Formal Languages & Autometa Theory

Lecture Material on
PUSHDOWN AUTOMATA

PNRL Chandra Sekhar

Department of Computer Science & Engineering GST, GITAM

October 11, 2023

The following material will give you an overview of what you learn in the class

1 Introduction

Pushdown automata are nondeterministic finite state machines augmented with additional stack memory, which is why the term "pushdown" is used, as elements are pushed down onto the stack. Pushdown automata are computational models like finite state machines and can do more than a finite state machine but less than a Turing machine.

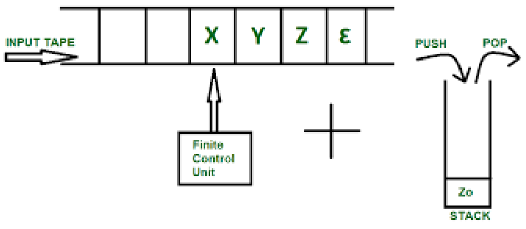


Figure 1: Pushdown Automaton

Pushdown automata accept context-free languages, which include the set of regular languages. Pushdown automata can be useful when considering parser design and any area where context-free grammars are used, such as in computer language design.

In pushdown automata, state transitions include adding a symbol to the string generated, as in FSMs. Still, state transitions can also include instructions about pushing and popping elements to and from the stack. At each transition, a pushdown automaton can push a symbol to the stack, pop a symbol from the stack, do both, or do no operations to the stack.

The pushdown automaton starts with an empty stack and accepts if it ends in an accepting state at the end or if the stack can be empty.

Pushdown automata can be modeled as a state transition diagram with added instructions about the stack. Where in the finite state machine, the arrows between states were labeled with a symbol that represented the input symbol from the string, a pushdown automaton includes information in the form of input symbol followed by the symbol that is currently at the top of the stack, followed by the symbol to replace the top of the stack with. Commas, slashes, or arrows sometimes separate these instructions.

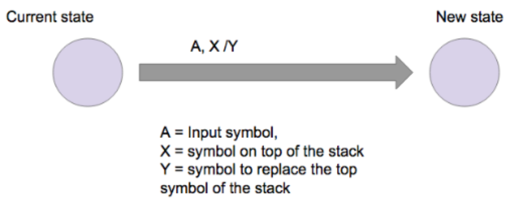


Figure 2: State Transitions on Pushdown Automaton

1.1 Formal Definition

A pushdown automaton is formally defined as a 7-tuple: $M(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ Where:

- Q is the finite set of states.
- Σ is the finite alphabet of the input alphabet.
- Γ is the finite stack alphabet.
- δ is the set of transition functions mapped as $QX(\Sigma \cup \{\epsilon\})X\Gamma \vdash (QX\Gamma)$.
- q_0 is the initial state of the PDA, $\in Q$
- Z_0 is the initial stack symbol, $\in \Gamma$
- F is the set of accepting states, $\subset Q$

1.2 Instataneous Description

Instantaneous Description (ID) is an informal notation describing how a PDA "computes" an input string and decides whether the string is accepted or rejected.

An ID is a triple (q_i, w, α), where:

- q_i is the current state of read head.
- 'w' is the position of the string still to be read on input.
- α is the string on the pushdown store. (It is customary to not write $w\&\alpha$ completely but to write the first symbol of w and the top most symbol of α)

The transition of a PDA can be defined as: $\delta(q_0, 0, Z_0) \vdash (q_1, 0Z_0)$

At each transition, a pushdown automaton can push a symbol to the stack, pop a symbol from the stack, do both, or do no operations to the stack.

1.3 Acceptance of a String

The acceptance of a string in PDA is of two types:

- **Final State:** At the end of the string, the read head is in one of the final states F.
- **Null Store:** The pushdown store is empty at the end of the string.

Problem 1

Design a pushdown automata for language $\{a^n b^n | n > 0\}$

Solution. Design M as follows:

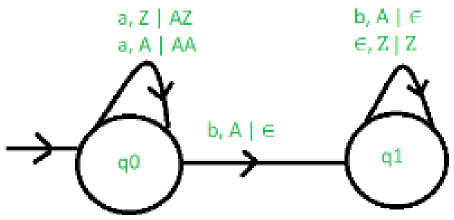


Figure 3: Pushdown Automaton for $a^n b^n$

where Q = { q0, q1 } and Σ = { a, b } and Γ = { A, Z } and δ is defined as:

- $\{\delta(q_0, a, Z) \vdash (q_0, AZ)$,
- $\delta(q_0, a, A) \vdash (q_0, AA)$,
- $\delta(q_0, b, A) \vdash (q_1, \epsilon)$,
- $\delta(q_1, b, A) \vdash (q_1, \epsilon)$,
- $\delta(q_1, \epsilon, Z) \vdash (q_1, \epsilon)\}$

