SCHOOL OF MATHEMATICAL AND COMPUTER SCIENCES

Computer Science

F28FS2

Formal Specification Mock exam

Semester 2 201314

Sometime before 22 May 2014

Duration: As long as you like (actual exam: 2 hours)

ANSWER THREE QUESTIONS

Some words on using this mock paper

Every concept in this paper appears in the lecture notes and exercises.

I have pitched the difficulty level of this mock above what you will face in the exam; however, exam conditions always make things seem more difficult, because of the stress.

If you understand and can do these questions, then you are certain to get a decent grade in the exam.

- You *must* attempt this entire paper *before* looking at the answers. Have you attempted the paper yet?

  ANSWER:
  If you see this text, you are looking at the version with model answers. Close this document and try to version without model answers, first.

- Then look up answers.
- Then do the paper again.
- Repeat until perfect.

Good luck.

1. Recall that $\mathbb{N}$ is the type of natural numbers, with elements $\{0, 1, 2, \dots\}$.

(a) Using precise English or correct Z notation, give examples of elements of the types below. Your examples must not be empty—if your answer is $\{\}$ or $\varnothing$ then it will score zero marks. Examples that are not clear or precise may also score zero marks.

- $\mathbb{P}(\mathbb{N} \times \mathbb{N})$.
  ANSWER:
  $\{0 \mapsto 0\}$
- $(\mathbb{P}\mathbb{N}) \times (\mathbb{P}\mathbb{N})$.
  ANSWER:
  $\varnothing \mapsto \varnothing$
- $\mathbb{N} \to \mathbb{N}$.
  ANSWER:
  The identity function, mapping $x : \mathbb{N}$ to itself.
- $\mathbb{P}((\mathbb{P}\mathbb{N}) \times (\mathbb{P}\mathbb{N}))$.
  ANSWER:
  $\{\varnothing \mapsto \varnothing\}$

(8)

(b) $f : \mathbb{N} \to \mathbb{N}$ is **bijective** when every element in $\mathbb{N}$ is mapped to by some unique element of $\mathbb{N}$ (so if $y : \mathbb{N}$ then there is some unique $x : \mathbb{N}$ such that $f(x) = y$).

Describe using precise English or Z notation or otherwise, one example of some bijective function in $\mathbb{N} \to \mathbb{N}$. (2)

ANSWER:
The identity function mapping $x : \mathbb{N}$ to itself. Other answers I would accept include $\lambda\, x : \mathbb{N}.x$ (λ-calculus notation), $\{0 \mapsto 0, 1 \mapsto 1, 2 \mapsto 2, \dots\}$ (semiformal English), and $\{x : \mathbb{N} \mid x \mapsto x\}$ (full Z notation).
More obscure answers include e.g. $\{0 \mapsto 1, 1 \mapsto 0, 2 \mapsto 3, 3 \mapsto 2, \dots\}$. But why bother?

(c) Write a Z predicate $bijective(f)$ which expresses that f is bijective. You may assume all standard connectives and quantifiers without comment, and you may assume function application, writing $f(x)$ without explanation. (4)

ANSWER:
$bijective(f) \stackrel{def}{=} \forall\, y : T \bullet \exists_1\, x : S \bullet f(x) = y$. *This just repeats in Z the English specification of bijectivity above.*
This would also be fine: $bijective(f) \stackrel{def}{=} \forall\, y : T \bullet (\exists\, x : S \bullet f(x) = y \land (\forall\, x' : S \bullet f(x') = y \Rightarrow x = x'))$.

(d) Suppose $S, T \subseteq \mathbb{N}$. We say that S and T **have the same size** when there exists a bijective function $f : S \to T$.

Specify in Z the set of all pairs of equally-sized sets of natural numbers: your answer must have type $\mathbb{P}(\mathbb{P}\mathbb{N} \times \mathbb{P}\mathbb{N})$.

You may assume a predicate $bijective$ (i.e. you may assume your answer to part (c) above). (3)

ANSWER:
$\{X \mapsto Y : \mathbb{PN} \times \mathbb{PN} \mid \exists f : X \to Y \bullet \mathit{bijective}(f)\} : \mathbb{P}(\mathbb{PN} \times \mathbb{PN})$.

