PAPER CODE NO. COMP513	EXAMINER : **Professor M Fisher** DEPARTMENT : Computer Science Tel. No. **46701/7663**

THE UNIVERSITY
of LIVERPOOL

SUMMER 2002 EXAMINATIONS

Master of Science : Year 1
Master of Science : Year 2

FORMAL METHODS

TIME ALLOWED : Two Hours and a Half

INSTRUCTIONS TO CANDIDATES

Answer ***four*** questions only

If you attempt to answer more than the required number of questions (in any section), the marks awarded for the excess questions will be discarded (starting with your lowest mark).

THE UNIVERSITY
of LIVERPOOL

1. This question concerns the basic structures used within Z specifications.

   (a) What is the representation of the sequence

   $$\langle 1, 3, 5, 7, 9 \rangle$$

   when given in terms of a function (i.e., as a set of maplets)? **[3]**

   (b) What is the representation of the bag

   $$\llbracket pig, cow, horse, hen, pig, horse, duck \rrbracket$$

   when given in terms of a function (i.e., as a set of maplets)? **[3]**

   (c) Given $c : \mathbb{P}\{a, b, d, g\}$, write down all the values c can possibly have. **[4]**

   (d) f is a function from $\{1, 2, 3, 4\}$ to $\{a, b, c\}$ that is defined as:

   $$f == \{1 \mapsto a, 2 \mapsto b, 3 \mapsto a, 4 \mapsto c\}$$

   Is f surjective or injective, and why? **[5]**

   (e) What is the value of

   $$\langle a, b, r, a, c, a, d, a, b, r, a \rangle \upharpoonright (\{a, b, c\} \setminus (\mathrm{dom}\{a \mapsto 2, d \mapsto 4\}))$$

   Explain your answer. **[5]**

   (f) If $B == \llbracket k, c, a, j, k, c, a, l, b \rrbracket$, then what is the value of

   $$(B \oplus \{c \mapsto 5\}) \setminus \{j \mapsto 1\}$$

   [5]

2. We wish to specify the relationship between exam marks and students, and already have the following state space schema (N.B., *PERSON* is the set of all people):

$$\begin{array}{|l}
\textit{MarkRecord} \\
\hline
\textit{students} : \mathbb{P}\,\textit{PERSON} \\
\textit{marks} : \textit{PERSON} \pfun 0 \mathinner{\ldotp\ldotp} 100 \\
\hline
\operatorname{dom} \textit{marks} \subseteq \textit{students} \\
\hline
\end{array}$$

(a) Write a Z specification for the operation

$$\textit{AddStudent}(\textit{name}? : \textit{PERSON})$$

which adds a new student (i.e. *name*?) to *MarkRecord*, but does not assign the student a mark. **[5]**

(b) Write a Z specification for the operation

$$\textit{AddMark}(\textit{name}? : \textit{PERSON}, \textit{mark}? : 0 \mathinner{\ldotp\ldotp} 100)$$

which assigns the mark (*mark*?) to the student (*name*?) in the *MarkRecord*. **[5]**

(c) Write a Z specification for the operation

$$\textit{CheckMark}(\textit{name}? : \textit{PERSON}, \textit{mark}! : 0 \mathinner{\ldotp\ldotp} 100)$$

which returns the mark (*mark*!) associated with the student (*name*?).

Note: the operation should be undefined if the given student has not already been assigned a mark. **[5]**

(d) Write a Z specification for the operation

$$\textit{Unmarked}(\textit{names}? : \mathbb{P}\,\textit{PERSON})$$

which returns the set of students that have not yet been assigned marks. **[5]**

(e) How would you modify the *CheckMark* operation above so that it is robust (i.e. it will be defined for any student name supplied)? Assume that a *REPORT* type exists for reporting errors. **[5]**

3. This question concerns temporal logic.

   (a) What type of structure is typically used to provide a model for propositional, discrete, linear temporal logic, with finite past, and why? **[3]**

   (b) Does the temporal formula

   $$(l \land a)\,\mathcal{U}\,(l \land\ (l \land b)\,\mathcal{U}\,c)$$

   imply $l\,\mathcal{U}\,c$? Explain your answer. **[5]**

   (c) Does the temporal formula

   $$\Box(p \Rightarrow \Diamond q) \land \Box p$$

   imply $\Box q$? Explain your answer. **[5]**

   (d) Consider the axiom

   $$(p \land\ \Box(p \Rightarrow \bigcirc p))\ \Rightarrow\ \Box p$$

   Translate this to classical first-order logic (with arithmetic) and explain what simple principle the above axiom characterises. **[8]**

