316 9-17-09

Ambiguity ⇒ More than one parse tree

· An unambiguous BNF for <E>

· Introduce precedence = * and / have higher precedence than + and −

· Introduce associativity = at the same precedence level, the operators associate to the right

<E> → <term> | <term> + <E> | <term> − <E>

<E> ⇒ <term> + <E> ⇒ <term> + <term> − <E> ⇒ <term> + <term> − <term> + <E>

⇒ ..... terminate by <E> → <term>

Any expression is a sequence of terms separated by +/−

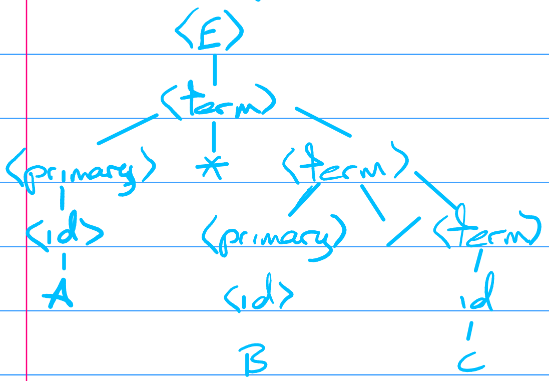
<term> → <primary> | <primary> * <term> | <primary> / <term>

<term> ⇒ <prim> * <term> ⇒ <prim> * <prim> / <term> ⇒ <prim> * <prim> / <prim> * <term> ⇒ .... terminated by <term> → <prim>

Any term is a sequence of primaries separated by */

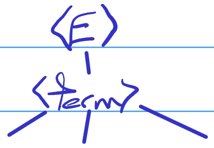
<primary> → <id> | <int> | <float> | "(" <E> ")"

construct the parse tree for A*B/C

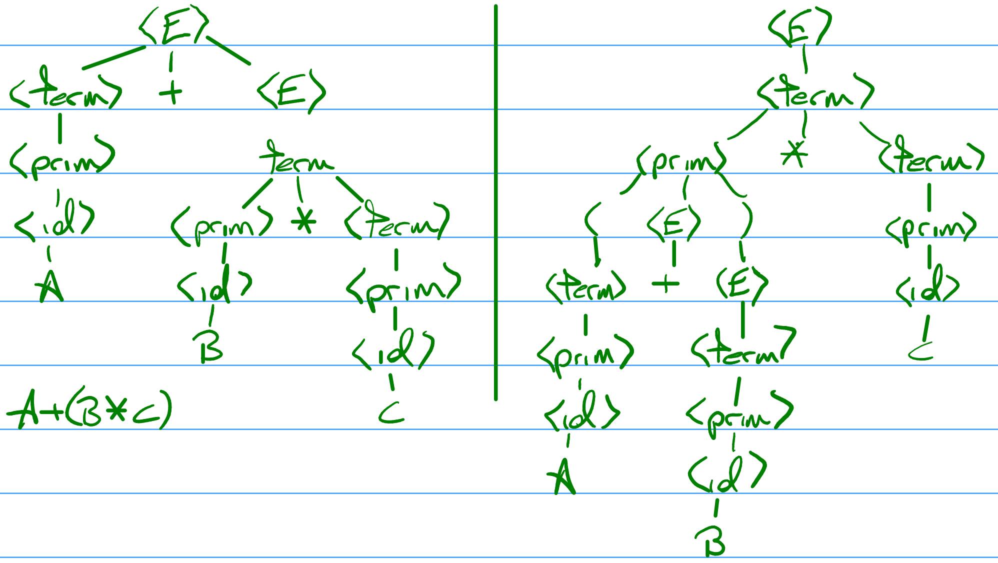


A*(B/C)

construct the parse tree for A+B+C



Construct the parse tree for A+B*C



To get left associating at the same procedure level, use left recursion:

<E> → <term> | <E> + <term> | <E> - <term>

<term> → <primary> | <term> * <primary> | <term> / <primary>

To give higher precedence to +/- over */ (although not used in math)

