in)$
4	q_1	b	a	$(q_1,\in)$
5	q_1	$\in$	Z_0	$(q_1, \in)$

For the string aabbb

$(q_0,\ aabbb,\ Z_0)$
$\vdash (q_0,\ abbb,\ aZ_0)$
$\vdash (q_0,\ bbb,\ aaZ_0)$
$\vdash (q_1,\ bb,\ aZ_0)$
$\vdash (q_1,\ b,\ Z_0)$

String **aabbb** is rejected as string **aabbb** is not read completely and stack is not empty.

Acceptance by PDA

Acceptance of input strings by PDA can be defined in terms of **final states** or in terms of **PDS(pushdown store).**

Let $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA.

The set accepted by final state is defined by

$$T(A) = \{ w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q_f, \Lambda, \alpha) \text{ for some } q_f \in F \text{ and } \alpha \in \Gamma^* \}$$

The set accepted by null store(or empty store)is defined by

$$N(A) = \{ w \in \Sigma^* \mid (q_0, w, Z_0) \vdash^* (q, \Lambda, \Lambda) \text{ for some } q \in Q \}$$

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

Logic:

W

Let

$A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

$Q=\{q_0,q_1\}$,

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

$\Gamma=\{a,Z_0\}$,

$F=\{q_1\}$

is a PDA.

δ (Transition Function) by **Final State** is

<u>**Move no**</u>	<u>**State**</u>	<u>**input**</u>	<u>**stack symbol**</u>	<u>**Move**</u>
1	q_0	a	Z_0	(q_0,aZ_0)
2	q_0	a	a	(q_0,aa)
3	q_0	b	a	$(q_1, \in)$
4	q_1	b	a	$(q_1,\in)$
5	q_1	$\in$	Z_0	$\mathbf{(q_1, Z_0)}$

Acceptance of a string aabb

$(q_0, aabb, Z_0)$
$\vdash (q_0, abb, aZ_0)$
$\vdash (q_0, bb, aaZ_0)$
$\vdash (q_1, b, aZ_0)$
$\vdash (q_1, \in, Z_0)$

δ (Transition Function) by **Empty stack or Empty store or Null stack** is

<u>**Move no**</u>	<u>**State**</u>	<u>**input**</u>	<u>**stack symbol**</u>	<u>**Move**</u>
1	q_0	a	Z_0	(q_0,aZ_0)
2	q_0	a	a	(q_0,aa)
3	q_0	b	a	$(q_1, \in)$
4	q_1	b	a	$(q_1\in)$
5	q_1	$\in$	Z_0	$\mathbf{(q_1, \in)}$

Acceptance of a string aabb

$(q_{0}, aabb, Z_0)$
$\vdash (q_{0}, abb, aZ_0)$
$\vdash (q_{0}, bb, aaZ_0)$
$\vdash (q_{1}, b, aZ_0)$
$\vdash (q_{1}, \in, Z_0)$
$\vdash (q_{1}, \in, \in)$

Ex: PDA to accept language of palindromes with the marker. i.e. $L=\{xcx^r \mid x \in \{a,b\}^*\}$

Let

$A= (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is a PDA.

where

$Q=\{q_0,q_1,q_f\}$,

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

$\Gamma=\{a,b,Z_0\}$,

$F=\{q_f\}$

δ (Transition Function) by **Final State** is

Move no	**State**	**input**	**stack symbol**	**Move**
1	q_0	a	Z_0	(q_0,aZ_0)
2	q_0	b	Z_0	(q_0,bZ_0)
3	q_0	a	a	(q_0,aa)
4	q_0	b	b	(q_0,bb)
5	q_0	a	b	(q_0, ab)
6	q_0	b	a	(q_0, ba)
7	q_0	c	Z_0	(q_1,Z_0)
8	q_0	c	a	(q_1,a)
9	q_0	c	b	(q_1,b)
10	q_1	a	a	$(q_1, \in)$
11	q_1	b	b	$(q_1, \in)$
12	q_1	$\in$	Z_0	**(q_f, Z_0)**

δ (Transition Function) by **Empty stack or Empty store or Null stack** is

Move no	**State**	**input**	**stack symbol**	**Move**
1	q_0	a	Z_0	(q_0, aZ_0)
2	q_0	b	Z_0	(q_0, bZ_0)
3	q_0	a	a	(q_0, aa)
4	q_0	b	b	(q_0, bb)
5	q_0	a	b	(q_0, ab)
6	q_0	b	a	(q_0, ba)
7	q_0	c	Z_0	(q_1, Z_0)
8	q_0	c	a	(q_1, a)
9	q_0	c	b	(q_1, b)
10	q_1	a	a	$(q_1, \in)$
11	q_1	b	b	$(q_1, \in)$
12	q_1	$\in$	Z_0	**$(q_1, \in)$**

Examples for practice

1. PDA for $L=\{a^n b^{2n} \mid n>0\}$
2. PDA for $L=\{a^n b^n c^m d^m \mid n, m>0\}$
3. PDA for $L=\{a^m b^n \mid m>n>=1\}$

Deterministic and non-deterministic PDA

DPDA:

transition function is :

$Q \times \Sigma \times \Gamma$ □ $Q \times \Gamma^*$

e.g. $\delta(q,a,Z)$ is either empty or a singleton.

