

CS310

Parsing with Context Free Grammars

Today's reference:

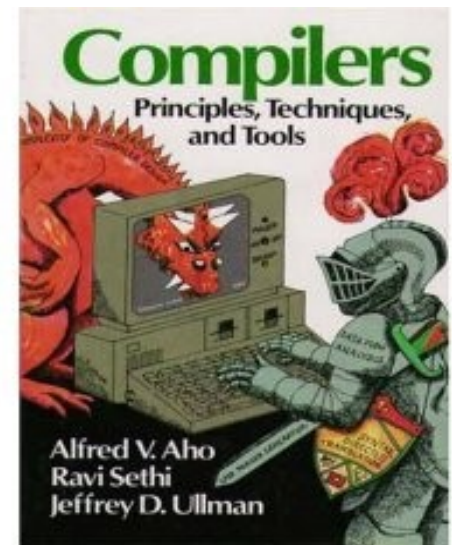
Compilers: Principles, Techniques, and Tools

by: Aho, Sethi, Ullman

aka: The Dragon Book

Section 4.4 – 4.8

Nov 3, 2010



Remove Left Recursion

- Immediate:
 $A \rightarrow Aa \mid a$
-

- Indirect:

$$S \rightarrow Aa \mid b$$

$$A \rightarrow Sd \mid Ac \mid \epsilon$$

$$\text{Recursion: } S \rightarrow Aa \rightarrow Sda$$

Algorithm (Dragon book, p 177 fig 4.7)

- Limits: no ϵ productions; no cycles
Arrange non-terminals in some order: $A_1 \dots A_n$

for $i = 1$ to n

 for $j = 1$ to $i-1$

 replace each production of the form

$A_i \rightarrow A_j Y$ by the productions

$A_i \rightarrow d_1 Y \mid d_2 Y \dots d_k Y$

 where $A_j \rightarrow d_1 \mid d_2 \mid \dots \mid d_k$ are all the
 current A_j productions

 end

 remove immediate recursion on A_i

end

Build Parse Table

$$\begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow (E) \mid \text{id} \end{array}$$

What transformations do we need to make?

What effects do these transformations have?

Parsing with JFLAP

- FIRST? FOLLOW? Parse Table?

- $S \rightarrow AcB$
- $A \rightarrow aAb$
- $A \rightarrow cBb$
- $B \rightarrow ccb$
- $B \rightarrow b$

The screenshot shows the JFLAP software interface. The main window is titled "JFLAP : <untitled1>" and has a menu bar with "File", "Input", "Test", "Convert", and "Help". The "Input" menu is open, showing options: "Build LL(1) Parse Table", "Build SLR(1) Parse Table", "Brute Force Parse", "Multiple Brute Force Parse", "User Control Parse", "CYK Parse", and "Multiple CYK Parse". The "Build LL(1) Parse" option is selected.

The "Build LL(1) Parse" dialog box is open, showing the following grammar rules:

S	→	AcB
A	→	aAb
A	→	cBb
B	→	ccb
B	→	b

The dialog also has a section for "Define FIRST sets. ! is the lambda character." with a table for FIRST and FOLLOW sets:

	FIRST	FOLLOW
A	{ }	{ }
B	{ }	{ }
S	{ }	{ }

Below this table is a larger table for the parse table, with columns for terminals 'a', 'b', 'c', '\$' and rows for non-terminals 'A', 'B', 'S'.

Parse

JFLAP : <untitled1>

File Input Test Convert Help

Editor Build LL(1) Parse LL(1) Parsing

Table Text Size

	a	b	c	\$
A	aAb		cBb	
B		b	ccb	
S	AcB		AcB	

Start Step Noninverted Tree

Input: cbb
Input Remaining: cbb\$
Stack: B

LHS		RHS
S	→	AcB
A	→	aAb
A	→	cBb
B	→	ccb
B	→	b

```
graph TD; S((S)) --- B((B));
```

Replacing S with AcB.

LL(2) Parse Table

- (1) $S \rightarrow AcB$
- (2) $A \rightarrow aAb$
- (3) $A \rightarrow aBb$
- (4) $B \rightarrow ccb$
- (5) $B \rightarrow b$

- Two lookahead symbols

	aa	ab	ac				
S							
A							

Bottom Up Parsing

- (1) $E \rightarrow E + E$
- (2) $E \rightarrow E * E$
- (3) $E \rightarrow (E)$
- (4) $E \rightarrow x$

- Shift-reduce parsing
 - used in many automatic parser generators,
- *Reduce* the string to the start symbol
- *Shift* a symbol from the string on to a stack

Stack	Input	Action
\$	x + x * x \$	shift
\$ x	+ x * x \$	reduce
\$ E		

Shift/Reduce Conflict

stmt \rightarrow IF expr THEN stmt
| IF expr THEN stmt ELSE stmt

Stack	Input	Action
... \$ IF expr THEN stmt	ELSE ... \$???

Sipser 2.24: $E = \{a^i b^j \mid i \neq j \text{ and } 2i \neq j\}$

$L = \{w \mid a^x a^z b^z b^x; x, z \geq 0\}$
