VICTORIA UNIVERSITY OF WELLINGTON

Te Whare Wananga o te Upoko o te Ika a Maui

EXAMINATIONS — 2001

END OF YEAR

COMP 202

Formal Methods of Computer Science

Time Allowed: 3 Hours

Instructions: Candidates should attempt **all** questions.

This exam will be marked out of 100.

Non-electronic foreign language dictionaries are permitted.

Question 1. Regular Languages. [34 marks]

(a) Write regular expressions that define the following languages:

(i) All strings of a's and b's such that all runs of a's are of even length. For example, strings such as *bbbaabaaaa*, *aaaabb*, and ϵ are in the language; *abbabaa* and *aaa* are not.

(ii) Strings of 0's and 1's having even parity, that is, an even number of 1's.

Hint: Think of even-parity strings as the concatenation of elementary strings with even parity, either a single 0 or a pair of 1's separated only by 0's.

[6 marks]

(b) Let Σ be the alphabet $\{a, b\}$.

Let $L \subset \Sigma^*$ be the language defined by the regular expression $a^*b^* \mid b^*a^*$.

(i) Design a DFA that recognises L. [4 marks]

(ii) Design a DFA that recognises the set of strings in Σ^* that form the complement of L, that is, $\Sigma^* - L$. [3 marks]

(c) Consider the following transition table.

		0	1
$\rightarrow p$	F	s	p
q		p	s
r		r	q
s		q	r

(i) Draw the transition diagram for the DFA corresponding to this table. Note that $\rightarrow$ signifies the start state, F signifies an accept state. [2 marks]

(ii) Apply the state elimination technique to this DFA to produce a regular expression defining the set of strings accepted by the DFA. Show all your working. [5 marks]

(d) Consider the regular expression $(a|b)^*b$.

(i) Convert this regular expression to an ϵ-NFA, using the bottom-up, inductive approach given in lectures. Present your solution as a transition diagram. [3 marks]

(ii) Number the states in your ϵ-NFA and use the subset construction to produce an equivalent DFA.

Present your solution as a transition diagram, showing clearly which subset of states from the ϵ-NFA corresponds to each state of the DFA. [5 marks]

(e) Let L be the regular language defined by the regular expression $(a|b)^*baaa$.

The strings in this set $\{\epsilon, b, ba, baa, baaa\}$ are all pairwise distinguishable with respect to L.

(i) Explain what it means for two strings to be *distinguishable* with respect to L.

(ii) Explain why L cannot be recognised by any DFA with fewer than five states.

[6 marks]

Question 2. Context Free Languages. [20 marks]

(a) Consider the context-free grammar

$$\begin{array}{lcl} S & \rightarrow & AB \\ A & \rightarrow & S \mid aA \mid aC \mid aa \\ B & \rightarrow & \epsilon \mid b \\ D & \rightarrow & \epsilon \mid d \end{array}$$

For each of the following constructs; give both a brief description, and an illustrative example using the context-free grammar shown above.

(i) undefined non-terminal symbol

(ii) useless symbol

(iii) circular definition

[6 marks]

(b) Consider the context-free grammar

$$S \rightarrow aS \mid aSbS \mid \epsilon$$

This *grammar* is ambiguous. Demonstrate this by showing two different parse trees for the string aab. [4 marks]

(c) Explain what it means for a context-free *language* to be ambiguous. [4 marks]

(d) Design a context-free grammar for the set $PAREN_2$ of balanced strings of parentheses of two types, ( ) and [ ]. For example, ([( ) [ ]] ([ ])) is in $PAREN_2$, but [( ]) is not.

Note that $PAREN_2$ is the smallest set of strings such that:

- $\epsilon \in PAREN_2$;
- if $x \in PAREN_2$, then so are (x) and $[x]$; and
- if x and y are in $PAREN_2$, then so is xy.

[6 marks]

Question 3. Context Free Grammars and Parsing. [22 marks]

The Hypertext Markup Language (HTML) has two major functions: describing the format of a document and creating links between documents. An HTML document consists of ordinary text interspersed with tags. The tags tell us something about the semantics of various strings within the document.

Here is a context-free grammar that describes some of the structure of the HTML language:

$$\begin{array}{lcl}
\textit{Doc} & \rightarrow & \textit{Element}\ \ \textit{Doc}\ \mid \epsilon \\[1em]
\textit{Element} & \rightarrow & \textbf{Text} \mid \\
 & & <\textbf{P}>\ \textbf{Text} \mid \\
 & & <\textbf{P}>\ \textbf{Text}\ </\textbf{P}> \mid \\
 & & <\textbf{OL}>\ \ \textit{List}\ \ </\textbf{OL}> \mid \cdots \\[1em]
\textit{List} & \rightarrow & \textit{List}\ \ \textit{ListItem}\ \mid \epsilon \\[1em]
\textit{ListItem} & \rightarrow & <\textbf{LI}>\ \ \textit{Doc}\ \mid \\
 & & <\textbf{LI}>\ \ \textit{Doc}\ \ </\textbf{LI}>
\end{array}$$

The strings in italics represent non-terminal symbols in the grammar. There are four non-terminal symbols:

$$\{\ \textit{Doc}\ ,\ \textit{Element}\ ,\ \textit{List}\ ,\ \textit{ListItem}\ \}$$

The start symbol is *Doc*.

