VICTORIA UNIVERSITY OF WELLINGTON

Te Whare Wananga o te Upoko o te Ika a Maui

EXAMINATIONS — 2004

TERM TEST

COMP 202

Formal Methods
of Computer Science
WITH ANSWERS

Time Allowed: 90 minutes

Instructions: There are **four** (4) questions.
Each question is worth **twenty-five** (25) marks.
Answer **all** the questions.
Show **all** your working.

Question 1.

The following while-program takes as input an integer x and a sequence $a[0..n]$ of integers. Subject to certain constraints on the inputs, it will return an integer i such that $a[i] = x$, if such an i exists, and $i = -1$ otherwise.

procedure *Find* (**in** x, a; **out** i) ;
    begin
$\{I_1\}$ $i := 0$;
$\{I_2\}$ **while** $a[i] < x$ **do**
$\{I_3\}$ $i := i + 1$
$\{I_4\}$ **od**;
$\{I_5\}$ **if** $a[i] = x$ **then**
$\{I_6\}$ **skip**
    else
$\{I_7\}$ $i := -1$
    fi
$\{I_8\}$ **end**

The program may be proved correct using the loop invariant

$$I_2 \triangleq (\forall j \in 0..i-1)a[j] < x$$

(a) State what properties a and x must have in order that the program behaves as described. [3 marks]

a is non-descending; x is less than or equal to the value of the last element of a.

(b) Write a formal specification (signature, precondition, and postcondition) that this program satisfies. [6 marks]

Input: integer x, integer sequence a
Output: integer i
Precondition: $I_1 \triangleq (\forall i \in 0..n-1)(a[i] \leq a[i+1]) \wedge x \leq a[n]$
Postcondition: $I_8 \triangleq x = a[i] \vee (i = -1 \wedge (\forall j \in 0..n)a[j] \neq x)$

(c) State an assertion I_5 that may be used to verify the program. **Hint**: What property of i does the **if** statement assume? What does the loop guarantee about the values of $a[j]$ for $j < i$? $j > i$? [4 marks]

$I_5 \triangleq x \leq a[i] \wedge (\forall j \in 0..i-1)a[j] < x$

(d) Show that the **if** statement is correct: that is, find assertions I_6 and I_7 such that I_6 implies I_8 and $I_7 \wedge i = -1$ implies I_8. [6 marks]

$I_6 \triangleq I_5 \wedge a[i] = x$, which implies I_8.
$I_7 \triangleq I_5 \wedge a[i] \neq x$, which implies $(\forall j \in 0..n)a[j] \neq x$ because a is in non-descending order. Together with $i = -1$, this implies I_8.

(e) State the **three** things that must be proved to verify the loop. [6 marks]

loop invariant holds initially: Assume $i = 0$; show I_2.
[[Proof: Trivial $\forall$]]

loop invariant maintained by body: Assume I_2 and $a[i] < x$; show I_2 with $i + 1$ in place of i.
[[Proof: $(\forall j \in 0..i-1)a[j] < x \wedge a[i] < x$ implies $(\forall j \in 0..i)a[j] < x$]]

postcondition holds on termination: assume I_2 and $a[i] \geq x$; show I_5.
[[Proof: immediate]]

Question 2.

(a) Let $\Sigma = \{a, b\}$. Give regular expressions which describe the following languages over Σ:

(i) All strings $\boxed{(\mathbf{a}+\mathbf{b})^* \text{ or } (\mathbf{a}^*\mathbf{b}^*)^* \text{ amongst others}}$

(ii) All strings of length less than 3 $\boxed{\mathbf{\Lambda}+\mathbf{a}+\mathbf{b}+(\mathbf{a}+\mathbf{b})(\mathbf{a}+\mathbf{b}) \text{ is one solution}}$

(iii) All strings which contain either bb or aa $\boxed{(\mathbf{a}+\mathbf{b})^*(\mathbf{aa}+\mathbf{bb})(\mathbf{a}+\mathbf{b})^*}$

[5 marks]

(b) State Kleene's Theorem. [6 marks]

A language is regular *iff* it is accepted by a FA.
A language is regular *iff* it is accepted by an NFA.
A language is regular *iff* it is described by a regular expression.
A language is regular *iff* it is generated by a regular grammar.

(c) Let L_1 and L_2 be regular languages. Show that the following languages are also regular:

(i) L_1^*

If L_1 is regular then by Kleene's theorem there is a regular expression $\mathbf{r}$ such that $L_1 = \text{Language}(\mathbf{r})$. Then $L_1^* = \text{Language}(\mathbf{r}^*)$. So L_1^* is described by a regular expression, so, by Kleene's theorem L_1^* is regular. A similar argument involving machines would do.