Let us trace the above PDA with a string w = aaabbb as follows:
$(q_0, aaabbb, Z_0) \implies (q_0, aabbb, aZ_0)$ – push a unto stack

- $\implies (q_0, abbb, aaZ_0)$ – push a unto stack
- $\implies (q_0, bbb, aaaZ_0)$ – push a unto stack
- $\implies (q_1, bb, aaZ_0)$ – pop a from top of stack
- $\implies (q_1, b, aZ_0)$ – pop a from top of stack
- $\implies (q_1, \epsilon, Z_0)$ – pop a from top of stack
- $\implies (q_1, \epsilon, \epsilon)$ – pop Z_0 from stack

  $\because$ at the end of the string, the stack becomes completely empty as well as it reaches to the final state q_1, the string w = aaabbb is accepted by the given PDA.

1.4 Language accepted by a PDA

- For a PDA $M(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, we define L(M), the language accepted by a PDA by the final state as:
  $L(M) = \{w|(q_0, w, Z_0) \vdash^* (p, \epsilon, \gamma)$ for some $p \in F \& \gamma \in \Gamma^*\}$
- For a PDA $M(Q, \Sigma, \Gamma, \delta, q_0, Z_0, \Phi)$, we define L(M), the language accepted by a PDA by Null Store as:
  $L(M) = \{w|(q_0, w, Z_0) \vdash^* (p, \epsilon, \epsilon)$ for some $p \in Q$ }

Problem 2

Practice designing PDA for the following languages and trace the acceptance of relevant strings.

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

b) L = $\{a^n b^{2n} | n \geq 0\}$

c) L = $\{0^n 1^n 2^m | n, m \geq 0\}$

d) L = $\{0^n 1^{n+2} | n \geq 0\}$

e) L = $\{wcw^R | w \in \{a, b\}^*\}$

2 Equivalence of PDA's

Let M_1 be a PDA by final state and is formally defined as a 7-tuple: $M(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ and its transition diagram as:

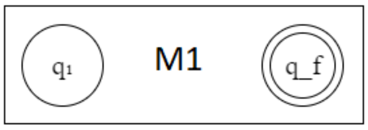


Figure 4: Pushdown Automaton for M_1

The acceptance of a string or configuration of the machine as: $(q_1, w, Z_1) \vdash^*_{M_1} (q_f, \epsilon, \gamma)$

Let M_2 be a PDA by null store and is formally defined as a 7-tuple: $M(Q, \Sigma, \Gamma, \delta, q_0, Z_0, \Phi)$ and its transition diagram as:

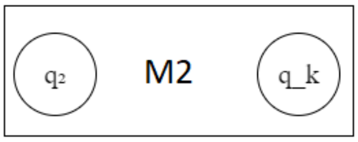


Figure 5: Pushdown Automaton for M_2

The acceptance of a string or configuration of the machine as: $(q_1, w, Z_1) \vdash^*_{M_1} (q_k, \epsilon, \epsilon)$

2.1 Equivalence of PDA by Final state to Null Store

Given M_1 PDA by final state which recognizes a string w, obtain an equivalent machine M_n to accept the same string w by null store as:

$M_n(Q_1 \cup \{q_0, q_k\}, \Sigma, \Gamma_1 \cup \{Z\}, \delta_1 \cup P, q_0, Z, \phi)$
where $P = \{\delta(q_0, \epsilon, Z) \vdash (q_1, Z_1Z), \delta(q_f, \epsilon, \gamma_1) \vdash (q_k, \epsilon), \delta(q_k, \epsilon, \gamma_i) \vdash (q_k, \epsilon)\}$
The corresponding transition diagram for M_n is:

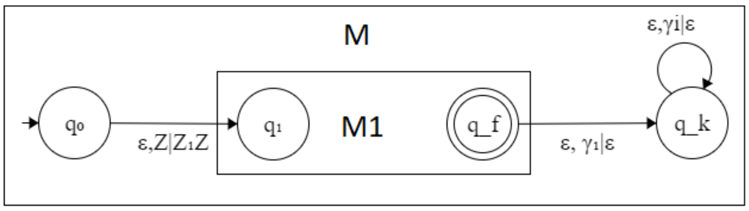


Figure 6: Pushdown Automaton for M_n

The configuration of the machine M_n is:
$\delta(q_0, w, Z) \vdash (q_1, w, Z_1Z) \vdash^*_{M_1} (q_f, \epsilon, \gamma) \vdash (q_k, \epsilon, \gamma_i) \vdash^* (q_k, \epsilon, \epsilon)$
Since both machines accepted the strings they are called as equivalent
$M_1 \equiv M_n \implies L(M_1) = L(M_n)$

