THE UNIVERSITY OF NEW SOUTH WALES

COMP2111
System Modelling and Design

Time allowed: 2 hours
Total number of questions: 30

This paper contains both multiple choice/multiple correct and short answer questions.

The multiple choice questions are not necessarily of equal value.

All short answer questions are of equal value. The value will be given with the question.

Multiple choice questions must be answered in pencil on the Generalised Answer Sheet.

Be careful: if you wish to delete a selection, make sure that you completely erase the old selection.

Short answers are to be written in the book provided.

Answers to multiple choice written in this book will not be read.

Answers to any questions written on this examination paper will not be read. A summary of the Event B mathematical toolkit is attached to this paper. During the examination, it may be detached from the paper if desired.

This paper may not be retained by the candidate.
Instructions for this paper

Please read carefully

This examination contains multiple-choice questions in which there may be more than one correct answer to any question.

**Marking scheme:** If there are \( N \) correct answers to all multiple choice questions in the paper then each correct selection to a question will earn you \( \frac{1}{N} \) of the total marks for multiple choice questions, and each incorrect selection will result in \( \frac{1}{2N} \) of the total marks for multiple choice question being deducted.

**For any one question, your cumulative mark for that question will never be less than zero.** So, if \( C \) and \( W \) are the number of correct and incorrect selections, respectively, in a question then the mark for that question will be \( \max(C - \frac{W}{2}, 0) \times 100\% \) of the total marks for the paper.

Examples: Assume a paper with 50 questions and five choices in each question. Assume the total number of correct choices across the whole paper is 100 and suppose the total mark for the paper is 100.

In one question you select

- 1 correctly and make no other choice: your mark will be \( (1/100) \times 100 = 1 \) mark.
- 1 correctly and 1 incorrectly: your mark will be \( (1 - 0.5)/100 \times 100 = 0.5 \) mark.
- 2 correct and 1 incorrect: your mark will be \( (1 + 1 - 0.5)/100 \times 100 = 1.5 \) mark.
- 1 correct and 2 incorrect: your mark will be \( (1 - 0.5 - 0.5)/100 \times 100 = 0 \) mark.
- 1 correct and 3 incorrect: your mark will be \( \max(1 - 1, 5, 0)/100 \times 100 = 0 \) mark.

**Note**

1) *Even though the phrasing of a question may imply more than one answer, there could be only one correct answer.*

2) *Answers that are only conditionally correct are not considered correct for the purposes of this examination.*
The number of correct multiple-choice answers in this paper is 57.

**Note carefully:** all of the B mathematics in the following questions is in marked-up form, unless stated otherwise.

1) Choose the best completions of the following statement.
   We frequently use nondeterminism in specification because:
   A) we want to delay design decisions until later in the refinement.
   B) we frequently need to specify nondeterministic operations.
   C) we need to provide alternatives for the user of the system.
   D) computers are nondeterministic.
   E) the requirements often allow a choice of behaviour, and the specification does not need to decide between those choices.

2) Which of the following are necessarily correct?
   A) a member of a sequence must have multiplicity, or frequency.
   B) a member of a sequence must have a position.
   C) a member of a bag must have a position.
   D) a member of a bag must have multiplicity, or frequency.
   E) sets are only required to provide membership.

3) Which of the following statements about events are necessarily correct?
   A) If the guard is satisfied then the event will be correct.
   B) The event will proceed on the assumption that the guard is satisfied.
   C) A guard ensures that the state invariant is maintained.
   D) If a guard is not satisfied then the event is skipped.
   E) Maintenance of the state invariant requires the guard to be satisfied.

4) Given a function \( f \) that is defined as \( f \in X \mapsto Y \), which of the following are correct?
   A) \( \text{dom}(f) \in \mathcal{P}(X) \)
   B) \( f \in X \leftrightarrow Y \)
   C) \( \text{dom}(f) = X \)
   D) \( \text{dom}(f) \subset X \)
   E) \( f^{-1} \in Y \mapsto X \)

5) Which of the following rules are correct?
   A) \( f \in X \mapsto Y \Rightarrow f \in X \mapsto Y \)
   B) \( f \in X \mapsto Y \Rightarrow f \notin X \mapsto Y \)
   C) \( f \in X \mapsto Y \Rightarrow \text{dom}(f) \subset X \)
   D) \( f \in X \mapsto Y \Rightarrow \text{ran}(f) = Y \)
   E) \( f \in X \mapsto Y \land x \in \text{dom}(f) \Rightarrow f([x]) = \{f(x)\} \)
6) Given \( n \in \mathbb{N} \) and \( f^n \) denotes the iteration of \( f \), which of the following rules are correct?

