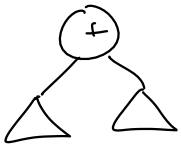
LL Parsing

- tree is constructed from the left
- input read left to right

→ we make decisions about how to build AST from top to the bottom

(...) + (....)



Today:

LR Parsing

→ tree is constructed "from the right" → we make decisions from the leaves up

---

LR parsing is based on pushdown automata

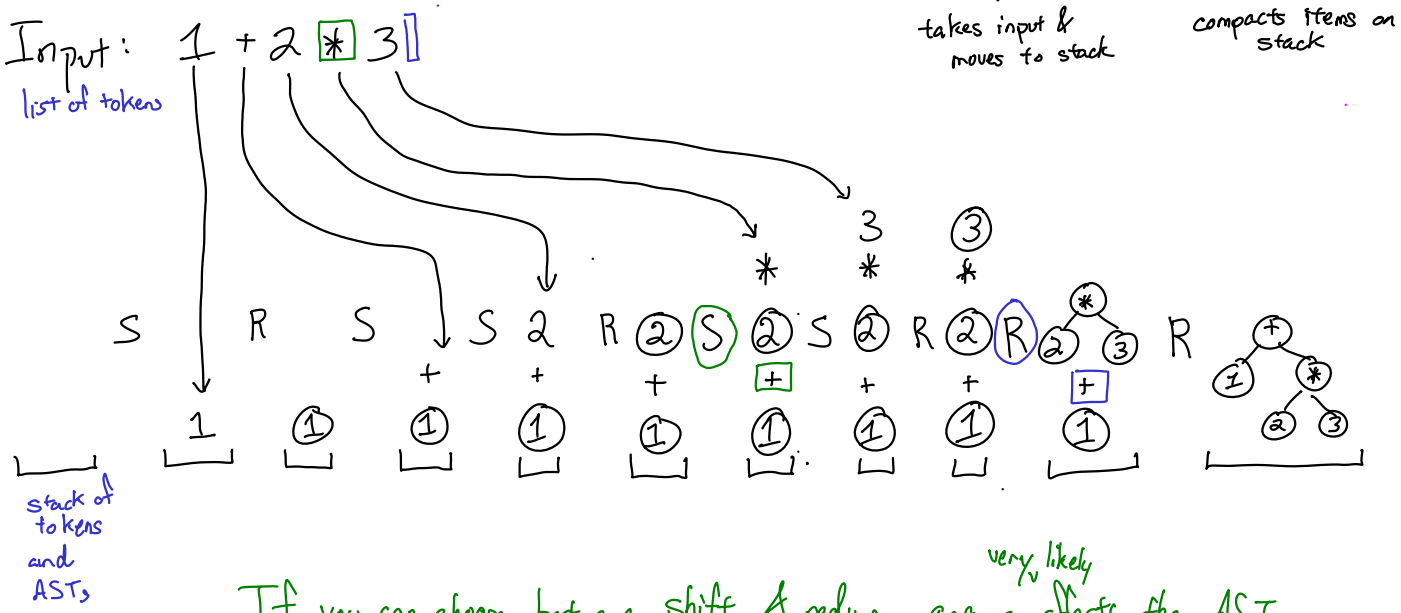
Here: Summary Algorithm (in spirit of LR)

Have: input string & stack

at each step, choose: "shift", "reduce"

"shift": takes input & moves to stack

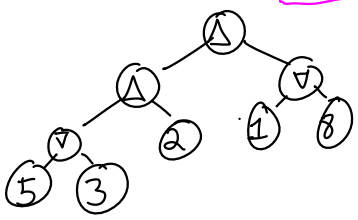
"reduce": compacts items on stack



If you can choose between shift & reduce, answer very likely affects the AST.

<expr> ::= ~~1~~
| <expr> Δ <expr>
| <expr> ∇ <expr>

This is an incomplete description.



5 ∇ 3 Δ 2 Δ 1 ∇ 8

.mly = "ocaml yacc" → "yet another compiler compiler"