Compiler Construction

Trees

Mr. Usman Wajid

usman.wajid@nu.edu.pk

National University
of Computer & Emerging Sciences

What is a Syntactic Tree?

Syntactic Tree

It represents the syntactic structure of a string according to some context-free grammar

What is a Syntactic Tree?

Syntactic Tree

It represents the syntactic structure of a string according to some context-free grammar

- Also known as Generation tree or Derivation tree

What is a Syntactic Tree?

Syntactic Tree

It represents the syntactic structure of a string according to some context-free grammar

- Also known as Generation tree or Derivation tree
- Consider the following CFG,
- $S \to AA$
  $A \to AAA \mid bA \mid Ab \mid a$
- The word ”baab” can be generated by the above CFG as,

What is a Syntactic Tree?

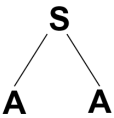
Syntactic Tree

It represents the syntactic structure of a string according to some context-free grammar

- Also known as Generation tree or Derivation tree
- Consider the following CFG,
- $S \rightarrow AA$
  $A \rightarrow AAA \mid bA \mid Ab \mid a$
- The word "baab" can be generated by the above CFG as,
- $S \rightarrow AA \rightarrow bAA \rightarrow bAAb \rightarrow baab$

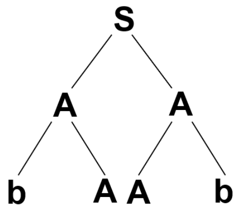
Syntactic Tree continued ...

- $S \rightarrow AA$
  $A \rightarrow AAA \mid bA \mid Ab \mid a$
- $S \rightarrow AA \rightarrow BAA \rightarrow BAAB \rightarrow baab$
- Drawing the downward lines from S to each character of this string as follows,



Syntactic Tree continued . . .

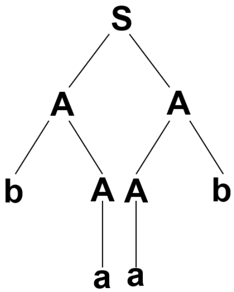
- $S \rightarrow AA$
  $A \rightarrow AAA \mid bA \mid Ab \mid a$
- $S \rightarrow AA \rightarrow BAA \rightarrow BAAB \rightarrow baab$
- Drawing the downward lines from S to each character of this string as follows,



Syntactic Tree continued . . .

- $S \rightarrow AA$
  $A \rightarrow AAA \mid bA \mid Ab \mid a$

- $S \rightarrow AA \rightarrow BAA \rightarrow BAAB \rightarrow baab$
- Drawing the downward lines from S to each character of this string as follows,



Total Language Tree

- For a given CFG, a tree with the start Symbol S as its root
- whose nodes are working strings of terminals and non-terminals
- The descendants of each node are all possible results of applying every production to the working string
- This tree is called **total language tree**

Total Language Tree example 1

- Consider the following CFG,

  $S \rightarrow aa \mid bX \mid aXX$
  $X \rightarrow ab \mid b$

Total Language Tree example 1

- Consider the following CFG,

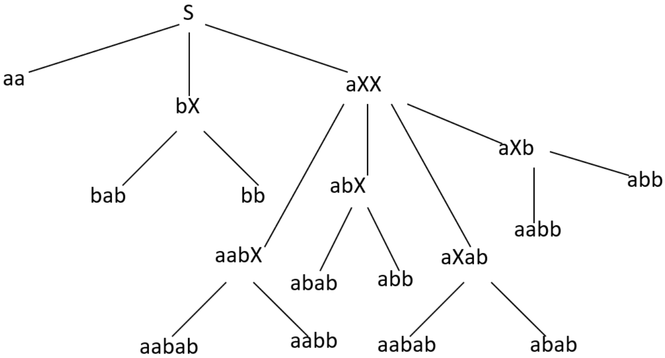
  $S \rightarrow aa \mid bX \mid aXX$
  $X \rightarrow ab \mid b$

- the the total language tree for the give CFG may be,

Total Language Tree example 1

- Consider the following CFG,

  $S \rightarrow aa \mid bX \mid aXX$
  $X \rightarrow ab \mid b$
- the the total language tree for the give CFG may be,