The boldface strings represent terminal symbols in the grammar. We assume the availability of a scanner, recognising HTML tags as terminal symbols, and recognising blocks of text not containing any HTML tags as corresponding to the abstract terminal **Text**. Thus, there are seven terminal symbols:

$$\{<\textbf{P}>, </\textbf{P}>, <\textbf{OL}>, </\textbf{OL}>, <\textbf{LI}>, </\textbf{LI}>, \textbf{Text}\}$$

(a) Explain what it means for a context-free grammar to be in LL(1) form, and show that the grammar given here is not in LL(1) form. [4 marks]

(b) Rewrite the grammar so that it is in LL(1) form, introducing any new non-terminals required. [6 marks]

(c) Show that your grammar is in LL(1) form by constructing the necessary *first* and *follow* sets. [6 marks]

(d) Draw the parse tree that would be produced by your grammar for the following piece of HTML text:

< **P** > A short list of things that I like:
< **OL** >
< **LI** > Fresh bread.
< **/OL** > [6 marks]

Question 4. Program Equivalence. [12 marks]

(a) Consider the following while program fragments:

```
if B₁ then              if B₁ then
  S₁                      S₁
else                      while B₂ do
  while B₂ do               S₂
    S₂                    od
  od                    else
fi                        if B₂ then
                            S₂
                          fi
                        fi
```

(i) Write a regular expression denoting the set of all potential execution paths for each of these program fragments (use s_1 to denote the set of paths through S_1, s_2 to denote the set of paths through S_2). [4 marks]

(ii) Decide whether the two program fragments are weakly equivalent, strongly equivalent, or neither weakly nor strongly equivalent. Give reasons for your conclusion. [2 marks]

(b) Many programming languages define Boolean operators in terms of conditional expressions, as follows:

$$\begin{aligned}
p \wedge q &\overset{\text{def}}{=} \textbf{if } p \textbf{ then } q \textbf{ else } \textit{false} \textbf{ fi}\\
p \vee q &\overset{\text{def}}{=} \textbf{if } p \textbf{ then } \textit{true} \textbf{ else } q \textbf{ fi}\\
p \Rightarrow q &\overset{\text{def}}{=} \textbf{if } p \textbf{ then } q \textbf{ else } \textit{true} \textbf{ fi}\\
p \equiv q &\overset{\text{def}}{=} \textbf{if } p \textbf{ then } q \textbf{ else } \neg q \textbf{ fi}
\end{aligned}$$

The way that most compilers translate constructs involving Boolean operators is based on the above definition. For example, if we allow Boolean operators in while programs but not in flowchart programs, we might translate an **if** statement

$$\textbf{if } P \wedge Q \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}$$

as:

```
     if P then 1 else 3;
1 :  if Q then 2 else 3;
2 :   skip;
     T_wf(S₁);
      goto 4;
3 :   skip;
     T_wf(S₂);
4 :   skip;
```

Give similar translations for the following statements:

(i) **if** $P \equiv Q$ **then** S_1 **else** S_2 **fi**

(ii) **while** $P \wedge Q$ **do** S **od**

[6 marks]

Question 5. Specification and Verification. [12 marks]

(a) Write a formal specification for the following problem.

Input: (A, n), where A is an odd-length list of distinct natural numbers (i.e. no number occurs twice in A), and n is the length of A.

Output: (B, z), where z is the median element in the list A (i.e. z is in A and there are exactly the same number of elements in A less than z as there are greater than z), and B is a list of integers, having the same length as A; each element of B represents the difference between z and the corresponding element of A.
[4 marks]

(b) The following is an algorithm to decide whether a particular value x is contained in an array A.

A precondition is given immediately after the input statement. A postcondition is given immediately before the output statement.

```
input $A, n, x$;
{ $(1 \leq n) \wedge (n = |A|)$ }
$j := 1$;
$b := false$;
while $j \neq (n + 1)$ do
    if $b$ then
        skip
    else
        if $A[j] = x$ then $b := true$; fi
    fi;
    $j := j + 1$;
od;
{ $b \Leftrightarrow (\exists i \in \{1, ..., n\})\ A[i] = x$ }
output b;
```

(i) Explain what is meant by the term "loop invariant" and how a loop invariant can be used in proving that a loop satisfies a given specification. [2 marks]

(ii) Consider the following possibility for a loop invariant for this algorithm:

$$(b \wedge (\exists i \in \{1, ..., j\})\ A[i] = x) \ \vee\ \neg b$$

Explain why this is an unsuitable choice as a loop invariant to use to verify the loop. [3 marks]

(iii) Give a loop invariant that could be used to verify the loop. [3 marks]

****