(e) Write $\mathbb{R}$ for the type of real numbers. If $x, y : \mathbb{R}$ then the **distance** between x and y, written $|x-y|$, is the unique non-negative element of $\{x-y, y-x\}$. A subset $S \subseteq \mathbb{R}$ is **dense** in $\mathbb{R}$ when for every $x : \mathbb{R}$ and $\epsilon : \mathbb{R}$ there exists a $y \in S$ that is less than ϵ distant from x, so that $|x-y| < \epsilon$ (for instance, the rational numbers $\mathbb{Q}$ are dense in $\mathbb{R}$, and the integers $\mathbb{Z}$ are *not* dense in $\mathbb{R}$).

Express in Z, giving full types, the set of dense subsets of $\mathbb{R}$. You may assume $|x-y|$ without specifying it. (3)

ANSWER:
$\{S : \mathbb{PR} \mid \forall x, \epsilon : \mathbb{R} \bullet \exists y : S \bullet |x-y| < \epsilon\} : \mathbb{PPR}$.
Notice the type: you get a point just for writing $\mathbb{PPR}$ (and lose a point for forgetting to do so). If I write "giving full types", then I mean it.

2. Define types $STATE ::= Live \mid Dead$ and $CELL = \mathbb{Z} \times \mathbb{Z}$.

Conway's game of life is played on an infinite two-dimensional grid of cells which can either be *live* or *dead*. Model this as a function $boardState : CELL \to STATE$.

(a) Write a schema $BoardState$ with precisely one schema variable $boardState$. The state predicate should reflect that every possible value of $boardState$ is a valid board state. (2)

ANSWER:

$$\begin{array}{|l} BoardState \\ \hline boardState : CELL \to STATE \\ \hline \end{array}$$

(b) Write the schemas $\Delta BoardState$ and $\Xi BoardState$ in full. (4)

ANSWER:

$$\begin{array}{|l} \Delta BoardState \\ \hline boardState, boardState' : CELL \to STATE \\ \hline \end{array}$$

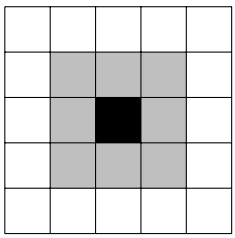
$$\begin{array}{|l} \Xi BoardState \\ \hline \Delta BoardState \\ \hline boardState' = boardState \\ \hline \end{array}$$

(c) Write a schema *InitBoard* which inputs a variable $seed? : CELL \to STATE$, and sets the board up according to $seed?$. (2)

ANSWER:

$$\begin{array}{|l} InitBoard \\ \hline \Delta BoardState \\ seed? : CELL \to STATE \\ \hline boardState' = seed? \\ \hline \end{array}$$

(d) The cell (x, y) is **adjacent** or a **neighbour** to the cell (x', y') when they share an edge or a corner. For instance, the grey squares below are adjacent to the black square:



Write a predicate $adjacent$ on $CELL \times CELL$ such that $adjacent((x, y), (x', y'))$ is true precisely when (x, y) and (x', y') are adjacent. (3)

ANSWER:
Here's how I'd do it:

$$adjacent((x, y), (x', y')) \stackrel{def}{=} (x-x')^2 + (y-y')^2 \in \{1, 2\}$$

or

$$adjacent((x, y), (x', y')) \stackrel{def}{=} 1 \leq (x-x')^2 + (y-y')^2 \leq 2$$

(e) Specify $liveNeighbours : CELL \to (CELL \to STATE) \to \mathbb{N}$ which given any c and $boardState$ will return the number of neighbours of c for which $boardState(c)$ is $Live$. Take care to put in correct quantifiers as appropriate. (4)

ANSWER:
$\forall\, c : CELL \bullet liveNeighbours(c) = \#\{c' : CELL \mid adjacent(c, c') \land boardState(c') = Live\}.$

(f) The state of the board evolves as a clock ticks. At each evolution, the state after a clock ticks is related to the state before, by the following transitions:

- A live cell with fewer than 2 live neighbours dies.
- A live cell with 2 or 3 live neighbours lives to the next generation.
- A live cell with more than 3 live neighbours dies.
- A dead cell with exactly 3 live neighbours becomes a live cell.