A) \( f \in X \rightarrow X \Rightarrow f^n \in X \rightarrow X \)
B) \( f \in X \mapsto X \Rightarrow f^n \in X \mapsto X \)
C) \( f \in X \mapsto X \Rightarrow f^n \in X \mapsto X \)
D) \( f \in X \mapsto X \Rightarrow f^n \in X \mapsto X \)
E) \( f \in X \mapsto X \Rightarrow f^n \in X \mapsto X \)

7) Assume \( FRUIT \) is a carrier set. Given the specification:

\[
\begin{align*}
\text{apples} & \subseteq FRUIT \\
\text{oranges} & \subseteq FRUIT \\
\text{salad} & \in FRUIT \leftrightarrow FRUIT \\
\text{box} & \in 1..\text{boxsize} \rightarrow FRUIT
\end{align*}
\]

which of the following expressions are well-typed?

A) \( \text{oranges} \leq \text{salad} \)
B) \( \text{salad} \leq \text{apples} \)
C) \( \text{box} \; ; \; \text{salad} \)
D) \( \text{box} \approx \text{salad} \)
E) \( \text{salad} \approx \text{box} \)

8) Consider the following enumerated set specifications:

\[
\begin{align*}
SETS \\
MEDAL & = \{ \text{Gold}, \text{Silver}, \text{Bronze} \} \\
COMPETITOR & = \{ \text{Cathy}, \text{Jenny}, \text{Susie}, \text{Inga}, \text{Lisa} \}
\end{align*}
\]

We require a variable \( \text{medals} \) that represents the mappings from a competitor (a member of \( COMPETITOR \)) to a medal (member of \( MEDAL \)) that has been won by the competitor in some event. Assume that:

- all medals are won and
- no two competitors are awarded the same medal.

Select constraints, not necessarily the strongest, that \( \text{medals} \) must satisfy.

A) \( \text{medals} \in COMPETITOR \rightarrow MEDAL \)
B) \( \text{medals} \in COMPETITOR \Rightarrow MEDAL \)
C) \( \text{medals} \in COMPETITOR \mapsto MEDAL \)
D) \( \text{medals} \in COMPETITOR \mapsto MEDAL \)
E) \( \text{medals} \in COMPETITOR \mapsto MEDAL \)
9) The two assumptions made in Question 8 do not allow for two competitors tying for a place. To be more realistic we could use the assumptions:

- if two competitors tie for a place then both competitors receive the same medal and the next medal (if any) is not awarded;
- at most two people tie for a place;
- otherwise all medals are won.

Select constraints, not necessarily the strongest, that medals must satisfy.

A) \( \text{medals} \in \text{COMPETITOR} \leftrightarrow \text{MEDAL} \)
B) \( \text{medals} \in \text{COMPETITOR} \mapsto \text{MEDAL} \)
C) \( \text{medals} \in \text{COMPETITOR} \mapsto \rightarrow \text{MEDAL} \)
D) \( \text{medals} \in \text{COMPETITOR} \rightarrow \rightarrow \text{MEDAL} \)
E) \( \text{medals} \in \text{COMPETITOR} \mapsto \rightarrow \text{MEDAL} \)

10) In a sales information system for a bookshop, a variable \( \text{sales} \) is specified by

\[
\text{sales} \in \text{DAY} \mapsto (\text{BOOK} \mapsto \mathbb{N}_1)
\]

where \( \text{DAY} \) is a set of days and \( \text{BOOK} \) is a set of books. The expression, \( \text{sales}(\text{day})(\text{book}) \) gives the number of copies of \( \text{book} \) sold on \( \text{day} \), if any copies were sold.

Given that \( \text{day} \in \text{dom(\text{sales})} \) which of the following gives the set of books sold on a particular \( \text{day} \).

A) \( \text{union(\text{dom(\text{sales}(\text{day}))})} \)
B) \( \text{dom(\text{sales}(\text{day}))} \)
C) \( \text{dom((\{\text{day}\} \triangle \text{sales})(\text{day}))} \)
D) \( \text{dom(\text{union(\text{sales}[[\text{day}]])}}) \)
E) \( \text{dom(\text{union(\text{sales}(\text{day}))})} \)

11) Continuing the example in Question 10 above.

Which of the following expressions specifies the set of all books contained in \( \text{sales} \).

A) \( \text{dom(\text{ran(\text{sales})})} \)
B) \( \sum \text{book} \cdot \text{book} \in \text{dom(\text{ran(\text{sales})})} \mid \text{ran(\text{sales})(\text{book})} \)
C) \( \sum \text{book} \cdot \text{dom(\text{ran(\text{sales})})} > 0 \mid \text{ran(\text{sales})(\text{book})} \)
D) \( \text{union(\text{dom(\text{ran(\text{sales})})})} \)
E) \( \text{dom(\text{union(\text{ran(\text{sales})})})} \)

where \( \sum x \cdot P \mid E = \text{“sum E(x) over all x satisfying P(x)”} \). This is not part of the EventB toolkit, but could be defined as a constant function.

12) Complete the following statement:

Discharging all proof obligations for a machine ensures that . . .

