CS 208 : Automata Theory and Logic DATE: 21st March, 2024

Lecture - 29

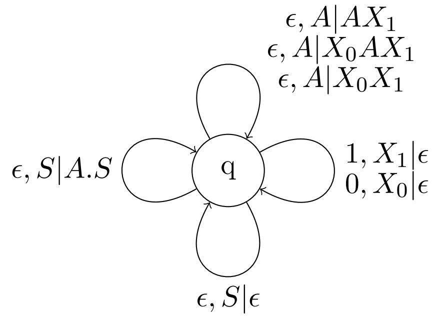
Topic: Chomsky Normal Form for CFG's

Scribed by: Karthikeya (22B1040)
Checked and compiled by: Lakshya

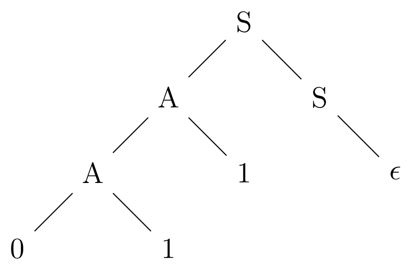
1 Drawing a graph representing a CFG

We can draw a push-down automata representing a language derived from a context-free grammar.
$\Sigma = \{0, 1\}$ - Alphabet
$\Gamma = \{S, A, X_0, X_1\}$ - Stack Symbols; X_0, X_1 are for terminals

Grammar:
$S \rightarrow A \cdot S | \epsilon$
$A \rightarrow A \cdot 1 | 0 \cdot A \cdot 1 | 01$

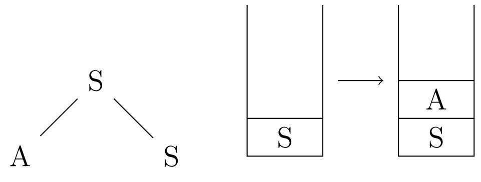


Let us take an example of a string in this language - 011. The parse tree for this string is



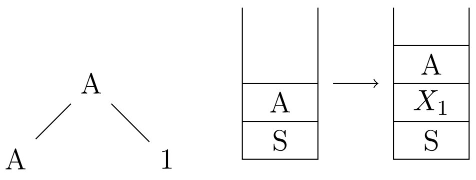
Steps:

- S $\rightarrow$ A$\cdot$ S. This transition is represented by the edge $\epsilon, S|A.S$ (We are not reading any alphabet, popping S and pushing A,S into the stack).

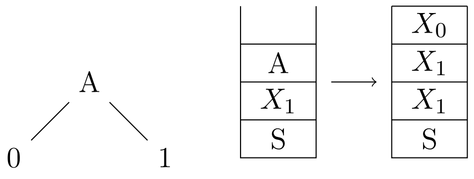


Note that here we are pushing S before A because we are wishing to get the left part of the string (A) first.
After getting the string A in this traversal, A will pop off and then we will continue to find the string S. How will A pop ?? Read further to know.

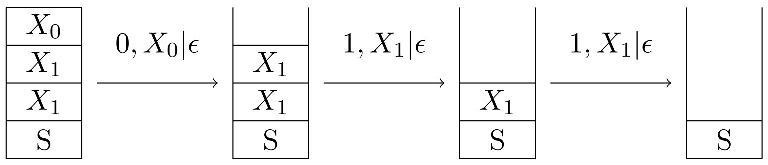
- A → A·1. This transition is represented by the edge $\epsilon, A|AX_1$.



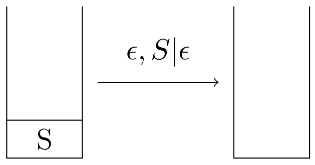
- A → 0·1. This transition is represented by the edge $\epsilon, A|X_0X_1$.



- Now, we will start to read the letters in the string and clear all the top X_0, X_1's in the stack. The transitions are represented by edges as $0, X_0|\epsilon$, $1, X_1|\epsilon$.



- S → ϵ. This tyransition is represented by the edge $\epsilon, S|\epsilon$.



2 Converting a Grammar into Chomsky Normal Form (CNF)

Take an example CFG:

$$S \to A \cdot B \cdot S|0A1$$
$$\text{A} \to 01|1A0|\epsilon$$
$$\text{B} \to 1B|BB0$$
$$\text{C} \to A0|01$$

Let us take S as the start symbol. We need the language associated with the symbol S.

1. For the language associated with the symbol S, we don't need the symbol C. So, we can delete the last rule of the grammar ($C \to A0|01$).

$$S \rightarrow A \cdot B \cdot S | 0A1$$
$$\text{A} \rightarrow 01|1A0|\epsilon$$
$$\text{B} \rightarrow 1B|BB0$$

We can find out this type of useless non-terminals by just drawing a graph for the non-terminals, joining the vertices if a non-terminal is used to derive another one.

$$\begin{tikzcd}
& A \arrow[loop above] & \\
S \arrow[loop left] \arrow[ur] \arrow[dr] & & C \arrow[ul] \\
& B \arrow[loop right] &
\end{tikzcd}$$

2. If we encounter B in the parse tree somehow, we cannot return from B because we cannot derive any terminal from B (B is derived only from B and does not have a derivation of a terminal). So, B and all the rules containing B can be removed.

$$S \rightarrow 0A1$$
$$\text{A} \rightarrow 01|1A0|\epsilon$$

The symbols which are unreachable from the start and which can't derive any terminating strings are useless and can be removed.
Non-terminals which can derive terminals or which can derive other non-terminal which indeed derive terminals are useful symbols.

