UNIVERSITY OF BRISTOL

January Examination Period

FACULTY OF ENGINEERING

Second Year Examination for the Degrees
of
Bachelor of Science
Master of Engineering

COMS20007J
Programming Languages and Computation

TIME ALLOWED:
3 Hours

This paper contains *three* questions, worth *45*, *35* and *20* marks respectively. Answer *all* questions. The maximum for this paper is *100 marks*. Credit will be given for partial answers.

Other Instructions:

You may bring a single A4 page (= 1 side of an A4 sheet) of your own notes with you, which you may consult freely during the examination.

YOU MAY START IMMEDIATELY

Q1. This question is about regular languages.

(a) Consider each of the following regular expressions over the alphabet $\Sigma = \{a, b, c\}$:

1. $((b+c)^*a(b+c)^*a(b+c)^*)^*$
2. $\Sigma^*abb\Sigma^*$
3. $\Sigma^*(abb + baa)\Sigma^*$
4. $b\Sigma^*b$
5. $(\Sigma\Sigma)^*$
6. $(b+c)^*$
7. $abb\Sigma^*$

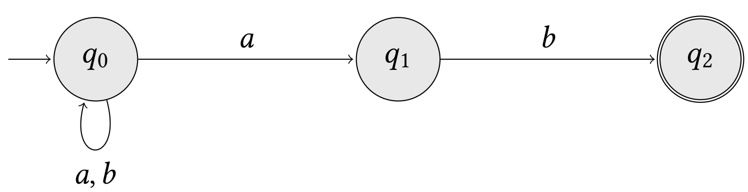
Match each of the following descriptions of languages to the regular expression above that denotes it:

i. The language of all words that start and end with b
ii. The language of all words that contain abb as a substring
iii. The language of all words that start with abb.
iv. The language of all words that do not contain a.
v. The language of all even length words.
vi. The language of all words containing an even number of a.
vii. The language of all words that either contain abb or bba

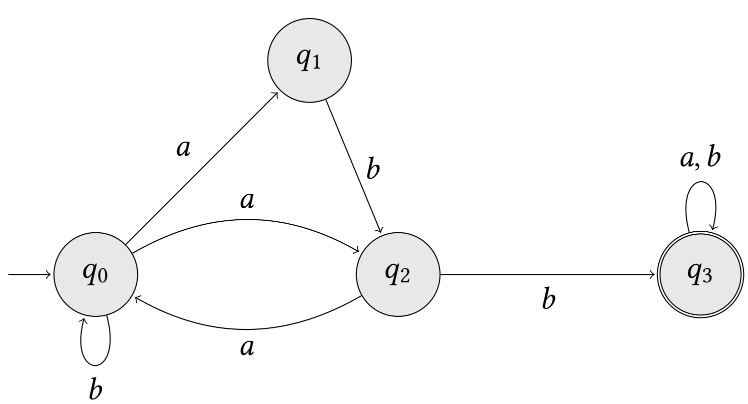
[7 marks]

(b) For each of the following automata, give a word that is accepted by the automaton *and* a *deterministic* automaton that recognises the same language.

i.



ii.



[10 marks]

(c) As in the Week 3 Problem Sheet, Question 9, we shall encode pairs of natural numbers by sequences of vectors of bits, with least significant bit first (left-most). Let Σ be the following set of binary vectors:

$$\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}$$

We use each word w over alphabet Σ to encode a pair of natural numbers, written $[[w]]$, which is defined by the following recursive function:

$$\begin{aligned} [[\epsilon]] &= (0, 0) \\ [[\begin{pmatrix}b_1\\b_2\end{pmatrix} \cdot w]] &= (2 * m + b_1,\ 2 * n + b_2) \\ & \qquad\qquad \textit{where } (m,\ n) = [[w]] \end{aligned}$$

For example:

$$[[\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}]] = (4,\ 13)$$
$$[[\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}]] = (26,\ 3)$$
$$[[\begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}]] = (1,\ 23)$$

For each of the following, give a finite state automaton that recognises the (encoding of the) language and has at most 3 states. For this problem, your automata should *not* accept the empty word.

i. $\{w \mid [[w]] = (n, m) \wedge n \leq m\}$
ii. $\{w \mid [[w]] = (n, m) \wedge m = 2 * m\}$

[8 marks]

