The University of Nottingham

SCHOOL OF COMPUTER SCIENCE

A LEVEL 2 MODULE, SPRING SEMESTER 2016-2017

G54FOP FOUNDATIONS OF PROGRAMMING

Time allowed 2 hours

Candidates may complete the front cover of their answer book and sign their desk card but must NOT write anything else until the start of the examination period is announced

Answer all 4 questions

Marks available for sections of questions are shown in square brackets.

No calculators are permitted in this examination.

Dictionaries are not allowed with one exception. Those whose first language is not English may use a standard translation dictionary to translate between that language and English provided that neither language is the subject of this examination. Subject specific translation dictionaries are not permitted.

No electronic devices capable of storing and retrieving text, including electronic dictionaries, may be used.

DO NOT turn your examination paper over until instructed to do so

Question 1: [25%]

UNTYPED λ-CALCULUS

(a) [8%]

Explain what a reduction strategy is. Define the *call-by-name* and *call-by-value* reduction strategies.

Explain their difference by showing the reduction steps in the evaluation of the following term in both strategies:

$$(\lambda x.x\, x)\, ((\lambda y.y)\, z).$$

(b) [8%]

For each of the following λ-terms, state if it is normalizable and, in case it is, write its normal form.

1. $(\lambda x.x\, (\lambda y.y\, x))\, (\lambda u.u)$
2. $(\lambda x.\lambda y.y\, (x\ y))\ (\lambda z.z)\ (\lambda u.u\ u)$
3. $(\lambda x.\lambda y.x\, (y\, x))\ (\lambda z.z)\ (\lambda u.u\ u)$
4. $(\lambda x.\lambda y.(y\ x)\ y)\ (\lambda z.z)\ (\lambda u.\lambda v.u)$

(c) [9%]

Lists are constructed by using two combinators, nil for the empty list and cons for the construction of a new list from a head and a tail:

$$\begin{aligned}\mathsf{nil} &= \lambda f.\lambda x.x\\ \mathsf{cons}\, h\, t &= \lambda f.\lambda x.f\, h\, (t\, f\, x)\end{aligned}$$

Write the (normal form) representation of the list $[\overline{3}, \overline{2}, \overline{0}]$ in this encoding (you can leave the numerals unevaluated).

Explain in what sense lists are represented *as iterators*.

Write a λ-term sum that computes the sum of the elements of a list of numbers (you can assume the addition combinator plus to be given). It must satisfy:

$$\begin{aligned}\mathsf{sum}\ \mathsf{nil} &\rightsquigarrow^* \overline{0}\\ \mathsf{sum}\ (\mathsf{cons}\, h\, t) &\rightsquigarrow^* \mathsf{plus}\, h\, (\mathsf{sum}\, t)\end{aligned}$$

Question 2: [25%]

SIMPLY TYPED λ-CALCULUS

(a) [8%]

Define the notions of weak and strong normalization. Give an example in the untyped λ-calculus of a term that weakly normalizes but doesn't strongly normalize.

(b) [8%]

The following are incomplete terms in the simply typed λ-calculus with a base type o. Give types to the abstracted variables and state what the type of the term is.

1. $\lambda x :?.\ \lambda y :?.\ \lambda z :?.\ x\ y\ (z\ y)$
2. $\lambda u :?.\ \lambda v :?.\ v\ (u\ v)$
3. $\lambda x :?.x\,(\lambda z :?.z)$

(c) [9%]

Remember the definitions of the basic types of Booleans and Natural Numbers in $\lambda\!\rightarrow$:

$$\mathsf{Bool} = \mathsf{o} \to \mathsf{o} \to \mathsf{o}$$
$$\mathsf{Nat} = (\mathsf{o} \to \mathsf{o}) \to \mathsf{o} \to \mathsf{o}$$

Write terms of $\lambda\!\rightarrow$ that implements the zero test and conditional branching:

$$\mathsf{isZero} : \mathsf{Nat} \to \mathsf{Bool}, \qquad \mathsf{if} : \mathsf{Bool} \to \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat};$$

such that

$$\mathsf{isZero}\,\overline{0} \rightsquigarrow^* \mathsf{true} \qquad \mathsf{if\ true}\,n\,m \rightsquigarrow^* n$$
$$\mathsf{isZero}\,\overline{n+1} \rightsquigarrow^* \mathsf{false} \qquad \mathsf{if\ false}\,n\,m \rightsquigarrow^* m.$$

Question 3: [25%]

CoInductive Types

(a) [8%]

Define what an coalgebra for a functor F is.

