

**CSE4213 Sample Examination
2002
Instructions to Candidates**

1. Examination time 2 hours
2. Answer all questions
3. There are 10 questions
4. Each question is worth 8 marks
5. Total marks 80
6. Calculators are not allowed
7. Use left hand page for rough working; this page will NOT be marked unless explicitly requested.
8. The *Concise Summary of the B mathematical toolkit* is supplied.

1. B has been used to design the signalling and control systems for driverless trains on the Paris Metro system. Explain why.

(8 marks)

Answer: Users (= railway companies) of such a system need to be confident that the software is both safe and correct. B allows the development of software from mathematical specifications. It can be mathematically (= mechanically) proven that the specification is consistent, and that the code developed from it meets the specification. This ensures a high degree of confidence in the correctness of the code. However, it is still necessary to establish the correctness of the specification itself by human input.

2. Explain the use of non-determinism in B specifications. Give an example to illustrate your answer.

(8 marks)

Answer: In specifications, it is important to focus upon **what** the behaviour is, not **how** it is to be implemented. Accordingly, when specifying the result to be computed, we look for the *weakest constraint* on that result. This gives the widest latitude to the implementor, and admits of the greatest number of correct implementations.

Such weak constraints are given by non-deterministic constructs, such as the ANY construct. This construct allows any value to be bound to an indicated (local) variable that satisfies a predicate, as in

```
ANY  bg
WHERE bg  $\notin$  BAGS - bags
THEN bags := bags  $\cup$  {bg}
```

which allows any choice of new bag *bg* of type *BAGS* as long as it is not one of the currently used bags in the set *bags*.

3. The INCLUDES, PROMOTES and EXTENDS clause in a B machine specification allow more complex specifications to be built from simple ones.
- The INCLUDES clause names machine whose operations are available for use in the including machine, **but** these operations are not part of the including machine's operations.
 - The EXTENDS clause names machines whose operations are available for use in the including machine, **and** these operations are also part of the including machine's operations.
 - The PROMOTES clause names operations of an included machine that are also made available as part of the including machines operations.

Motivate the distinctions between these clauses from a software engineering perspective.

(8 marks)

Answer:

- (a) The clauses allow the construction of complex specifications from simple ones, by building on the operations of included/extended machines.
- (b) All of these are motivated by the *Information Hiding Principle*
- (c) INCLUDES means that the primitive operations are not automatically visible, and can be hidden
- (d) PROMOTES gives selective control over primitive operations that may be required to be exported from the including machine
- (e) EXTENDS is syntactic sugaring in the case where all operations of the included machine are to be visible outside the including machine.

4. Give the Predicate Transformations of the following expressions:

(a) $[skip] x > y$

(b) $[x := a] \forall p.f(x) = p$

(c) $[x := y || y := x] x > y$

(d) $[x \in \{2, 4, 6\} \Rightarrow x := x/2] 2 \times x$

(8 marks)

Answer:

(a) $x > y$

(b) $\forall p.f(a) = p$

(c) $y > x$

(d) $x \in \{2, 4, 6\} \Rightarrow x$

5. The following B machine specification fragment is taken from DEMO2, which deals with a simple Lift system. Given that

mov is the set of LIFTS in motion;

dir is a total function of LIFTS to DIRECTIONS ($\{up, dn\}$), and describes in which direction a particular lift is travelling;

flr is a total function of LIFTS to FLOORS (*bottomflr..topflr*), and describes which floor each lift is at;

out is a relation from LIFTS to FLOORS, and describes which buttons inside the lifts have been pressed; and

in is a relation from FLOORS to DIRECTIONS, and describes which buttons outside the lifts have been pressed:

explain the first four (4) preconditions of the operation **Continue_Up** in English.

Continue_Up (*ll*) $\hat{=}$

PRE

ll \in *mov* \wedge

dir (*ll*) = *up* \wedge

flr (*ll*) < *topfloor* \wedge

ll \mapsto *flr* (*ll*) \notin *out* \wedge

flr (*ll*) \mapsto *up* \notin *in* \wedge

attr_up (*ll*)

THEN

flr (*ll*) := *flr* (*ll*) + 1

END ;

(8 marks)

Answer:

ll \in *mov* The lift specified in the parameter must be moving.

dir (*ll*) = *up* The direction of the lift specified in the parameter must be *up*.

flr (*ll*) < *topfloor* The floor that this lift is at must be less than the *topfloor*, that is, there is room to continue upwards.

ll \mapsto *flr* (*ll*) \notin *out* No one has pressed the button inside this lift to get out at this floor.

6. In a railway system, trains are made up by joining a number of carriages together, and numbered from the front of the train. At the front of the train is a locomotive, required to haul the train.

In specifying a train despatch system, we need to model such trains. Give B fragments to define variables and invariants that would specify such trains. You may assume the sets *LOCOS* and *CARRIAGES*.

