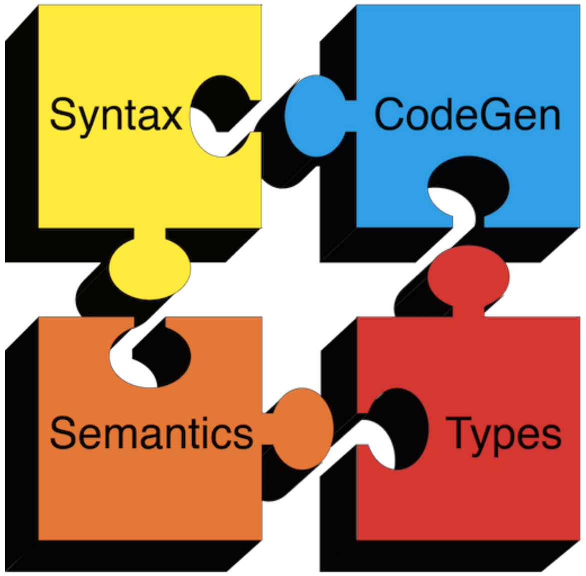


Compilers

Ambiguity

Alex Aiken

Ambiguity

- Grammar

$$E \rightarrow E + E \mid E * E \mid (E) \mid id$$

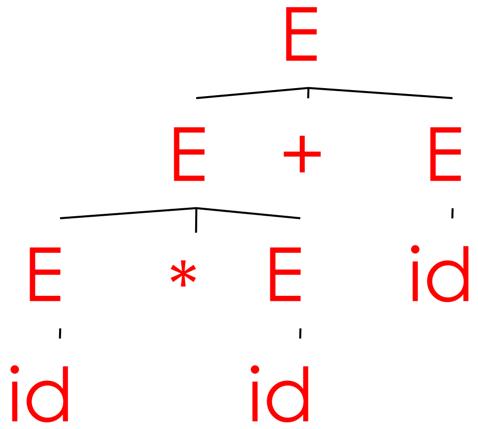
- String

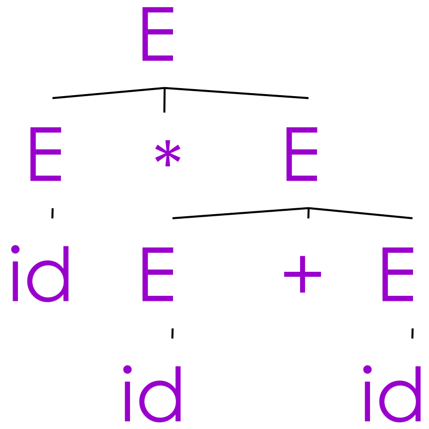
$$id * id + id$$

Alex Aiken

Ambiguity

This string has two parse trees





Ambiguity

- A grammar is *ambiguous* if it has more than one parse tree for some string
  - Equivalently, there is more than one right-most or left-most derivation for some string

- Ambiguity is **BAD**
  - Leaves meaning of some programs ill-defined

Ambiguity

Which of the following grammars are ambiguous?

- ☐ $S \rightarrow SS \mid a \mid b$
- ☐ $E \rightarrow E + E \mid id$
- ☐ $S \rightarrow Sa \mid Sb$
- ☐ $E \rightarrow E' \mid E' + E$

  $E' \rightarrow -E' \mid id \mid (E)$

Ambiguity

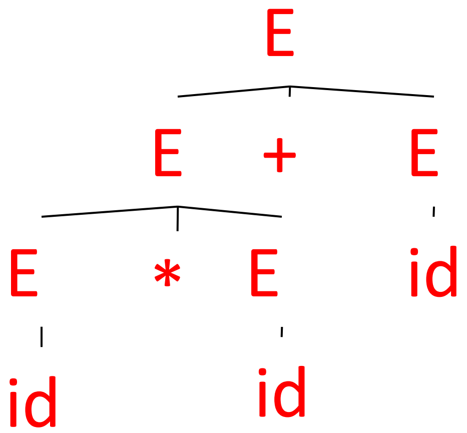
- There are several ways to handle ambiguity