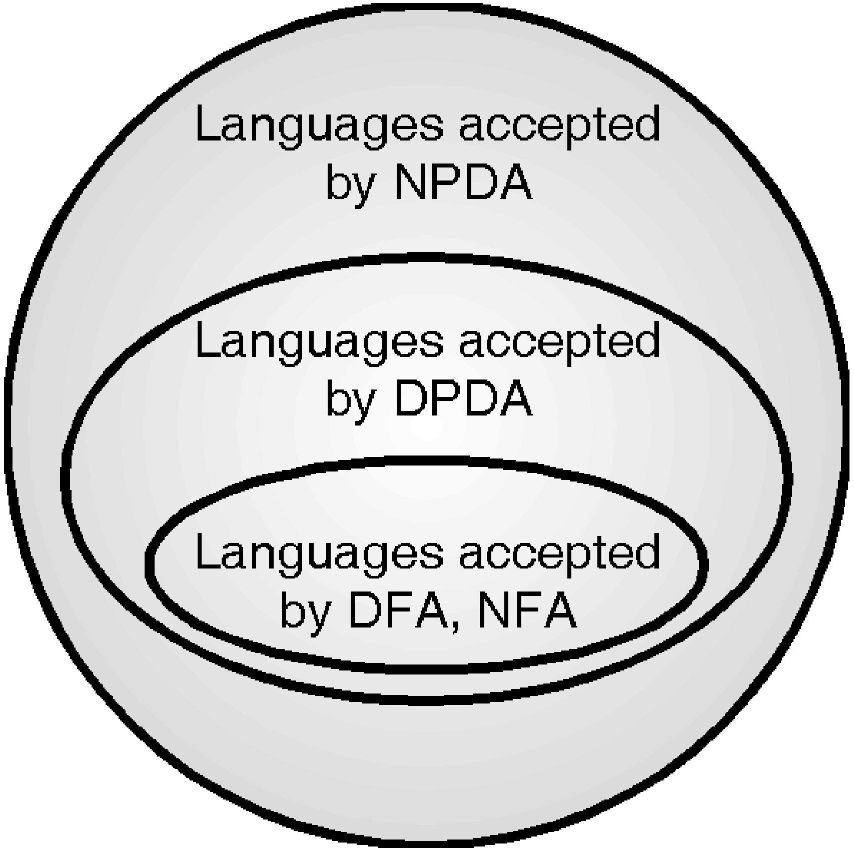
$\delta(q,a,Z) \neq \emptyset$

NPDA:

$Q \times \Sigma \cup \{\epsilon\} \times \Gamma$ □ finite subsets of $Q \times \Gamma^*$

e.g. $\delta(q,a,Z) = \{(p1,\gamma 1), (p2, \gamma 2)\ldots\ldots(pm, \gamma m)\}$

DPDA and NPDA



NPDA and DPDA

- For every NPDA, there may not exist an equivalent DPDA.
- The NPDA can accept any CFL, while DPDA is a special case of NPDA that accepts only a subset of the CFLs accepted by the NPDA.
- Thus, DPDA is less powerful than NPDA.

NPDA to accept language of palindromes without the marker.

Let

A= (Q, Σ, Γ , δ, q_0, Z_0, F) is a PDA.

where

Q={q_0,q_1,q_f},

Σ={a,b},

Γ={a,b,Z_0},

F={q_f}

NPDA to accept language of all palindrome strings

Move no	**State**	**input**	**stack symbol**	**Move**
1	q_0	a	Z_0	$\{(q_0,aZ_0), (q_1, Z_0)\}$
2	q_0	b	Z_0	$\{(q_0,bZ_0), (q_1, Z_0)\}$
3	q_0	a	a	$\{(q_0,aa),(q_1,a)\}$
4	q_0	b	a	$\{(q_0,ba), (q_1,a)\}$
5	q_0	a	b	$\{(q_0, ab), (q_1,b)\}$
6	q_0	b	b	$\{(q_0, bb), (q_1,b)\}$
7	q_0	$\in$	Z_0	$\{(q_1,Z_0)\}$
8	q_0	$\in$	a	$\{(q_1,a)\}$
9	q_0	$\in$	b	$\{(q_1,b)\}$
10	q_1	a	a	$\{(q_1, \in)\}$
11	q_1	b	b	$\{(q_1, \in)\}$
12	q_1	$\in$	Z_0	$\mathbf{\{(q_f,} Z_0\mathbf{)\}}$

CFG to PDA

Theorem: **If L is a CFL then we can construct a PDA A accepting L by empty store ie. L=N(A).**

Proof: We construct A by making use of productions in G.

Let L=L(G) where G=(V, T, P, S) is a CFG.

We construct PDA A as

$A= (Q, \Sigma, \Gamma, \delta, q, Z_0, F)$

where $\Sigma=T$

Γ is (V U T)

$Z_0 = S$

$F = \Phi$

δ is defined as

R_1 : $\delta(q, \in, A) = \{(q, \alpha) \mid A \square \alpha$ is in $P\}$

R_2 : $\delta(q, a, a) = \{(q, \in)\}$ for every a in Σ.

CFG to PDA

1 .Construct a PDA for the CFG

$S \rightarrow 0BB$

$B \rightarrow 0S \mid 1S \mid 0$

Test whether 010^4 is in N(A).

We construct PDA A as

$A = (Q, \Sigma, \Gamma, \delta, q, Z_0, F)$

$Q = \{q\}$

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

$\Gamma = \{S, B, 0, 1\}$

$Z_0 = S$

$F = \Phi$

Move no	**State**	**input**	**stack symbol**	**Move**
1	q	∈	S	{(q,0BB)}
2	q	∈	B	{(q,0S), (q,1S), (q,0)}
3	q	0	0	{(q, ∈)}
4	q	1	1	{(q,∈)}

CFG to PDA

1 .Construct a PDA for the CFG

S□ 0BB

B□ 0S | 1S |0

Test whether 010^4 is in N(A).

2. Convert the grammar

S□ aSb | A

A□ bSa |S| ∈

To a PDA that accepts the same language by empty stack.

Thank You…!!!