Sample Solutions for Chapter 8

Exercise 1: Table Generation (1)

Given the following grammar, create the parser table for bottom-up syntax analysis.

```
A = A B c.
A = a.
B = b.
```

Is this grammar LR(0), LR(1) or LALR(1)? Give reasons for your answer.

Solution

The productions are first numbered consecutively and a pseudo-production for S' is added:

```
0  S' = A #
1  A = A B c
2  A = a
3  B = b
```

The table generation then looks as follows:

```
0   S = . A #                  shift  a    1
    A = .a          / #b       shift  A    2
    A = . A B c     / #b

1   A = a .         / #b       red    #,b  2  (A = a)

2   S = A . #                  acc    #
    A = A . B c     / #b       shift  b    3
    B = . b         / c        shift  B    4

3   B = b .         / c        red    c    3  (B = b)

4   A = A B . c     / #b       shift  c    5

5   A = A B c .     / #b       red    #,b  1  (A = A B c)
```

This results in the following parser table:

	a	b	c	#	S	A	B
0	s1					s2	
1		r2		r2			
2		s3		acc			s4
3			r3				
4			s5				
5		r1		r1			

The grammar is LR(0) because there is no state in which both a shift and a reduce action can be executed and because in each reduce state, a reduction can only take place according to a single production. This means that the parser can decide on the correct reduction in every reduce state without a lookahead symbol.

Since the grammar is LR(0), it is also LR(1): If a distinction can be made between shift and reduce without a lookahead symbol, this is even more possible with 1 lookahead symbol.

The grammar is also LALR(1). No states have been merged, so the LALR(1) property cannot deviate from the LR(1) property.