SEMINAR 5

1) se da gramatica G cu productiile

E->TR R-> +TR | \*TR | lambda T->n

1: 2: 3: 4: 5:

Sa se construiasca tabela LR(1) pentru G.

Extindem G: E’->E; adugam simbolul terminal nou #

Se calculeaza multimile canonice LR(1)

I0= E’->.E;# —>I1 (goto(I0,E)) // aplic closure pt E

E->.TR;# (#=First(lambda.#)) —>I2 // aplic closure pt T

T->.n; +|\*|# (=First(R#)) —>I3

I1= E’->E.;#

I2= E->T.R;# —>I4 // aplic closure pt R

R->.+TR;#—>I5

R->.\*TR;# —>I6

R->.;#

I3= T->n.;+|\*|#

I4= E->TR.;#

I5= R->+.TR;# —>I7

T->.n;+|\*|# —>I3

I6= R->\*.TR;# —>I8

T->.n;+|\*|# —>I3

I7= R->+T.R;# —>I9

R->.+TR;#—>I5

R->.\*TR;# —>I6

R->.;#

I8= R->\*T.R;# —>I10

R->.+TR;#—>I5

R->.\*TR;# —>I6

R->.;#

I9= R->+TR.;#

I10= R->\*TR.;#

Construim tabela LR(1)

Tabela action tabela goto

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| M | + | \* | n | # | E | T | R |
| 0 | error | error | Shift 3 | error | 1 | 2 | error |
| 1 | ... |  |  | accept |  |  |  |
| 2 | Shift 5 | Shift 6 |  | Reduce 4 |  |  | 4 |
| 3 | Reduce 5 | Reduce 5 |  | Reduce 5 |  |  |  |
| 4 |  |  |  | Reduce 1 |  |  |  |
| 5 |  |  | Shift 3 |  |  | 7 |  |
| 6 |  |  | Shift 3 |  |  | 8 |  |
| 7 | Shift 5 | Shift 6 |  | Reduce 4 |  |  | 9 |
| 8 | Shift 5 | Shift 6 |  | Reduce 4 |  |  | 10 |
| 9 |  |  |  | Reduce 2 |  |  |  |
| 10 |  |  |  | Reduce 3 |  |  |  |

Tabela M nu are intrari multiple <=> G este LR(1)

Se analizeaza sirul n\*n

(0, n\*n#, lambda) -> (shift 3) (0n3, \*n#, lambda) ->(reduce 5) (0T2, \*n#, 5) >

2=goto(0,T)

(shift 6) (0T2\*6, n#, 5) -> (Shift 3) (0T2\*6n3, #, 5) -> (Reduce 5) (0T2\*6T8, #, 55 )

8=goto(6,T)

-> (Reduce 4)(0T2\*6T8R’10’, #, 455) -> (Reduce 3) (0T2R4, #, 3455) -> (Reduce 1) (0E1, #, 13455) -> accept

2) Se da gramatica G2: E-> E+E | E\*E | n

1 2 3

Sa se construiasca tabela SLR(1) pentru G2.

Extindem G: E’->E; adugam simbolul terminal nou #

Calculam Follow(X), X neterminal; se initializeaza Follow(simbol\_de\_start)={#}

|  |  |
| --- | --- |
| Follow(E) | #, +, \* |

Multimile canonice LR(0)

I0= E’->.E —>I1

E->.E+E —>I1

E->.E\*E —>I1

E->.n —>I2

I1= E’->E.

E->E.+E —>I3

E->E.\*E —>I4

I2= E->n.

I3= E->E+.E —>I5 I5= E->E+E.

E->.E+E —>I5 E->E.+E —>I3

E->.E\*E —>I5 E->E.\*E —>I4

E->.n —>I2

I4= E->E\*.E —>I6 I6= E->E\*E.

E->.E+E —>I6 E->E.+E —>I3

E->.E\*E —>I6 E->E.\*E —>I4

E->.n —>I2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| M2 | + | \* | n | # | E |
| 0 | error | error | Shift 2 | error | 1 |
| 1 | Shift 3 | Shift 4 | error | accept | error |
| 2 | Reduce 3 | Reduce 3 |  | Reduce 3 |  |
| 3 | error | error | Shift 2 | error | 5 |
| 4 | error | error | Shift 2 |  | 6 |
| 5 | Reduce 1/ shift 3 | Reduce 1/ shift 4 | error | Reduce 1 | error |
| 6 | Reduce 2/ shift 3 | Reduce 2/ shift 4 | error | Reduce 2 | error |

Tabela M2 are intrari multiple <=> G2 nu este SLR(1) (G2 este ambigua)

Pentru cazul particular al gramaticilor ambigue, ca cea de mai sus, care genereaza expresii cu operatori binari, putem elimina conflictele pornind de la prioritatile si asociativitatea operatorilor:

+, \* asoc la stanga

\*prioritar fata de +

Reduce 1/ shift 3 pt ‘+’ in starea 5 --> alegem reduce 1 (+ asoc la stg)

Reduce 1/ shift 4 pt ‘\*’ in starea 5 --> alegem shift 4 (\* pri +)

Reduce 2/ shift 3 pt ‘+’ in starea 6 --> alegem reduce 2

Reduce 2/ shift 4 pt ‘\*’ in starea 6 --> reduce 2 (\* asoc la stg)