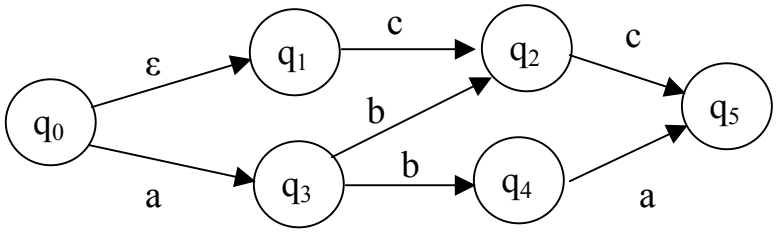
Language Processors Sample Exam Paper

Answer question 1 (40%) and 2 of questions 2-4 (30% each)

Question 1:

(a) Describe three different types of language processors [2]
(b) Describe the difference between type-0 and type-1 grammars in Chomsky's hierarchy of grammars [2]
(c) Convert the non-deterministic FA below to its equivalent DFA; explain your steps [4]



(d) Describe the difference between S- and L-attributed grammars [2]
(e) Consider the following grammar for arithmetic expressions:

E -> E + T | T
T -> T * F | F
F -> (E) | Digit
Digit -> [0..9]

Eliminate all instances of immediate left recursion, and write your derived grammar [6]

(f) Consider the augmented grammar:

- E' -> E
- E -> E + T | T
- T -> T * F | F
- F -> (E) | id

If K is the set of items {[E' -> .E]}, compute closure(K) [4]

Question 2:

Construct the DFA for the regular expression (a|b)*abb by:
(a) Constructing the NFA using Thompson's algorithm [10]
(b) Constructing the equivalent DFA using the subset construction algorithm**; Clearly explain your steps** [10]

Question 3:

(a) Construct a grammar [use BNF/EBNF syntax] that can generate the language containing the four simple arithmetic operations (addition, subtraction, division, multiplication) on single digits (no parentheses) using infix notation [e.g.: 9+2*5]. [10]
(b) Construct a finite state automaton that, given a string of the grammar above, will determine whether it is valid; clearly mark all final (accepting) states [10]

Question 4:

(a) Consider the grammar

- E -> E + E
- E -> E * E
- E -> (E)
- E -> id

Simulate the bottom up parsing of the input string 5+3*2, showing at each step the remaining input string, the contents of the stack, the action performed, and if it is a reduce action, the production rule used. Mark any S/R and R/R conflicts [10]

(b) Consider the following program fragment, which is to be compiled on a machine that has four registers R1, R2, R3 and R4

```
{
        temp_2ab = 2*a*b;
        temp_aa = a*a;
        temp_bb = b*b;
        x = temp_aa + temp_2ab + temp_bb;
        y = temp_aa – temp_2ab + temp_bb;
}
```

Assume that with respect to this block:

- a and b are live on entry to this block, and dead upon exit
- x and y are live upon exit.
- temp_aa, temp_2ab and temp_bb are dead upon exit

1. Draw the interference graph for the above program segment [5]
2. Using the register_allocation_by_graph_coloring method, allocate registers R1, R2, R3, R4 to the variables above. [5]