   (e) How do *branching* temporal logics differ from linear temporal logics, and what additional operators do they typically provide? **[4]**

4. Below is a temporal specification for a simple message-passing system consisting of two components, A and B.

$$Spec_A\colon \Box \left[\begin{array}{rrcl} & \textbf{start} & \Rightarrow & p \\ \wedge & p & \Rightarrow & \bigcirc q \\ \wedge & q & \Rightarrow & \bigcirc p \\ \wedge & q & \Rightarrow & \bigcirc send\_msg \end{array}\right] \qquad Spec_B\colon \Box \left[\begin{array}{rrcl} & rcv\_msg & \Rightarrow & \bigcirc g \\ \wedge & f & \Rightarrow & \bigcirc g \\ \wedge & g & \Rightarrow & \bigcirc f \end{array}\right]$$

(a) What is the behaviour of $Spec_A$, i.e. how often is $send\_msg$ made true? **[7]**

(b) In

$$Spec_A \wedge Spec_B \wedge \Box[send\_msg \Rightarrow rcv\_msg]$$

what is the last formula meant to specify? **[3]**

(c) If we wish to specify that a message send will be followed, at some time in the future, by a message receipt, what formula should we modify in the above specification and what should it be changed to? **[5]**

(d) What is a *safety* property, and what general form of temporal formulae characterise such properties? **[5]**

(e) What is a *liveness* property, and what general form of temporal formulae characterise such properties? **[5]**

5. This question concerns the foundations of model checking.

(a) Given a finite state structure, M, represented as a finite-state automaton, and a temporal formula, φ, how would we use the *automata-theoretic* approach to model checking to establish $M \models \varphi$? **[10]**

(b) What is *on the fly* model checking, and why might it be beneficial? **[7]**

(c) Describe two current problems with the model checking approach in general. **[8]**

6. Consider the following Promela code describing a three process system where:

- process A sends information to process B via channel a2b,
- process B sends information to process C via channel b2c, and
- process C sends information to process A via channel c2a.

```
proctype A (chan in, out)
{
        int total;
        total = 0;                          /* initial state */
S1:     total = (total+1)%8;
        out!total;
        printf("A sent %d\n", total);
        in?total;
        assert(total != 1);                    /* assertion */
        printf("A received %d\n", total);
        if
        :: (total != 0) -> goto S1;
        :: (total == 0) -> out!total
        fi }

proctype B (chan in, out)
{
        int total;
S1:     in?total;
        printf("B received %d\n", total);
        if
        :: (total != 0) -> total = (total+1)%8;  out!total;
                           printf("B sent %d\n", total);
                           goto S1;
        :: (total == 0) -> out!total
        fi }

proctype C (chan in, out)
{
        int total;
S1:     in?total;
        printf("C received %d\n", total);
        if
        :: (total != 0) -> total = (total+1)%8;  out!total;
                           printf("C sent %d\n", total);
                           goto S1;
        :: (total == 0) -> out!total
        fi }

init {  chan a2b = [1] of { int };
        chan b2c = [1] of { int };
        chan c2a = [1] of { int };
        atomic { run A(c2a, a2b); run B(a2b, b2c); run C(b2c, c2a) }
     }
```

THE UNIVERSITY
of LIVERPOOL

Note that `%` is the modulo arithmetic operator. So, for example `(17%8)=1` and `(15%8)=7`.

(a) If we execute this program what sequence of outputs can we expect? **[8]**

(b) Will the assertion in process `A` succeed? Explain your answer. **[7]**

(c) What will happen if we change the assertion in process `A` to be

```
assert(total != 3)
```

[5]

(d) If we wanted to verify that it is *not* the case that the `total` variable has a non-zero value infinitely often, what temporal formula would we wish to check? **[5]**

Additional material for students in exam

Glossary of Z notation

Names

a,b	identifiers
d,e	declarations (e.g., $a : A; b, ... : B ...$)
f,g	functions
m,n	numbers
p,q	predicates
s,t	sequences
x,y	expressions
A,B	sets
C,D	bags
Q,R	relations
S,T	schemas
X	schema text (e.g., d, $d \mid p$ or S)

Definitions

$a == x$	Abbreviated definition
$a ::= b \mid ...$	Data type definition (or $a ::= b\langle\!\langle x \rangle\!\rangle \mid ...$)
$[a]$	Introduction of a given set (or $[a, ...]$)
$a\_$	Prefix operator
$\_a$	Postfix operator
$\_a\_$	Infix operator

