CSCI 301 Lecture 26: LL(1) Parsing

You have a grammar G.

You have removed left recursion.

You have removed common prefixes.

Is it LL(1)? Maybe.

Running Example: Reverse Polish Notation (postfix)

Infix: 1 + (4 - 2) * 3

Polish (Prefix): (+ 1 (* (- 4 2) 3))

Reverse Polish (Postfix): 1 4 2 - 3 * +

1 (4 2 - 3 * +)

1 (2 3 *) +

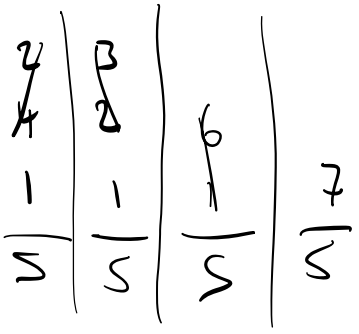
1 6 +

7

RPN: Look ma, no parentheses (or PEMDAS)! 100

Evaluation with a stack:

1 4 2 - 3 * +



Ex. PtA

RPN Grammar

- no left recursion
- no common prefixes

Example:

Parse 12_3_+

$S \to DNT$

| NT
| DNT
| 2NT
| 2T
| 2_S
| 2_DNT
| 2_3NT
| 2_3T
| 2_3_S
| 2_3_PS
| 2_3_+S
| 2_3_+

RPN expression ← $S \to \_S \mid PS \mid DNT \mid \varepsilon$

$T \to \_S \mid PS \mid \varepsilon$

number ← $N \to DN \mid \varepsilon$

digit ← $D \to 0 \mid 1 \mid 2 \mid \cdots \mid 9$

operator ← $P \to + \mid - \mid * \mid /$

Goal: Construct a Parse table:

leftmost nonterminal	↓ D	P	_	$ ← end of input
S →	DN			
T →				
N →				
D				
P				

lookahead symb

How? FIRST and FOLLOW

$S \to \_S \mid PS \mid DNT \mid \varepsilon$
$T \to \_S \mid PS \mid \varepsilon$
$N \to DN \mid \varepsilon$
$D \to 0 \mid 1 \mid 2 \mid \cdots \mid 9$
$P \to + \mid - \mid * \mid /$

Let A be a variable in V.

FIRST(A) is the set of terminals any derivation from A can start with.

$$FIRST(A) = \{x \in \Sigma : A \Rightarrow^* x\alpha \text{ for any } \alpha \in (\Sigma \cup V)^*\}$$

Calculating FIRST:

1. If A is a terminal, $FIRST(A) = \{A\}$
2. If A is a nonterminal and $A \to Y_1 Y_2 \cdots Y_k$, add $FIRST(Y_1)$ to $FIRST(A)$ ↖ nullable? $Y_1 \Rightarrow^? \varepsilon$

If $Y_1 \Rightarrow^* \varepsilon$, we need to add $FIRST(Y_2)$ too

3. If $Y_1 \ldots Y_q$ are nullable, add $FIRST(Y_{q+1})$ to first(A)

FOLLOW

What if $A \Rightarrow^* \varepsilon$?

FOLLOW(A): all terminals that can come after A in a derived string

(show RDN parse table)

Recursive Descent Parsing

- Each Nonterminal is a function
- It applies rules by calling functions from RHS
- It returns the rest of the string still to be parsed

Input: 44_9_+

	Parsed	Rest
→ S(44_9_+)	ε	44_9_+
D(44_9_+)	4	4_9_+
→ N(4_9_+)	4	4_9_+
D(4_9_+)	44	_9_+
N(_9_+)		_9_+
T(_9_+)		