Luke Paireepinart Homework 2 – Crafting Compilers

(a)

Grammar:

S -> iSeS

| iS

| a

Nonterminals: Follow:  
S i, e, a, EOF

Items:

S -> •iSeS

S -> i•SeS

S -> iS•eS

S -> iSe•S

S -> iSeS•

S ->•iS

S -> i•S

S -> iS•

S -> •a

S -> a•

Closure(S -> •iSeS) = {S -> •iSeS, S -> •iS, S-> •a}

= i0

GOTO(i0, iSeS) = {S->i•SeS, S-> i•a} = i1

GOTO(i0, iS) = {S -> i•S, S-> i•a} = i2

GOTO(i0, a) = {S->a} = i3

S0 =[ S -> •iSeS, a], [S ->•iS, a], [S -> •a, EOF]

S1 = [S -> i•SeS, a]

S2 = [S -> i•S, a]

Table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | ACTION | | | GOTO |
| State | i | a | e | EOF | S |
| 0 | s1 | r1 | s3 |  | 2 |
| 1 | r1 |  |  |  | 0 or 3 |
| 2 | s0 | r1 | s0 | acc | 2 |
| 3 | r1 | s2 | s0 |  | 0 or 3 |

(b) The conflict occurs because when an ‘i’ is on the stack, you don’t know whether to reduce to the state that leads to the terminal ‘e’ or not. For example, at ‘i’ you can only look ahead to the nonterminal S, which is an ‘i’ or an ‘a’ in either case. You don’t know if after the ‘i’ or ‘a’ if there will be another ‘i’ or ‘a’, or if there will be an ‘e’, because you only have 1 step of lookahead. This is a reduce/reduce conflict.

Another thing that’s not necessarily a conflict but if the grammar is written as three separate rules, it will be more clear which rules are being used for reductions (rather than having every reduction be by rule1.)

The way to fix the ambiguity is to add another production so that there is no longer a conflict.

S -> iSB | a

B -> empty | eS