CFG2: Ambiguity

Context-Free Grammars

CMPT 379: Compilers

Instructor: Anoop Sarkar

anoopsarkar.github.io/compilers-class

Ambiguity

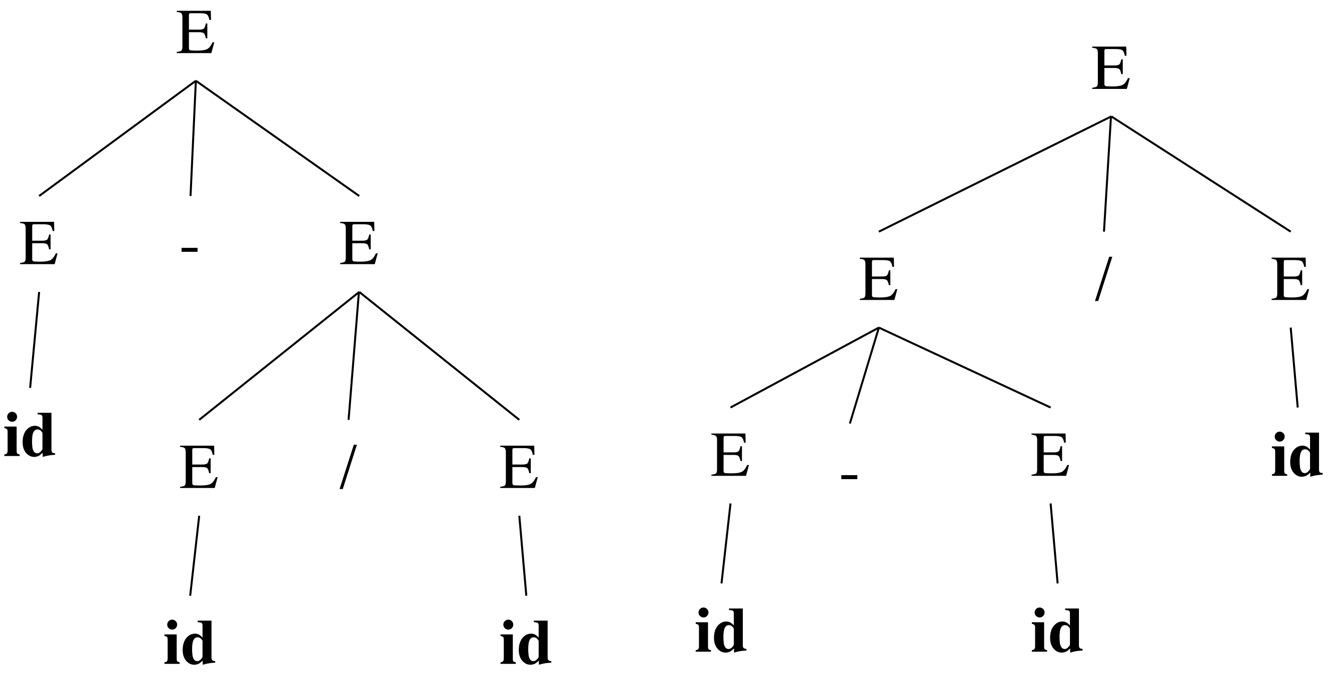
$E \rightarrow E - E$

$E \rightarrow E \,/\, E$

$E \rightarrow (E)$

$E \rightarrow \textbf{id}$

id - id / id



Ambiguity

- Grammar is ambiguous if more than one parse tree is possible for some sentences
  - There is more than one leftmost (or rightmost) derivations
- Ambiguity is not acceptable in programming languages
  - Leaves meaning of some programs ill-defined
  - Unfortunately, it's undecidable to check whether a given CFG is ambiguous
  - Some CFLs are inherently ambiguous (do not have an unambiguous CFG)

Ambiguity

- Handle ambiguity:
  - Rewrite the grammar unambiguously
  - Augment parser by enforcing precedence and associativity
- Consider the original ambiguous grammar:

