Parsing III

(Top-down parsing: recursive descent & *LL(1)*)

Roadmap (Where are we?)

Previously

We set out to study parsing

- Specifying syntax
  - → Context-free grammars
  - → Ambiguity
- Top-down parsers
  - → Algorithm & its problem with left recursion
  - → Left-recursion removal

Today

- Predictive top-down parsing
  - → The LL(1) condition
  - → Simple recursive descent parsers
  - → Table-driven LL(1) parsers

Picking the "Right" Production

If it picks the wrong production, a top-down parser may backtrack

Alternative is to look ahead in input & use context to pick correctly

How much lookahead is needed?

- In general, an arbitrarily large amount
- Use the Cocke-Younger, Kasami algorithm or Earley's algorithm

Fortunately,

- Large subclasses of CFGs can be parsed with limited lookahead
- Most programming language constructs fall in those subclasses

Among the interesting subclasses are *LL(1)* and *LR(1)*