Syntactic specification

Parse trees

Derivation and parse trees

- A graphical representation for the derivation can be created
  - This is called a parse tree
  - Each interior node of a parse tree is labeled by a nonterminal A
  - The children of the node A are also labeled from left to right by the symbols of the right hand side of the productions
  - The leaves of the trees are labeled by terminals (and nonterminals)
    - Ordered from left to right, they constitute a sentential form called a yield or frontier

Building a parse tree

- A->XYZ
- E->E+E|E*E|(E)-E|Id
- Consider the string –(id + id) and build the parse tree

Building a parse tree

- E->E+E|E*E|(E)-E|Id
- Consider the string **(id + id * id)**
- Derive the string using leftmost derivation
- Create a corresponding parse tree

Ambiguity

- A grammar that produce more than one parse tree for some sentence is said to be ambiguous
  - More than one leftmost derivation for some sentence
  - More than one rightmost for some sentence

Eliminating ambiguity

- An ambiguous grammar can be converted into an unambiguous one by
  - Introducing precedence and associativity

Precedence	level
- (unary minus)	
↑	
* /	
+ -	

Associativity

- The $\uparrow$ operator will be right associative
- i.e., a $\uparrow$ b $\uparrow$ c is computed as a $\uparrow$ (b $\uparrow$ c)
- The other binary operators will be left associative
- i.e., a – b – c is computed as (a-b) – c

Method to eliminate ambiguity

- Introduce one nonterminal for each precedence level
- A sub-expression that is essentially indivisible shall be called an element
- element- is either a single identifier or a parenthesized expression

  element->(expression)|id

Method to eliminate ambiguity cont..

- Primaries – are elements with zero or more of the operator of the highest precedence, the unary minus

  Primary-> - primary | element

- Factors- are sequences of one or more primaries connected by exponentiation signs

  factor->primary ↑ factor | primary

Method to eliminate ambiguity cont..

- terms - are sequences of one or more factors connected by multiplicative , * and /
- Expressions- are sequences of one or more terms connected by additive operators, + and -
- Term->term * factor |term / factor |factor
- Expression->expression + term | expression – term |term

Method to eliminate ambiguity cont..

expression->expression + term | expression – term |term

term->term * factor |term / factor |factor

factor->primary ↑ factor | primary

primary-> - primary | element

element->(expression)|id

Applying Operator precedence and associativity

Terminals={ id,+,-,*,/,↑,(,)}

Nonterminals={term,factor,primary,element,expression,multiop, addop}

Start symbol=primary

primary→ - primary | element

element → (expression)|id

expression-→ expression addop term |term

term → term multiop factor |factor

factor → primary ↑ factor | primary

multiop → * | /

Addop→ + | -

Derivation using non-ambiguous grammar

- Derive the string (id + id * id) using leftmost derivations