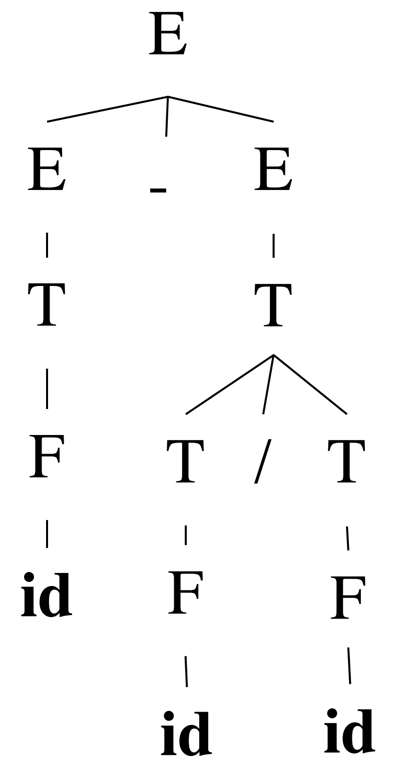
$E \rightarrow E - E \qquad E \rightarrow E \,/\, E$

$E \rightarrow (E) \qquad E \rightarrow \textbf{id}$

- How can we change the grammar to get only one tree for the input **id - id / id**

Precedence

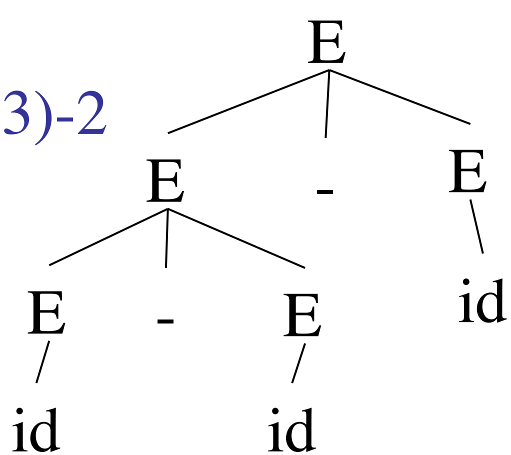
- Original ambiguous grammar:
  - $E \rightarrow E - E$ $\quad E \rightarrow E / E$
  - $E \rightarrow (E)$ $\quad E \rightarrow \textbf{id}$
- Use different non-terminals for each Precedence level: (start from lowest level)
  - $E \rightarrow E - E$ $\quad E \rightarrow T$
  - $T \rightarrow T / T$ $\quad T \rightarrow F$
  - $F \rightarrow \textbf{id}$ $\quad F \rightarrow (E)$
- Input: id - id / id



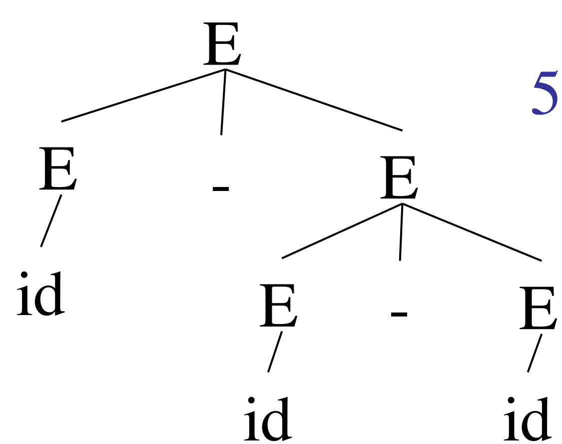
Associativity

- The grammar capture operator precedence
  - E → E - E   E → T
  - T → T / T   T → F
  - F → **id**   F → (E)
- Still ambiguous!! id - id – id   5-3-2
  - "-" is left associative (operations are grouped from left)

(5-3)-2



5-(3-2)