2.2 Equivalence of PDA by Null store to Final State

Given M_2 PDA by null store which recognizes a string w, obtain an equivalent machine M_f to accept the same string w by final as:
$M_f(Q_2 \cup \{q_0, q_f\}, \Sigma, \Gamma_2 \cup \{Z\}, \delta_2 \cup P, q_0, Z, \{q_f\})$
where $P = \{\delta(q_0, \epsilon, Z) \vdash (q_1, Z_2Z), \delta(q_k, \epsilon, Z) \vdash (q_f, Z)\}$
The corresponding transition diagram for M_f is:

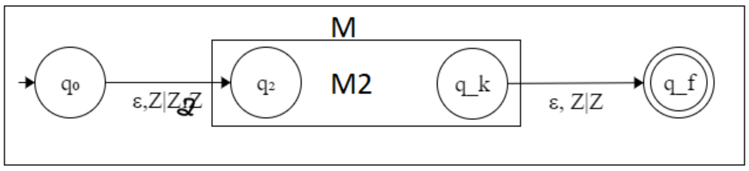


Figure 7: Pushdown Automaton for M_f

The configuration of the machine M_n is:
$\delta(q_0, w, Z) \vdash (q_2, w, Z_2Z) \vdash^*_{M_2} (q_k, \epsilon, Z) \vdash (q_f, \epsilon, Z)$
Since both machines accepted the strings they are called as equivalent
$M_2 \equiv M_f \implies L(M_2) = L(M_f)$

3 Deterministic and Non-Deterministic PDA

A deterministic pushdown automaton is formally defined as a 7-tuple: $M(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ such that:

- $\forall q \in Q, a \in \Sigma_\epsilon$ and $X \in \Gamma$ then $\delta(q, a, X)$has at most one element. In other words, the PDA has only one move from the state q on input symbol a when X is on top of the stack.
- $\forall q \in Q, X \in \Gamma$, if $\delta(q, \epsilon, X) \neq \Phi$ then $\delta(q, a, X) = \Phi$, $\forall a \in \Sigma$. In other words, if there is a move on ϵ from a state q, there should be no other move on any input symbol.

For any PDA to be deterministic, both the above conditions must be satisfied; otherwise, if any condition fails, then the PDA is non-deterministic. In other words, if there are multiple moves from the same state on the same input symbol as well as from the same state there are transitions on both input symbols as well as ϵ then such PDA is called non-deterministic.

The PDA in problem 1 is deterministic as the two conditions are satisfied.

Problem 3

Design PDA for L=$\{ww^R | w \in \{a, b\}^*\}$ and show it is non-deterministic.

Solution. The language L = {aa,bb,abba,baba, abaaba,.....} where each string is a palindrome. Nod design a PDA to accept such types of strings as:

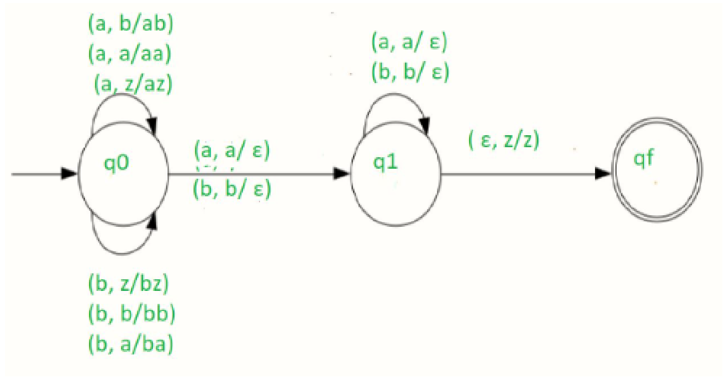


Figure 8: Pushdown Automaton for ww^R

where the transition are :

- $\{\delta(q0, a, z) \vdash (q0, az)$,
- $\delta(q0, a, a) \vdash \{(q0, aa), (q1, \epsilon)\}$,
- $\delta(q0, b, z) \vdash \{(q0, bz), (q1, \epsilon)\}$,
- $\delta(q0, b, b) \vdash (q0, bb)$,
- $\delta(q0, a, b) \vdash (q0, ab)$,
- $\delta(q0, b, a) \vdash (q0, ba)$,

- $\delta(q1, a, a) \vdash (q1, \epsilon)$,
- $\delta(q1, b, b) \vdash (q1, \epsilon)$,
- $\delta(q1, \epsilon, z) \vdash (qf, z)\}$

To trace out the string "abbbba" the ID is as follows: $\delta(q0, abbbba, z) \vdash \delta(q0, bbbba, az) \vdash \delta(q0, bbba, baz) \vdash \delta(q0, bba, bbaz) \vdash \delta(q1, ba, baz) \vdash \delta(q1, a, az) \vdash \delta(q1, \epsilon, z) \vdash \delta(q2, \epsilon, z)$

$\because$ after reading the string "abbba", the pda reaches to a final state, the string is accepted.

