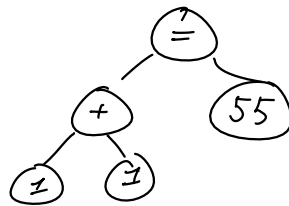


string \rightarrow token list

token list \rightarrow expr



$\langle \text{expr} \rangle ::= \begin{cases} \text{after}(\langle \text{expr} \rangle) \\ \langle \text{expr} \rangle + \langle \text{expr} \rangle \end{cases}$

Left-recursive grammar is one in which it is possible to follow from a rule back to itself without reaching a terminal on the left.
(token)

These parsers we are building are called LL parsers.

$\langle \text{expr} \rangle ::= \begin{cases} \text{after}(\langle \text{expr} \rangle) \\ \langle \text{expr} \rangle + \langle \text{expr} \rangle \end{cases}$

$\langle \text{expr} \rangle ::= \langle \text{prim_expr} \rangle \mid \langle \text{prim_expr} \rangle (+ \langle \text{prim_expr} \rangle)^+$

$\langle \text{expr} \rangle ::= \text{after}(\langle \text{tail} \rangle)$

$\langle \text{prim_expr} \rangle ::= \text{after}(\text{terminal})$

$\langle \text{tail} \rangle ::= \epsilon \mid + \langle \text{expr} \rangle$