Recursion

- Grammar is recursive in nonterminal X if:
  - $X \Rightarrow^{+} \ldots\ X\ \ldots$
    - $\Rightarrow^{+}$ means in one or more steps, X derives a sequence of symbols that includes X
- Grammar is left recursive in X if:
  - $X \Rightarrow^{+} X\ \ldots$
    - In one or more steps, X derives a sequence of symbols that starts with X
- Grammar is right recursive in X if:
  - $X \Rightarrow^{+} \ldots X$
    - In one or more steps, X derives a sequence of symbols that ends with X

Fix Associativity

- Left and right recursive in non-terminals E and T
  - $E \rightarrow E - E$ $\quad\quad E \rightarrow T$
  - $T \rightarrow T / T$ $\quad\quad T \rightarrow F$
  - $F \rightarrow$ **id** $\quad\quad F \rightarrow (E)$
- Express operator associativity:
  - For left associativity use left recursion
  - For right associativity use right recursion
- Unambiguous grammar
  - $E \rightarrow E - T$ $\quad\quad E \rightarrow T$
  - $T \rightarrow T / F$ $\quad\quad T \rightarrow F$
  - $F \rightarrow$ **id** $\quad\quad F \rightarrow (E)$

Dangling else ambiguity

- Original Grammar (amibguous)

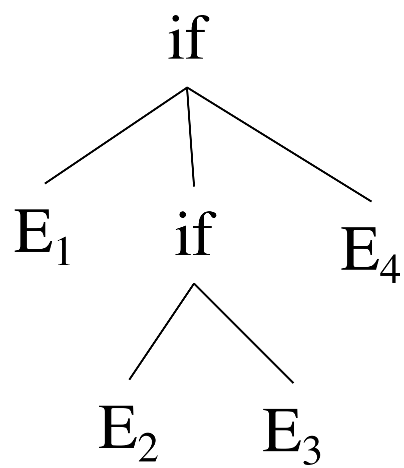
  Stmt $\rightarrow$ **if** Expr **then** Stmt **else** Stmt

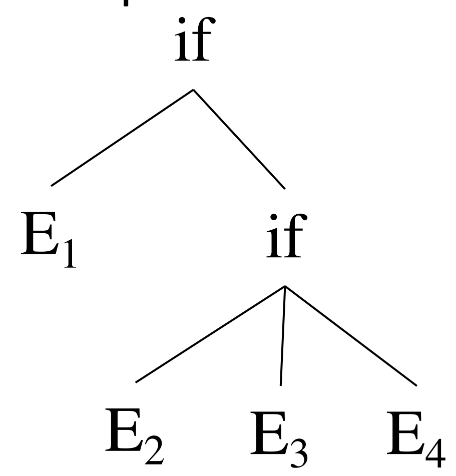
  Stmt $\rightarrow$ **if** Expr **then** Stmt

  Stmt $\rightarrow$ Other

- if E_1 then if E_2 then E_3 else E_4

else matches the closest unmatched then





Dangling else ambiguity

- Original Grammar (ambiguous)

  Stmt → **if** Expr **then** Stmt **else** Stmt

  Stmt → **if** Expr **then** Stmt

  Stmt → Other

else matches the closest unmatched then

- Unambiguous grammar

  Stmt → MatchedStmt /*all then are matched*/

  Stmt → UnmatchedStmt /*some then are unmatched*/

  MatchedStmt → **if** Expr **then** MatchedStmt **else** MatchedStmt

  MatchedStmt → Other

  UnmatchedStmt → **if** Expr **then** Stmt

  UnmatchedStmt → **if** Expr **then** MatchedStmt **else** UnmatchedStmt

Dangling else ambiguity

- Check unambiguous dangling-else grammar with the following inputs:
  - if Expr then if Expr then Other else Other
  - if Expr then if Expr then Other else Other else Other
  - if Expr then if Expr then Other else if Expr then Other else Other