A) the machine is correct with respect to the requirements.
B) *the events of the machine are deterministic.*
C) *all events maintain the machine invariant.*
D) *the initialisation establishes the machine invariant.*
E) *the machine can be implemented.*
13) A video rental store rents videos to members.

The set of members is modelled by the variable $\text{members}$, and the set of videos owned by the store is modelled by the variable $\text{videos}$. These variables are specified by

$$\text{members} \subseteq \text{MEMBER} \land \text{videos} \subseteq \text{VIDEO}$$

The policy of the store is that

$a$ video is rented to a single member, but a member can rent any number of different videos.

The relation between rented video and the member who is renting it is to be modelled by the variable $\text{rented}$. Which of the following are appropriate specifications?

A) $\text{rented} \in \text{members} \rightarrow \text{videos}$
B) $\text{rented} \in \text{videos} \rightarrow \text{members}$
C) $\text{rented} \in \text{videos} \\rightarrow \text{members}$
D) $\text{rented} \in \text{videos} \rightarrow \text{members}$
E) $\text{rented} \in \text{members} \rightarrow \text{videos}$

14) For some video store, not necessarily the same as in 13, $\text{rented}$ is specified as

$$\text{rented} \in \text{videos} \rightarrow \text{members}$$

For renting a video, we wish to specify an event $\textbf{Rent}$ as follows

**Event**  $\textbf{Rent}$  $\triangleq$

- any member
- video

**when** guards

where $\text{member}$ denotes the member who is renting and $\text{video}$ denotes the video that is being rented.

The action of the event is given as:

$$\text{rented(} \text{video} \text{)} := \text{member} \; .$$

Which of the following are required for a minimal set of guards?

A) $\text{member} \notin \text{ran(} \text{rented} \text{)}$
B) $\text{member} \in \text{members}$
C) $\text{video} \in \text{videos}$
D) $\text{rented(} \text{video} \text{)} \neq \text{member}$
E) $\text{video} \notin \text{dom(} \text{rented} \text{)}$
15) Continuing the example in Question 13.

The rental store allows two members to swap their videos, with the store records modified to reflect the new rental situation.

We need an event:

**Event**  $\text{Swap} \triangleq$

\[
\begin{align*}
\text{any } m_1, m_2 \\
\quad \text{when } v_1 \in \text{dom } \text{rented} \land v_2 \in \text{dom } \text{rented} \\
\quad m_1 = \text{rented}(v_1) \land m_2 = \text{rented}(v_2)
\end{align*}
\]

**then**  $\text{do-swap}$

Which of the following can be used for $\text{do-swap}$?

A) \(\text{rented}(v_1) := m_2 \parallel \text{rented}(v_2) := m_1\)

B) \(\text{rented} := \text{rented} \cup \{v_1 \mapsto m_2, v_2 \mapsto m_1\}\)

C) \(\text{rented}(v_1), \text{rented}(v_2) := m_2, m_1\)

D) \(\text{rented} := \text{rented} \leftarrow \{v_1 \mapsto m_2, v_2 \mapsto m_1\}\)

E) \(\text{rented} := \text{rented} \leftarrow \{v_1 \mapsto m_2\} \leftarrow \{v_2 \mapsto m_1\}\)

16) Consider modelling a supermarket stock control system.

There is a requirement that:

- all products in stock must have a price

A model uses the following:

- The set $\text{PRODUCT}$ models all possible products.
- The variable $\text{price}$ models the prices of some set of products.
- The variable $\text{stock}$ models the number of items in stock for some set of products.

Additionally, the variable $\text{products}$ modelling some set of products may be used.

Which of the following adequately satisfy the above requirement: (you may assume that price is adequately modelled by $\mathbb{N}$)

A) \(\text{price} \in \text{PRODUCT} \rightarrow \mathbb{N} \land \text{stock} \in \text{PRODUCT} \rightarrow \mathbb{N}_1\)

B) \(\text{price} \in \text{PRODUCT} \rightarrow \mathbb{N} \land \text{stock} \in \text{PRODUCT} \rightarrow \mathbb{N}_1 \land \text{dom}(\text{stock}) \subseteq \text{dom}(\text{price})\)

C) \(\text{price} \in \text{PRODUCT} \rightarrow \mathbb{N} \land \text{stock} \in \text{PRODUCT} \rightarrow \mathbb{N}_1 \land \text{dom}(\text{price}) \subseteq \text{dom}(\text{stock})\)

D) \(\text{products} \subseteq \text{PRODUCT} \land \text{price} \in \text{products} \rightarrow \mathbb{N} \land \text{stock} \in \text{products} \rightarrow \mathbb{N}_1\)

E) \(\text{products} \subseteq \text{PRODUCT} \land \text{price} \in \text{products} \rightarrow \mathbb{N} \land \text{stock} \in \text{products} \rightarrow \mathbb{N}_1\)
17) A specification for a seating plan has a set of guests, $guests$, a set of tables, $tables$ and positions at each table modelled as a subrange of natural numbers. The seating is modelled by the variable $seating$ specified as follows

$$seating \in guests \rightarrow (tables \times Seats)$$

where $Seats = 1..maxSeat$.