(d) Let Σ be an alphabet and suppose $M_1 = (Q_1, \Sigma, \delta_1, p_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, p_2, F_2)$ are deterministic finite state automata. Describe a finite state automaton that recognises the following language:

$$\{a_1 b_1 a_2 b_2 \cdots a_n b_n \mid n \geq 0 \wedge a_1 a_2 \cdots a_n \in L(M_1) \wedge b_1 b_2 \cdots b_n \in L(M_2)\}$$

(i.e. those words of even length where concatenating the letters at even numbered positions yields a word accepted by M_1 and concatenating the letters at odd-numbered positions yields a word accepted by M_2). *[6 marks]*

(e) Let $\#_0(w)$ be the number of '0' letters in word w and $\#_1(w)$ be the number of '1' letters in word w. For example, $\#_0(01101) = 2$ and $\#_1(01101) = 3$.

Prove that the language $\{w \in \{0, 1\}^* \mid \#_0(w) = \#_1(w)\}$ is not regular.

[6 marks]

(cont.)

(f) Given two languages A and B over a common alphabet Σ, define:

$$A \blacktriangleleft B := \{w \in \Sigma^* \mid \exists v.\, wv \in A \wedge v \in B\}$$

Suppose A is regular. Show that there is a finite automaton recognising $A \blacktriangleleft B$ (irrespective of whether or not B is regular).

[8 marks]

Q2. This question is about the While language.

(a) For each of the following, indicate whether it is a syntactically valid Boolean expression in the While language. You may assume that x, y and z are variables.

i. `! true`

ii. `(!x) = true`

iii. `true && 1 = 1`

iv. `true && (false || !x=3)`

v. `x < y < z`

[5 marks]

(b) For each of the following arithmetic expressions a, give the number it evaluates to $[[a]]^{\mathcal{A}}([x \mapsto 3, y \mapsto 5])$ when evaluated in state $[x \mapsto 3, y \mapsto 5]$:

i. `23`

ii. `x + x`

iii. `(3 - x) + z`

iv. `5 * (x + y)`

v. `1 + y * z`

[5 marks]

(c) i. Consider the following While program.

```
n := 1
r := 0
while (n <= x) {
  n := 2 * n
  r := r + 1
}
```

Describe the 7th configuration in its execution trace starting in initial state $[\mathrm{x} \mapsto 4]$.

ii. Write a program in While that always terminates in a state where variable z has value x^y, where x (resp. y) is the initial value of variable x (resp. y). You may assume that we only run this program in initial states where y is non-negative.

iii. Consider the nth Fibonacci number fib(n), defined inductively as follows.

$$\mathrm{fib}(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ \mathrm{fib}(n-2) + \mathrm{fib}(n-1) & \text{otherwise} \end{cases}$$

Write a While program that computes the nth Fibonacci number for any given $n \in \mathbb{N}$. Your program should read the value of n from variable n in its initial state (you can assume it is always non-negative), and write the value of $\mathrm{fib}(n)$ into variable result in its final state. You can use as many additional variables as desired.

[15 marks]

(d) This question is about compiling arithmetic expressions to machine code. The abstract syntax for the machine language is slightly simpler than that for the While language, although it retains a recursive (inductive) definition since its control-flow is still structured.

Abstract machine instructions, typically C, are either:

- PUSH v whenever $v \in \mathbb{Z}$;
- LOAD x whenever x is a valid While variable identifier;
- ADD, SUB, or MUL

The set of abstract machine programs P, is the smallest set such that:

- ϵ, the empty program, is in P.
- $C\,;\,\pi$, the program whose first instruction is C and after that is program π, is in P whenever $\pi \in P$.

We will use π to stand for an arbitrary machine program.

The small-step semantics of the machine is specified over configurations $\langle \pi,\ s,\ \sigma \rangle$ composed of a machine language program π, a state σ (mapping variable names to values in $\mathbb{Z}$), and a stack of integer values, the top of which serves as working memory both for arithmetic operators and control-flow statements.