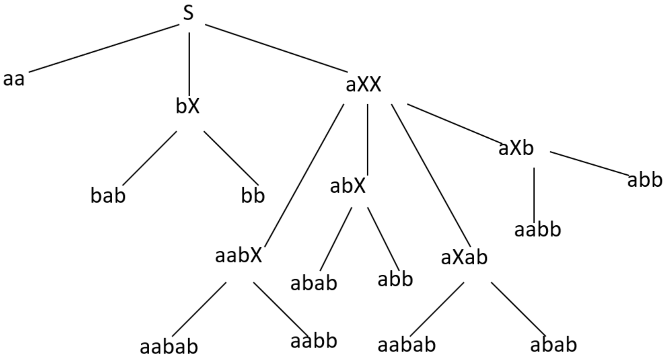


Total Language Tree example 1

- Consider the following CFG,

  $S \rightarrow aa \mid bX \mid aXX$
  $X \rightarrow ab \mid b$

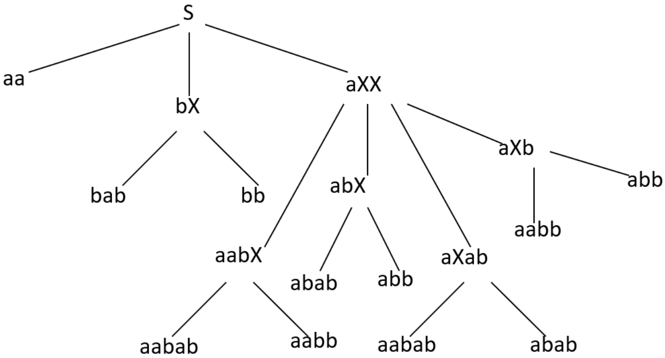
- the the total language tree for the give CFG may be,
- Ignoring the repetitive words, the total words generated by the above CFG is,



Total Language Tree example 1

- Consider the following CFG,

  S $\rightarrow$ aa | bX | aXX
  X $\rightarrow$ ab | b
- the the total language tree for the give CFG may be,
- Ignoring the repetitive words, the total words generated by the above CFG is,
- {aa, bab, bb, aabab, aabb, abab, abb}



Total Language Tree example 2

- Consider the following CFG,

  $S \rightarrow X \mid b$
  $X \rightarrow aX$

Total Language Tree example 2

- Consider the following CFG,

  $S \rightarrow X \mid b$
  $X \rightarrow aX$

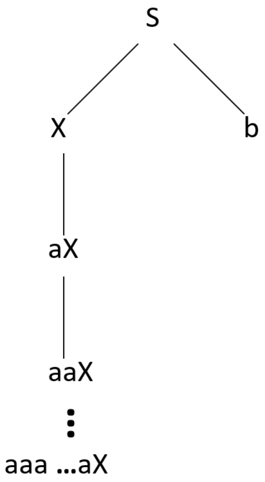
- then the total language tree for the give CFG is,

Total Language Tree example 2

- Consider the following CFG,

  $S \rightarrow X \mid b$
  $X \rightarrow aX$

- then the total language tree for the give CFG is,



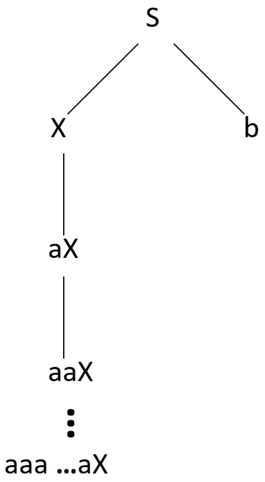
Total Language Tree example 2

- Consider the following CFG,

  $S \rightarrow X \mid b$
  $X \rightarrow aX$

- then the total language tree for the give CFG is,

- It is to be noted that the only word generated by this language is, $\{b\}$



RE and CFG

- Any language described by an RE, can also be described by a CFG

RE and CFG

- Any language described by an RE, can also be described by a CFG
- It is rather difficult to write grammar directly

RE and CFG

- Any language described by an RE, can also be described by a CFG
- It is rather difficult to write grammar directly
- Finite Automaton (FA) can be converted into corresponding CFG

FA to CFG conversion

- To convert FA to a grammar, the following rules should be used,

FA to CFG conversion

- To convert FA to a grammar, the following rules should be used,
  - Each state 'i' of the FA creates a non-terminal symbol 'A'

FA to CFG conversion

- To convert FA to a grammar, the following rules should be used,
  - Each state ’i’ of the FA creates a non-terminal symbol ’A’
  - If a state ’i’ has a transaction to the state ’j’ on a symbol ’a’, i.e., δ(i,a) = j, then introduce a production rule of the following,
    $A_i \rightarrow aA_j$

FA to CFG conversion

- If state 'i' goes to 'j' on input 'ϵ', then introduce a production rule of the form $A_i \rightarrow A_j$

FA to CFG conversion

- If state 'i' goes to 'j' on input 'ϵ', then introduce a production rule of the form $A_i \rightarrow A_j$
- If state 'i' is an accepting state, then introduce a production rule of the form $A_i \rightarrow \epsilon$

FA to CFG conversion

- If state 'i' goes to 'j' on input 'ϵ', then introduce a production rule of the form $A_i \rightarrow A_j$
- If state 'i' is an accepting state, then introduce a production rule of the form $A_i \rightarrow \epsilon$
- If state 'i' is the start state, then A_i is the start symbol (non-terminal) of the grammar.