Since the transitions have multiple actions:

- $\delta(q0, a, a) \vdash \{(q0, aa), (q1, \epsilon)\}$,
- $\delta(q0, b, z) \vdash \{(q0, bz), (q1, \epsilon)\}$,

the above PDA is considered to be non-deterministic.

3.1 Difference between DPDA and NPDA

Table 1: Difference between DPDA and NPDA

DPDA (Deterministic Pushdown Automata)	NPDA (Non-deterministic Pushdown Automata)
Less powerful than NPDA	More powerful than DPDA
every DPDA can be converted to a corresponding NPDA	Not possible to convert every NPDA to a corresponding DPDA.
DPDA accepts DCFL(Deterministic Context-free Language), a subset of NCFL(Non-deterministic Context-free Language) accepted by NPDA.	The language NPDA accepts is called NCFL(Non-deterministic Context-free Language).
DCFL $\subseteq$ NFCL	NCFL $\not\subseteq$ DCFL

4 Correspondence between CFG and PDA

CFG and PDA are equivalent in power: a CFG generates a context-free language and a PDA recognizes a context-free language. So, one is given we can convert into other to recognize the same context-free language.

i.e., if a grammar G is context-free, we can build an equivalent nondeterministic PDA which accepts the language that is produced by the context-free grammar G. A parser can be built for the grammar G. Also, if P is a pushdown automaton, an equivalent context-free grammar G can be constructed where L(G) = L(P)

This is a useful application where it allows a CFG to be used to specify a programming language and the equivalent PDA to be used to implement its compiler.

4.1 Equivalence of CFG to PDA

Algorithm: If L is a context-free language obtained from the grammar G, obtain an equivalent PDA M to recognize the same L. **Method to convert given CFG G into PDA M: Input:** CFG G = (V, T, P, S)

Output: PDA M = $(Q, \Sigma, \Gamma, \delta, q_0, Z, F)$

- Bring the productions of the G into GNF (Optional).
- Let q be the only state of the non deterministic PDA M.
- The start symbol of CFG S will act as initial pushdown store symbol of the PDA
- All non-terminals or variables of the CFG will be the stack symbols 'Γ' of the PDA and all the terminals of the CFG will be the input symbols 'Σ' of the PDA.
- For each production in the form $A \to \alpha$ in the CFG G (where $A \in V, \alpha \in (V \cup T)^*$) add a transition to M as $\delta(q, \epsilon, A) \vdash (q, \alpha)$.
- For each production in the form $A \to a\alpha$ in the CFG G (where $A \in V, \alpha \in V^*$) add a transition to M as $\delta(q, a, A) \vdash (q, \alpha)$.
- For every terminal $a \in T$ add the transition $(q, a, a) \vdash (q, \epsilon)$

Problem 4

Obtain an equivalent non deterministic PDA from the given CFG $G(\{S, X\}, \{a, b\}, P, S)$ where where the productions P= $\{S \to XS|\epsilon, A \to aXb|Xb|ab\}$.

Solution. Let the equivalent PDA M = $(\{q\}, \{a, b\}, \{a, b, X, S\}, \delta, q, S, \Phi)$ where δ is defined as follows:

- For all productions $A \to \alpha$
- $S \to XS|\epsilon \implies \delta(q, \epsilon, S) \vdash \{(q, XS), (q, \epsilon)\}$
- $A \to aXb|Xb|ab \implies \delta(q, \epsilon, A) \vdash \{(q, aXb), (q, Xb), (q, ab)\}$
- For all Terminals
- $\delta(q, a, a) \vdash (q, \epsilon), \delta(q, b, b) \vdash (q, \epsilon)$

Problem 5

Convert the following grammars into equivalent PDA's 1. G1 = $\{S \to 0S1|A, A \to 1A0|S|\epsilon\}$
2. G2 = $\{S \to 0BB, B \to 0S|1S|0\}$
3. G3 = $\{S \to aSb|a|b|\epsilon\}$
4. G4 = $\{S \to ASB|ab, A \to aA|a, B \to bB|b\}$

5. G5 $= \{S \rightarrow 0CC|ab, C \rightarrow 1S|0S|0\}$

4.2 Equivalence of PDA to CFG

Given a PDA M$(Q, \Sigma\Gamma, \delta, q_0, Z_0, \Phi)$ accepting a language L(M) by null store, the corresponding CFG G(V, T, P, S) can be generated as follows:

- **Rule1:** The set of Variables (V) is constructed as follows:

  $V = \{S\} \cup \{[q_i, Z, q_j]\} \qquad \forall q_i, q_j \in Q \& Z \in \Gamma$

- **Rule2:** The production fo start symbol S are defined as follows:

  $S \rightarrow [q_0, Z_0, q]$ where q_0 is the initiate state of PDA M,

  The S production is created for every $q \in Q$

- **Rule3:** The transition function involving the pop symbol from the pushdown store such as $\delta(q_i, a, Z) \vdash (q, \epsilon)$ leads to the production

  $[q_i, Z, q] \rightarrow a$

- **rule4:** The transition function that involves pushing a symbol unto the stack my either keep the pushdown store unchanged or push a string of symbols unto the stake as represented by the following transition:

  $\delta(q_i, a, Z) \vdash (q, Zz_1Z_2....Z_m]$ leads to the production

  $[q_j, Z, q] \rightarrow a[q_0, z_1, q_1][q_1, Z_2, q_2], ...[q_m - 1, Z_m, Q_m]$

  where $q_1, q_2....q_m$ is the possible list of states for the PDA.