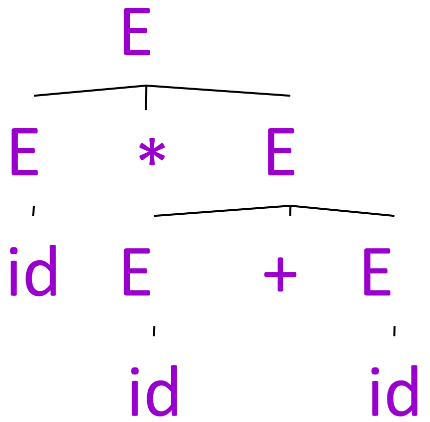
- Most direct method is to rewrite grammar unambiguously

$$E \rightarrow E' + E \mid E'$$
$$E' \rightarrow \text{id} * E' \mid \text{id} \mid (E) * E' \mid (E)$$

- Enforces precedence of $*$ over $+$

Ambiguity





Ambiguity

- Consider the grammar

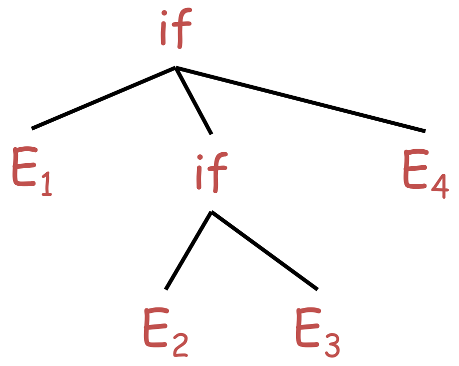
$$
\begin{aligned}
E \rightarrow\ & \text{if } E \text{ then } E \\
\mid\ & \text{if } E \text{ then } E \text{ else } E \\
\mid\ & \text{OTHER}
\end{aligned}
$$

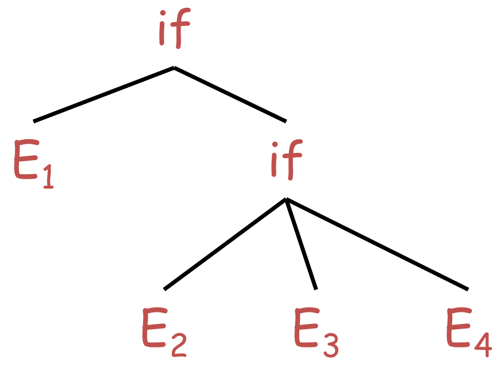
Ambiguity

- The expression

  if E_1 then if E_2 then E_3 else E_4

  has two parse trees





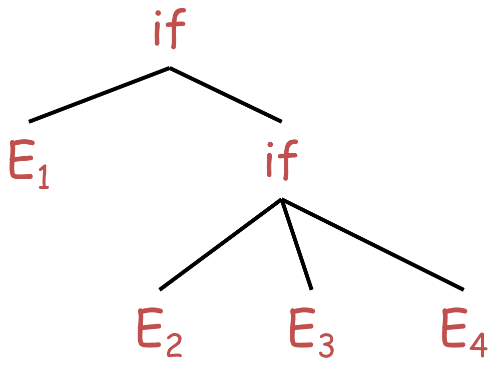
Ambiguity

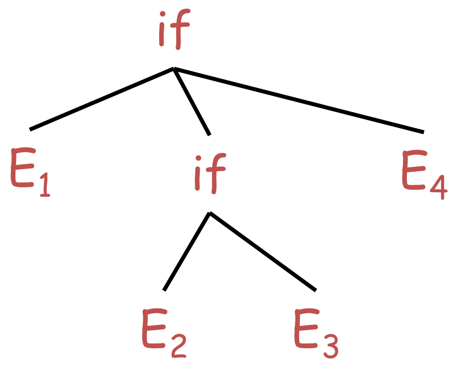
- else matches the closest unmatched then

```
E → MIF                    /* all then are matched */
  | UIF                    /* some then is unmatched */
MIF → if E then MIF else MIF
    | OTHER
UIF → if E then E
    | if E then MIF else UIF
```

