PAPER CODE NO.	EXAMINER : Michael Fisher
COMP313/COMP513	DEPARTMENT : Computer Science Tel. No. 4-6701

THE UNIVERSITY
of LIVERPOOL

MAY 2005 EXAMINATIONS

Bachelor of Science : Year 3
Master of Science : Year 1

Formal Methods

TIME ALLOWED : $2\frac{1}{2}$ hours

INSTRUCTIONS TO CANDIDATES

Answer **four** questions only.

If you attempt to answer more questions 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) In Z, functions, sequences and bags can all be represented as sets of maplets. Explain how and give simple examples to illustrate your answer. **[4]**

   (b) If $f == \{a \mapsto 1, b \mapsto 3, c \mapsto 4\}$ and $g == \{a \mapsto 3, b \mapsto 3, d \mapsto 5, e \mapsto 1\}$ then what is the value of $(\operatorname{ran} f) \setminus (\operatorname{ran} g)$? **[6]**

   (c) If $f(x) = ((x \times x) + 3)$ then what is the value of $2 \otimes (\operatorname{map} f\ \langle 2, 4, 5, 7 \rangle)$? **[6]**

   (d) If $Z == [\![red, red, white, blue, green, blue]\!]$ then what is the value of $(Z \oplus \{white \mapsto 3\}) \oplus \{red \mapsto 1\}$? **[6]**

   (e) Write down a logical formula to represent the fact that, for any Natural Number (n) you could choose, we can always find two integers, one bigger then n, one smaller than n. **[3]**

2. We must write a Z specification of the members of a family and have developed the initial state space schema below (where *PERSON* is the set of all people):

$$
\begin{array}{|l}
\hline
\textit{FamilyRecord} \\
\hline
family : \mathbb{P}\, PERSON \\
age : PERSON \nrightarrow 0\,..\,120 \\
\hline
\operatorname{dom} age = family \\
\hline
\end{array}
$$

   (a) Write a Z specification for the operation

   $$Add(name? : PERSON, age? : 0\,..\,120)$$

   which adds a new family member (*name*?) of age (*age*?) to the *FamilyRecord*. **[7]**

   (b) Write a Z specification for the operation

   $$CheckAge(name? : PERSON, age! : 0\,..\,120)$$

   which returns the age (*age*!) associated with the family member (*name*?).

   Note: the operation should be undefined if the given person is not a family member. **[5]**

   (c) How would you modify the *CheckAge* operation above so that it is robust (i.e. it will be defined for any name supplied)? Assume that ~~a~~ *REPORT* type exists for reporting errors. **[6]**

   (d) Write a Z specification for the operation $Young(names! : \mathbb{P}\, PERSON)$ which returns the set of family members who are less than 20 years old. **[7]**

3. [About fundamentals of Temporal Logic.]

   (a) We wish to say that

   > "at some point in the future, either a will always be true or b will be true in the next moment."

   How might we represent this in temporal logic? **[3]**

   (b) How does temporal logic extend classical logic? In your answer give an example of a statement that is more naturally represented in temporal, rather than classical, logic. **[8]**

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

   (d) Consider the semantics of propositional, discrete, linear temporal logic, and show why the formula $\square(p \Rightarrow \bigcirc p)$ implies the formula $p \Rightarrow \square p$ **[10]**

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

$$Spec_P\colon\ \square\left[\begin{array}{lrcl} & \textbf{start} & \Rightarrow & a \\ \wedge & a & \Rightarrow & \bigcirc b \\ \wedge & b & \Rightarrow & \bigcirc c \\ \wedge & d & \Rightarrow & \bigcirc e \end{array}\right] \qquad Spec_Q\colon\ \square\left[\begin{array}{lrcl} & x & \Rightarrow & \bigcirc y \\ \wedge & y & \Rightarrow & \Diamond w \end{array}\right]$$

   (a) What is the behaviour of $Spec_P$ on its own? **[6]**

   (b) Given $Comms = \square(b \Rightarrow \Diamond x)$ what is the behaviour of $Spec_P \wedge Spec_Q \wedge Comms$ **[7]**

   (c) Given $Comms = \square(b \Rightarrow \Diamond x) \wedge \square(w \Rightarrow \Diamond d)$ what is the behaviour of

$$Spec_P \wedge Spec_Q \wedge Comms$$

   [7]

   (d) Explain, informally, why

$$Spec_P \wedge Spec_Q \wedge \square(b \Rightarrow \Diamond x) \wedge \square(w \Rightarrow \Diamond a)$$

   implies $\square\Diamond c$ **[5]**

THE UNIVERSITY
of LIVERPOOL

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) Describe two problems with the standard model checking approach and explain what techniques are being developed to tackle these. **[10]**

   (c) If the model checking process fails then what information is returned? What does this say about the execution of the system being modelled? **[5]**

6. In this question, we consider the Promela language. In answering the sub-parts of the question, please give simple examples to illustrate your answers.

   (a) In Promela, how are processes defined and executed? **[7]**

   (b) Channels are used to communicate between Promela processes. How does the size of the channel affect the behaviour of the processes reading from, or writing to, the channel? **[6]**

   (c) *Assertions* and *Never Claims* are used to carry out verification of temporal properties. What is the difference between these approaches in terms of their coding, their implementation and the temporal formulae they typically represent? **[12]**

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
$p \land q$	Logical conjunction
$p \lor q$	Logical disjunction
$p \Rightarrow q$	Logical implication ($\neg\, p \lor q$)
$p \Leftrightarrow q$	Logical equivalence ($p \Rightarrow q \land q \Rightarrow p$)
$\forall X \bullet q$	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 ($A \subseteq B \land A \neq B$)
$\{x, y, ...\}$	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
$\mathbf{if}\ p\ \mathbf{then}\ x\ \mathbf{else}\ y$	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
$A \mathbin{\lhd\!\!\!-} R$	Domain anti-restriction
$A \rhd R$	Range restriction
$A \mathbin{\rhd\!\!\!-} R$	Range anti-restriction
$R(\!\mid A \mid\!)$	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 ($(\mathrm{dom}\ R \mathbin{\lhd\!\!\!-} Q) \cup R$)
$a \underline{R}\ b$	Infix relation

Functions

$A \mathrel{+\!\!\!\!\!\rightarrow} B$	Partial functions
$A \longrightarrow B$	Total functions
$A \mathrel{\rightarrowtail\!\!\!\!\!\!+\;} B$	Partial injections
$A \rightarrowtail B$	Total injections
$A \mathrel{+\!\!\!\!\!\twoheadrightarrow} B$	Partial surjections
$A \twoheadrightarrow B$	Total surjections
$A \mathrel{\rightarrowtail\!\!\!\!\!\twoheadrightarrow} B$	Bijective functions
$A \mathrel{+\!\!\!\!\!+\!\!\!\!\!\rightarrow} B$	Finite partial functions
$A \mathrel{\rightarrowtail\!\!\!\!\!\!+\!\!\!\!\!+\;} B$	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 ($A \nrightarrow \mathbb{N}_1$)
$[\![\,]\!]$	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
$x \inbag C$	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

S / d / p	**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.
d / p	**Axiomatic definition.** The definitions may be non-unique. The predicate part is optional. The definitions apply globally in the document.
$[a, ...]$ / d / p	**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
$S \fatsemi T$	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