- **Rule5:** After obtaining all the productions, identify the variables and productions that are not participating in the grammar and remove them.

  Finally, rename all the remaining variables and present the obtained grammar?

Problem 6

Convert the following PDA M to a CFG G. M $= (q_0, q_1, 0, 1, X, Z_0, \delta, q_0, Z_0, \Phi)$ with δ:

- $\delta(q_0, 0, Z_0) \vdash (q_0, XZ_o)$ (1a)
- $\delta(q_0, 0, X) \vdash (q_0, XX)$ (1b)
- $\delta(q_0, 0, X) \vdash (q_1, \epsilon)$ (2a)
- $\delta(q_1, 1, X) \vdash (q_1, \epsilon)$ (2b)
- $\delta(q_1, \epsilon, X) \vdash (q_1, \epsilon)$ (2c)
- $\delta(q_1, \epsilon, Z_0) \vdash (q_1, \epsilon)$ (2d)

Solution. We need to obtain a CFG G (V,T,P,S) as follows: From the given PDA $M(\Sigma)$, T = 0,1. Let us make the Start Symbol as S. The possible variables (V) are:

$V = (S, [q_0, X, q_0], [q_0, X, q_1], [q_1, X, q_0], [q_1, X, q_1], [q_0, Z_0, q_0], [q_0, Z_0, q_1], [q_1, Z_0, q_0], [q_1, Z_0, q_1])$
The remaining are productions. Let us apply the rules from 2 to 4 as follows:

Based on rule 2, the S productions are: $S \to [q_0, Z_0, q_0]$,$S \to [q_0, Z_0, q_1]$

Based on rule 3, the production where the transition involves popping of a symbol from the stack are:

The transition $\delta(q_0, 0, X) \vdash (q_1, \epsilon)$ leads to the production $[q_0, X, q_1] \to 0$.

The transition $\delta(q_1, 1, X) \vdash (q_1, \epsilon)$ leads to the production $[q_1, X, q_1] \to 1$.

The transition $\delta(q_1, \epsilon, X) \vdash (q_1, \epsilon)$ leads to the production $[q_1, X, q_1] \to \epsilon$.

The transition $\delta(q_1, \epsilon, Z_0) \vdash (q_1, \epsilon)$ leads to the production $[q_1, Z_0, q_1] \to \epsilon$.

Based on rule 4, the production where the transition involves pushing of a symbol unto the stack are:

The transition $\delta(q_0, 0, Z_0) \vdash (q_0, XZ_o)$ leads to the following productions:

- $[q_0, Z_0, q_0] \to 0[q_0, X, q_0][q_0, Z_0, q_0]$ (1a1)
- $[q_0, Z_0, q_0] \to 0[q_0, X, q_1][q_1, Z_0, q_0]$ (1a2)
- $[q_0, Z_0, q_1] \to 0[q_0, X, q_0][q_0, Z_0, q_1]$ (1a3)
- $[q_0, Z_0, q_1] \to 0[q_0, X, q_1][q_1, Z_0, q_1]$ (1a4)

The transition $\delta(q_0, 0, X) \vdash (q_0, XX)$ leads to the following productions:

- $[q_0, X, q_0] \to 0[q_0, X, q_0][q_0, X, q_0]$ (1b1)
- $[q_0, X, q_0] \to 0[q_0, X, q_1][q_1, X, q_0]$ (1b2)
- $[q_0, X, q_1] \to 0[q_0, X, q_0][q_0, X, q_1]$ (1b3)
- $[q_0, X, q_1] \to 0[q_0, X, q_1][q_1, X, q_1]$ (1b4)

Now, we can revisit the obtained production and remove which are not necessary:

The variables $[q_1, X, q_0]$ and $[q_1, Z_0, q_0]$ never appear as a LHS of productions, drop the respective production which contains them: 1a2, 1b2.

Similarly, $[q_0, X, q_0]$ and $[q_0, Z_0, q_0]$ always appear on the RHS of any of their own productions (or just on the RHS), so we can drop these productions: 1a1, 1a3, 1b1, 1b3 as well as the first production from the start symbol.