Ambiguity

- The expression if E_1 then if E_2 then E_3 else E_4





Choose the unambiguous version of the given ambiguous grammar: $S \rightarrow SS \mid a \mid b$

○ $S \rightarrow Sa \mid Sb \mid \varepsilon$

○ $S \rightarrow SS'$
$S' \rightarrow a \mid b$

○ $S \rightarrow S \mid S'$
$S' \rightarrow a \mid b$

○ $S \rightarrow Sa \mid Sb$

Ambiguity

- Impossible to convert automatically an ambiguous grammar to an unambiguous one

- Used with care, ambiguity can simplify the grammar
  - Sometimes allows more natural definitions
  - We need disambiguation mechanisms

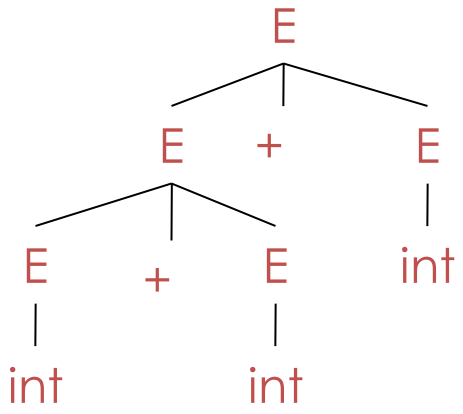
Ambiguity

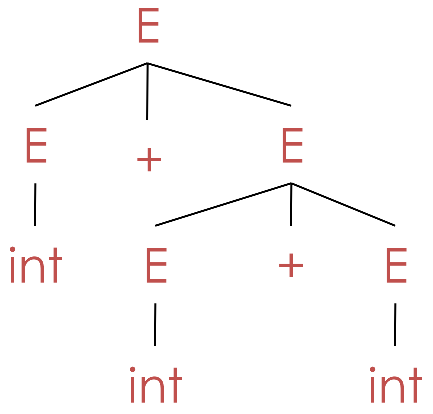
- Instead of rewriting the grammar
  - Use the more natural (ambiguous) grammar
  - Along with disambiguating declarations

- Most tools allow precedence and associativity declarations to disambiguate grammars

Ambiguity

- Consider the grammar $E \rightarrow E + E \mid \text{int}$
- Ambiguous: two parse trees of int + int + int

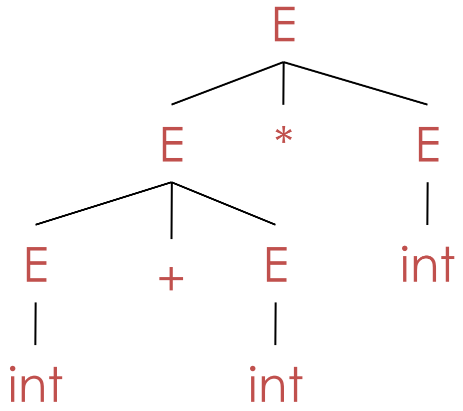


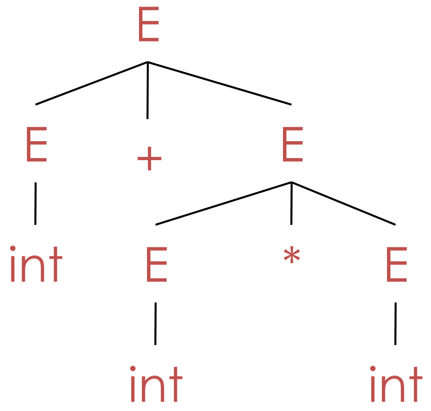


- Left associativity declaration: %left +

Ambiguity

- Consider the grammar $E \rightarrow E + E \mid E * E \mid \text{int}$
  - And the string int + int * int





- Precedence declarations: %left +

  %left *