Precedence and Associativity Declaration

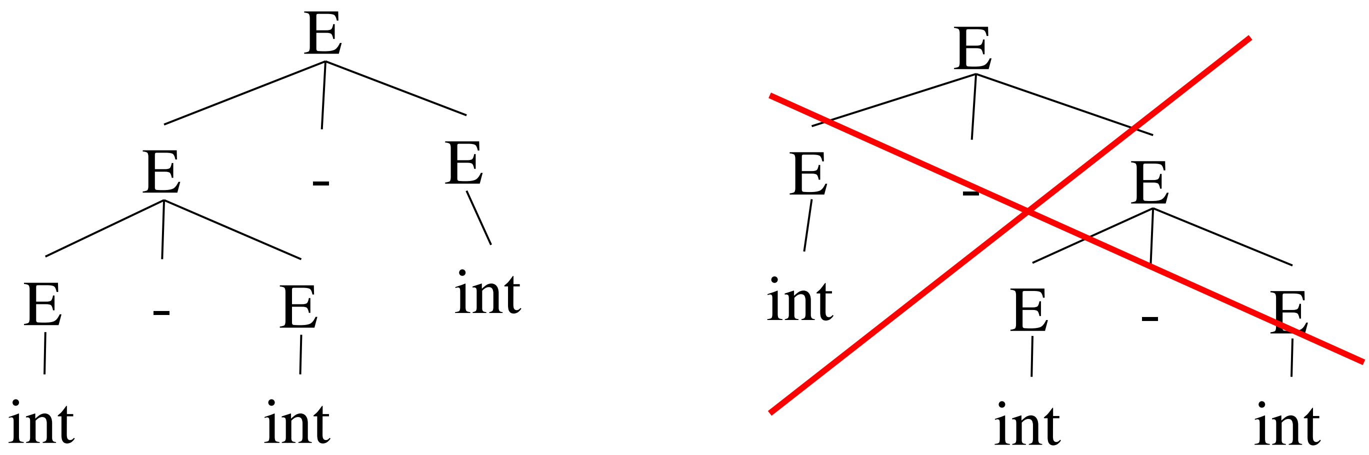
- Impossible to automatically convert an ambiguous grammar to an unambiguous one
- Used with care, ambiguity can simplify the grammar
  - Sometimes allow more natural definitions
  - We need disambiguation mechanisms

Precedence and Associativity Declaration

- Instead of re-writing the grammar
  - Use the more natural (ambiguous) grammar
  - Along with disambiguation declarations
- Most tools allow precedence and associativity declaration to disambiguate grammars

Associativity Declaration

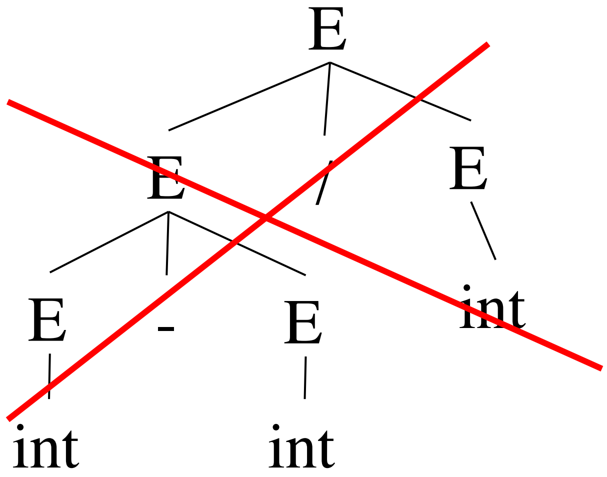
- Consider the grammar:
  - E → E - E | int
- Ambiguous: two parse trees int - int - int



- Left associativity declaration: %left -

Precedence Declaration

- Consider the grammar:
  - E → E - E | E / E | int
- Ambiguous: two parse trees int - int / int