We use Haskell list notations for stacks (this means their top is denoted to the left), using a lowercase letter s to denote an abstract stack, and [] to denote an empty stack.

$$
\begin{array}{rcl}
\langle \mathtt{PUSH}\ v;\ \pi, s, \sigma \rangle & \rightarrow & \langle \pi, v : s, \sigma \rangle \\
\langle \mathtt{LOAD\ \ x};\ \pi, s, \sigma \rangle & \rightarrow & \langle \pi, \sigma(\mathrm{x}) : s, \sigma \rangle \\
\langle \mathtt{ADD};\ \pi, i_2 : i_1 : s, \sigma \rangle & \rightarrow & \langle \pi, (i_1 + i_2) : s, \sigma \rangle \text{ when } i_1, i_2 \in \mathbb{Z} \\
\langle \mathtt{SUB};\ \pi, i_2 : i_1 : s, \sigma \rangle & \rightarrow & \langle \pi, (i_1 - i_2) : s, \sigma \rangle \text{ when } i_1, i_2 \in \mathbb{Z} \\
\langle \mathtt{MUL};\ \pi, i_2 : i_1 : s, \sigma \rangle & \rightarrow & \langle \pi, (i_1 * i_2) : s, \sigma \rangle \text{ when } i_1, i_2 \in \mathbb{Z}
\end{array}
$$

Figure 1: Semantics for the machine's arithmetic instructions

i. Give the complete trace of the following program configuration:

$$\langle \mathtt{PUSH\ 3;\ LOAD\ \ x;\ ADD;}\ \epsilon,\ [],\ [x \mapsto 2] \rangle$$

ii. Construct a machine language program π such that:

$$\langle \pi,\ [],\ \sigma \rangle \rightarrow^* \langle \epsilon,\ [[\mathtt{1\ +\ (x\ *\ y)}]]^{\mathcal{A}}(\sigma) : [],\ \sigma \rangle$$

(cont.)

In other words, executing the machine language program π starting from an empty stack and in any state σ, yields a stack with one element, which is exactly the interpretation of $1 + (x * y)$ under σ.

iii. Define a function C from While arithmetic expressions to machine programs in such a way that, for all arithmetic expressions $a \in \mathcal{A}$:

$$\langle C(a),\ [],\ \sigma\rangle \to^* \langle \epsilon,\ [[a]]^{\mathcal{A}}(\sigma) : [],\ \sigma\rangle$$

In other words, executing the machine language program $C(a)$ starting from an empty stack and in any state σ, yields a stack with one element, which is exactly the interpretation $[[a]]^{\mathcal{A}}(\sigma)$ of a under σ.

You may find the following machine program concatenation operator $\oplus$ useful when defining this function. For all instructions $C_1, C_2, \ldots, C_n$ and $C'_1, C'_2, \ldots, C'_m$ it satisfies:

$$(C_1;\ C_2;\ \cdots;\ C_n;\ \epsilon) \oplus (C'_1;\ C'_2;\ \cdots;\ C'_m;\ \epsilon) = (C_1;\ C_2;\ \cdots;\ C_n;\ C'_1;\ C'_2;\ \cdots;\ C'_m;\ \epsilon)$$

[10 marks]

Q3. This question is about computability.

(a) A perfect square is a number which is of the form n^2 for some $n \in \mathbb{N}$. Show that the set

$$U = \{n \in \mathbb{N} \mid\ n \text{ is a perfect square }\}$$

is decidable.

[2 marks]

(b) Let $f : A \to B$ and $g : B \to C$. Show that if $g \circ f : A \to C$ is injective, then so is f.

[2 marks]

(c) Let us say that a state σ is 2022-*bounded* just if, for all variables x, $-2022 \le \sigma(x) \le 2022$.

One of the following two predicates P and Q is semi-decidable and the other is not. Determine which one is semi-decidable and justify your answer.

$$P = \{\ulcorner S \urcorner \mid \text{for all 2022-bounded } \sigma,\ S \text{ terminates when started in } \sigma\}$$
$$Q = \{\ulcorner S \urcorner \mid \text{there is 2022-bounded } \sigma \text{ and } S \text{ does not terminate when started in } \sigma\}$$

[8 marks]

(d) Show that the following predicate is undecidable:

$$P = \{\langle \ulcorner S_1 \urcorner, \ulcorner S_2 \urcorner \rangle \mid \text{for all } n \in \mathbb{N}\text{: } [\![S_1]\!]_x(n) \simeq 1 \text{ iff } [\![S_2]\!]_x(n) \simeq k \text{ where } k \neq 1\ \}$$

[8 marks]

END OF PAPER