Write a schema $BoardTransition$ specifying one step in the evolution of the board. (5)

ANSWER:

BoardTransition

$\Delta BoardState$

$$\begin{aligned}\forall\, c : CELL \bullet\ & (boardState(c) = Live \land liveNeighbours(c, boardState) < 2) \Rightarrow boardState'(c) = Dead \\ & (boardState(c) = Live \land liveNeighbours(c, boardState) \in \{2, 3\}) \Rightarrow boardState'(c) = Live \\ & (boardState(c) = Live \land liveNeighbours(c, boardState) > 3) \Rightarrow boardState'(c) = Dead \\ & (boardState(c) = Dead \land liveNeighbours(c, boardState) = 3) \Rightarrow boardState'(c) = Live \\ & (boardState(c) = Dead \land liveNeighbours(c, boardState) \neq 3) \Rightarrow boardState'(c) = Dead\end{aligned}$$

The interested student can find a Life simulator online; search for *Golly*.

3. (a) You are asked to produce one example each of programs that satisfy the specifications $\top$ and $\bot$ ('true' and 'false'). What do you answer? (2)

ANSWER:
The noop program (which does nothing) is an example of a program that makes $\top$ be 'true'. No program makes $\bot$ be 'true', so no example exists; $\bot$ is unsatisfiable.

(b) Give one concrete example of *informal* specification and one concrete example of *formal* specification, taken from programming practice. (2)

ANSWER:
Examples of informal specification: a comment in the code or suggestive variable names or file names. Example of formal specification: types in C, program headers, the Z language, or indeed the code of computer programming languages.

(c) Explain Goedel's incompleteness theorem and its relevance to Z specification. (4)

ANSWER:
Goedel's incompleteness theorem is a mathematical proof that it is impossible, in the general case, to write a program that will input a pair (code of program,Z specification) and say whether the code satisfies the spec. So fully automated machine checking of code correctness is mathematically impossible in the general case. Nevertheless, useful special cases exist where full automation is possible, e.g. type checking.

A **code audit** is when code is checked line by line by people who did not write that code, asking: "Why is this line here. What does it do? Is it correct?".

(d) Explain to what extent formal specification in Z and a code audit are competing, or complementary, methods of reducing errors in code. (4)

ANSWER:
I'm looking for evidence that the student appreciates what Z is and what its limitations are. So for instance: A code audit might benefit from a Z specification of what the code is supposed to do. In practice such specifications are more likely to be written in English. However, if what the program does is highly mathematical (rocket guidance, floating point arithmetic, logic reasoning e.g. on package dependencies, database operations) then a more formal mathematical specification might well form part of that English.
Code audits are expensive and require qualified staff, but they may still be less expensive than full Z specification.
Code audits are not *good for automatically verifying large tracts of code. You could not, for instance, fully audit a microchip design or an entire OS. On the other hand you* can *use formal specification to specify common bugs and scan large portions of code looking for them. (In a sense, this is what tools such as* `grep` *do.)*
Code audits will not necesssarily help discover non-obvious interactions of widely separated systems that humans might miss for the sheer size of the interactions. That is, a code audit might verify that program A increments a counter and program B decrements a counter, but that is no guarantee that A and B might not then interact to loop indefinitely.

(e) "Better to ship buggy code today, than perfect code tomorrow—we can fix the bugs in updates."

Discuss, giving one concrete example where this might be appropriate, and one concrete example where this is might be inadvisable, with justification for each. (4)

ANSWER:
This is right if you are e.g. building something like Facebook and it is critical to build a user base early and benefit from networking effects before your competitors do. Time is critical, and safety is not.
This is wrong if you are e.g. Jaguar designing the embedded software for a new model of car for the Indian market. A buggy product here could be deadly, and also it would sully Jaguar's reputation for quality in India and so could damage the brand there for a generation. Time is less important, safety is more important, and maintaining reputation is key to success.

