W-Assignment 2

Klaus Møllnitz
nhg665

December 8, 2013

1 Writing Context-Free Grammars

In the following subquestions, we are working with the alphabet:

$$\sum = \{a, b, c\} \tag{1}$$

1.1 a

Words have to match this regular expression a^*b^* and have to match more a's than b's.

$$S = aS \tag{2}$$
$$S = aT \tag{3}$$
$$T = aTb \tag{4}$$
$$T = \epsilon \tag{5}$$
$$\tag{6}$$

1.2 b

Words that are palindromes.

$$S = \epsilon \tag{7}$$
$$S = a \tag{8}$$
$$S = b \tag{9}$$
$$S = c \tag{10}$$
$$S = aSa \tag{11}$$
$$S = bSb \tag{12}$$
$$S = cSc \tag{13}$$
$$\tag{14}$$

A conflict free grammar for palindromes does not exist, since it belongs to a class of grammars which is not LR(1).

2 LL(1)-Parser Construction

We have the following grammar:

$$Z \to b|XYZ \tag{15}$$
$$Y \to \epsilon|c \tag{16}$$
$$X \to Y|a \tag{17}$$

2.1 a. Calculating *Nullable*

I need to calculate *Nullable* of all the nonterminals. I assume that all the equations in 18 is false, as a starting point. Then I use the algorithm 2.4 in Springer 2011, [1].

$$\begin{aligned} &\mathit{Nullable}(Z) \\ &\mathit{Nullable}(Y) \\ &\mathit{Nullable}(X) \end{aligned} \tag{18}$$

I then write down the expressions of Nullable for all nonterminals and terminals.

$$\begin{aligned} \mathit{Nullable(Z)} &= \mathit{Nullable(b)} \vee \mathit{Nullable(XYZ)} \\ \mathit{Nullable(Y)} &= \mathit{Nullable}(\epsilon) \vee \mathit{Nullable(c)} \\ \mathit{Nullable(X)} &= \mathit{Nullable(Y)} \vee \mathit{Nullable(a)} \\ \\ \mathit{Nullable}(b) &= \mathit{Nullable(b)} = false \\ \mathit{Nullable}(XYZ) &= \mathit{Nullable(X)} \wedge \mathit{Nullable(Y)} \wedge \mathit{Nullable(Z)} \\ \mathit{Nullable}(c) &= \mathit{Nullable(c)} = false \\ \mathit{Nullable}(\epsilon) &= \mathit{true} \\ \mathit{Nullable(Y)} &= \mathit{Nullable(Y)} \\ \mathit{Nullable(a)} &= \mathit{Nullable(a)} = false \end{aligned} \tag{19}$$

Then I use fixed point iterations to solve the equations.

Right-hand side	Init	It. 1	It. 2	It. 3	It. 4	It. 5
b	$false$	$false$	$false$	$false$	$false$	$false$
XYZ	$false$	$false$	$false$	$false$	$false$	$false$
ϵ	$false$	$true$	$true$	$true$	$true$	$true$
c	$false$	$false$	$false$	$false$	$false$	$false$
Y	$false$	$false$	$false$	$true$	$true$	$true$
a	$false$	$false$	$false$	$false$	$false$	$false$
Nonterminal						
Z	$false$	$false$	$false$	$false$	$false$	$false$
Y	$false$	$false$	$true$	$true$	$true$	$true$
X	$false$	$false$	$false$	$false$	$true$	$true$

(20)

The result is then (21).

$$\begin{aligned} \mathit{Nullable}(Z) &= False \\ \mathit{Nullable}(Y) &= True \\ \mathit{Nullable}(X) &= True \end{aligned} \tag{21}$$

2.2 a. Calculating *FIRST*

To calculate the *FIRST* set, I am uing the algorithm 2.5 in Springer 2011, [1].

$$
\begin{aligned}
FIRST(Z) &= FIRST(b) \cup FIRST(XYZ) \\
FIRST(Y) &= FIRST(\epsilon) \cup FIRST(c) \\
FIRST(X) &= FIRST(Y) \cup FIRST(a) \\
\\
FIRST(b) &= \{b\} \\
FIRST(XYZ) &= FIRST(X) \cup FIRST(Y) \cup FIRST(Z) \\
FIRST(c) &= \{c\} \\
FIRST(\epsilon) &= \emptyset \\
FIRST(a) &= \{a\}
\end{aligned} \tag{22}
$$

Then I use fixed point iterations to solve the equations.

Right-hand side	Init	It. 1	It. 2	It. 3	It. 4	It. 5
b	$\emptyset$	$\{b\}$	$\{b\}$	$\{b\}$	$\{b\}$	$\{b\}$
XYZ	$\emptyset$	$\emptyset$	$\emptyset$	$\{a,b,c\}$	$\{a,b,c\}$	$\{a,b,c\}$
ϵ	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
c	$\emptyset$	$\{c\}$	$\{c\}$	$\{c\}$	$\{c\}$	$\{c\}$
Y	$\emptyset$	$\emptyset$	$\emptyset$	$\{c\}$	$\{c\}$	$\{c\}$
a	$\emptyset$	$\{a\}$	$\{a\}$	$\{a\}$	$\{a\}$	$\{a\}$
Nonterminal						
Z	$\emptyset$	$\emptyset$	$\{b\}$	$\{b\}$	$\{a,b,c\}$	$\{a,b,c\}$
Y	$\emptyset$	$\emptyset$	$\{c\}$	$\{c\}$	$\{c\}$	$\{c\}$
X	$\emptyset$	$\emptyset$	$\{a\}$	$\{a\}$	$\{a,c\}$	$\{a,c\}$

(23)

The final FIRST set is gonna be as shown in (24).

