Parsing — Part II

(Top-down parsing, left-recursion removal)

Parsing Techniques

Top-down parsers *(LL(1), recursive descent)*

- Start at the root of the parse tree and grow toward leaves
- Pick a production & try to match the input
- Bad "pick" ⇒ may need to backtrack
- Some grammars are backtrack-free *(predictive parsing)*

Bottom-up parsers *(LR(1), operator precedence)*

- Start at the leaves and grow toward root
- As input is consumed, encode possibilities in an internal state
- Start in a state valid for legal first tokens
- Bottom-up parsers handle a large class of grammars

Top-down Parsing

A top-down parser starts with the root of the parse tree

The root node is labeled with the goal symbol of the grammar

Top-down parsing algorithm:

Construct the root node of the parse tree

Repeat until the fringe of the parse tree matches the input string

1. *At a node labeled A, select a production with A on its lhs and, for each symbol on its rhs, construct the appropriate child*
2. *When a terminal symbol is added to the fringe and it doesn't match the fringe, backtrack*
3. *Find the next node to be expanded* *(label ∈ NT)*

- The key is picking the right production in step 1
  - → *That choice should be guided by the input string*