(f) "The Heartbleed Bug happened because of a stupid programmer."

Discuss, giving four distinct and clear reasons, at least one of which must be for the position above, and one must be against it. (4)

ANSWER:
It would be unfair to label the programmer as 'stupid', though they certainly made a significant and elementary programming mistake.
However, humans do make such mistakes, and the rigorous proofing, auditing, and validation which such important code should have been subjected to, did not trap this error. To finger one programmer for this mistake would be unfair and would not address the real problems.
And anyway: why is such code being written in a language (C) that is so notoriously vulnerable to memory overflow errors? And why is the code not run sandboxed by design? And so on. The whole programming industry has to take responsibility for that one.
Heartbleed is a symptom of design, culture, and systems problems, not the fault of a lone programmer.

4. Recall that an integer **matrix** is an m by n table of integers. For instance,

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & -1 & 2 \end{pmatrix}$$

is a 2×3 matrix ($m = 2$ rows and $n = 3$ columns).

Model matrixes using an ML type `matrix = (int list) list`. So the matrix above would be represented as `val M = [ [1,2,3], [0,-1,2] ]`.

You may assume a function `length : 'a list -> int` which returns the length of its argument, and `hd : 'a list -> 'a` returns the head of a list.[1]

(a) 1. To what integer does `length [ [1,2,3], [0,-1,2] ]` evaluate? (1)

ANSWER:
2

2. To what integer does `length (hd ([ [1,2,3], [0,-1,2] ]))` evaluate? (1)

ANSWER:
3

3. Does there exist some type `S` and `l:S list` such that `length l` will evaluate to `-1`? (1)

ANSWER:
No. Come on. Don't let me mess with your mind.

4. Write an example of a type `S` and an `l:S list` such that `length l` evaluates to `0`. (1)

ANSWER:
`int` *and* `[]` *(the empty list).*

(b) Write an ML program `shape : matrix -> int*int` such that if `M` is an $m \times n$ matrix, then `shape M` will return `(m,n)`.
We do not care about behaviour if `M` is not a matrix. (2)

ANSWER:
```
fun shape M = (length M,length(hd M));
```

(c) Write a program `makeList : int -> matrix` that inputs `n` and outputs a list of n zeroes. (3)

ANSWER:
```
fun makeList 0 = [] | makeList n = 0::(makeList (n-1));
```

(d) Write a program `makeMatrix : int -> int -> matrix` that inputs `m` and `n` and outputs an $m \times n$ matrix of zeroes. (3)

[1] Make sure you can implement basic list operations such as these, yourself.

ANSWER:

```
fun makeMatrix 0 n = [] | makeMatrix m n = (makeList n)::(makeMatrix (m-1)
n);
```

(e) Write a program `returnIndex : int -> int -> matrix -> int` that inputs `m` and `n` and `M` and outputs the integer entry at the mth row and nth column.
We do not care about behaviour if no such entry exists or if `M` is not a matrix. You are free to define helper functions if this is convenient. (4)

ANSWER:

```
fun listItem 1 (h::t) = h | listItem n (h::t) = listItem (n-1) t;
fun returnIndex 1 n M = listItem n (hd M)
| returnIndex m n M = returnIndex (m-1) n (tl M);
```

(f) Not every list of list of integers represents a matrix. Write an ML program `isMatrix : int -> int -> matrix -> bool` such that `isMatrix m n M` returns `true` if `M` is an $m \times n$ matrix, and `false` otherwise.
We do not care if your answer is more polymorphic than necessary (so works for `'a list list` instead of `int list list`). We do not care about behaviour if `m` or `n` are negative. (4)

ANSWER:

```
fun isMatrix 0 n l = (l=[])
| isMatrix m n [] = (m=0)
| isMatrix m n (h::t) = ((length h) = n) andalso isMatrix (m-1) n t;
```

Some might just write `isMatrix m n M = (shape M)=(m,n)`. *That is not a fully correct answer, since* `shape` *only gives correct results if it is given a matrix. Be careful: 4 marks towards the end of a question are unlikely to be attained so easily.*

END OF PAPER