Select the most precise English explanation of the above specification.

A) The guests are seated around tables.
B) There is a seat allocated for guests at each table.
C) Every guest has a seat allocated at each table.
D) Every guest has a seat allocated at a table.
E) Guests are allocated to tables around which there are seats.

18) Continuing the example in Question 17, we need to specify that no two guests are seated at the same position on any table.

Which of the following could be used?

A) $\forall (g1, g2). (g1 \in \text{dom}(seating) \land g2 \in \text{dom}(seating) \land g1 \neq g2 \Rightarrow seating(g1) \neq seating(g2))$
B) $\forall (g1, g2). (seating(g1) \neq seating(g2))$
C) $\forall (g1, g2). (g1 \in \text{dom}(seating) \land g2 \in \text{dom}(seating) \Rightarrow seating(g1) \neq seating(g2))$
D) $\forall (g1, g2). (\text{ran}(seating(g1)) \neq \text{ran}(seating(g2)))$
E) $\text{card}(seating) = \text{card}(\text{ran}(seating))$

19) Continuing the example in Question 17, we wish to specify an event, $\text{NewGuest}$ that adds a new guest, provided there is space left at the current tables, without adding more tables. The event has parameter $guest$, and guard $guest \in GUEST \land guest \notin guests$.

We also require a guard that ensures that more seats are available. Which of the following could be used?

A) $\text{card}(seating) < maxSeat \ast \text{card}(tables)$
B) $\text{card}(guests) < maxSeat \ast \text{card}(tables)$
C) $\text{card}(guests) < maxSeat \ast \text{card}(\text{ran}(seating))$
D) $\text{card}(\text{ran}(seating)) < maxSeat$,
E) $\text{ran}(seating) \neq tables \times Seats$
20) Continuing the *NewGuest* event in Question 19, we wish to allocate a seat at a table to a guest.

We do this adding extra parameters *tb* and *ps*, representing a table and a position at that table.

Which of the following could be used for the guard?

A) \( tb \in \text{tables} \land ps \in \text{Seats} \land \)
\( tb \mapsto ps \in (\text{tables} \times \text{Seats}) \)

B) \( tb \in \text{tables} \land ps \in \text{Seats} \land \)
\( tb \mapsto ps \notin \text{ran}(\text{seating}) \)

C) \( tb \in \text{tables} \land ps \in \text{Seats} \land \)
\( tb \mapsto ps \in (\text{tables} \times \text{Seats} – \text{ran}(\text{seating})) \)

D) \( tb \in (\text{tables} – \text{dom}(\text{ran}(\text{seating}))) \land \)
\( ps \in (\text{Seats} – \text{ran}(\text{ran}(\text{seating}))) \)

E) \( tb \in \text{tables} \land tb \notin \text{dom}(\text{ran}(\text{seating})) \land \)
\( ps \in \text{Seats} \land ps \notin \text{ran}(\text{ran}(\text{seating})) \)

21) Which of the following statements about guards are necessarily correct?

A) Satisfying the guards ensures that the event will be correct.

B) If the guard is satisfied, then the event will proceed.

C) A guard ensures that the state invariant is maintained.

D) Maintenance of the state invariant requires the guard to be satisfied.

E) The event will only proceed if the guard is satisfied.
22) Let $[\text{Abort}]R = false$ and $[\text{Magic}]R = true$ for all $R$.
Which of the following are correct?

A) Abort is feasible.
B) Magic is feasible.
C) Abort can be refined to anything.
D) Anything can be refined to Magic.
E) A feasible construct can always be refined to a feasible construct.

23) The specification for an airline booking system contains a variable $flights$ modelled as follows:

$$flights \in DATE \rightarrow FLIGHT$$

where $DATE$ and $FLIGHT$ are carrier sets.

In a refinement, this variable is modelled by two sequences:

$$day \in SEQ(DATE)$$
$$flight \in SEQ(FLIGHT)$$

Which of the following would be a suitable part of a refinement relation?

A) $flights = (day^{-1}; flight)$
B) $flights = (flight^{-1}; day)$
C) $flights = day \times flight$
D) $flights = \{d,f,n \cdot d = day(n) \land f = flight(n) \mid d \mapsto f\}$
E) None of the above.

24) Consider the events in figure 1
Assume $x \in 1..5$
Which of the following are correct?

A) ii) is refined by i).
B) i) is refined by ii).
C) ii) is refined by iii), but the refinement is infeasible.
D) i) is refined by iii), but the refinement is infeasible.
E) ii) is refined by iii), and the refinement is feasible.