What characterizes the *final* F-coalgebra? What is an *anamorphism*? [You can give the defining property of the anamorphism either by an equation or by drawing a commutative diagram.]

(b) [8%]

Recall the informal definition of the type of streams of bits:

$$\begin{aligned}&\mathsf{codata}\ \mathsf{BitStr} : \mathsf{Set}\\ &\quad \mathsf{c}_0 : \mathsf{BitStr} \to \mathsf{BitStr}\\ &\quad \mathsf{c}_1 : \mathsf{BitStr} \to \mathsf{BitStr}\end{aligned}$$

We use the notation $1 \lhd 0 \lhd \cdots$ for $\mathsf{c}_1\,(\mathsf{c}_0 \cdots)$.

Write the type as a final coalgebra: $\mathsf{BitStr} = \nu F$. What should the functor F be?

$$F\,X = ?$$

Let $\langle \nu F, \mathsf{out} \rangle$ be the final coalgebra. How do you define the constructors in terms of out? [Use the inverse of the coalgebra: $\mathsf{in} = \mathsf{out}^{-1}$.]

$$\mathsf{c}_0 = ? \qquad \mathsf{c}_1 = ?$$

(c) [9%]

Assume you have a primality test function $\mathsf{prime} : \mathsf{Nat} \to \mathsf{Bool}$.

Define the stream of *primality bits* of the natural numbers: the ith element of the stream is 1 if i is a prime number, 1 otherwise:

$$\begin{aligned}&\mathsf{prstr} : \mathsf{BitStr}\\ &\mathsf{prstr} = 0 \lhd 0 \lhd 1 \lhd 1 \lhd 0 \lhd 1 \lhd 0 \lhd 1 \lhd 0 \lhd 0 \lhd \cdots\end{aligned}$$

(Elements at indices $2, 3, 5, 7$ are 1 because $2, 3, 5, 7$ are primes.)

[Use an auxiliary function $\mathsf{prstr}_{\mathsf{from}} : \mathbb{N} \to \mathsf{BitStr}$, defined as the anamorphism of a coalgebra, that gives the primality bits from a given starting number.]

Question 4: [25%]

INDUCTIVE TYPES AND SYSTEM F

The type of binary trees with leaves labelled by elements of a type A is informally defined like this:

$$\begin{array}{l}\mathsf{data}\ \mathsf{Tree}_A : \mathsf{Set} \\ \quad \mathsf{leaf} : A \to \mathsf{Tree}_A \\ \quad \mathsf{node} : \mathsf{Tree}_A \to \mathsf{Tree}_A \to \mathsf{Tree}_A\end{array}$$

(a) [8%]

Write this type as an initial algebra: $\mathsf{Tree}_A = \mu F$. What should the functor F be?

$$F\,X = ?$$

Write down the introduction, elimination and reduction rules for μF.

(b) [8%]

Write a SYSTEM F realization of the type Tree_A. How are the constructors leaf and node implemented?

(c) [9%]

Write a SYSTEM F term that implements the function that computes the mirror image of a tree. It must satisfy the following typing and reduction properties:

$$\begin{array}{l}\mathsf{mirror} : \mathsf{Tree}_A \to \mathsf{Tree}_A \\ \mathsf{mirror}\,(\mathsf{leaf}\,a) \rightsquigarrow^* \mathsf{leaf}\,a \\ \mathsf{mirror}\,(\mathsf{node}\,t_1\,t_2) \rightsquigarrow^* \mathsf{node}\,(\mathsf{mirror}\,t_1)\,(\mathsf{mirror}\,t_2)\end{array}$$