Lecture 5

Syntax Analysis I

Introduction to Parsing

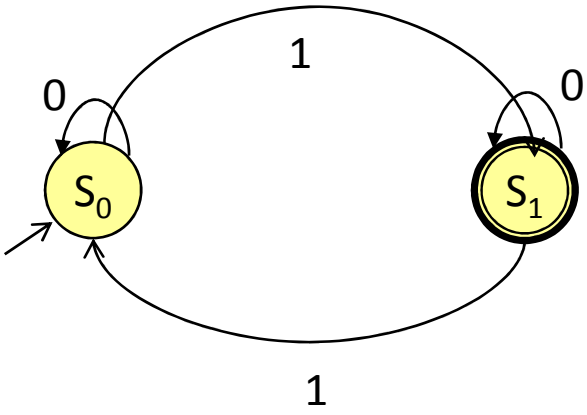
Regular Languages

- Weakest formal languages widely used
- Many applications
  - Lexical Analysis
- Some languages are not regular
- Consider the language

$$\{ (^{i})^{i} \mid i \geq 0 \}$$

Limitation of Regular Expression

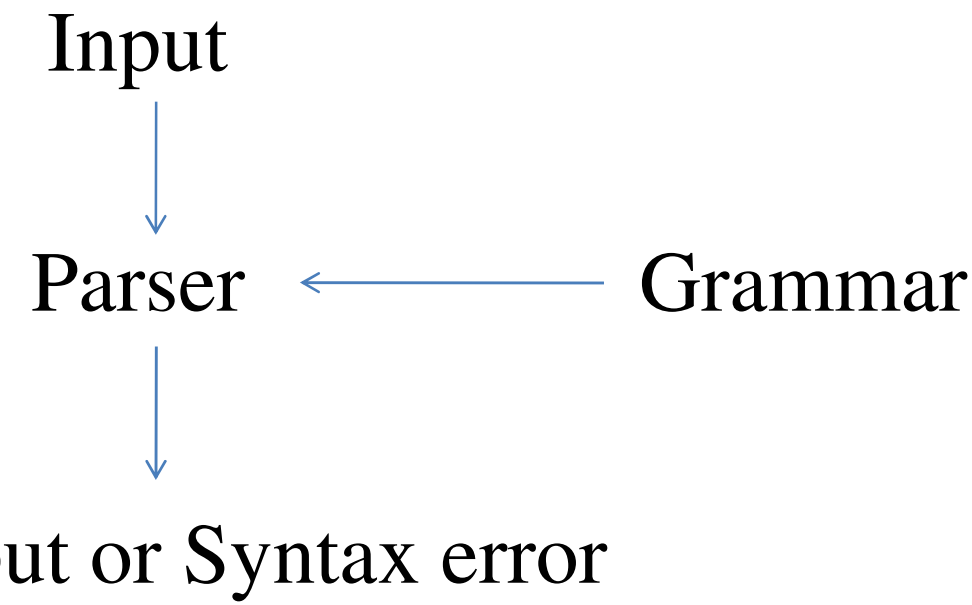
- RE can't express balance of the nested parenthesis
- Also can't express nested loop structure
- What can regular languages express?