(ii) L_1L_2

If L_1 and L_2 are regular then by Kleene's theorem there are regular expressions r and s such that $L_1 = \text{Language}(\mathbf{r})$ and $L_2 = \text{Language}(\mathbf{s})$. Then $L_1L_2 = \text{Language}(\mathbf{rs})$. So L_1L_2 is described by a regular expression, so, by Kleene's theorem L_1L_2 is regular. A similar argument involving machines would do.

(iii) $L_1 + L_2$

If L_1 and L_2 are regular then by Kleene's theorem there are regular expressions r and s such that $L_1 = \text{Language}(\mathbf{r})$ and $L_2 = \text{Language}(\mathbf{s})$. Then $L_1 + L_2 = \text{Language}(\mathbf{r}+\mathbf{s})$. So $L_1 + L_2$ is described by a regular expression, so, by Kleene's theorem $L_1 + L_2$ is regular. A similar argument involving machines would do.

[6 marks]

(d) Let M_1 and M_2 be NFA's which accept the languages L_1 and L_2. Explain how to construct an FA which accepts the language consisting of strings which are in neither L_1 nor L_2. [8 marks]

We are asked to construct a FA which accepts some language, given some NFA's.
First, what is the language? Strings which are in neither L_1 nor L_2 are in the complement of the union of L_1 and L_2, i.e. we are being asked to find an FA which accepts $\overline{L_1 + L_2}$.
Given the NFA's M_1 and M_2 we can construct NFA's with Λ transitions $M_{1'}$ and $M_{2'}$ which accept L_1 and L_2 respectively. The only change is a trivial one to the transition function.
Next we create a new NFA with Λ transitions M_3 which accepts $L_1 + L_2$. The machine M_3 is created from $M_{1'}$ and $M_{2'}$, by adding a new start state, with Λ transitions to the start states of $M_{1'}$ and $M_{2'}$. Accepting states of $M_{1'}$ and $M_{2'}$ become accepting states of M_3.
Then we turn M_3 into an FA M_4, probably by converting it to an intermediate NFA. We are careful to check that the transition function for M_4 is total. (It will be unless we deliberately remove the "black hole" state.) So M_4 is an FA which accepts $L_1 + L_2$.
Now we construct M_5 from M_4 by making the accepting states of M_5 be the complement of the accepting states of M_4. So M_5 an FA which accepts $\overline{L_1 + L_2}$

Question 3.

(a) A finite automaton (FA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$. Describe

- Q, A finite set of states.
- Σ, A finite alphabet of symbols.
- δ,

  A function $Q \times Sigma \to Q$. The transition function. Describes how to move from state to state on reading a symbol.
- q_0, and An element of Q, the start state.
- F A subset of Q, the accepting states.

[5 marks]

(b) Explain how an FA defines a language. [3 marks]

The set of strings accepted by the FA is the language it defines. A string is accepted if it labels a path from the start state to an accepting state of the machine.

(c) A generalised non-deterministic finite automaton (GNFA) is a 5-tuple $(Q, \Sigma, \delta, q_s, q_f)$. Describe

- Q, A finite set of states.
- Σ, A finite alphabet of symbols.
- δ, A function $Q - q_f \times Q - q_s \to \mathrm{RegularExpression}$. The transition function. Describes how to move from state to state on reading a string.
- q_s, and An element of Q, the start state.
- q_f An element of Q, the accepting state.

[5 marks]

(d) Explain how a GNFA defines a language. [3 marks]

The set of strings accepted by the GNFA is the language it defines. A string is accepted if it labels a path from the start state to the accepting state of the machine.

(e) Let M be a finite automaton. Explain how to find a regular expression which describes the language accepted by M. [9 marks]

Two steps:

1. convert M to a GNFA N which accpets the same language as M
2. reduce N to a two state GNFA which accpets the same language as N

1: Construct N from M. Add two new states q_s and q_f. Add transitions labelled Λ from q_s to the start state of M and from the accepting states of M to q_f. If there is any state in M with transitions to any state of M on more than one symbol replace this with a single transition labelled with the sum of the symbols. Now add sufficient transitions to ensure that there is a transition from every state in $Q - q_f$ to every state in $Q - q_s$. Label these new transitions with $\emptyset$.
2: Remove states from N until only two are left. Pick a state S_{rem} (not q_s or q_f) to remove. For every pair of states S_i and S_j in the reduced machine the transition between them will be labelled $\delta(S_i, S_j) + \delta(S_i, S_{\text{rem}})\delta(S_{\text{rem}}, S_{\text{rem}})^*\delta(S_{\text{rem}}, S_j)$, where δ is the transition function of the machine being reduced.
When only two states remain the regular expression accepted by the original FA is $\delta(q_s, q_f)$

Question 4.