- Precedence declaration: %left –

  %left /

Other Ambiguous Grammars

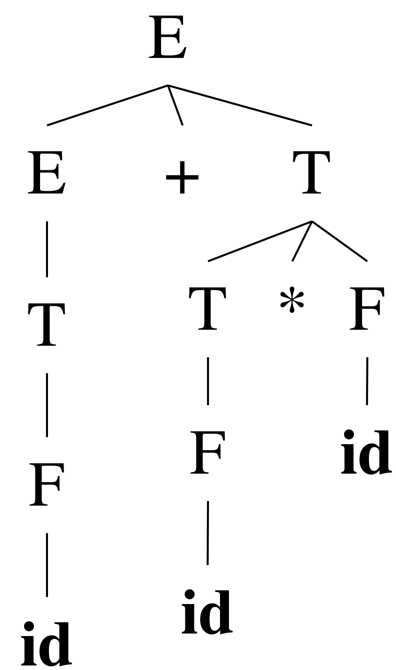
- Consider the grammar

$$
\begin{aligned}
R \rightarrow\ & R\ \text{‘|’}\ R \\
\mid\ & R\ R \\
\mid\ & R\ \text{‘*’} \\
\mid\ & \text{‘(’}\ R\ \text{‘)’} \\
\mid\ & a \\
\mid\ & b
\end{aligned}
$$

- What does this grammar generate?
- What is the parse tree for *a|b*a*
- Is this grammar ambiguous?

Ambiguity

- Original ambiguous grammar:
  - E → E + E E → E * E
  - E → (E) E → - E
  - E → id
- Unambiguous grammar:
  - E → E + T T → T * F
  - E → T T → F
  - F → (E) $\boxed{\text{F} \rightarrow \text{- E}}$
  - F → id
- Input: id + id * id



Warning! Is this unambiguous?
Check derivations for – id + id

Compare with F → - F

Dangling else ambiguity

- Original Grammar (ambiguous)

  Stmt → **if** Expr **then** Stmt **else** Stmt

  Stmt → **if** Expr **then** Stmt

  Stmt → Other

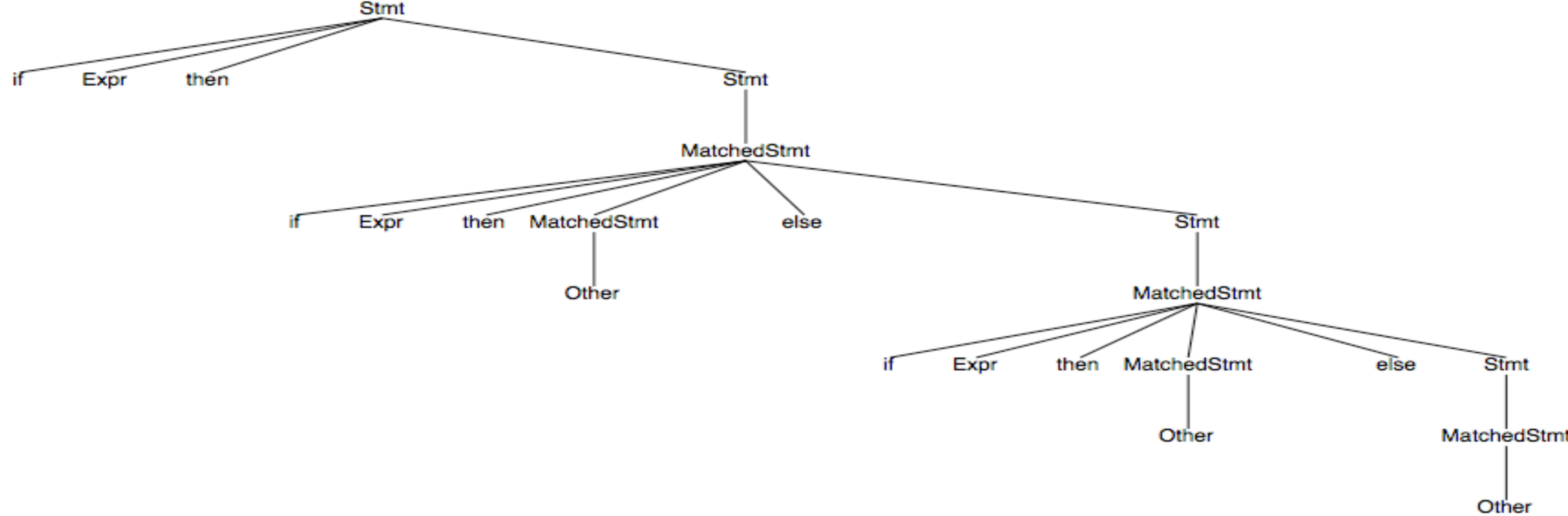
- Modified Grammar (unambiguous?)

