

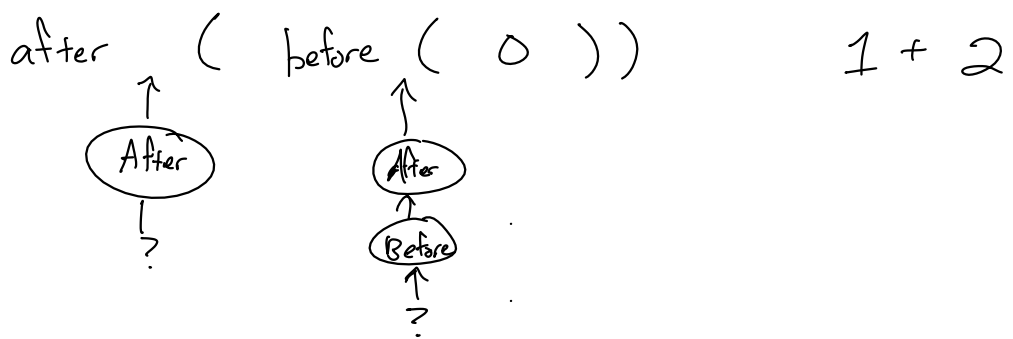
LL Parsing

Order of consumption (input) \uparrow

"LT parsing" — $L \equiv T$
left \equiv top

Order of construction \uparrow

Building from the top of AST down



LR Parsing

Bottom-up construction

$\langle \text{expr} \rangle ::= \langle \text{expr} \rangle + \langle \text{expr} \rangle$
 $|\langle \text{expr} \rangle * \langle \text{expr} \rangle$
 $|\langle \text{integer} \rangle$

$\text{prec}(+) < \text{prec}(*)$

Input (tokens)	1+2*3	+2*3	+2*3	2*3	*3	*3	3		
Stack (tokens, ASTs)		1	E 1	E 1 +	E 1 + 2	E 1 + 2 *	E 1 + 2 * 3	E 1 + 2 * 3	
			Shift					Reduce	

$\langle \text{expr} \rangle \cdot + \langle \text{expr} \rangle :$
 $\langle \text{expr} \rangle * \langle \text{expr} \rangle \cdot$

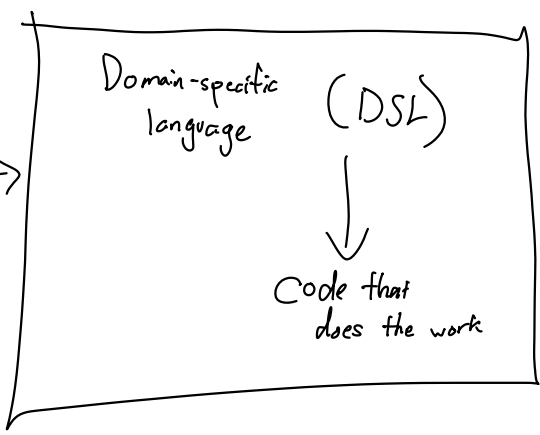
Input	+4	+4	+4
Stack	3 * 2 + 1	E * 2 3 + E 1	+ 1 * 2 3

.m/y
ocaml-related

yacc

yet
another
compiler
compiler

is



ocaml yacc



menhir

let x = 5 in 3 + x

→ let x = 5 in (3 + x)
(let x = 5 in 3) + x