Lab Nine

Alex Smith

alex.smith1@Marist.edu

May 2, 2019

1 Crafting a Compiler

1.1 Exercise 5.5

Transform the following grammar into LL(1) form using the techniques presented in Section 5.5:

1	DeclList	→ DeclList ; Decl
2		| Decl
3	Decl	→ IdList : Type
4	IdList	→ IdList , id
5		| id
6	Type	→ ScalarType
7		| array (ScalarTypeList) of Type
8	ScalarType	→ id
9		| Bound .. Bound
10	Bound	→ Sign intconstant
11		| id
12	Sign	→ +
13		| −
14		| λ
15	ScalarTypelist	→ ScalarTypeList , ScalarType
16		| ScalarType

First, remove any left recursion by swapping the position of the non-terminals on the right side of the production.
1. DeclList → Decl ; DeclList
2. DeclList → Decl
3. Decl → IdList : Type
4. IdList → id , IdList
5. IdList → id
6. Type → ScalarType
7. Type → array(ScalarTypeList) of Type

8. ScalarType → id
9. ScalarType → Bound .. Bound
10. Bound → Sign intconstant
11. Bound → id
12. Sign → +
13. Sign → -
14. Sign → λ
15. ScalarTypeList → ScalarType , ScalarTypeList
16. ScalarTypeList → ScalarType

Next, use left factoring to let the operator be the first set where needed.

1. DeclList → Decl DeclList2
2. DeclList2 → ; DeclList
3. DeclList2 → λ
4. Decl → IdList : Type
5. IdList → id IdList2
6. IdList2 → , IdList
7. IdList2 → λ
8. Type → ScalarType
9. Type → array(ScalarTypeList) of Type
10. ScalarType → id
11. ScalarType → Bound .. Bound
12. Bound → Sign intconstant
13. Bound → id
14. Sign → +
15. Sign → -
16. Sign → λ
17. ScalarTypeList → ScalarType ScalarTypeList2
18. ScalarTypeList2 → , ScalarTypeList
19. ScalarTypeList2 → λ

At this point, the only conflict will be with ScalarType and Bound as they both have a first set of id and will be considered at the same time (Although ScalarTypeList2 and IdList2 both have a first set including , they will not be considered at the same time in the parse so this will not cause an issue). In order to resolve this, we can add productions to change the .. operator to be its own production as shown and removing the id production from ScalarType as shown below:

ScalarType → Bound Bound2
Bound2 → .. Bound
Bound2 → λ

2 Dragon Book

2.1 Exercise 4.5.3

Give bottom-up parses for the following input strings and grammars:
a) The input 000111 according to the grammar of Exercise 4.5.1.
Grammar: S $\rightarrow$ 0 S 1 | 0 1
Stack: empty, Input: 000111. Shift
Stack: 0, Input: 00111. Shift
Stack: 00, Input: 0111. Shift
Stack: 000, Input: 111. Shift
Stack: 0001, Input: 11. Reduce 01 to S
Stack: 00S, Input: 11. Shift
Stack: 00S1, Input: 1. Reduce 0S1 to S
Stack: 0S, Input: 1. Shift
Stack: 0S1, Input: empty. Reduce 0S1 to S
Stack: S, Input: empty. Accept!

b) The input aaa * a + + according to the grammar of Exercise 4.5
Grammar: S $\rightarrow$ S S + | S S * | a
Stack: empty, Input:aaa*a++. Shift
Stack: a, Input: aa*a++. Reduce a to S
Stack: S, Input: aa*a++. Shift
Stack: Sa, Input: a*a++. Reduce a to S
Stack: SS, Input:a*a++. Shift
Stack: SSa, Input:*a++. Reduce a to S
Stack: SSS, Input:*a++. Shift
Stack: SSS*, Input:a++. Reduce SS* to S
Stack: SS, Input: a++. Shift
Stack: SSa, Input: ++. Reduce a to S
Stack: SSS, Input: ++. Shift
Stack: SSS+, Input:+. reduce SS+ to S
Stack: SS, Input: +. Shift
Stack: SS+, Input: empty. Reduce SS+ to S
Stack: S, Input:empty. Accept!

2.2 Exercise 4.6.5

Show that the following grammar:

S $\rightarrow$ A a A b | B b B a
A $\rightarrow$ ϵ
B $\rightarrow$ ϵ

is LL(1) but not SLR(1).

This grammar is LL(1) because if the first symbol is a, the first production of S will be used and if the first symbol is b, the second production of S will be used. Although both nonterminals have the same first set, the first set is epsilon so this first set will not be considered.

This grammar is not SLR(1) because it will be unclear when to reduce to A or B since it can be reduced an infinite number of times due to needing no input. So at each step (assuming reduce has more priority over shift) it will attempt to reduce without knowing the proper amount of times to reduce or if to reduce to A or B since they are the same.