<E> → <term> | <term> * <term> / <E>

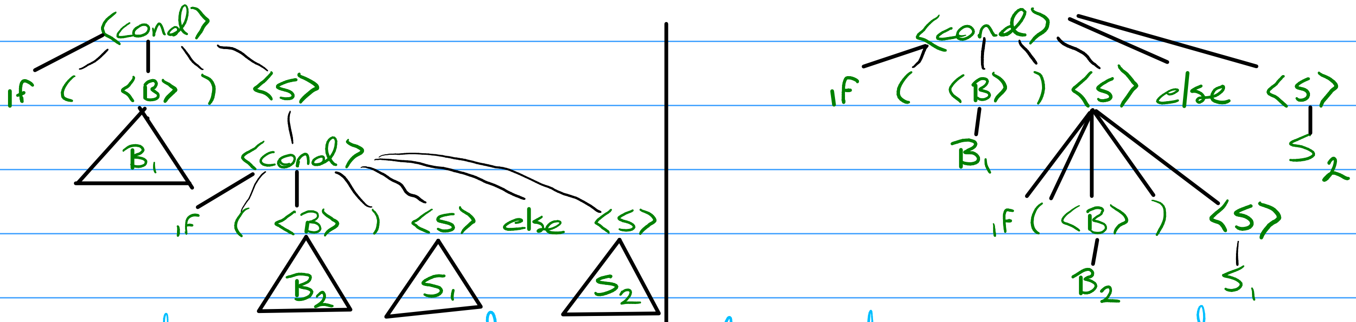
<term> → <prim> | <prim> + <term> | <prim> - <term>

<u>Conditionals</u>

<S> → .... | <cond> | ....

<cond> → if "(" <B> ")" <S> | if "(" <B> ")" <S> else <S>

construct 2 parse trees for if <B> if <B_2> S_1 else S_2



Intermediate code would compute

if(B_1){if (B_2) S_1 else S_2}

Intermediate code would compute

if (B_1) {if (B_2) S_1} else S_2

This is the default grouping in C++/Java, and almost all languages.

The following is an unambiguous BNF incorporating the rule:

each else matches with the closest preceding unmatched if

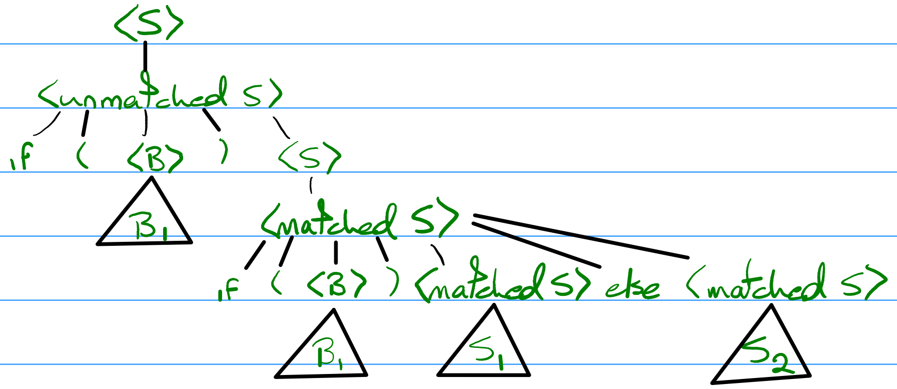
Introduce <matched S> for the statements containing no 1-branch conditional directly nested with other conditionals.

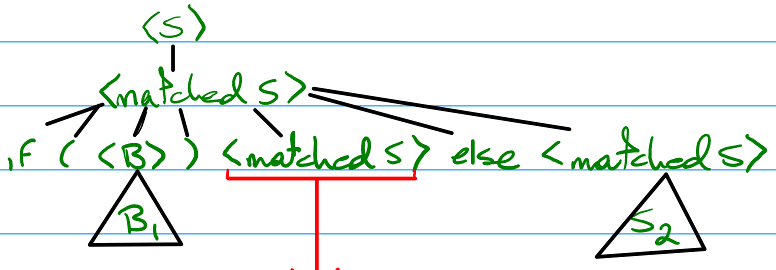
<S> → <matched S> | <unmatched S>

<matched S> → if "(" <B> ")" <matched S> else <matched S> | .... other kinds of non-conditional statements

(assume S_1 and S_2 are simple statements, like assignments)

<unmatched S> → if "(" <B> ")" <S> | if "(" <B> ")" <matched S> else <unmatched S>





can't generate if(B_2) S_1

This BNF is NOT used in practice.

- Not so easy to understand
- The parser cannot make the right choice without heavy look-ahead analysis.