After removing the productions, the remaining are:

- $S \to [q_0, Z_0, q_1]$
- $[q_0, Z_0, q_1] \to 0[q_0, X, q_1][q_1, Z_0, q_1]$ (A)
- $[q_1, Z_0, q_1] \to \epsilon$. (B)
- $[q_0, X, q_1] \to 0[q_0, X, q_1][q_1, X, q_1]$ (C)
- $[q_0, X, q_1] \to 0$. (C)
- $[q_1, X, q_1] \to 1$. (D)
- $[q_1, X, q_1] \to \epsilon$. (D)

After renaming, the productions (P) for the obtained grammar G({S,A,B,C,D},{0,1},P,S) as follows:

- $S \to A$
- $A \to 0CB$
- $B \to \epsilon$
- $C \to 0CD|0$
- $D \to 1|\epsilon$

Problem 7

Convert the following PDA's to equivalent CFG.

1. $M = \{$
   - $\delta(q_0, a, Z_0) \vdash (q_0, aZ_0)$,
   - $\delta(q_1, a, a) \vdash (q_2, \epsilon)$,
   - $\delta(q_0, a, a) \vdash (q_0, aa)$,
   - $\delta(q_2, a, a) \vdash (q_2, \epsilon)$,
   - $\delta(q_0, c, a) \vdash (q_a, a)$,
   - $\delta(q_2, \epsilon, Z_0) \vdash (q_2, \epsilon)\}$
2. $M = \{$
   - $\delta(q_0, 1, Z_0) \vdash (q_0, KZ_0)$,
   - $\delta(q_0, 0, K) \vdash (q_1, K)$,
   - $\delta(q_0, \epsilon, Z_0) \vdash (q_0, \epsilon)$,
   - $\delta(q_0, 1, K) \vdash (q_0, KK)$,
   - $\delta(q_1, 0, K) \vdash (q_1, \epsilon)$,
   - $\delta(q_1, 0, Z_0) \vdash (q_0, Z_0)\}$

5 Decision Algorithms for CFL

The following are the fundamental questions that need to e answered about CFL's.

5.1 Deciding whether the CFL is non-empty or not

Let a CFG G(V,T,P,S) be corresponds to CFL L. L(G) is non empty if and only if that start symbol generates or derives some strings of terminals.

i.e., if S $\overset{*}{\Longrightarrow}$ w - then L(G) is non-empty otherwise L(G) is empty.

5.2 Deciding whether the CFL is finite or infinite

First bring the given CFG into CNF. Then draw a direct acyclic graph for the CFG in such a way that all the variables are nodes and

if there is a production $A \to BC$ then draw an edge for $A \to B \& A \to C$.

In the obtained DAG, if there is a cycle found, then the L(G) is infinite otherwise if there is no cycle then the L(G) is finite.

5.3 Deciding wheter a given string is in the CFL or not

For a given CFG G(V,T,P,S) and a string $w \in T^*$, is $w \in L(G)$?

To find it in a better way the scientist named Cocke-Younger-Kasami proposed an algorith also known as CYK algorithm. To apply the algorithm the grammar must be in CNF. It uses dynamic programming where it is based on the principle that the solution to problem [i, j] can be constructed from solution to sub-problem [i, k] and solution to sub problem [k, j]. Its time complexity is $O(n^3|w|)$

For that a triangular table where

- Each row of the table corresponds to one particular length of sub strings.
- Bottom most row corresponds to strings of length-1.
- Second row from bottom corresponds to strings of length-2.
- Third row from bottom corresponds to strings of length-3.
- Top most row from bottom corresponds to the given string of length-n.

Example: For a given string "x" of length four units, the triangular table looks like-

7cm

In the above table each cell of x_{ij} with it V_{ij} where x_{ij} represents the sub-string of x starting from location i and has length j.

Example: For the string x = abcd is a strin, there are n(n+1)/2 possible sub-strings. i.e., 4 x (4+1) / 2 = 10. They are:

x14 = abcd			
x13 = abc	x23 = bcd		
x12 = ab	x22 = bc	x32 = cd	
x11 = a	x21 = b	x31 = c	x41 = d

Table 2: CYK Table

and V_{ij} represents a set of variables in the grammar which can derive the sub string x_{ij}. If the set of variables consists of the start symbol, then it becomes sure-

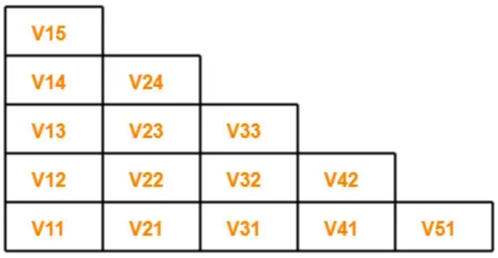
- Sub string xij can be derived from the given grammar.
- Sub string xij is a member of the language of the given grammar.