25) Consider the sequence refinement shown in machines Seq_ctx, Seq, List_ctx and SeqR shown in figures 4,5,6,7.
Which of the following describes a property specified by the invariant of SeqR.

A) last is never in dom(next).
B) if the sequence has more than one item then the pointer to the first item is in dom(next).
C) next is a bijective function.
D) next is an injective function.
E) last is always in the ran(next).
i) Event \( \text{evt1} \equiv \)

\[
\text{when } \text{grd1} : x \in \{1, 2, 3\} \\
\text{then act1} : \text{result} := 0 \\
\text{end}
\]

Event \( \text{evt2} \equiv \)

\[
\text{when } \text{grd1} : x \in \{3, 4, 5\} \\
\text{then act1} : \text{result} := 1 \\
\text{end}
\]

ii) Event \( \text{evt3} \equiv \)

\[
\text{when } \text{grd1} : x \in \{1, 2\} \\
\text{then act1} : \text{result} := 0 \\
\text{end}
\]

Event \( \text{evt4} \equiv \)

\[
\text{when } \text{grd1} : x \in \{3, 4, 5\} \\
\text{then act1} : \text{result} := 1 \\
\text{end}
\]

iii) Event \( \text{evt5} \equiv \)

\[
\text{when } \text{grd1} : x \in \{1, 2\} \\
\text{then act1} : \text{result} := 0 \\
\text{end}
\]

Event \( \text{evt6} \equiv \)

\[
\text{when } \text{grd1} : x \in \{4, 5\} \\
\text{then act1} : \text{result} := 1 \\
\text{end}
\]

Figure 1: Simple Events
Answers to the following questions are to be written in the Examination Book. Each question is worth 4 marks.

26) This question is concerned with the Coin context machine shown in figure 2. This machine provides some functions for handling bags of coins, here called \textit{COINS}.

We wish to add a function \textit{COINSVAL} which takes a bag of coins and gives the total value of the coins in the bag.

In the book provided give all details of what you would need to specify this function in the context machine.

27) This question is concerned with the Coin context and Vending machines shown in figures 2 and 3.

Please answer the following questions in the book provided:

a) The event \textit{GiveChange} should set the variable \textit{change} to a bag of coins, as specified by \textit{COINS}. The change should be correct, but is not required to be minimal in terms of the number of coins. Also of course the \textit{coinbox} should contain the chosen coins.

b) Also, describe —it does not have to be formal— any extra guards for the event that would ensure the event is feasible.

28) Continuing question 25.

The question is concerned with the \textit{refinement relation} that demonstrates that the list refinement shown in figure 7 simulates the sequence shown in figure 5.

In the answer book:

a) Write your refinement relation in the provided book.

b) With each of your predicates give a brief statement of what property you are describing.

Both the formal and informal parts of the refinement relation will be assessed.

29) This question refers to the RailTicket assignment on page 17

In the examination book:

- List as many design problems in this solution as you can, giving brief explanations of why they are problems.

30) This question refers to the Traffic Light development shown in figures 8, 9, 10 on page 23

We plan to add another event \textit{ToGreenPlus}, which changes the light in direction \textit{adir} to \textit{Green}, as in \textit{ToGreen}, but also changes the light in all other directions that consequently do not conflict with other \textit{Green} lights.

In the answer book:

a) Write the specification of the new event \textit{ToGreenPlus}.

b) Sketch a plan for the refinement of the new event using as much as possible of the current refinement.
CONTEXT  Coin_ctx

SETS

COIN  A set of coin denominations

CONSTANTS

ONE   one dollar
TWO   two dollar
FIVE  five dollar
VALUE  mapping from denomination to value
COINS  the set of bags of coins
COINSVALUE  yields the value of a bag of coins
SUBCOINS  SUBBAG(b1 ↦ b2) = TRUE ⇒ b1 is a subbag of b2

AXIOMS

axm1:  \( COIN = \{ONE, TWO, FIVE\} \)
axm2:  \( ONE \neq TWO \)
axm3:  \( ONE \neq FIVE \)
axm4:  \( TWO \neq FIVE \)
axm5:  \( VALUE \in COIN \rightarrow \mathbb{N} \)
axm6:  \( VALUE(ONE) = 1 \)
axm7:  \( VALUE(TWO) = 2 \)
axm8:  \( VALUE(FIVE) = 5 \)
axm9:  \( COINS = COIN \rightarrow \mathbb{N}_1 \)
axm10: \( COINSVALUE \in COINS \rightarrow \mathbb{N} \)
axm11: \( \forall cs \cdot cs \in COINS \Rightarrow COINSVALUE(cs) = cs(ONE) + cs(TWO) \ast 2 + cs(FIVE) \ast 5 \)
axm12: \( SUBCOINS \in COINS \times COINS \rightarrow BOOL \)
axm13: \( \forall c_1, c_2 \cdot c_1 \in COINS \land c_2 \in COINS \Rightarrow SUBCOINS(c_1 \mapsto c_2) = \text{bool}(dom(c_1) \subseteq dom(c_2) \land \forall c \cdot c \in dom(c_1) \Rightarrow c_1(c) \leq c_2(c)) \)

