CS 442/542

Parsing 1

Parsing

- Is the source program syntactically correct?
  - Given a sequence of symbols and a grammar find a derivation that produces the sequence of symbols
- Input: sequence of words or tokens recognized by the scanner
- Output: parse tree or syntax tree or some other representation of the source program
- In the main project for this course you will generate MIPS assembler language code during the parsing process

Parsing Terminology

- Context Free Grammar (CFG)
- Push Down Automata (PDA)
- Sentence
- Derivation
- Top down parsing
- Bottom up parsing

Context Free Grammar (CFG)

- A CFG is a 4 tuple (T, NT, S, P) where
  - T is set of terminals
  - NT is a set of nonterminals
  - S is one of the nonterminals called the goal or start symbol
  - P is a set of productions (also called rewriting rules) of the form NT -> (T U NT)*

Sentence and Derivation

- Sentence
  - A sequence of symbols that can be derived from the grammar
- Derivation
  - A sequence of rewriting rules that starts with the start symbol and ends with a sentence in the language
- Sentential form
  - A sequence of symbols that can occur in one step of a valid derivation
- Rightmost derivation
  - A derivation where each step rewrites the rightmost nonterminal
- Leftmost derivation
  - A derivation where each step rewrites the leftmost nonterminal

Example Grammar

- E -> (E) | E Op id | id
- Op -> + | - | * | /
- Rightmost derivation of (a+b) * c
  - E -> E Op id -> E * id -> (E) * id -> (E Op id) * id -> (E + id) * id -> (id + id) * id
- Leftmost derivation of (a + b) * c
  - E -> E Op id -> (E) Op id -> (E Op id) Op id -> (id Op id) Op id -> (id + id) Op id -> (id + id) * id

Parse Tree

- Also known as a concrete syntax tree
- Tree representation of the parsing process
- In a complete parse tree of a syntactically correct source program the leaves are the terminals of the grammar representing the symbols and syntactic categories of the source program

Example Parse Tree of (a+b)*c

$$\begin{tikzcd}
 & & & & & & & E \arrow[dll] \arrow[d] \arrow[drr] & & \\
 & & & & & E \arrow[dlll] \arrow[dll] \arrow[dr] & & Op \arrow[d] & & \langle id, c\rangle \\
 & & (& E \arrow[dll] \arrow[dr] \arrow[drrrrr] & & &) & * & & \\
 & E \arrow[dl] & & & Op \arrow[d] & & & & \langle id, b\rangle & \\
\langle id, a\rangle & & & & + & & & & &
\end{tikzcd}$$

Another Example Grammar (partial)

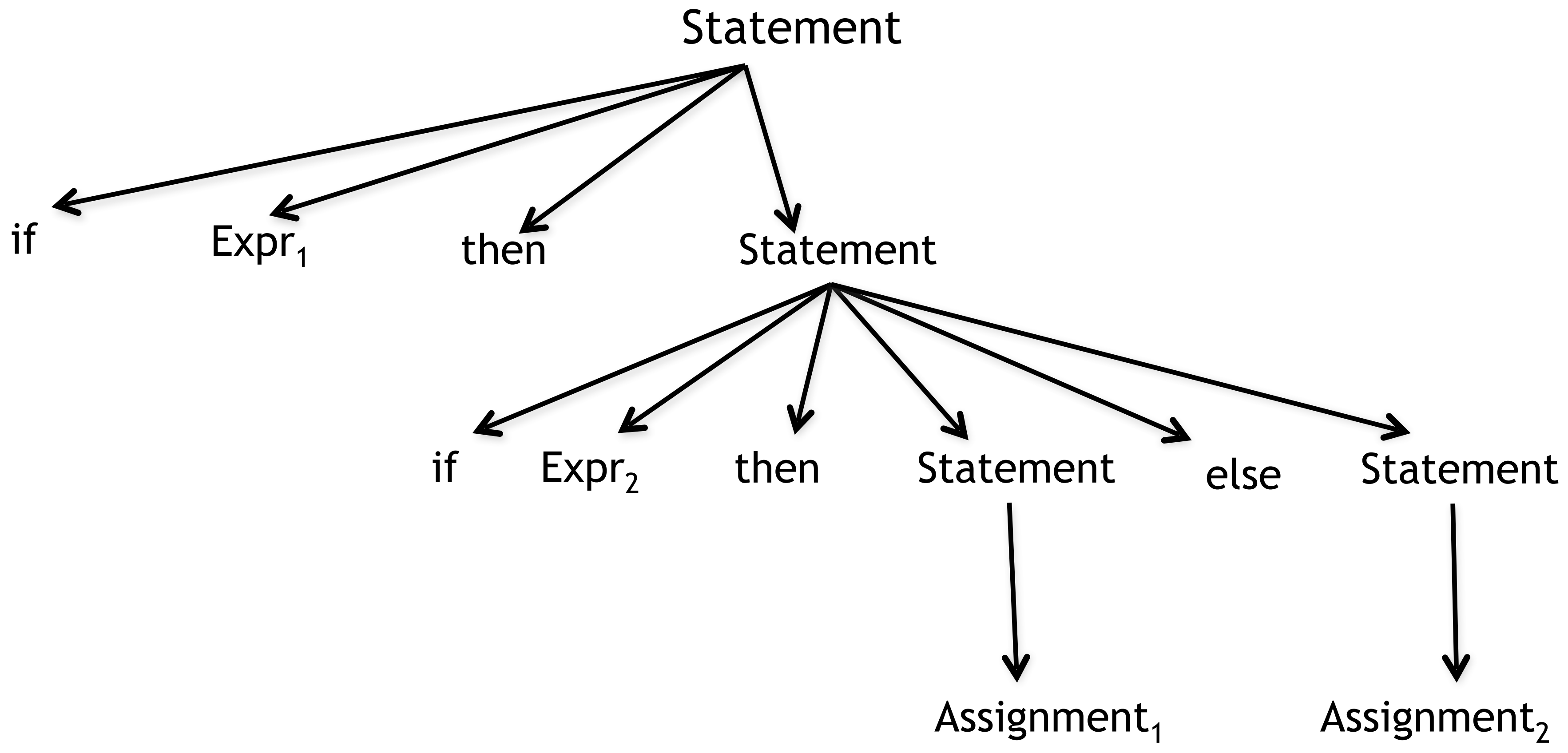
- Statement -> if Expr then Statement else Statement

