THE UNIVERSITY OF CALGARY

FACULTY OF SCIENCE

FINAL EXAMINATION

COMPUTER SCIENCE 417

December, 2006 | Time: 2 hrs.

Instructions

The exam contains questions totalling 100 points. Answer all questions. This exam is closed book.

30 marks

1. Given the following datatype for terms:

```
data Term = Var Int
          | Node String [Term]
```

and following datatype for exceptions:

```
data Possibly a = Error String | Value a
```

(a) Explain what a *unifier* of two terms is.

(b) Explain what the *most general unifier* of two terms is.

(c) Create an instance of the exception monad for the possibly type.

(d) Write a function to implement the occurs check:

$$\texttt{check} :: \texttt{Int}- > \texttt{Term}- > \texttt{Possibly}(\texttt{Term})$$

using the `Possibly` monad to return an informative error.

(e) Write a function to find the most general unifier of two terms.

35 marks

2. (a) In λ-calculus (with respect to α, β equality) let:

$$\begin{aligned} \mathsf{S} &= \lambda xyz.xz(yz) \\ \mathsf{K} &= \lambda xy.x \\ \mathsf{I} &= \lambda x.x \end{aligned}$$

Show that

i. $\mathsf{SKK} = \mathsf{I}$;

ii. $(\mathsf{SII})(\mathsf{SII})$ does not have a normal form;

iii. $\lambda xy.y$ can be expressed using K and I.

(b) What is a fixed point combinator? Given an example of a fixed point combinator and show that it has the desired property.

(c) Explain how one may represent binary trees in the λ-calculus.

(d) Assuming that one has a representation of numbers and of their basic functions (such as addition and multiplication) describe how to encode the following recursive program in the λ-calculus:

```
gdc n m = if n*m = 0 then n+m
          else if n < m then gcd (m-n) n
          else if n > m then gcd (m-n) m
```

(e) Which of the following are true? Explain your reasoning.

- It is decidable whether a term in the λ-calculus is in normal form.
- It is decidable whether a term in the λ-calculus has a normal form.
- If a term in the λ-calculus does not have a normal form it is cannot be solvable.
- It is decidable whether a term in the λ-calculus which is in normal form is solvable.
- It is undecidable whether a term in the λ-calculus has a head normal form.
- It is even udecidable whether a term in the λ-calculus, which has a normal form, is equal to `true`.
- A rewriting system is always terminating.
- A rewriting system in which all critical divergences can be resolved is always confluent.
- In an orthogonal rewriting system a leftmost outermost reduction strategy will always find the normal form if one exists.
- In the simply typed λ-calculus (without fixed points) one can express all computable functions.

$$\frac{}{x : P, \Gamma \vdash x : P}\ \text{proj}$$

$$\frac{x : P, \Gamma \vdash t : Q}{\Gamma \vdash \lambda x.t : P \to Q}\ \text{abst}$$

$$\frac{\Gamma \vdash f : P \to Q \quad \Gamma \vdash t : P}{\Gamma \vdash (ft) : Q}\ \text{app}$$

Table 1: Type judgements

35 marks

3. (a) Given the type judgements in table 1 give the lambda term which corresponds to the following proof (in which $\Gamma := A, A \to B, B \to C$):

$$\dfrac{\dfrac{}{\Gamma \vdash B \to C}\ \text{proj} \qquad \dfrac{\dfrac{}{\Gamma \vdash A}\ \text{proj} \quad \dfrac{}{\Gamma \vdash A \to B}\ \text{proj}}{\Gamma \vdash B}\ \text{app}}{\Gamma \vdash C}\ \text{app}$$

(b) Using the judgements for type inference in table 2:

i. Show that the term, $(\lambda x.xx)$, cannot be typed in the simply typed lambda calculus.

ii. Show that $\mathsf{S} = \lambda xyz.xz(yz)$ can be typed in the simply typed lambda calculus and provide the type.

$$\frac{}{x : P, \Gamma \vdash x : Q} \rhd P = Q$$

$$\frac{x : P, \Gamma \vdash t : R}{\Gamma \vdash \lambda x.t : Q} \; P, R \rhd Q = P \to R$$

$$\frac{\Gamma \vdash f : R \quad \Gamma \vdash t : P}{\Gamma \vdash (ft) : Q} \; P, R \rhd R = P \to Q$$

Table 2: Type judegements with type equations