(a) Let M_3 be an NFA with Λ transitions $(Q, \Sigma, \delta, q_0, F)$, where:

$$\begin{aligned} Q &= \{S_0, S_1, S_2, S_3\} \\ \Sigma &= \{a, b\} \\ q_0 &= S_0 \\ F &= \{S_3\} \end{aligned}$$

δ	a	b	Λ
S_0	$\{S_0, S_1\}$	$\{S_2\}$	$\{S_3\}$
S_1	$\{S_1, S_2\}$	$\{\}$	$\{S_1\}$
S_2	$\{S_3\}$	$\{S_1, S_2\}$	$\{\}$
S_3	$\{S_2\}$	$\{S_1\}$	$\{S_0\}$

(i) Draw M_3. [4 marks]

(ii) Find $M_4 = (Q', \Sigma', \delta', q_0', F')$, an NFA which accepts the same language as M_3.
[8 marks]

Key points: M_4 has same states, alphabet, start state as M_3. Accepting states of M_4 are those of M_3 plus S_0, as there is a Λ transition from S_0 to S_3 (i.e. M_3 accepts Λ. Transiton function altered to replace paths which consist of a combination of Λs and single symbol σ by arcs labelled only by σ.

$$\begin{aligned} Q &= \{S_0, S_1, S_2, S_3\} \\ \Sigma &= \{a, b\} \\ q_0 &= S_0 \\ F &= \{S_0, S_3\} \end{aligned}$$

δ	a	b
S_0	$\{S_0, S_1, S_2, S_3\}$	$\{S_1, S_2\}$
S_1	$\{S_1, S_2\}$	$\{\}$
S_2	$\{S_0, S_3\}$	$\{S_1, S_2\}$
S_3	$\{S_0, S_1, S_2\}$	$\{S_1, S_2\}$

(b) Let M_5 be an NFA $(Q, \Sigma, \delta, q_0, F)$, where:

$$\begin{aligned} Q &= \{S_0, S_1, S_2, S_3\} \\ \Sigma &= \{a, b\} \\ q_0 &= S_0 \\ F &= \{S_0\} \end{aligned}$$

δ	a	b
S_0	$\{S_1, S_2\}$	$\{\}$
S_1	$\{S_3\}$	$\{S_0, S_2\}$
S_2	$\{S_3\}$	$\{\}$
S_3	$\{S_3\}$	$\{S_0\}$

(i) Draw M_5. [4 marks]

(ii) Find $M_6 = (Q', \Sigma', \delta', q_0', F')$, an FA which accepts the same language as M_5. [9 marks]

Key points: Alphabet remains the same. States of M_6 will be sets of states of M_5. Start state of M_6 will be set containing just S_0. Accepting states of M_6 will be states which contain an accepting states of M_5. Construct the states of M_6 and its transition function in parallel by staring from $\{S_0\}$, and generating new states from $\delta(S_0, \sigma), \sigma \in \Sigma$. Continue the process until no new states are generated.

$$\begin{aligned}
Q' &= \{\{S_0\}, \{S_1, S_2\}, \{\}, \{S_3\}, \{S_0, S_2\}, \{S_1, S_2, S_3\}\} \\
\Sigma' &= \{a, b\} \\
q_0' &= \{S_0\} \\
F' &= \{\{S_0\}, \{S_0, S_2\}\}
\end{aligned}$$

δ	a	b
$\{S_0\}$	$\{S_1, S_2\}$	$\{\}$
$\{S_1, S_2\}$	$\{S_3\}$	$\{S_0, S_2\}$
$\{\}$	$\{\}$	$\{\}$
$\{S_3\}$	$\{S_3\}$	$\{S_0\}$
$\{S_0, S_2\}$	$\{S_1, S_2, S_3\}$	$\{\}$
$\{S_1, S_2, S_3\}$	$\{S_3\}$	$\{S_0, S_2\}$

The language involved is $\mathrm{Language}((\mathbf{aa}^*\mathbf{b})^*)$.
The minimal FA (with a total transition function) which accepts this language is:

$$\begin{aligned}
Q'' &= \{T_0, T_1, T_2\} \\
\Sigma'' &= \{a, b\} \\
q_0'' &= T_0 \\
F'' &= \{T_0\}
\end{aligned}$$

δ	a	b
T_0	T_1	T_2
T_1	T_1	T_0
T_2	T_2	T_2

****