Seminar 5

1) Sa se construiasca tabela LR(1) pentru gramatica G cu productiile:

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

1: 2: 3: 4: 5:

Extindem: S->E introducem simbolul nou #

I0= S->.E;# goto(I0,E)=I1

E->.TR;# (=First(#)) goto(I0,T)=I2

T->.n; +|\*|# (=First(R#)) —>I3

I1 S->E.;#

I2 E->T.R;#—>I4

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 de tip LR(1)

Sa se analizeze sirul n\*n

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

2=goto(0,T)

(shift 6) (0T2\*6, n#, 5) -> (shift 3) (0T2\*6n3. #. 5) -> (reduce 5) (0T2\*6T8, #, 5 5)->

(reduce 4) (0T2\*6T8R’10’, #, 4 5 5) -> (reduce 3) (0T2R4, #, 3 4 5 5) ->

(reduce 1) (0E1, #, 1 3 4 5 5) -> (accept)

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

1: 2 3

Sa se construiasca tabela SLR(1) pentru G2.

Extindem G2: E’->E: adaugam simbolul terminal #

Calculam Follow(X), X neterminal; se initializeaza Follow(E)={#}

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

E->E+E Follow(E)+=First(+E.Follow(E))={+}

E->E\*E Follow(E)+=First(\*E.Follow(E))={\*}

Se calculeaza 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

E->.E+E —>I5

E->.E\*E —>I5

E->.n —>I2

I4= E->E\*.E —>I6

E->.E+E —>I6

E->.E\*E —>I6

E->.n —>I2

I5= E->E+E.

E->E.+E —>I3

E->E.\*E —>I4

I6= E->E\*E.

E->E.+E —>I3

E->E.\*E —>I4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | + | \* | n | # | E |
| 0 | error | error | Shift 2 | error | 1 |
| 1 | Shift 3 | Shift 4 |  | accept |  |
| 2 | Reduce 3 | Reduce 3 |  | Reduce 3 |  |
| 3 |  |  | Shift 2 |  | 5 |
| 4 |  |  | Shift 2 |  | 6 |
| 5 | Reduce 1/  Shift 3 | Reduce 1/  Shift 4 |  | Reduce 1 |  |
| 6 | Reduce2/ Shift 3 | Reduce 2/  Shift 4 |  | Reduce 2 |  |

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

Pentru cazul acesta particular: se elimina conflictele tinand cont de asociativitatea lui +, \* (stanga) si de faptul ca \* prioritar fata de +

Reduce 1/ shift 3 pt ‘+’ in starea 5 —> se alege reduce 1

Reduce 1/ shift 4 pt ‘\*’ in starea 5 —> se alege shift 4

Reduce 2/ shift 3 pt ‘+’ in starea 6 —> se alege reduce 2

Reduce 2/ shift 4 pt ‘\*’ in starea 6 —> se alege reduce 2

T->.;