Problem 8

For the given grammar, check the acceptance of string w = baaba using CYK Algorithm. G=$\{S \to AB|BC, A \to BA|a, B \to CC|b, C \to AB|a\}$

Solution. First, let us draw the triangular table for the given string x = baaba. Since the length of given string = $|x|$ = 5, the possible sub strings = (5 x 6) / 2 = 15.

The triangular table looks like:


V15
V14
V24
V13
V23
V33
V12
V22
V32
V42
V11
V21
V31
V41
V51

Now fill the cells of the matrix row by row as follows: Filling of each row means finding the possibility its parent while constructing a parse tree from bottom-up (starting from leaves)

Filling first row

let $x_{11}, x_{12}, x_{13}, x_{14}, x_{15}$ = {b,a,a,b,a} be the sub-strings of length 1. (leaves of parse tree)

- For sub-string $x_{11} = b$ and $\because B \to b$ so, the possible parent(s) $V_{11} = \{B\}$
- For sub-string $x_{12} = a$ and $\because A \to a$ & $C \to a$ so, the possible parent(s) $V_{21} = \{A, C\}$
- For sub-string $x_{13} = a$ and $\because A \to a$ & $C \to a$ so, the possible parent(s) $V_{31} = \{A, C\}$
- For sub-string $x_{14} = b$ and $\because B \to b$ so, the possible parent(s)$V_{41} = \{B\}$
- For sub-string $x_{15} = a$ and $\because A \to a$ & $C \to a$ so, the possible parent(s)$V_{51} = \{A, C\}$

Filling Second row

Now the sub-strings of $x_{21}, x_{22}, x_{22}, x_{24}$ = {ba,aa,ab,ba}

- The sub-string $x_{21} = ba$ obtained from the variables V_{11}, V_{12}.

  V_{11}, V_{12}= $\{B\}.\{A, C\} = \{BA, BC\}$.

  From G the corresponding productions which derives $\{BA, BC\}$ are $A \to BA\&S \to BC$.

  So, the possible parent(s) $V_{12} = \{S, A\}$
- The sub-string $x_{22} = aa$ obtained from the variable V_{12}, V_{13}.

  $V_{12}, V_{13} = \{A, C\}.\{A, C\} = \{AA, AC, CC\}$.

  From G the corresponding productions which derives $\{AA, AC, CC\}$ are $B \to CC$.

  So, the possible parent(s) $V_{22} = \{b\}$

- For sub-string $x_{23} = ab$ obtained from the variable V_{13}, V_{14}.

  V_{13}, V_{14}= $\{A, C\}.\{B\} = \{AB, CB\}$.

  From G the corresponding productions which derives $\{AB, CB\}$ are

  $S \to AB \& C \to AB$.

  So, the possible parent(s) $V_{32} = \{S, C\}$

- For sub-string $x_{24} = ba$ obtained from the variable V_{14}, V_{15}.

  V_{14}, V_{15}= $\{B\}.\{A, C\} = \{BA, BC\}$.

  From G the corresponding productions which derives $\{BA, BC\}$ are $A \to BA \& S \to BC$.

  So, the possible parent(s) $V_{42} = \{S, A\}$

Filling Third row
Now the sub-strings of x_{31}, x_{32}, x_{33} = {baa,aab,aba}

- The sub-string $x_{31} = baa$ obtained from the different combinations of sub-strings such as {b,aa} and {ba ,a} and the corresponding variables are $\{V_{11}, V_{22}\}$ and $\{V_{21}, V_{13}\}$

  $\{V_{11}, V_{22}\} = \{B\}.\{B\} = \{BB\}$ and $\{V_{13}, V_{21}\} = \{S, A\}.\{A, C\} = \{sA, SC, AA, AC\}$

  From G the corresponding productions which derives $\{BB\} \& \{sA, SC, AA, AC\}$ are {}

  So, the possible parent(s) $V_{31} = \{\Phi\}$

- The sub-string $x_{32} = aab$ obtained from the different combinations of sub-strings such as {a,ab} and {aa ,b} and the corresponding variables are $\{V_{12}, V_{23}\}$ and $\{V_{14}, V_{22}\}$

  $\{V_{12}, V_{23}\} = \{A, C\}.\{S, C\} = \{AS, AC, CS, CC\}$ and $\{V_{14}, V_{22}\} = \{B\}.\{B\} = \{BB\}$

  From G the corresponding productions which derives $\{AS, AC, CS, CC\} \& \{BB\}$ are $\{B \to CC\}$

  So, the possible parent(s) $V_{32} = \{B\}$

- The sub-string $x_{33} = aba$ obtained from the different combinations of sub-strings such as {a,ba} and {ab,a} and the corresponding variables are $\{V_{13}, V_{24}\}$ and $\{V_{15}, V_{23}\}$

  $\{V_{13}, V_{24}\} = \{A, C\}.\{S, A\} = \{AS, AA, CS, CA\}$ and $\{V_{15}, V_{23}\} = \{A, C\}.\{S, C\} = \{AS, AC, CS, CC\}$