RE to CFG: Example 1

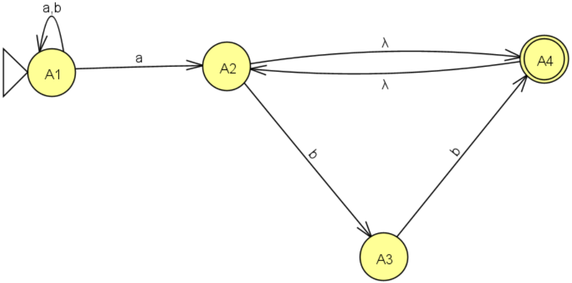
- Regular expression for the FA is $(a|b) * a(bb)*$

RE to CFG: Example 1

- Regular expression for the FA is $(a|b) * a(bb)*$
- Its corresponding grammar will be,
  Σ = {a, b}
  N = $\{A_1, A_2, A_3, A_4\}$
  S = $\{A_1\}$

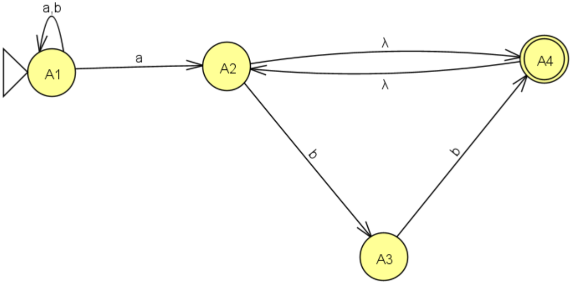
RE to CFG: Example 1

- Regular expression for the FA is $(a|b) * a(bb)*$
- Its corresponding grammar will be,
  Σ = {a, b}
  N = $\{A_1, A_2, A_3, A_4\}$
  S = $\{A_1\}$
- Its corresponding NFA will be,



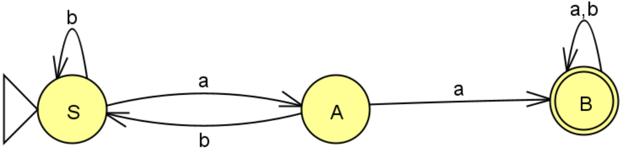
RE to CFG: Example 1

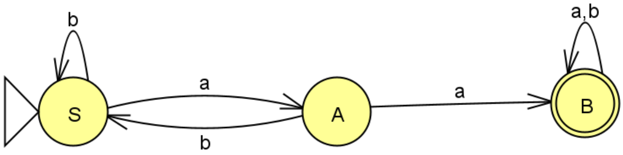
- Regular expression for the FA is $(a|b)*a(bb)*$
- Its corresponding grammar will be,
  Σ = {a, b}
  N = $\{A_1, A_2, A_3, A_4\}$
  S = $\{A_1\}$
- Its corresponding NFA will be,



$$A_1 \rightarrow aA_1$$
$$A_1 \rightarrow bA_1$$
$$A_1 \rightarrow aA_2$$
$$A_2 \rightarrow bA_3$$
$$A_2 \rightarrow A_4$$
$$A_3 \rightarrow bA_4$$
$$A_4 \rightarrow A_2$$
$$A_4 \rightarrow \epsilon$$

RE to CFG: Example 1





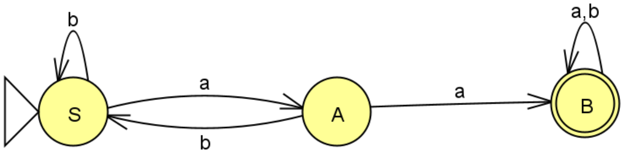
- The corresponding CFG may be,

$$S \rightarrow bS \mid aA$$

$$A \rightarrow aB \mid bS$$

$$B \rightarrow aB \mid bB \mid \epsilon$$

RE to CFG: Example 1



- The corresponding CFG may be,

  $S \rightarrow bS \mid aA$

  $A \rightarrow aB \mid bS$

  $B \rightarrow aB \mid bB \mid \epsilon$

- It may be noted that the number of terminals in the above CFG is equal to the number of states of corresponding FA
- where S corresponds to the initial state
- each transition defines a production

RE to CFG: Example 3

- Construct FA and grammar for the following RE,

$$(a|b)^* bbb(a|b)^*$$