END

Figure 2: Coin_ctx
MACHINE Vending
SEES Coin_ctx
VARIABLES
    coinbox
topay
paid
change
INVIANTS
    inv1: coinbox ∈ COINS
    inv2: topay ∈ N
    inv3: paid ∈ N
    inv4: change ∈ COINS
EVENTS
Initialisation
    begin act1: coinbox ∈ COIN ↦→ N₁
    act2: topay := 0
    act3: paid := 0
    act4: change := Ø
    end
Event GiveChange ≜
    when grd1: paid ≥ topay
    end
END

Figure 3: Vending machine

CONTEXT Seq_ctx
SETS
    TOKEN
CONSTANTS
    SEQ
AXIOMS
    axm1: SEQ = \{ s · s ∈ N₁ ↦→ TOKEN ∧ finite(s) ∧ dom(s) = 1 .. card(s)|s \}
THEOREM
    thm1: ∀ s · s ∈ SEQ ⇒ s ∈ dom(s) ↦→ TOKEN
END

Figure 4: Seq_ctx
MACHINE Seq
SEES Seq_ctx
VARIABLES
  seq
  size
  tokens
INVARIANTS
  inv1: seq ∈ SEQ
  inv2: size = card(seq)
  inv3: tokens = ran(seq)
EVENTS
  Initialisation
  begin
    act1: seq := ∅
    act2: size := 0
    act3: tokens := ∅
  end
END

Figure 5: Sequence machine

CONTEXT List_ctx
EXTENDS Seq_ctx
CONSTANTS
  ITER
  CLOSE
AXIOMS
  axm1: ITER ∈ (TOKEN ↔ TOKEN) → (N → (TOKEN ↔ TOKEN))
  axm2: ∀ r. r ∈ TOKEN ↔ TOKEN ⇒ ITER(r)(0) = id(TOKEN)
  axm3: ∀ r. n. r ∈ TOKEN ↔ TOKEN ∧ n ∈ N₁ ⇒ ITER(r)(n) = (ITER(r)(n - 1); r)
  axm4: CLOSE ∈ (TOKEN ↔ TOKEN) → (TOKEN ↔ TOKEN)
  axm5: ∀ r. r ∈ TOKEN ↔ TOKEN ⇒ id(TOKEN) ⊆ CLOSE(r)
  axm6: ∀ r. r ∈ TOKEN ↔ TOKEN ⇒ (CLOSE(r); r) ⊆ CLOSE(r)
END

Figure 6: List_ctx
MACHINE SeqR
REFINES Seq
SEES List_ctx
VARIABLES
  seq
  first
  last
  next
INvariants
  inv1: size ≠ 0 ⇒ first ∈ tokens
  inv2: size ≠ 0 ⇒ last ∈ tokens
  inv3: next ∈ tokens \ {last} ⇒ tokens \ {first}
  inv4: size ≠ 0 ⇒ first = seq(1)
EVENTS
Initialisation
  begin act1: seq := ∅
    act4: first ∈ TOKEN
    act5: last ∈ TOKEN
    act6: next := ∅
  end
END

Figure 7: Sequence refinement machine
CONTEXT RailTicket_ctx

SETS

STATION

CONSTANTS

maxtickets
REALSTATION
PRICE
NoWhere
TownHall
Central

AXIOMS

axm7: maxtickets ∈ N
axm6: REALSTATION = STATION \ {NoWhere}
axm1: PRICE ⊆ N
axm2: STATION = \{NoWhere, TownHall, Central\}
axm3: NoWhere ≠ TownHall
axm4: NoWhere ≠ Central
axm5: TownHall ≠ Central

END
MACHINE TicketMachine
SEES RailTicket_ctx
VARIABLES

stations
ticketprice
tickets
chosen chosen station
nochosen number of tickets required
topay amount left to pay
paid amount paid
moneybox all money paid

INVARIANTS

inv1: stations ⊆ REALSTATION
inv2: finite(stations)
inv3: ticketprice ∈ stations ↦ PRICE
inv4: tickets ∈ stations → 0..maxtickets
inv5: chosen ∈ stations ∪ {NoWhere}
inv6: topay ∈ Z
inv7: nochosen ∈ N
inv8: paid ∈ N
inv9: moneybox ∈ N
inv10: paid ≤ moneybox
inv11: topay ≤ ticketprice(chosen) * nochosen ∧ chosen ∈ stations
inv12: nochosen ≤ tickets(chosen) ∧ chosen ∈ stations