(8 marks)

Answer:

VARIABLES

trains, locos, carriages

INVARIANT

$trains \subseteq \mathbb{N}_1 \wedge$

$locos \in \mathbb{N}_1 \rightsquigarrow LOCOS \wedge$

$carriages \in \mathbb{N}_1 \rightsquigarrow seq\ CARRIAGES \wedge$

$\forall c1, c2. c1 \in carriages \wedge c2 \in carriages \wedge c1 \neq c2 \Rightarrow$

$ran\ ran\ c1 \cap ran\ ran\ c2 = \{\}$

There are a number of trains in the system. Each train is identified by a natural number. Each train has a single loco, not shared by any other train, and a unique sequence of carriages, where each carriage can only be attached to one train.

7. Describe the difference between a *precondition* and a *guard*.

(8 marks)

Answer: A precondition states the *assumptions* made in the specification of an operation. If any precondition is not satisfied, the operation is unconstrained and any behaviour meets the specification. This is equivalent to the logical axiom that by assuming false, one can prove anything!

PRE precondition **THEN** substitution **END**

A guard, on the other hand, is an explicit test to see whether a condition is satisfied. Only if the condition is satisfied is the guarded specification applied. If the condition is not met, then this aspect of the specification is empty, and no behaviour (the null behaviour) is specified. A guarded specification in which the condition is not met is equivalent to *skip*.

IF condition **THEN** substitution **END**

8. To prove that $f \triangleleft g \in s \rightarrow t$, it is sufficient to prove the following lemmas:

L1 f and g are total functions

L2 $\text{dom}(f) \cup \text{dom}(g) = s$

L3 $\text{ran}(f) \subseteq t$ and $\text{ran}(g) \subseteq t$

Show therefore that if $f \in u \rightarrow t$, $u \cup \{a\} = s$, and $b \in t$, that $f \triangleleft \{a \mapsto b\} \in s \rightarrow t$.
(8 marks)

Answer:

- (a) f is a total function (from hypothesis: L1a)
- (b) $\{a \mapsto b\} \in \{a\} \rightarrow \{b\}$ and is therefore a total function, = g in original postulation (from set theory: L1b)
- (c) $\text{dom}(g) = \{a\}$ (from (b))
- (d) $\text{dom}(f) \cup \text{dom}(g) = u \cup \{a\}$ (from (c))
- (e) $\text{dom}(f) \cup \text{dom}(g) = s$ (from (d) and hypothesis: L2)
- (f) $\text{ran}(f) = t$ (from hypothesis), therefore $\text{ran}(f) \subseteq t$ (from set theory: L3a)
- (g) $\text{ran}(g) = \{b\}$ (from (b))
- (h) but $b \in t$ (from hypothesis), therefore $\text{ran}(g) \subseteq t$ (from set theory: L3b)
- (i) all the lemmas are proved, therefore QED.

9. From the **SimpleBank** machine, we have the following fragments:

```
...
VARIABLES
    accounts , balance
INVARIANT
    accounts  $\subseteq$  ACCOUNT  $\wedge$ 
    balance  $\in$  accounts  $\rightarrow$   $\mathbb{N}$ 
INITIALISATION
    accounts , balance := { } , { }
    ...
OPERATIONS
    Withdraw ( amount , account )  $\hat{=}$ 
        PRE amount  $\in$   $\mathbb{N}$   $\wedge$  account  $\in$  accounts  $\wedge$  amount  $\leq$  balance ( account )
        THEN
            balance := balance  $\Leftarrow$  { account  $\mapsto$  balance ( account ) - amount }
        END ;
```

Part of one of the proof obligations is to establish

$$\textit{balance}(\textit{account}) - \textit{amount} \in \mathbb{N}$$

Prove this.

(8 marks)

Answer:

- (a) $\textit{balance}(\textit{account}) \in \mathbb{N}$ (From the invariant $\textit{balance} \in \textit{accounts} \rightarrow \mathbb{N}$, and the precondition $\textit{account} \in \textit{accounts}$)
- (b) $\textit{amount} \in \mathbb{N}$ (From the precondition)
- (c) $\textit{amount} < \textit{balance}(\textit{account})$ (From the precondition)
- (d) Therefore, since if $a \in \mathbb{N}$ and $b \in \mathbb{N}$ and $b < a$ then $a - b \in \mathbb{N}$, QED

10. State one principle of software engineering you have learnt, and explain how B does (or does not!) give support to this principle.

(8 marks)

Answer: One principle is the Principle of Semantic Emphasis, which states that specifications should describe the *what* and not the *how*. *Mechanisms* are for *implementations*, not specifications. On the other hand, a specification should focus on the *policy* that the specified software is to follow. B specifications give ample flexibility to the software designer to follow this principle.