Informatics Institute of Technology

IIT School of Computing

BEng/ BEng (Hons) Software Engineering

Level 6 - Semester 1 - Examination

6SENG005C - Formal Methods

Time Allowed: 90 Minutes January 2024

ADDITIONAL MATERIALS

Summary of the B-Method's Abstract Machine Notations (AMN)

INSTRUCTIONS TO CANDIDATES

1. This paper contains 10 questions on 7 pages (including this page).
2. The total marks obtainable for this examination is 100. The marks assigned for each question are included in square brackets.
3. This is a closed book examination.
4. Answer **ALL** questions.
5. Use the **ANSWER BOOKLETS** to answer the questions.
6. Start to answer **a new question** on **new page**.
7. If a page is not printed, please inform the supervisor immediately.

———————- END OF INSTRUCTIONS ———————-

Question 1 [Total Marks Allocated: 10 Marks]

Consider the following sets.

Vegetables = {Beet, Eggplant, Pumpkin, Potatoe, Pepper, Radish}
RedVegetables <: Vegetables & RedVegetables = {Beet, Pepper, Radish}
YellowVegetables <: Vegetables & YellowVegetables = {Pepper, Pumpkin}

Evaluate the following expressions. [2 marks each]

a) RedVegetables /\ YellowVegetables

b) RedVegetables - YellowVegetables

c) card(RedVegetables \/ YellowVegetables)

d) card({RedVegetables, YellowVegetables})

e) POW(YellowVegetables)

Question 2 [Total Marks Allocated: 10 Marks]

Consider the following sets.

A <: NAT & $A = \{2,4,6,8\}$
B <: NAT & $B = \{1,3,5,7,9\}$

For each of the following formulas, explain briefly what they mean and decide which ones are true. [2 marks each]

a) $(A <: B)$ OR $(B <: A)$

b) $!(n).(n : A => n + 1 : B)$

c) $\#(n).(n : B \ \& \ n > 7)$

d) $!(n).(n : A => \#(m).(m : B \ \& \ m > n))$

e) $\#(n).(n : A \ \& \ !(m).(m : B => m > n))$

Question 3 [Total Marks Allocated: 10 Marks]

Consider the following sets,

Fruit = {Apple, Banana, Cherry, Avocado, Grape}
Colour = {Green, Red, Yellow}
Part = {Peel, Pit, Pulp}

and the relations

colours : Fruit < − > Colour & colours = {Apple |− >Green, Apple|− > Red, Apple|− > Yellow, Banana|− >Yellow, Cherry|− >Red, Avocado|− >Green, Grape|− >Green, Grape|− >Red}

inedible : Fruit < − > Part & inedible = {Banana|− >Peel, Cherry|− >Pit, Avocado|− >Pulp}

Determine the following. [2 marks each]

a) The kind of relation of **inedible** (Is it a (partial) function? If so, is it injective and/or surjective?)

b) dom(inedible)

c) inedible∼

d) (inedible∼ ; colours)

e) colours[{Apple}]

Question 4 [Total Marks Allocated: 10 Marks]

Consider the following sequences.

X : seq(NAT) & X = [8,2,3]
Y : seq(NAT) & Y = [1,5,2]

Determine the following. [2 marks each]

a) Y ˆ X

b) rev(X)

c) Y /|\ 2

d) X < − 5

e) tail(Y)

Question 5 [Total Marks Allocated: 10 Marks]

Write a B machine called Committee satisfying the following requirements:

It allows us to seat the committee members Anne, Marry, Jack, Peter, and Marlon at a table with 5 places. Initially none of them are seated.

Some committee members dislike each other. Marry dislikes both Anne, and Jack, and Peter dislikes Marlon (and the feelings are mutual).

It must allow us to try to assign any committee member who is not yet seated to the next (i.e. lowest-numbered) free seat. However, they must not end up sitting next to a committee member they dislike. If seating any committee member would result in such a pair, the operation should instead report a failure.

