LR5: Precedence and Associativity

LR Parsing

CMPT 379: Compilers

Instructor: Anoop Sarkar

anoopsarkar.github.io/compilers-class

S/R & ambiguous grammars

- Lx(k) Grammar vs. Language
  - Grammar is Lx(k) if it can be parsed by Lx(k) method – according to criteria that is specific to the method.
  - A Lx(k) grammar may or may not exist for a language.
- Even if a given grammar is not LR(k), shift/reduce parser can *sometimes* handle them by accounting for ambiguities
  - Example: ‘dangling’ else
    - Preferring shift to reduce means matching inner ‘if’

Dangling ‘else’

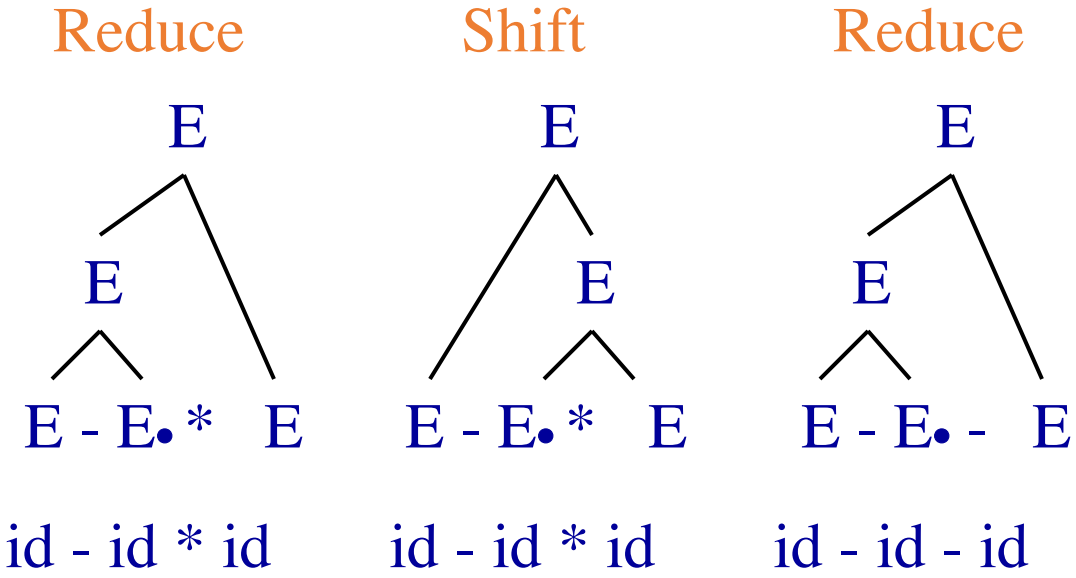
1. S $\rightarrow$ if E then S
2. S $\rightarrow$ if E then S else S

- Viable prefix “`if E then if E then S`”
  - Then read `else`
- Shift “`else`” (means eventually reduce using rule 2)
- Reduce (reduce using rule 1)
- Dangling else as written above is ambiguous
- Prefer shift over reduce resolves the ambiguity, but there’s no LR(k) grammar

Precedence & Associativity

- Consider

$$E \rightarrow E - E \mid E * E \mid id$$



Precedence Relations

- Let A $\rightarrow$ *w* be a rule in the grammar
- And *b* is a terminal
- In some state *q* of the LR(1) parser there is a shift-reduce conflict:
  - either reduce with A $\rightarrow$ *w* or shift on *b*
- Write down a rule, either:

  A $\rightarrow$ *w*, < *b* or A $\rightarrow$ *w*, > *b*

Precedence Relations

- A $\rightarrow$ *w*, < *b* means rule has less precedence and so we shift if we see *b* in the lookahead
- A $\rightarrow$ *w*, > *b* means rule has higher precedence and so we reduce if we see *b* in the lookahead
- If there are multiple terminals with shift-reduce conflicts, then we list them all:

  A $\rightarrow$ *w*, > *b*, < *c*, > *d*

Precedence Relations

- Consider the grammar

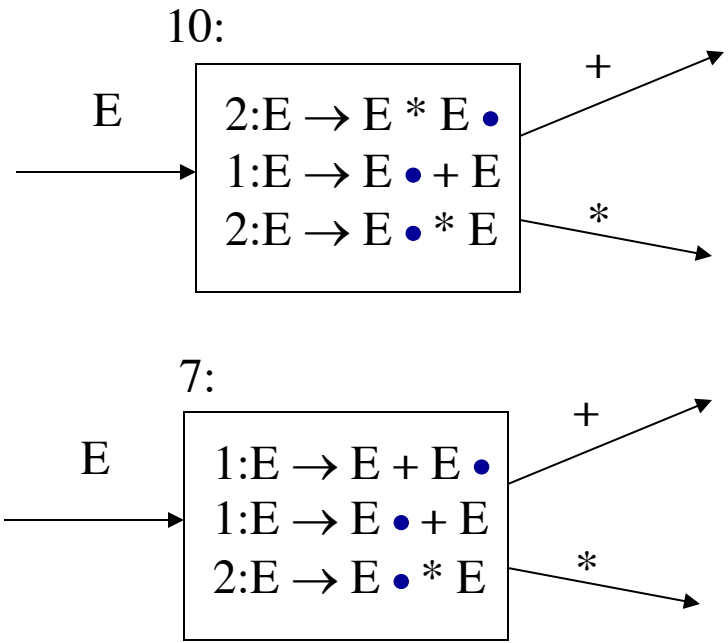
  $E \rightarrow E + E \mid E * E \mid (E) \mid a$
- Assume left-association so that E+E+E is interpreted as (E+E)+E
- Assume multiplication has higher precedence than addition
- Then we can write precedence rules/relns:

  $E \rightarrow E + E, > +, < *$

  $E \rightarrow E * E, > +, > *$

Precedence & Associativity

Implemented in yacc/bison using ordered list of `%left` or `%right` declarations



$E \rightarrow E + E, > +, < *$

$E \rightarrow E * E, > +, > *$

	+	*
7	R1	Shift
10	R2	R2

Parsing - Summary

- Top-down vs. bottom-up
- Lookahead: FIRST and FOLLOW sets
- LL(1) – Parsing: $O(n)$ time complexity
  - recursive-descent and table-driven predictive parsing
- LR(k) – Parsing : $O(n)$ time complexity
  - LR(0), SLR(1), LR(1), LALR(1)
- Resolving shift/reduce conflicts
  - using precedence, associativity