$$
\begin{aligned}
FIRST(b) &= \{b\} \\
FIRST(XYZ) &= \{a,b,c\} \\
FIRST(\epsilon) &= \emptyset \\
FIRST(c) &= \{c\} \\
FIRST(Y) &= \{c\} \\
FIRST(a) &= \{a\}
\end{aligned} \tag{24}
$$

2.3 b. Calculating FOLLOW

I'm calculating the FOLLOW set according to the algorithm in the book page 59. To calculate the FOLLOW set, we first have to add an extra start production:

$$S = Z\$ \tag{25}$$

For each nonterminals, X, Y and Z, I locate all occurences on the right hand side of the productions, which leads me to following tables of constraints. First I calculate constraints for the nonterminal Z:

Production	*Constraints*
$S \to Z\$$	$FIRST(\{\$\}) = \{\$\} \subseteq FOLLOW(Z)$
$Z \to XYZ$	$FOLLOW(Z) \subseteq FOLLOW(Z)(\beta = \epsilon,$ this can be omitted since Z=Z)

(26)

Then I proceed calculating Y:

Production	*Constraints*
$X \rightarrow Y$	$FOLLOW(X) \subseteq FOLLOW(Y)(\beta = \epsilon)$
$Z \rightarrow XYZ$	$FIRST(Z) \subseteq FOLLOW(Y)(\beta = Z \text{ which is not nullable})$

(27)

Lastly I calculate X:

Production	*Constraints*
$Z \rightarrow XYZ$	$FIRST(YZ) \subseteq FOLLOW(X)(\beta = YZ, \text{nullable(YZ)} = \text{false})$

(28)

Since Y is nullable, I can write

$$FIRST(YZ) = FIRST(Y) \cup FIRST(Z) = \{a, b, c\} \tag{29}$$

The constraints table with FIRST set derived is then:

Production	*Constraints*
$S \rightarrow Z\$$	$\{\$\} \subseteq FOLLOW(Z)$
$X \rightarrow Y$	$FOLLOW(X) \subseteq FOLLOW(Y)$
$Z \rightarrow XYZ$	$\{a, b, c\} \subseteq FOLLOW(X), \{a, b, c\} \subseteq FOLLOW(Y)$

(30)

Follow-set	Init	It. 1	It. 2
$FOLLOW(Z)$	$\emptyset$	$\{\$\}$	$\{\$\}$
$FOLLOW(Y)$	$\emptyset$	$\{a, b, c\}$	$\{a, b, c\}$
$FOLLOW(X)$	$\emptyset$	$\{a, b, c\}$	$\{a, b, c\}$

(31)

2.4 c. Look ahead and parser table

I calculate the lookahead set for alle productions by using the information provided in the handed out slides about Syntax Analysis (slide 19).

Production	Lookahead set
$Z \rightarrow b$	$FIRST(b)$
$Z \rightarrow XYZ$	$FIRST(XYZ)$
$Y \rightarrow \epsilon$	$FIRST(\epsilon) \cup FOLLOW(Y)$
$Y \rightarrow c$	$FIRST(c)$
$X \rightarrow Y$	$FIRST(Y) \cup FOLLOW(X)$
$X \rightarrow a$	$FIRST(a)$

(32)

$$la(Z \rightarrow b) = \{b\} \tag{33}$$
$$la(Z \rightarrow XYZ) = \{a, b, c\} \tag{34}$$
$$la(Y \rightarrow \epsilon) = \{a, b, c\} \tag{35}$$
$$la(Y \rightarrow c) = \{c\} \tag{36}$$
$$la(X \rightarrow Y) = \{a, b, c\} \tag{37}$$
$$la(X \rightarrow a) = \{a\} \tag{38}$$

2.4.1 Parse table

Stack	a	b	c	$
Z	$XYZ, 2$	$XYZ, 2; b, 1$	$XYZ, 2$	*error*
Y	$\epsilon, 3$	$\epsilon, 3$	$\epsilon, 3; c, 4$	*error*
X	$Y, 5; a, 6$	$Y, 5$	$Y, 5$	*error*
a	*pop*	*error*	*error*	*error*
b	*error*	*pop*	*error*	*error*
c	*error*	*error*	*pop*	*error*
$	*error*	*error*	*error*	*accept*

(39)

3 SLR Parser Construction

3.1 Left-recursion

I made up the following grammar with left-recursion.

$$\begin{aligned} S &\to Sa \\ S &\to \epsilon \end{aligned} \tag{40}$$

I have made the parse table by using mosmlyac, with the -v flag. This introduce a new terminal as starting symbol, and a nonterminal called entry:

	\001	a	ϵ	*end*	S	*%entry%*
S_0	S_1					Go S_2
S_1		red. 2			Go S_3	
S_2				*accept*		
S_3		S_4		red. 3		
S_4			red. 1			

(41)

3.2 Parse table of grammar with right-recursion

The equivalent right-recursive grammar of (40) is as shown in (42).

$$\begin{aligned} S &\to aS \\ S &\to \epsilon \end{aligned} \tag{42}$$

The parse table of this grammar generated with mosmlyac can be seen below.

	\001	a	ϵ	*end*	S	*%entry%*
S_0	S_1					Go S_2
S_1		S_3		red. 2	Go S_4	
S_2				*accept*		
S_3		S_3		red. 2	Go S_5	
S_4			red. 3			
S_5			red. 1			

(43)

From these two parse tables, it can be shown that the *right-recursive* equivalent parser would build up a stack before doing the reduce actions, whereas the *left-recursive*, would simply reduce imediately.

References

[1] Torben Ægidius Mogensen, *Introduction to Compiler Design*. Springer London, 2011.