Question 6 [Total Marks Allocated: 10 Marks]

Given the following B machine:

```
MACHINE TwoSets

    VARIABLES
       First, Second

   INVARIANT
      First <: NAT & Second <: NAT

   INITIALISATION
      First := {} || Second  := {}

   OPERATIONS
      addFirst(nn) =
      PRE nn : NAT
      THEN
            First := First \/ {nn}
      END;

      addSecond(nn) =
      PRE nn : NAT
      THEN
            Second := Second \/ {nn}
      END
END // TwoSets
```

Add an operation lastUnique which returns the largest element of **first** which is not in **second**. Make sure to use a suitable precondition if needed.

Question 7 [Total Marks Allocated: 10 Marks]

Consider this incomplete machine for a simple "car bingo" game:

```
MACHINE CarBingo

SETS
      BRANDS = {BMW, Citroen, Ferrari, Fiat, Ford,
                          Renault, Rover};
      COLOURS = {Black, White, Red, Orange, Yellow,
                          Green, Blue, Purple};
      RESPONSE = {Yes, No}

VARIABLES
      seenCars
```

a) The seenCars variable should keep track of the cars (i.e. combinations of a brand and a colour) seen so far. Provide the invariant and initialisation for it. [2 marks]

b) Add an operation coloursSeen(brand) which returns the number of colours that the given car brand has been seen in. For example, if we have seen a blue, green, and yellow Ferrari so far then coloursSeen(Ferrari) should return 3. [4 marks]

c) Add an operation hasWinningBrand which returns Yes if we have seen cars of at least 3 different colours of any one brand, and No othewise. [4 marks]

Question 8 [Total Marks Allocated: 10 Marks]

The following is an example of the general structure of an abstract machine's *operation*:

```
xx <-- oo( yy ) =
         PRE  PC
         THEN
              Body
         END
```

Explain the role that each part (xx, oo, yy, PC, Body) plays in specifying the operation.

[2 marks each]

Question 9 [Total Marks Allocated: 10 Marks]

Consider the following incomplete B machine that specifies a single train's rail route from Colombo to Matara. The route is a sequence of capital cities, starting from the departure city to the destination city.
It is a one-way rail journey, so no city should occur on the route more than once.

```
MACHINE CeylonTrains

 SETS
   CITY = { Colombo, Dehiwala, Panadura, Wadduwa, Kalutara,
             Ambalangoda, Hikkaduwa, Galle, Matara} ;
   MESSAGE = { City_Added, Departure_City_Removed,
              ERROR_City_Already_In_Route, ERROR_Route_Empty,
              ERROR_Train_Route_Full}

VARIABLES
   railroute

INVARIANT
     // TO DO

INITIALISATION
     // TO DO

OPERATIONS
   report <-- AppendStationToRoute( city ) =
     // TO DO

   report <-- RemoveDepartureStationFromRoute =
     // TO DO

END // EuroTrains
```

Define the missing parts:

a) The INVARIANT defining the variable. [2 marks]

b) The INITIALISATION of the variable [2 marks]

c) The operation AppendStationToRoute which adds a city to the end of the route. A message should be output indicating that this was done successfully or if not indicating what the error was. [3 marks]

d) The operation RemoveDepatureStation which removes the first (departure) city station from the route. A message should be output indicating that this was done successfully or if not indicating what the error was. [3 marks]

Question 10 [Total Marks Allocated: 10 Marks]

Consider the following incomplete B machine that specifies a “Nought and Crosses” game, that is a 4×4 grid where the two players take turns to place a nought (O) or a cross (X).

```
MACHINE NoughtsCrosses

SETS
      SYMBOLS = { XX, OO } ;
      REPORTS = { Not_On_Grid, Occupied, Placed_Symbol}

CONSTANTS
      Grid

PROPERTIES
      // TO DO

VARIABLES
      Noughts, Crosses
//Keep track of the positions of all O and all X

INVARIANT
      // TO DO

INITIALISATION
      // TO DO
```

Complete this machine by defining the following: [2 marks each]

a) The PROPERTIES defining the Grid constant.

b) The INVARIANT for the variables

c) The INITIALISATION for the variables

d) An operation $report <--placeSymbol(xx, yy, symbol)$ which tries to place the symbol (OO or XX) in the square and reports whether it did so. If the mark can not be placed because the square is occupied or not on the grid then it should output an error message.

e) An operation $squares <-- freeSquares$ which returns the set of free squares, i.e. those that do not have a OO or XX assigned to them.

——————- END OF PAPER ——————-