EVENTS

Initialisation

begin act1: stations := Ø
act2: ticketprice := Ø
act3: tickets := Ø
act4: chosen := NoWhere
act5: topay := 0
act6: nochosen := 0
act7: paid := 0
act8: moneybox := 0
end
**Event**  \( InitPrice \) \( \triangleq \) \( \text{Set initial price} \)

\[
\begin{align*}
\text{any station} & \quad \text{price} \\
\text{when } \text{grd1: station} & \in \text{REALSTATION} \\
\text{grd3: price} & \in \text{PRICE} \\
\text{then act1: ticketprice(station) := price} \\
\end{align*}
\]

**Event**  \( ChangePrice \) \( \triangleq \) \( \text{Change price} \)

\[
\begin{align*}
\text{any station} & \quad \text{price} \\
\text{when } \text{grd1: station} & \in \text{REALSTATION} \\
\text{grd2: price} & \in \text{PRICE} \\
\text{grd3: station} & \in \text{stations} \\
\text{then act1: ticketprice(station) := price} \\
\end{align*}
\]

**Event**  \( AddTickets \) \( \triangleq \) \( \text{Add more tickets} \)

\[
\begin{align*}
\text{any station} & \quad \text{count} \\
\text{when } \text{grd1: station} & \in \text{REALSTATION} \\
\text{grd2: count} & \in \mathbb{N}_1 \\
\text{grd3: station} & \in \text{stations} \\
\text{grd4: tickets(station) + count} & \leq \text{maxtickets} \\
\text{then act1: tickets(station) := tickets(station) + count} \\
\end{align*}
\]
Event Choose =

any station
number
when grd2: station ∈ REALSTATION
  grd3: number ∈ $\mathbb{N}_1$
  grd1: station ∈ stations
  grd4: chosen ∈ stations
  grd5: chosen $\neq$ NoWhere
  grd6: nochosen ≤ tickets(chosen)
then act1: nochosen := nochosen + number
  act2: chosen := station
end

Event Pay =

any amount
when grd1: amount ∈ $\mathbb{N}_1$
then act1: topay := topay − amount
  act2: paid := amount
  act3: moneybox := moneybox + amount
end

Event Cancel =

when grd1: chosen ∈ stations
then act1: chosen := NoWhere
  act2: paid := 0
  act3: topay := 0
  act4: nochosen := 0
  act5: moneybox := moneybox − paid
end
CONTEXT  TrafficLights_ctx

SETS
  LIGHTS
  DIRECTION

CONSTANTS
  Red
  Green
  Amber
  CONFLICT
  adir

AXIOMS
  axm1: LIGHTS = \{Red, Green, Amber\}
  axm2: Red ≠ Green
  axm3: Red ≠ Amber
  axm4: Green ≠ Amber
  axm5: finite(DIRECTION)  DIRECTION is a finite set of directions
  axm6: CONFLICT ∈ DIRECTION ⇔ DIRECTION  CONFLICT relates conflicting directions
  axm7: CONFLICT ∩ id(DIRECTION) = ∅  a direction cannot conflict with itself
  axm8: CONFLICT⁻¹ = CONFLICT  conflicts are symmetric
  axm9: adir ∈ DIRECTION

END

Figure 8: Traffic Light Context machine

Event  GiveTickets ≜
  when grd1: chosen ∈ stations
    grd3: tickets(chosen) − nochosen ≤ maxtickets
    grd4: tickets(chosen) − nochosen ≥ 0
    grd2: nochosen ≤ tickets(chosen)
  then act1: tickets(chosen) := tickets(chosen) − nochosen
          act2: nochosen := 0
          act3: chosen := NoWhere
          act4: topay := 0
          act5: paid := 0

END
MACHINE  ChangeLight
SEES   TrafficLights_ctx

VARIABLES
  lights

INvariants
  inv1: lights ∈ DIRECTION → {Red, Green}
  inv2: ∀d· d ∈ DIRECTION ∧ lights(d) = Green ⇒ lights[CONFLICT[[d]]] ⊆ {Red}

Safety

EVENTS

Initialisation
  begin act1: lights : |lights' ∈ DIRECTION → {Red, Green} ∧ ∀d· d ∈ DIRECTION ∧ lights'(d) = Green ⇒ lights'[CONFLICT[[d]]] ⊆ {Red} |
  end

Event  ToGreen ⇐
  begin act1: lights := lights ⊥ {adir → Green} ⊥ {CONFLICT[[adir]] × {Red}}
  end

END

Figure 9: ChangeLight machine
MACHINE ChangeLightR1
REFINES ChangeLight
SEES TrafficLights_ctx

VARIABLES

xlights Extended lights, Red, Green and Amber lights
delay delay between Amber and Red or Red and Green

