LL Parsing

input is read from left to right

trees are constructed from top down

<expr> ::= <n> |
after (<expr>) |
<var> |
let <var> = <expr> in <expr>

\+ intuitive

\- can't handle left-recursive grammar

1. "parser generator" — a tool for writing parsers
2. LR parser

LR Parser — handle left recursion! operates using a pushdown automaton

trees built from bottom up

input read from left to right

(gist)

two actions: 1. reduce — compact items on stack
2. shift — move input token onto stack

information: 1. stack
2. tokens
3. rules in a 2d grid