  | if Expr then Statement

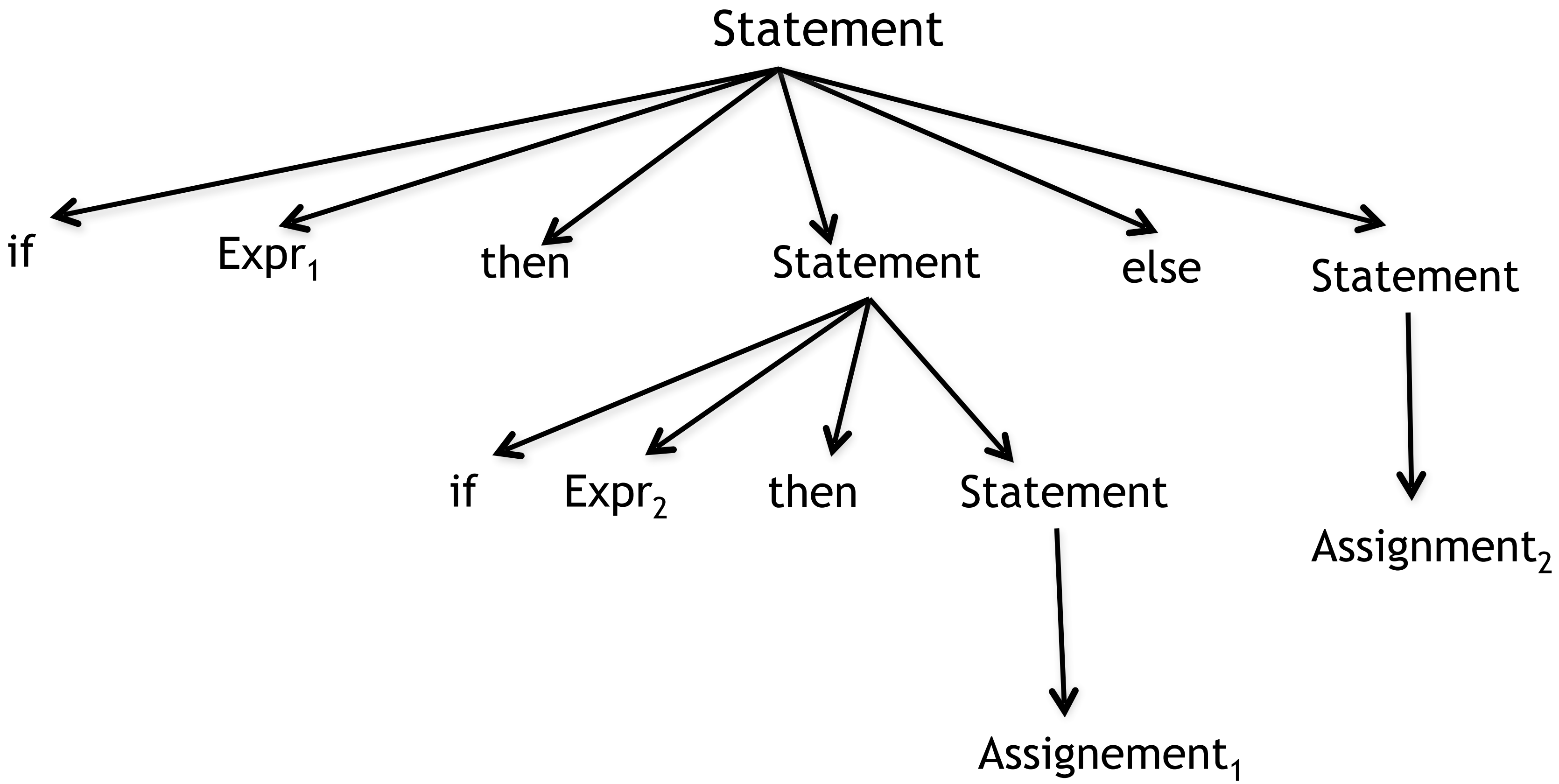
  | Assignment

  | …other statements…
- Ambiguous grammar
  - A grammar that has more that one rightmost (leftmost) derivation for the same sentence
  - A grammar that has more that one parse tree for the same sentence

Parse Tree for
if Expr_1 then if Expr_2 then Assignment_1 else Assignment_2



Another Parse Tree for if Expr$_1$ then if Expr$_2$ then Assignment$_1$ else Assignment$_2$



Top Down Parsing

- Build the parse tree from the root to the leaves
- Recursive descent parsing
- LL(1) grammar

Bottom up parsing

- Build the parse tree from the leaves to the root
- LR(1) grammar
  - SLR(1)
  - LALR(1)
    - LALR(1) parser generator
      - YACC
      - Bison