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?

- 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.

- $\mathbb{P}(\mathbb{N} \times \mathbb{N})$.
- $(\mathbb{P}\mathbb{N}) \times (\mathbb{P}\mathbb{N})$.
- $\mathbb{N} \to \mathbb{N}$.
- $\mathbb{P}((\mathbb{P}\mathbb{N}) \times (\mathbb{P}\mathbb{N}))$. (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)

(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)

(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)

(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)

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

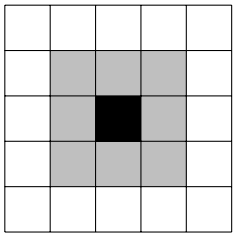
Conway's game of life is played on an infinite two-dimensional grid of cells which can either be *live* or *dead*. Model this as a function $boardState : CELL \to STATE$.

(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)

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

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

(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)

(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)

(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)

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)

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

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

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)

(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)

(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)

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)

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

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

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

(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)

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

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

(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)

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

(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)

END OF PAPER