Logic

$true$	Logical true constant
$false$	Logical false constant
$\neg\ p$	Logical negation
$p \wedge q$	Logical conjunction
$p \vee q$	Logical disjunction
$p \Rightarrow q$	Logical implication ($\neg\ p \vee q$)
$p \Leftrightarrow q$	Logical equivalence ($p \Rightarrow q \ \wedge\ q \Rightarrow p$)
$\forall X \bullet q$	Universal quantification
$\exists X \bullet q$	Existential quantification
$\exists_1 X \bullet q$	Unique existential quantification
$\mathrm{let}\ a == x; ... \bullet p$	Local definition

Sets and expressions

$x = y$	Equality of expressions
$x \neq y$	Inequality ($\neg\ (x = y)$)
$x \in A$	Set membership
$x \notin A$	Non-membership ($\neg\ (x \in A)$)
$\varnothing$	Empty set
$A \subseteq B$	Set inclusion
$A \subset B$	Strict set inclusion ($A \subseteq B \wedge A \neq B$)
$\{x, y, ...\}$	Set of elements
$\{X \bullet x\}$	Set comprehension
$\lambda X \bullet x$	Lambda-expression – function
$\mu X \bullet x$	Mu-expression – unique value
$\mathrm{let}\ a == x; ... \bullet y$	Local definition
$\mathrm{if}\ p\ \mathrm{then}\ x\ \mathrm{else}\ y$	Conditional expression
$(x, y, ...)$	Ordered tuple
$A \times B \times ...$	Cartesian product
$\mathbb{P}\ A$	Power set (set of subsets)
$\mathbb{P}_1\ A$	Non-empty power set
$\mathbb{F}\ A$	Set of finite subsets
$\mathbb{F}_1\ A$	Non-empty set of finite subsets
$A \cap B$	Set intersection
$A \cup B$	Set union
$A \setminus B$	Set difference
$\bigcup A$	Generalized union of a set of sets
$\bigcap A$	Generalized intersection of a set of sets
$first\ x$	First element of an ordered pair
$second\ x$	Second element of an ordered pair
$\#A$	Size of a finite set

Relations

$A \leftrightarrow B$	Relation ($\mathbb{P}(A \times B)$)
$a \mapsto b$	Maplet ((a, b))
$\mathrm{dom}\ R$	Domain of a relation
$\mathrm{ran}\ R$	Range of a relation
$\mathrm{id}\ A$	Identity relation
$Q \mathbin{;} R$	Forward relational composition
$Q \circ R$	Backward relational composition ($R \mathbin{;} Q$)
$A \lhd R$	Domain restriction
$A \ntriangleleft R$	Domain anti-restriction
$A \rhd R$	Range restriction
$A \ntriangleright R$	Range anti-restriction
$R(\!\lvert A \rvert\!)$	Relational image
$iter\ n\ R$	Relation composed n times
R^n	Same as $iter\ n\ R$
$R^{\sim}$	Inverse of relation (R^{-1})
R^*	Reflexive-transitive closure
R^+	Irreflexive-transitive closure
$Q \oplus R$	Relational overriding ($(\mathrm{dom}\ R \ntriangleleft Q) \cup R$)
$a \ \underline{R}\ b$	Infix relation

Functions

$A \rightarrowtail\!\!\!\!\!\!+ B$	Partial functions
$A \longrightarrow B$	Total functions
$A \rightarrowtail\!\!\!\!\!\!+\!\!\!\!\to B$	Partial injections
$A \rightarrowtail B$	Total injections
$A \twoheadrightarrow\!\!\!\!\!\!+ B$	Partial surjections
$A \twoheadrightarrow B$	Total surjections
$A \rightarrowtail\!\!\!\!\twoheadrightarrow B$	Bijective functions
$A \twoheadrightarrow\!\!\!\!\!\!+\!\!\!+ B$	Finite partial functions
$A \rightarrowtail\!\!\!\!\!\!+\!\!\!+ B$	Finite partial injections
$f\ x$	Function application (or $f(x)$)