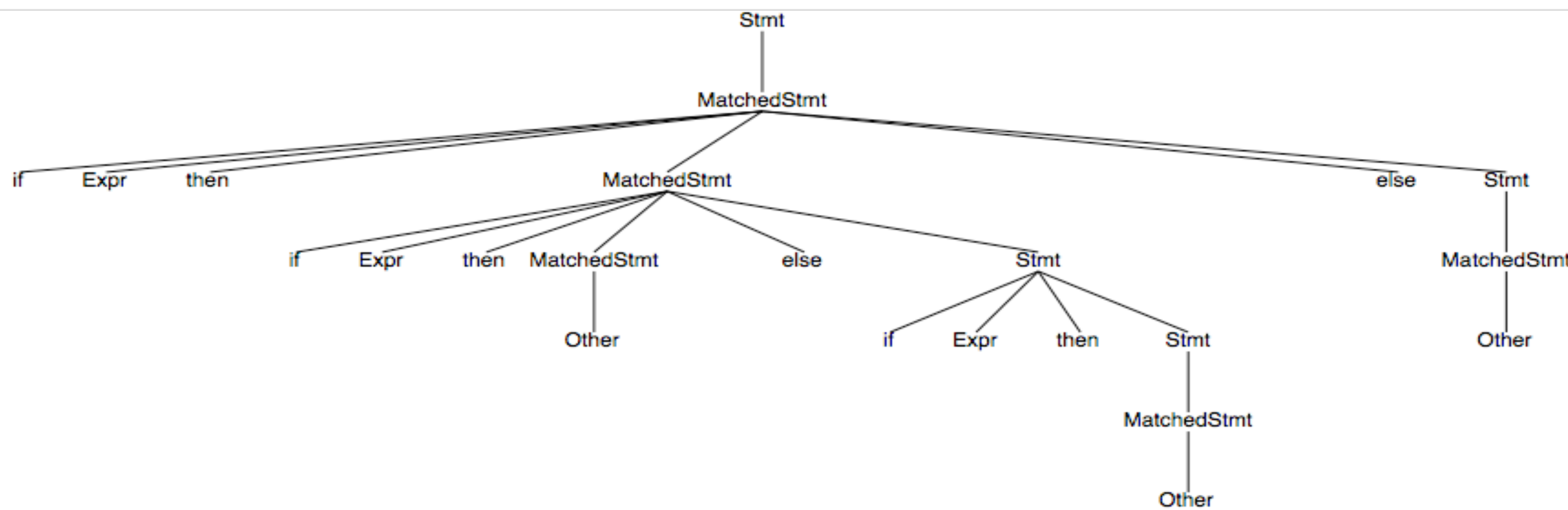
  Stmt → **if** Expr **then** Stmt

  Stmt → MatchedStmt

  MatchedStmt → **if** Expr **then** MatchedStmt **else** Stmt

  MatchedStmt → Other


Stmt
if
Expr
then
Stmt
MatchedStmt
if
Expr
then
MatchedStmt
else
Stmt
Other
MatchedStmt
if
Expr
then
MatchedStmt
else
Stmt
Other
MatchedStmt
Other


Stmt
MatchedStmt
if
Expr
then
MatchedStmt
else
Stmt
if
Expr
then
MatchedStmt
else
Stmt
MatchedStmt
Other
if
Expr
then
Stmt
Other
MatchedStmt
Other