INvariants

inv1: xlights ∈ DIRECTION → LIGHTS
inv2: ∀d· d ∈ DIRECTION ∧ xlights[d] ⊆ {Green, Amber}
     ⇒ xlights[CONFLICT[d]] ⊆ {Red}
inv3: CONFLICT[adir] ≪ lights = CONFLICT{adir} ≪ xlights
     Only change lights for adir and CONFLICT{adir}

inv4: xlights(adir) = Green ⇒ lights = xlights
inv5: delay ⊆ DIRECTION

EVENTS

Initialisation

begin with lights′ : lights′ = xlights′
act1: xlights : xlights[d] ∈ DIRECTION → {Red, Green}
     ∧ (∀d· d ∈ DIRECTION
     ∧ xlights′(d) = Green ⇒ xlights′[CONFLICT[d]] ⊆ {Red})
act2: delay := ∅
end

Event ToGreen ≜
refines ToGreen

when grd1: xlights[CONFLICT{adir}] ⊆ {Red}
grd2: adir ∉ delay
then act1: xlights := xlights ≪ {adir ⇒ Green}
end
**Event**  \( ToAmber \) \( \triangleq \)

any dir
when \( grd1 \): \( dir \in \text{DIRECTION} \)
\( grd2 \): \( dir \in \text{CONFLICT}\{\text{adir}\} \)
\( grd3 \): \( xlights(dir) = \text{Green} \)
\( grd4 \): \( xlights(adir) \neq \text{Green} \)
\( grd5 \): \( dir \notin \text{delay} \)
then act1: \( xlights(dir) := \text{Amber} \)
act2: \( delay := delay \cup \{dir\} \)
end

**Event**  \( ToRed \) \( \triangleq \)

any dir
when \( grd1 \): \( dir \in \text{DIRECTION} \)
\( grd2 \): \( dir \in \text{CONFLICT}\{\text{adir}\} \)
\( grd3 \): \( xlights(dir) = \text{Amber} \)
\( grd4 \): \( dir \notin \text{delay} \)
\( grd5 \): \( xlights(adir) \neq \text{Green} \)
then act1: \( xlights(dir) := \text{Red} \)
act2: \( delay := delay \cup \{adir\} \)
end

**Event**  \( Delay \) \( \triangleq \)

any dir
when \( grd1 \): \( dir \in \text{delay} \)
then act1: \( delay := delay \setminus \{dir\} \)
end

**VARIANT**

\[ 4 * \text{card}(xlights^{-1}\{\text{Green}\}) + 2 * \text{card}(xlights^{-1}\{\text{Amber}\}) + \text{card}(delay) \]

**END**
1 Answers

1.1 Multiple Choice

1 A, E
2 B, D, E
3 D
4 A, B
5 A, E
6 A, B, C, D, E
7 A, C
8 B, C, E
9 A, B
10 B, C, D
11 E
12 C, D
13 B
14 A, B, C, E
15 D, E
16 B, D
17 D
18 A, E
19 A, B, E
20 B, C
21 E
22 A, C, D, E
23 A, D
24 B, D, E
25 A, B, C, D

1.2 Short Answer

26) Given COIN is a set of coin denominations, for example

\[ COIN = \{ONE, TWO, FIVE\} \]

where each coin denomination has a value given by VALUE

\[ VALUE \in COIN \rightarrow \text{NAT}_1 \]

(assuming that no coin has value 0)

then we can have a set of functions COINS representing a bag of COIN — that is a collection of multiple instances of the elements of COIN

\[ COINS \equiv COIN \rightarrow \mathbb{N} \]

Now we want to be able to determine the value of a bag of COIN

\[ COINSVAL \in COINS \rightarrow \mathbb{N} \]
the following definition of COINSVAL is for a general finite set COIN where we don’t know the elements of the set.

\[ \forall cs \cdot cs \in COINS \land \text{ran}(cs) = 0 \Rightarrow \text{COINSVAL}(cs) = 0 \]

\[ \forall cs, c \cdot cs \in COINS \land c \in cs \land cs(c) \neq 0 \Rightarrow \]

\[ \text{COINSVAL}(cs) = cs(c) \ast \text{VALUE}(c) + \text{COINSVAL}(cs \setminus \{c \mapsto 0\}) \]

This is a more general form of COINSVALUE accidentally included in the sample paper.

27) • You would need to add a parameter change
• and guards
  - change \in COINS
  - COINSVAL(change) = paid - topay
  - and you need a guard that says that such a bag of coins exists in the coinbox. This can be done formally using SUBCOINS

28) each element of sequence is in the list
\[ \forall i \cdot i \in 0 \ldots \text{size} \Rightarrow (\text{ITER}(\text{next})(i - 1))(\text{first}) = \text{seq}(i) \]
Explanation: (\text{ITER}(\text{next})(i - 1)) applies next \(i - 1\) times, and that is applied to first.

all the elements in the list are exactly the elements of tokens
\[ \text{CLOSE}(\text{next})[\text{first}] = \text{tokens} \]
Explanation: if you follow all the paths form first you get exactly all of the elements of tokens.

29) No answer.

30) This question is too difficult and would never be asked.