DEPARTMENT OF COMPUTER SCIENCE
CARLOS III UNIVERSITY OF MADRID

Computer Science
Language Processors

Rules

- The duration of the test is **60 minutes**
- Questions will not be answered during the test
- One cannot re-enter the classroom after leaving it
- The answers must be written using a pen (not a pencil)

1.- Given the grammar:

```
S ::= AD
A ::= bB | ƛ
B ::= Ca | D
C ::= a | ƛ
D ::= b | c
```

A) Construct the parsing table for the SLR(1) parser and represent the DFA that is obtained using this methodology.

Augmented Grammar

S' → S
(1) S → AD
(2, 3) A → bB | λ
(4, 5) B → Ca | D
(6, 7) C → a | λ
(8, 9) D → b | c

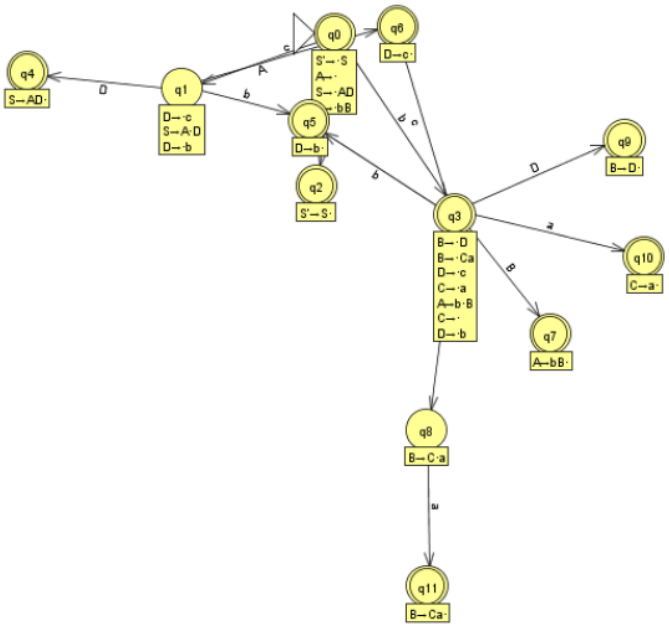
LR(0) canonical set

I0 = { [S' →·S], [S→ ·AD], [A→·bB] , [A→ λ·]}
I1 = goto (I0, S) = {[S' →S·]}
I2 = goto (I0, A) = {[S→ A·D], [D→ ·b], [D→·c]}
I3 = goto (I0, b) = {[A→ b·B] , [B→·Ca], [B→·D], [C→·a], [C→ λ·], [D→ ·b], [D→·c]}
I4 = goto(I2, D) = {[S→ AD·]}
I5 = goto(I2, b) = {[D→ b·]}
I6 = goto(I2, c) = {[D→ c·]}
I7 = goto(I3, B) = {[A→ bB·]}
I8 = goto(I3, C) = {[B→C·a]}
I9 = goto(I3, D) = {[B→D·]}
I10 = goto(I3, a) = {[C→a·]}

goto(I3, b) = I5
goto (I3, c) = I6

I11 = goto(I8, a) = {[B→Ca·]}

We can represent the DFA as follows:



Finally, to construct the SLR(1) parsing table we need to know:

FOLLOW(A) = {b, c} FOLLOW(C) = {a} FOLLOW(S) = {$}
FOLLOW(D) = {b,c,$} FOLLOW(B) = {b,c}

STATES	ACTIONS				GOTO				
	a	b	c	$	S	A	B	C	D
0		S3 R3	R3		1	2			
1				ACCEPT					
2		S5	S6						4
3	S10 R7	S5	S6				7	8	9
4				R1					
5		R8	R8	R8					
6		R9	R9	R9					
7		R2	R2						
8	S11								
9		R5	R5						
10	R6								
11		R4	R4						

B) Construct the parsing table for the LR(1) parser.

The states of the LR(1) parser are the following:

I0 = { [S' →·S,$], [S→ ·AD,$], [A→·bB,b/c] , [A→ λ·, b/c]}

I1 = goto (I0, S) = {[S' →S·,$]}
I2 = goto (I0, A) = {[S→ A·D,$], [D→ ·b,$], [D→·c,$]}
I3 = goto (I0, b) = {[A→ b·B,b/c] , [B→·Ca,b/c], [B→·D,b/c], [C→·a,a], [C→ λ·,a], [D→ ·b,b/c], [D→·c,b/c]}

I4 = goto(I2, D) = {[S→ AD·,$]}
I5 = goto(I2, b) = {[D→ b·,$]}
I6 = goto(I2, c) = {[D→ c·,$]}

I7 = goto(I3, B) = {[A→ bB·,b/c]}
I8 = goto(I3, C) = {[B→C·a,b/c]}
I9 = goto(I3, D) = {[B→D·,b/c]}
I10 = goto(I3, a) = {[C→a·,a]}
I11 = goto(I3, b) = {[D→ b·,b/c]}
I12 = goto(I3, c) = {[D→ c·,b/c]}

I13 = goto(I8, a) = {[B→Ca·,b/c]}

FIRST(A) = {λ,b} FIRST(C) = {λ,a} FIRST(S) = {b,c}
FIRST(D) = {b,c} FIRST(B) = {a,b,c}

STATES	ACTIONS				GOTO				
	a	b	c	$	S	A	B	C	D
0		S3 R3	R3		1	2			
1				ACCEPT					
2		S5	S6						4
3	S10 R7	S11	S12				7	8	9
4				R1					
5				R8					
6				R9					
7		R2	R2						
8	S13								
9		R5	R5						
10	R6								
11		R8	R8						
12		R9	R9						
13		R4	R4						

C) Construct the parsing table for the LALR(1) parser.

The LALR(1) parsing table can be constructed by using the LR(1) states and merging those states that are identical if the lookaheads are ignored, i.e., two states being merged must have the same number of items and the items have the same core (i.e., the same productions, differing only in lookahead). The lookahead on merged items is the union of the lookahead from the states being merged. This way, the states that can be merged are:

- States 5 and 11 = State 511
- States 6 and 12 = State 612

The resulting parsing table for the LALR(1) after joining these states is:

STATES	ACTIONS				GOTO				
	a	b	c	$	S	A	B	C	D
0		S3 R3	R3		1	2			
1				ACCEPT					
2		S5	S6						4
3	S10 R7	S11	S12				7	8	9
4				R1					
511		R8	R8	R8					
612		R9	R9	R9					
7		R2	R2						
8	S13								
9		R5	R5						
10	R6								
13		R4	R4						

D) Extract conclusions about which methods can be used or not and show how the input string *aabece* would be analyzed using the correct ones.

The three methodologies cannot be used due to there are not conflicts in the corresponding parsing tables. As it can be seen, the LR(1) is the parsing technique with the large number of states and the parsing tables for the LALR(1) and SLR(1) parsing techniques are the same.

E) Complete the parsing table for the SLR(1) parser to provide the user with detailed information about the different errors that can be detected during the parsing process.

Possible information that the SLR(1) parser can provide to inform the user when a string cannot be parsed due to a error (empty cells in the parsing table) is shown following:

STATES	ACTIONS					GOTO	
	a	b	c	e	$	S	D
0	S2	S3	*Waiting a or b*	*Waiting a or b*	*Waiting a or b*	1	
1	*Waiting end of string*	*Waiting end of string*	*Waiting end of string*	*Waiting end of string*	ACCEPT		
2	S2	S3	*Waiting a or b*	*Waiting a or b*	*Waiting a or b*	4	