- RE can count *mod k*
- But RE can't count a number

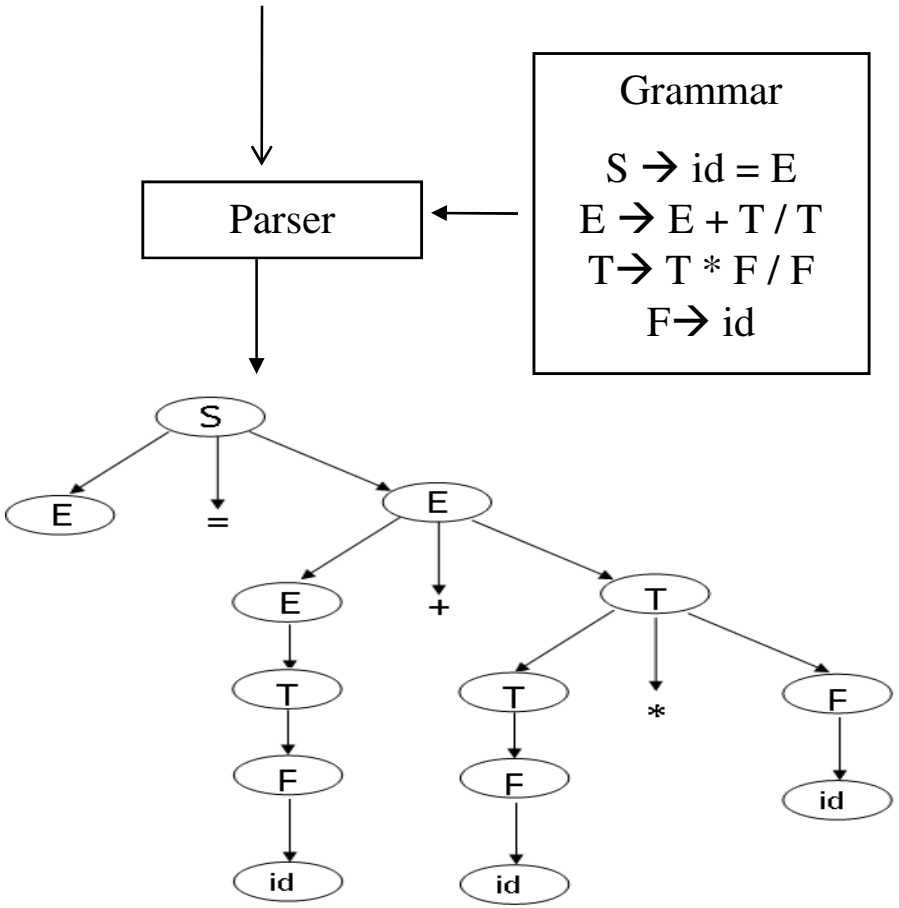
Introduction to Parser

- What the parser do?
  - Input to the parser: Sequence of tokens from lexer
  - Output of the parser: parse tree of the program



Introduction to Parser

Input: x = a + b * c → id = id + id * id



Not all strings of tokens are valid programs.

Parser must distinguishes between valid and invalid strings of tokens.

Requirements

- A language is needed for describing valid strings of tokens
- A method for distinguishing valid strings from the invalid strings
- Programming languages have natural recursive structure
  - *if (expr) expr else expr*

Context Free Grammar

- Formally a ***CFG G = (T, N, S, P),*** where:
  - ***T*** is the set of terminal symbols in the grammar (i.e., the set of tokens returned by the lexical analyzer)
  - ***N,*** the non-terminals, are variables that denote sets of (sub)strings occurring in the language. These impose a structure on the grammar.
  - **S** is the goal symbol, a distinguished non-terminal in N denoting the entire set of strings in L(G).
  - ***P*** is a finite set of productions specifying how terminals and non-terminals can be combined to form strings in the language. Each production must have a single non-terminal on its left hand side.

CFG – An Example

- Production : $X \rightarrow Y_1 \ldots \ldots Y_N$ where

  $X \in N$ *and* $Yi \in N \cup T \cup \{\varepsilon\}$

- The language for balanced parenthesis, i.e. $\{(^i)^i \mid i \geq 0\}$, CFG productions are

  $S \rightarrow (S) \mid S$

  $S \rightarrow \varepsilon$

  where set of non-Terminals (N) = {S}, set of Terminals (T)= {(,)} and S is the start symbol

Productions represents some rules

- Begin with a string with only the start symbol S

- Replace non terminal X in the string by the RHS of the some production
  - Let $X \rightarrow Y_1 \ldots \ldots Y_n$ and there is a string
  - $x_1 \ldots \ldots x_i\, X\, x_{i+1} \ldots \ldots x_N$; after using above production rule, it can be written as
  - $x_1 \ldots \ldots x_i\, Y_1 \ldots \ldots Y_n\, x_{i+1} \ldots \ldots x_N$

- If $\alpha_0 \rightarrow \alpha_1 \rightarrow \alpha_2 \rightarrow \ldots \ldots \alpha_n \rightarrow \alpha_0 \rightarrow \alpha_n$

CFG – An Example

- A possible CFG for arithmetic operations:
- **E → E + E | E * E | (E) | id**
- Input string: id * id + id

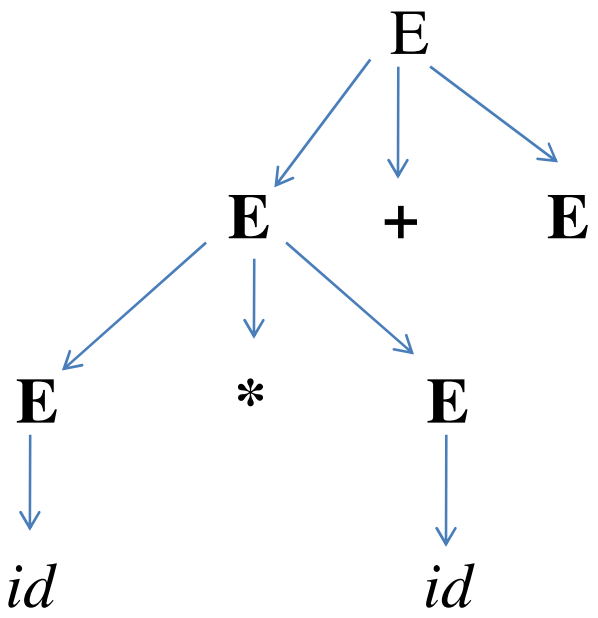
E → E + E

E * E + E

id * E + E

id * id + E

id * id + id



- *Left-Most Derivation*
- Inorder traversal of the leaves give the original input string
- All derivations of a string should yield the same parse tree

Thanks