IO Item Generation

Prepared By:
Assoc. Lec. Ahmed A. Alsamman
University of Mosul/ College of Computer Science and Mathematics/ Computer Science Dept.

Generating LR(0) Items

Example:

Given Grammar G:

S→AA
A→aA
A→b

Step 1: Generate augmented Grammar G'

Augmented grammar G':

S'→S
S→AA
A→aA
A→b

Generating LR(0) Items – Cont.

Step 2: Generate I0 item

Follow the closure function

```
function closure ( list of production rules Item )
begin
Repeat
  I := Item;
  for each production P in I of the form P -> .Aβ do
     for each production A -> β in the Grammar do
              if (A -> .β) ∉ I
       Item := Item + (A -> .β)
    end for
  end for
Until no new production added to Item
return Item
end closure
```

I0 Item: Closure (S' →.S)

S' → .S
S → .AA
A→ .aA
A→ .b

```
1  //######################    I0 item generator   ########################
2  //#################### CODED BY AHMED ALI ALSAMMAN ####################
3  // Input: Grammar G
4  // Output: I0 item
5  using System;
6  using System.Collections.Generic;
7  using System.Linq;
8
9  namespace I0
10 {
11     class Program
12     {
13         static List<string> G = new List<string>() { "E→E+T", "E→T", "T→T*F",
             "T→F", "F→(E)", "F→i" };
14         static List<char> closured = new List<char>();// to store closured NT
15         static void Main(string[] args)
16         {
17             List<string> I0 = new List<string>();
18             Console.OutputEncoding = System.Text.Encoding.UTF8; // to write non
                 ascii characters on the console
19             string augG = G[0][0] + "'→" + G[0][0]; // augmented grammar G':
               E'→E
20             string prod = augG.Substring(0, 3) + '.' + augG.Substring(3); //
               E'→.E
21
22             I0 = closure(new List<string>() { prod }); // perform closure on
               E'->.E;
23             print(I0);
24           }
25
26          static List<string> closure(List<string> Item)
27         {
28             int prevCount;
29             do
30             {
31                 prevCount = Item.Count();
32                 List<string> I = new List<string>(Item);
33                 foreach (string prod in I)
34                 {
35                     char symbol = prod[prod.IndexOf('.')+1]; // next char to the .
36
37                     if (Char.IsUpper(symbol) && !closured.Contains(symbol)) //
                       perform closure
38                     {
39                         foreach (string gProd in G)
40                           if (gProd[0] == symbol) // lookup left hand side
                         ofgrammar G for symbol
41                             Item.Add(gProd.Substring(0, 2) + "." +
                         gProd.Substring(2));
42                         closured.Add(symbol);// mark symbol as closured
43                     }
44                 }
45             } while (prevCount < Item.Count()); // repeat if there is a new
               production rule
46             return Item;
47         }
```

```
48
49
50          static void print(List<string> Item)
51          {
52              Console.WriteLine("I0\n------");
53              foreach (string pr in Item)
54                  Console.WriteLine(pr);
55
56              Console.WriteLine();
57          }
58
59      }
60  }
61
```


Thank You!


Any
Questions