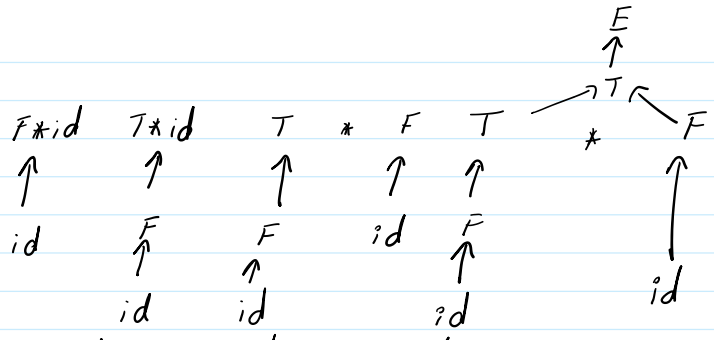
Automata Theory 5

05 October 2023 15:17

Bottom Up Parsing (LR Parsing)

$E \to E+T \mid T$
$T \to T*F \mid F$
$F \to E \mid id$



$id * id$ $\quad E \to T \to T*F \to T*id \to F*id \to id*id$

Handle Parsing

'handle is a substring matching the body of a punctuation st its reduction represents one step along the reverse rightmost derivation

(right sentential form)

RSF	Handle	Reducing P
id_1+id_2	id_1	$F \to id$
$F*id_2$	F	$T \to F$
$T+id_2$	id_2	$F \to id_2$
$T*F$	$T*F$	$F \to T*F$
T	T	$E \to T$

$S \xrightarrow[rm]{*} \alpha A w \xrightarrow[rm]{*} \alpha\beta w$

β after α is an input string we say β is a handle of $\alpha\beta w$

leftmost substring in a production is not necessarily a handle.

Shift Reduce Parsing - LR(1)

we have a stack, a current stack symbol / input symbol we can either

shift : push current symbol on top of stack

reduce: reduce stack to a parent production

Stack	Input	Action
$\$$	$id_1*id_2\$$	shift
$\$id_1$	$*id_2\$$	reduce
$\$F$	$*id_2\$$	reduce
$\$T$	$*id_2\$$	shift
$\$T*$	$id_2\$$	shift
$\$T*id_2$	$\$$	reduce
$\$T*F$	$\$$	reduce
$\$T$	$\$$	reduce

$\$$ - stack bottom and end of input

Shift reduce conflicts:
At some step in parsing, you can't decide whether to shift or reduce
grammars with such conflicts fall out of LR(k)

Lemma- an ambiguous grammar is never LR

$ I * id_2	$	reduce
$ T * F	$	reduce
$ T	$	reduce
$ E		accept

Lemma- an ambiguous grammar is never LR

stmt → if expr. then stmt
| if expr then stmt else stmt
| other

if expr. then $stmt_1$ else $stmt_2$

E → E+E
| E * E
| id

id + id + id