Parsing - 3

- **More on SLR(1) parsing**
  - **Definition of First and Follow sets**
  - **How to calculate First and Follow sets?**
  - **Example SLR(1) parser without conflicts**

How to calculate Follow Sets

- **If nonterminal X produces the empty string it is called *nullable.***
  - **This property can be calculated independently of the next two sets.**
- ***Follow[X]* is the set of terminals which immediately follow X in some derivation**
  - **t Follow[X] if some derivations contains XYZt where both Y and Z derive the empty string**
- ***First*[γ] is defined to be the set of terminals that can begin a string derived from γ**

Algorithm

Reformulation of algorithm in Appel, Alg 3.13, p 51.

repeat
{for each production $X \rightarrow Y_1 Y_2 \ldots Y_m$ do
{if for 1<= j <= m, all the Y_j are nullable or j=0,
then X is nullable;
for 1<= j <= m/*form First sets*/
{if $Y_1 \ldots Y_{j-1}$ are nullable
then First[X]=First[X] First[Y_j];
}

Algorithm, cont.

repeat

{for each production $X \rightarrow Y_1 Y_2 \ldots Y_m$ do

...

for 1<= j <= m /*Follow set calculation; disting symbol*/
/*has $ in its Follow set*/

{ if $Y_{j+1} \ldots Y_m$ are nullable or j=m,
then Follow[Y_j]=Follow[Y_j] Follow[X];

for j+1<= n<= m do

{if $Y_{j+1} \ldots Y_{n-1}$ are nullable
then Follow[Y_j]=Follow[Y_j] First[Y_n]

}

}

}

until First, Follow, and nullable sets don’t change on an iteration of the repeat loop.

Example, Appel p 49

Z' → Z (1)

Z → X Y Z (2) | d (3)

Y → c (4) | ε (5)

X → Y (6) | a (7)

1. Calculate *nullable* nonterminals:

Y is nullable from rule (5)   X is also nullable by rule (6), but Z is not nullable by rule (2), because even if X and Y in XYZ are nullable, Z must eventually expand to d to stop the recursion in the production.

Example, cont.

Z’ → Z (1)
Z → X Y Z (2) | d (3)
Y → c (4) | ε (5)
X → Y (6) | a (7)

2. Calculate First sets

First(Z’) = First(Z) from (1);

First(Z) First(X), First(Z) First(Y), d First(Z) from (3)

First(Y) = {c} from (4), First(X) First(Y) from (6) and a First(X) from (7)

Therefore, First(X)={c,a}, First(Z)={c,a,d}=First(Z’)

Example, cont.

Z' → Z (1)

Z → X Y Z (2) | d (3)

Y → c (4) | ε (5)

X → Y (6) | a (7)

3. Calculate Follow sets. Recall X,Y nullable.

Follow(Z')=Follow(Z)   {$} from (1)

Follow(X)   First(Z)={c,a,d},Follow(Y)   First(Z) from (2)

Follow(X)=Follow(Y) from (6)

Therefore, Follow(Z')=Follow(Z)={$}.

Follow(X)=Follow(Y)={c,a,d}

SLR(1) Example

E’ → E
E → E   T | E + T | T
T → *id*

If A → α . *a* β in I_k and goto(I_k , *a*) = I_j table entry for (k,*a*) is sj for *a* terminal symbol.
If A → α . in I_k then table entry for (k,*b*) is r(rule#) where *b* is in Follow(A).
If S’ → S in I_k then table entry for (k,*$*) is accept.

Calculate Follow sets:
Follow(E’) = {$}, Follow(E) = {$ * +},
Follow(T) = {$ * +}

I_0 : E’ → .E
E → .E * T
E → .E + T
E → .T
T → .*id*

I_1 : E’ → E. {$}
E → E. * T
E → E. + T

I_3 : E → E + .T
T → .*id*

I_2 : E → E * . T
T → .*id*

I_4 : E → E + T. {$ * +}

I_6 : T → *id*. {$ * +}

I_5 : E → E * T. {$ * +}

I_7 : E → T. {$ * +}

Example - Parser Table

Draw the parser table to show there are no conflicts.

Notice although this grammar has the operator precedences wrong, it is NOT ambiguous

	+	*	id	$	goto E	goto T
0			s6		1	7
1	s3	s2		accept		
2			s6			5
3			s6			4
4	r(iii)	r(iii)		r(iii)		
5	r(ii)	r(ii)		r(ii)		
6	r(v)	r(v)		r(v)		
7	r(iv)	r(iv)		r(iv)		