Numbers

$\mathbb{Z}$	Set of integers
$\mathbb{N}$	Set of natural numbers $\{0, 1, 2, ...\}$
$\mathbb{N}_1$	Set of non-zero natural numbers ($\mathbb{N} \setminus \{0\}$)
$m + n$	Addition
$m - n$	Subtraction
$m * n$	Multiplication
$m \operatorname{div} n$	Division
$m \operatorname{mod} n$	Modulo arithmetic
$m \leq n$	Less than or equal
$m < n$	Less than
$m \geq n$	Greater than or equal
$m > n$	Greater than
$succ\ n$	Successor function $\{0 \mapsto 1, 1 \mapsto 2, ...\}$
$m \mathrel{..} n$	Number range
$min\ A$	Minimum of a set of numbers
$max\ A$	Maximum of a set of numbers

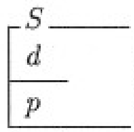
Sequences

$\operatorname{seq} A$	Set of finite sequences
$\operatorname{seq}_1 A$	Set of non-empty finite sequences
$\operatorname{iseq} A$	Set of finite injective sequences
$\langle\rangle$	Empty sequence
$\langle x, y, ...\rangle$	Sequence $\{1 \mapsto x, 2 \mapsto y, ...\}$
$s \frown t$	Sequence concatenation
$\frown / s$	Distributed sequence concatenation
$head\ s$	First element of sequence ($s(1)$)
$tail\ s$	All but the head element of a sequence
$last\ s$	Last element of sequence ($s(\#s)$)
$front\ s$	All but the last element of a sequence
$rev\ s$	Reverse a sequence
$squash\ f$	Compact a function to a sequence
$A \upharpoonleft s$	Sequence extraction ($squash(A \lhd s)$)
$s \upharpoonright A$	Sequence filtering ($squash(s \rhd A)$)
$s \operatorname{prefix} t$	Sequence prefix relation ($s \frown v = t$)
$s \operatorname{suffix} t$	Sequence suffix relation ($u \frown s = t$)
$s \operatorname{in} t$	Sequence segment relation ($u \frown s \frown v = t$)
$\operatorname{disjoint} A$	Disjointness of an indexed family of sets
$A \operatorname{partition} B$	Partition an indexed family of sets

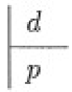
Bags

$\operatorname{bag} A$	Set of bags or multisets ($A \mathrel{+\!\!\!\rightarrow} \mathbb{N}_1$)
$[\![\,]\!]$	Empty bag
$[\![x, y, ...]\!]$	Bag $\{x \mapsto 1, y \mapsto 1, ...\}$
$count\ C\ x$	Multiplicity of an element in a bag
$C \sharp x$	Same as $count\ C\ x$
$n \otimes C$	Bag scaling of multiplicity
$x \sqsubseteq C$	Bag membership
$C \sqsubseteq D$	Sub-bag relation
$C \uplus D$	Bag union
$C \uminus D$	Bag difference
$items\ s$	Bag of elements in a sequence

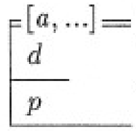
Schema notation



Vertical schema. New lines denote ';' and '$\wedge$'. The schema name and predicate part are optional. The schema may subsequently be referenced by name in the document.



Axiomatic definition. The definitions may be non-unique. The predicate part is optional. The definitions apply globally in the document.



Generic definition. The generic parameters are optional. The definitions must be unique. The definitions apply globally in the document.

$S \mathrel{\widehat{=}} [X]$	Horizontal schema
$[T; ... \mid ...]$	Schema inclusion
$z.a$	Component selection (given $z : S$)
θS	Tuple of components
$\neg S$	Schema negation
$\operatorname{pre} S$	Schema precondition
$S \wedge T$	Schema conjunction
$S \vee T$	Schema disjunction
$S \Rightarrow T$	Schema implication
$S \Leftrightarrow T$	Schema equivalence
$S \setminus (a, ...)$	Hiding of component(s)
$S \upharpoonright T$	Projection of components
$S \mathbin{;} T$	Schema composition (S then T)
$S \gg T$	Schema piping (S outputs to T inputs)
$S[a/b, ...]$	Schema component renaming (b becomes a, etc.)
$\forall X \bullet S$	Schema universal quantification
$\exists X \bullet S$	Schema existential quantification
$\exists_1 X \bullet S$	Schema unique existential quantification

Conventions

$a?$	Input to an operation
$a!$	Output from an operation
a	State component before an operation
a'	State component after an operation
S	State schema before an operation
S'	State schema after an operation
ΔS	Change of state (normally $S \wedge S'$)
ΞS	No change of state (normally $[S \wedge S' \mid \theta S = \theta S']$)

Jonathan P. Bowen
Oxford University Computing Laboratory
Wolfson Building, Parks Road, OXFORD OX1 3QD, UK
Email: `Jonathan.Bowen@comlab.ox.ac.uk`