UNIVERSITY OF TORONTO

Department of Electrical and Computer Engineering

ECE467– Compilers and Interpreters

Midterm Examination

Fall 2019

Course Instructor: Pooja Vashisth

2019/10/18

Name: ____

UTOR ID: ____

Student Number: ______________________________________

This exam contains 12 pages (including this cover page) and two parts of 10 marks each. Total of points is 20. Write your answers within this booklet.

Good luck!

PART 1: 10 marks . All questions are of one mark each in this part.

1. Local and loop optimization in turn provide motivation for

a) Data flow analysis

b) Constant folding

c) Pee hole optimization

d) DFA and constant folding

2. The term environment in programming language semantics is understood as

a) Function that maps a name to value held there

b) Function that maps a name to storage location

c) The function that maps a storage location to the value held there

d) None of the above

3. The number of tokens in the following C statement is
printf ("i = %d, &i = %x", i, &i);

a) 3

b) 26

c) 10

d) 21

4. Whether a given pattern constitutes a token or not depends on the

a) Source language

b) Target language

c) Compiler

d) All of these

5. The languages that need heap allocation in the runtime environment are:

a) Those that use global variables

b) Those that use dynamic scoping

c) Those that support recursion

d) Those that dynamic data structure

6. Compiler translates the source code to

a) Executable code

b) Machine code

c) Binary code

d) Both B and C

7. A language is regular if and only if accepted by DFA.

(True/False)

8. Text Search is an application of Finite Automaton.

(True/False)

9. Given the language L = {ab, aa, baa}, which of the following strings are in L*?

1) abaabaaabaa

2) aaaabaaaa

3) baaaaabaaaab

4) baaaaabaa

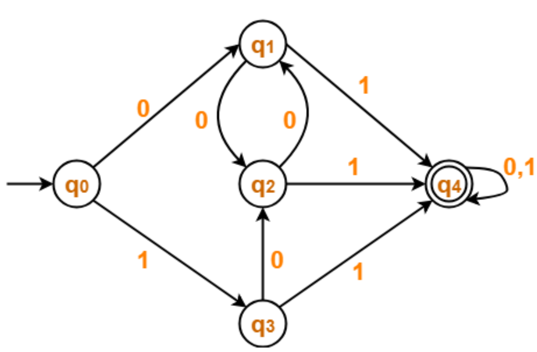
- a) 1, 2 and 3
- b) 2, 3 and 4
- c) 1, 2 and 4
- d) 1, 3 and 4

10. John is asked to make an automaton which accepts a given string for all the occurrence of '1001' in it. How many number of transitions would John use such that, the string processing application works?

- a) 9
- b) 11
- c) 12
- d) 15

PART 2 : 10 marks . All five questions are of two marks each in this part.

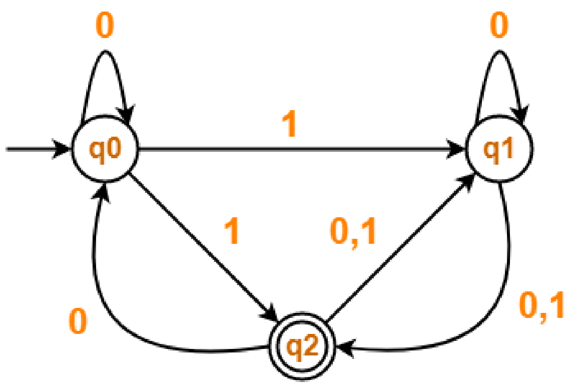
1. Minimize the given DFA using the Equivalence Theorem and write all the steps properly.



2. Given the alphabet {0,1} , write the regular expression that generates the language described by: *The set of all strings containing at least two 0's.*

3. Design NFA for the above regular expression and write all the steps properly.

4. Convert the following Non-Deterministic Finite Automata (NFA) to Deterministic Finite Automata (DFA) and write all the steps properly.



5. Divide the following C program:

```
{
  int a = 10, b = 20;
  printf("sum is :%d",a+b);
  return 0;
}
```

into appropriate lexemes. Which lexemes should get associated lexical values? What should those values be?

FORMULA CHEAT SHEET

TOKEN	INFORMAL DESCRIPTION	SAMPLE LEXEMES
if	characters `i`, `f`	`if`
else	characters `e`, `l`, `s`, `e`	`else`
comparison	`<` or `>` or `<=` or `>=` or `==` or `!=`	`<=`, `!=`
id	letter followed by letters and digits	`pi`, `score`, `D2`
number	any numeric constant	`3.14159`, `0`, `6.02e23`
literal	anything but `"`, surrounded by `"`'s	`"core dumped"`

Figure 3.2: Examples of tokens

Example 3.2 : The token names and associated attribute values for the Fortran statement

```
E = M * C ** 2
```

are written below as a sequence of pairs.

