UNIT 6: BOTTOM-UP PARSING TECHNIQUES

We want to incorporate a repetitive sentence into a high-level language. The sentence can be represented by the following regular expression:

repeat (identifier | number) >> sentence^{+} <<

A program consists of at least one statement, where statements can be assignments, conditionals, and loops.

NOTE: The symbols "|" and "+" are part of the regular expressions, the others are part of the language.

It is required:

1. Define the grammar G that would generate valid programs of this programming language. Consider the assignment and conditional statements as terminal symbols of the grammar.

2. Obtain the table of an SLR analyzer from the previous G grammar or an equivalent one that allows an SLR analysis.

David Griol Barres, Antonio Berlanga de Jesús, Jesús García Herrero, Juan Manuel Alonso Weber

SOLUTION:

A grammar to generate the language defined:

G = {assigment, condition, id, n, repeat, (,), <<, >>},{**S**, **S'**, **B**, **E**, **R**}, {**S**}
(1) **S**::= **E S'**
(2) **S'**::= **S**
(3) **S'**::= λ
(4) **E**::= assigment
(5) **E**::= condition
(6) **E**::= **B**
(7) **B**::= repeat (**R**
(8) **R**::= id) >> **S** <<
(9) **R**::= n) >> **S** <<

We add a new symbol S'' and a new production rule S''::=S.

State 0	Action	Go To
S''::= ·S		[0,S]=1
S::= ·ES'		[0,E]=2
E::= ·assigment	[0,asig]=D3	
E::= ·condition	[0,cond]=D4	
E::= ·B		[0,E]=5
B::= ·repeat (R	[0,repeat]=D6	
State 1	Action	Go To
S''::= S·	[1,$]=ACP	
State 2	Action	Go To
S::= E·S'		[2,S]=9
S'::= ·S		[2,S]=2
S'::= λ	[2,<<]=R3 [2,$]=R3	
S::= ·ES'		[2,S]=3
E::= ·assigment	[2,asig]=D3	
E::= ·condition	[2,cond]=D4	
E::= ·B		[2,E]=5
B::= ·repeat (R	[2,repeat]=D6	
State 3	Action	Go To
E::= assigment·	[3,asig]=R4 [3,cond]=R4 [3,repeat]=R4 [3,<<]=R4 [3,$]=R4	
State 4	Action	Go To
E::= condition·	[4,asig]=R5 [4,cond]=R5 [4,repeat]=R5 [4,<<]=R5 [4,$]=R5	
State 5	Action	Go To
E::= B·	[5,asig]=R6 [5,cond]=R6 [5,repeat]=R6 [5,<<]=R6 [5,$]=R6	
State 6	Action	Go To
B::= repeat · (R	[6,(]=D9	
State 7	Action	Go To
S::= ES'·	[7,<<]=R1 [7,$]=R1	
State 8	Action	Go To
S'::= S·	[8,<<]=R2 [8,$]=R2	
State 9	Action	Go To
B::= repeat (· R R::= · id) >> S << R::= · n) >> S <<	 [9,id]=D11 [9,n]=D12	[9,R]=10
State 10	Action	Go To
B::= repeat (R ·	[10,asig]=R7 [10,cond]=R7 [10,repeat]=R7 [10,<<]=R7 [10,$]=R7	
State 11	Action	Go To
R::=id ·) >> S <<	[11,id]=D13	
State 12	Action	Go To
R::= n ·) >> S <<	[12,n]=D14	
State 13	Action	Go To
R::=id) · >> S <<	[13,>>]=D15	
State 14	Action	Go To
R::= n) · >> S <<	[14,>>]=D16	
State 15	Action	Go To
R::=id) >> · S << S::= ·ES' E::= ·assigment E::= ·condition E::= ·B B::= ·repeat (R		[15,S]=17
State 16	Action	Go To
R::= n) >> · S << S::= ·ES' E::= ·assigment E::= ·condition E::= ·B B::= ·repeat (R		[16,S]=18
State 17	Action	Go To
R::=id) >> S · <<	[17,<<]=D19	
State 18	Action	Go To
R::= n) >> S · <<	[18,<<]=D20	
State 19	Action	Go To
R::=id) >> S << .		
State 20	Action	Go To
R::= n) >> S << ·		

David Griol Barres, Antonio Berlanga de Jesús, Jesús García Herrero, Juan Manuel Alonso Weber