3. We can eliminate the ϵ- production rule by explicitly stating another rule by using that ϵ.
Eg: If the grammar is

$$S \rightarrow 0A1$$
$$\text{A} \rightarrow 01|1A0|\epsilon$$

As ϵ is derived from A, wherever A appears in the right hand side, replace it with ϵ and write another rule, like this:

$$S \rightarrow 0A1|01$$
$$\text{A} \rightarrow 01|1A0|10$$

4. If there is a single non-terminal in the right side of a rule, we can simply add rules for that non-terminal replaced by it's left side non-terminal.
Eg: If the grammar is

$$S \rightarrow 0A1|01$$
$$\text{A} \rightarrow 01|1A0|10|D$$
$$\text{D} \rightarrow S|0AD$$

Here, each occurrence of D can be replaced by S as a new rule and the rule $D \rightarrow S$ can be eliminated. This type of production rules where a single non-terminal is there on the right side are called *unit productions*.

$$S \rightarrow 0A1|01$$
$$\text{A} \rightarrow 01|1A0|10|S|D$$
$$\text{D} \rightarrow 0AD|0AS$$

Here, again there are two unit productions $A \to S$, $A \to D$ which can be eliminated. Add new rules by replacing each occurrence of A with S and D and make new rules, delete $A \to S$ and $A \to D$.

$$S \to 0A1|01|0S1|0D1$$
$$\text{A} \to 01|1A0|10|1S0|1D0$$
$$\text{D} \to 0AD|0AS|0SD|0DD|0DS$$

5. If we have any rules that have more than two symbols in the right side, the we can add some other rules and symbols such that there are less than or equal to two symbols in the right hand side.
   Eg: If there is a rule like this: $S \to X_0SX_1$, then it can be converted to two rules:
   $S \to X_0S_1$, $S_1 \to SX_1$.

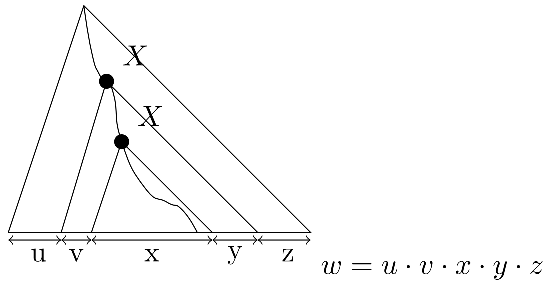
After applying all these steps to any Context-Free Grammar, we will get a grammar in which:

- There are no useless symbols.
- There is no rules which produce ϵ.
- There are no unit production rule.
- All rules are either of the form $A \to BC$ (Non-terminal to two non terminals), or the form $D \to X_0$ (Non-terminal to a terminal).

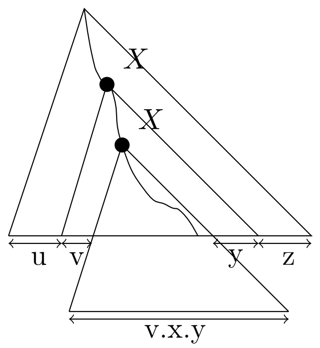
This form of grammar is called **Chomsky Normal Form** (CNF).

3 Pumping Lemma for CNF of CFGs

Let the CNF of CFG have n non-terminals. The derivation tree of any string in this grammar will be a binary tree (because each node will give rise to either two nodes (non-terminals) or a leaf (terminal)).
If there is a word w such that it's length $|w| > 2^n$, that means, there are 2^n leaves to the graph. Then the longest path from the root to a leaf must be $> n$. That means we will have a non-terminal repeated at least twice in this path. Let the repeated non-terminal be X. We can divide the word w into 5 parts as given in the parse tree:



As the string $v.x.y$ and x are derived from the same symbol X, we can replace x with $v.x.y$ and it also will belong to the language, and it's parse tree will be the same tree with the sub-tree from the lower X replaced by the sub-tree of the higher X.



$w' = u.v^2.x.y^2.z$

We can replace this x with v.x.y as many times as we want. So, every word of the form $u.v^i.x.y^i.z$ also belongs to the language described by the given grammar.
Formally:
For any context-free language L, there exists a constant p (the "pumping length") such that any string w in L of length at least p can be divided into five pieces w = u.v.x.y.z such that:

- $|v.x.y| \leq p$
- $v.x \geq 1$
- $\forall x \geq 0, u.v^i.x.y^i.z \in L$

Here, the value of p is 2^n, where n is the number of non-terminals in the grammar.
Eg:
Let us take a language $L = \{0^n1^n2^n | n \geq 0\}$, $\Sigma = \{0, 1, 2\}$.
Let us assume L is a regular language and let k = $2^{\#\text{non-terminals in the CNF grammar}}$.
Take an example of a string in the given language: $0^{2k}1^{2k}2^{2k}$. No matter how we break the string into 5 parts, we can always pump it so that the final string is not in the language. So, this language is not a context-free language.