  From G the corresponding productions which derives $\{AS, AA, CS, CA\} \& \{AS, AC, CS, CC\}$ are $\{B \to CC\}$

  So, the possible parent(s) $V_{32} = \{B\}$

Filling fourth row
Now the sub-strings of x_{41}, x_{42} = {baab,aaba}

- The sub-string $x_{41} = baab$ obtained from the different combinations of sub-strings such as {b,aab}, {ba ,ab}, {baa,b} and the corresponding variables are $\{V_{11}, V_{32}\}$, $\{V_{21}, V_{23}\}$, $\{V_{31}, V_{14}\}$

  $\{V_{11}, V_{32}\} = \{B\}.\{B\} = \{BB\}$

  $\{V_{21}, V_{23}\} = \{S, A\}.\{S, C\} = \{sS, SC, AS, AC\}$

  $\{V_{31}, V_{14}\} = \Phi.\{B\} = \Phi$

  From G the corresponding productions which derives $\{BB\}, \{sS, SC, AS, AC\}\&\Phi$ are {}

  So, the possible parent(s) $V_{41} = \{\Phi\}$

- The sub-string $x_{42} = aaba$ obtained from the different combinations of sub-strings such as {a,aba}, {aa ,ba}, {aab,a} and the corresponding variables are $\{V_{12}, V_{33}\}$, $\{V_{22}, V_{24}\}$, $\{V_{32}, V_{15}\}$

  $\{V_{12}, V_{33}\} = \{A, C\}.\{B\} = \{AB, CB\}$

  $\{V_{22}, V_{24}\} = \{B\}.\{S, A\} = \{BS, BA\}$

  $\{V_{32}, V_{15}\} = \{B\}.\{A, C\} = \{BA, BC\}$

  From G the corresponding productions which derives $\{AB, CB\}, \{BS, BA\}\&\{BA, BC\}$ are $\{S \to AB, A \to BA, C \to AB, S \to BC\}$

  So, the possible parent(s) $V_{42} = \{S, A, C\}$

Filling fifth row
Now the sub-string of x_{51} = {baaba}

- The sub-string $x_{51} = baaba$ obtained from the different combinations of sub-strings such as {b,aaba}, {ba ,aba}, {baa,ba}, {baab,a} and the corresponding variables are $\{V_{11}, V_{42}\}$, $\{V_{21}, V_{33}\}$, $\{V_{31}, V_{24}\}$, $\{V_{41}, V_{15}\}$

  $\{V_{11}, V_{42}\} = \{B\}.\{S, A, C\} = \{BS, BA, BC\}$

  $\{V_{21}, V_{33}\} = \{S, A\}.\{B\} = \{sB, AB\}$

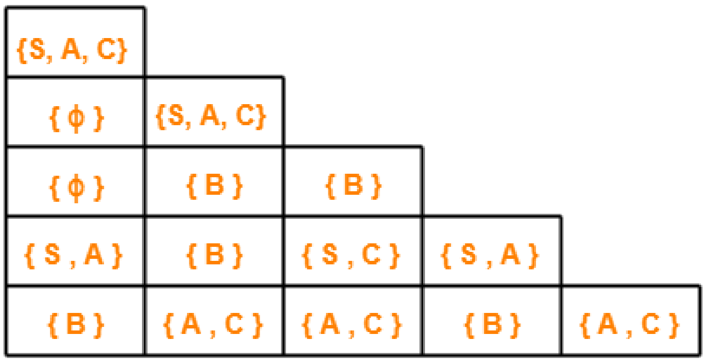
  $\{V_{31}, V_{24}\} = \Phi.\{S, A\} = \Phi$

  $\{V_{41}, V_{15}\} = \Phi.\{A, C\} = \Phi$

  From G the corresponding productions which derives $\{BS, BA, BC\}, \{sB, AB\}$ are $\{A \to BA, S \to BC, C \to AB\}$

  So, the possible parent(s) $V_{51} = \{S, A, C\}$

After filling the entries of triangular table :



{S, A, C}				
{ ϕ }	{S, A, C}			
{ ϕ }	{ B }	{ B }		
{ S , A }	{ B }	{ S , C }	{ S , A }	
{ B }	{ A , C }	{ A , C }	{ B }	{ A , C }

Since the root entry such as V_{51}={S,A,C} contains start symbol, the string x=baaba is belongs to the given CFL or can be derived from the grammar G.

Problem 9

Solve the following problems using CYK algorithm.

1. G1=$\{S \rightarrow XY, T \rightarrow ZT|a, X \rightarrow TY, Y \rightarrow YT|b, Z \rightarrow TZ|b\}$ with a string x=abab.
2. G2=$\{S \rightarrow AB|BB, A \rightarrow CC|AB|a, B|CA|b, C \rightarrow BA|AA|b\}$ with a string x=baaba.