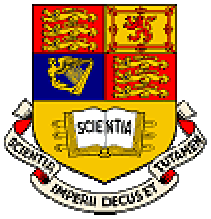


E 2.5 Language Processors

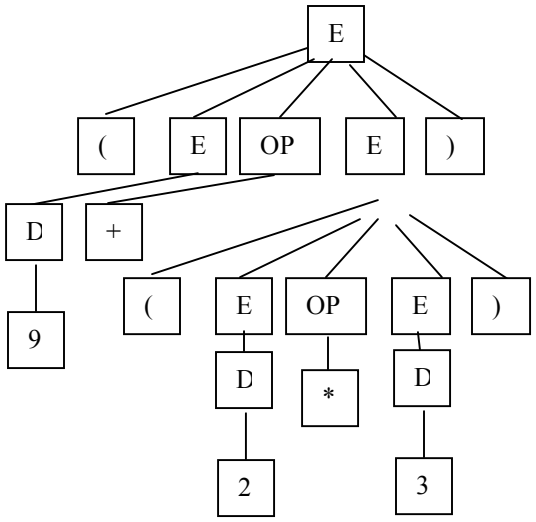
Department of Electrical & Electronic Engineering
Imperial College London

Language Processors Tutorial Exercises Set 1
SAMPLE SOLUTIONS

Grammars

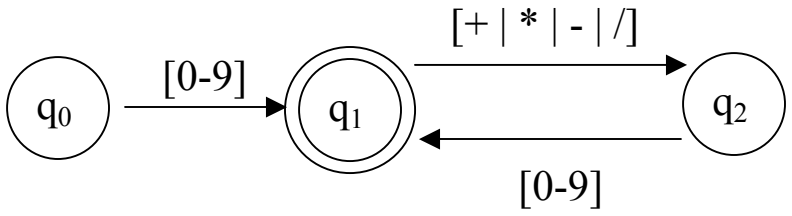
1. A grammar is:
   E -> D | (E OP E)
   OP -> + | - | * | /
   D -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
   (Note: There are more solutions of course)
2. The grammar is context-free (type-2) – its not regular since rule 1 is neither left- or right-linear
3. The derivation of (9+(2*3)) and the parse tree are shown below:

   E
   1@1 (E OP E)
   … (D OP E)
   (9 OP E)
   (9 + E)
   (9 + (E OP E))
   (9 + (D OP E))
   (9 + (D OP D))
   (9 + (3 OP D))
   (9 + (3 * D))
   (9 + (3 * 2))

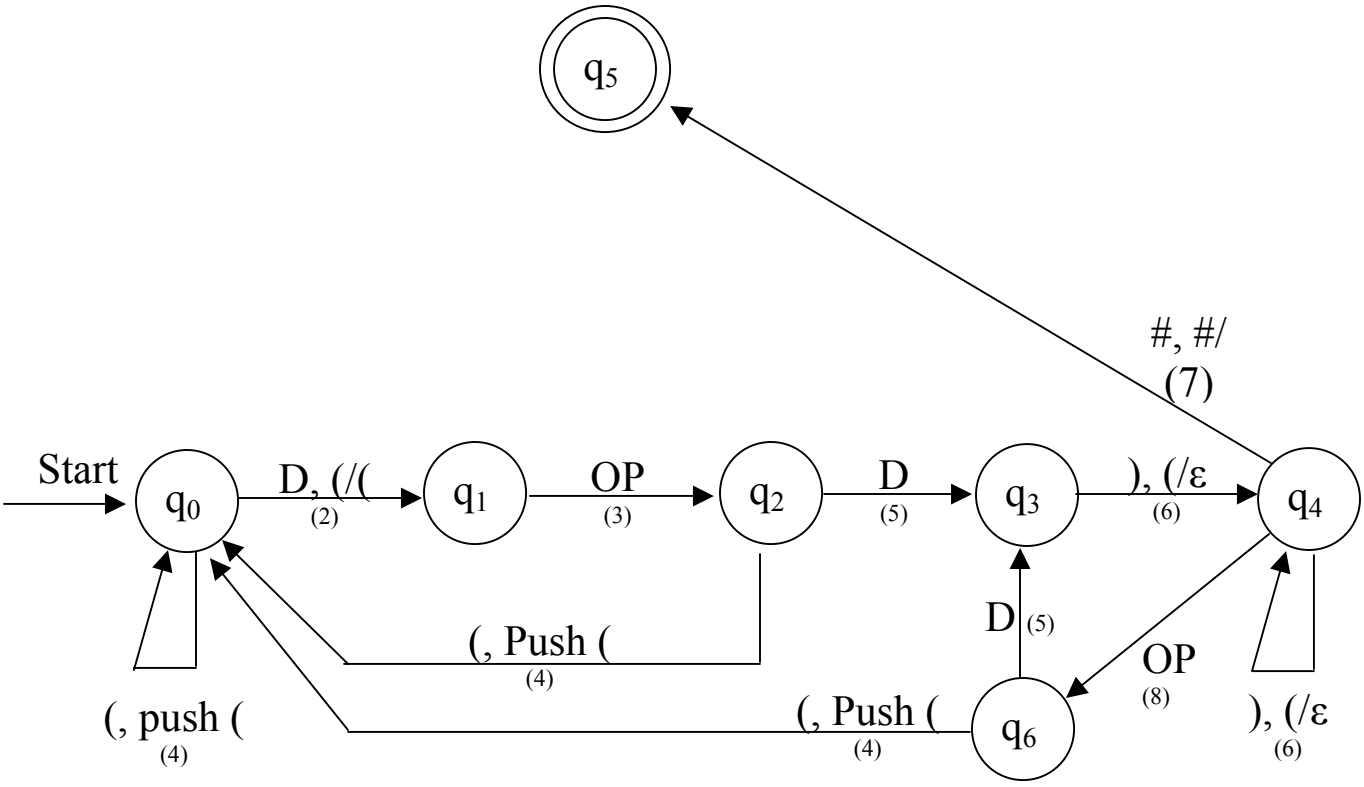


4. If we ignore parentheses, a regular grammar can be used:
   E -> D
   E -> D Rest
   Rest -> + E | - E | * E | / E
   D -> 0 | 1 | 2 | … | 9

5. The equivalent regular expression is [0-9]((+|-|*|/)[0-9])*
6. The finite automaton for q1:



7. The Push-down automaton for Q1 is:



Notes:

1. [# wrt the input indicates end of input, and wrt the stack, it indicates an empty stack – we start with (expression)# and we should enter an accepting state if we have #/# situation]
2. If we receive a digit and we have open parentheses, we consume the digit and move to q1
3. If we receive an operator (any of +,-,*,/), we consume it and move to q2 – the contents of the stack are not relevant and are not taken into account
4. We keep accumulating the parentheses; I use "push (" as an shortcut for all possibilities. i.e. (, #/(and (, (/((and (,)/)(
5. We consume the digit and move on; the contents of the stack are not relevant
6. For each closing parenthesis, we pop an opening one
7. If there is no more input, and we have nothing left on the stack, we move to final accepting state.
8. A closing parenthesis might be followed by another operator, bringing us to q_6