<**id**, pointer to symbol-table entry for `E`>
<**assign_op**>
<**id**, pointer to symbol-table entry for `M`>
<**mult_op**>
<**id**, pointer to symbol-table entry for `C`>
<**exp_op**>
<**number**, integer value 2>

OPERATION	DEFINITION AND NOTATION
Union of L and M	$L \cup M = \{s \mid s \text{ is in } L \text{ or } s \text{ is in } M\}$
Concatenation of L and M	$LM = \{st \mid s \text{ is in } L \text{ and } t \text{ is in } M\}$
Kleene closure of L	$L^* = \cup_{i=0}^{\infty} L^i$
Positive closure of L	$L^+ = \cup_{i=1}^{\infty} L^i$

Figure 3.6: Definitions of operations on languages

LAW	DESCRIPTION
$r \mid s = s \mid r$	$\mid$ is commutative
$r \mid (s \mid t) = (r \mid s) \mid t$	$\mid$ is associative
$r(st) = (rs)t$	Concatenation is associative
$r(s \mid t) = rs \mid rt;\ (s \mid t)r = sr \mid tr$	Concatenation distributes over $\mid$
$\epsilon r = r\epsilon = r$	ϵ is the identity for concatenation
$r^* = (r \mid \epsilon)^*$	ϵ is guaranteed in a closure
$r^{**} = r^*$	* is idempotent

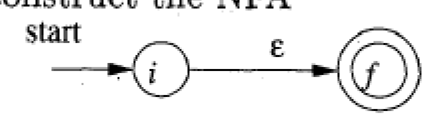
Figure 3.7: Algebraic laws for regular expressions

EXPRESSION	MATCHES	EXAMPLE
c	the one non-operator character c	`a`
$\backslash c$	character c literally	`\*`
"s"	string s literally	`"**"`
.	any character but newline	`a.*b`
^	beginning of a line	`^abc`
$	end of a line	`abc$`
[s]	any one of the characters in string s	`[abc]`
[^s]	any one character not in string s	`[^abc]`
$r*$	zero or more strings matching r	`a*`
$r+$	one or more strings matching r	`a+`
$r?$	zero or one r	`a?`
$r\{m,n\}$	between m and n occurrences of r	`a[1,5]`
r_1r_2	an r_1 followed by an r_2	`ab`
$r_1 \mid r_2$	an r_1 or an r_2	`a|b`
(r)	same as r	`(a|b)`
r_1/r_2	r_1 when followed by r_2	`abc/123`

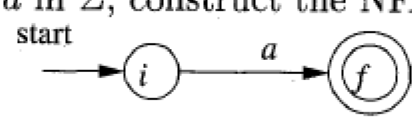
Figure 3.8: Lex regular expressions

The McNaughton-Yamada-Thompson algorithm to convert a regular expression to an NFA.

For expression ϵ construct the NFA



For any subexpression a in Σ, construct the NFA



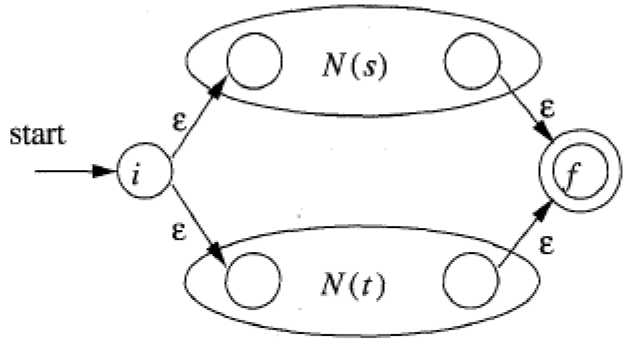


Figure 3.40: NFA for the union of two regular expressions

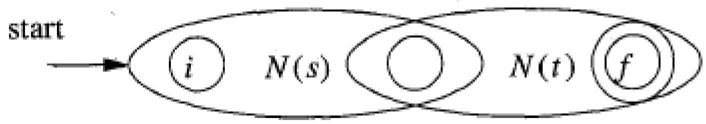


Figure 3.41: NFA for the concatenation of two regular expressions

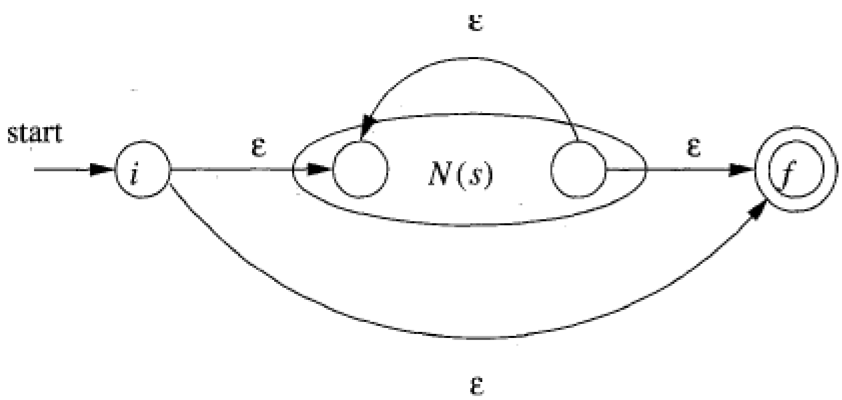


Figure 3.42: NFA for the closure of a regular expression