UNIVERSITY OF EDINBURGH

COLLEGE OF SCIENCE AND ENGINEERING

SCHOOL OF INFORMATICS

INFR11114 TYPES AND SEMANTICS FOR PROGRAMMING LANGUAGES

Thursday 15$^{\underline{\text{th}}}$ December 2016

14:30 to 16:30

INSTRUCTIONS TO CANDIDATES

Answer QUESTION 1 and ONE other question.

Question 1 is COMPULSORY.

All questions carry equal weight.

CALCULATORS MAY NOT BE USED IN THIS EXAMINATION

Year 4 Courses

Convener: I. Murray
External Examiners: A. Cohn, A. Donaldson, S. Kalvala

THIS EXAMINATION WILL BE MARKED ANONYMOUSLY

1. THIS QUESTION IS COMPULSORY

This question uses the library definition of `list` in Coq, which includes the functions `rev` and `app` (++). You may use any facts about `rev` and `app` that you find with `SearchAbout`, except in (b) you may not use `rev_app_distr`, which is the theorem you are trying to prove

Here is an informal definition of the predicate `pal`.

$$\texttt{pal\_empty}\frac{}{\texttt{pal []}} \qquad \texttt{pal\_unit}\frac{}{\texttt{pal [}x\texttt{]}} \qquad \texttt{pal\_step}\frac{\texttt{pal } xs}{\texttt{pal } (\texttt{[}x\texttt{]} \mathbin{++} xs \mathbin{++} \texttt{[}x\texttt{]})}$$

(a) Formalise the definition above. [*8 marks*]

(b) Prove the following.

$$\texttt{Theorem rev\_app\_distr}' : \forall (X : \texttt{Type})\ (xs\ ys : \texttt{list}\ X),$$
$$\texttt{rev}\ (xs \mathbin{++} ys) = (\texttt{rev}\ ys) \mathbin{++} (\texttt{rev}\ xs).$$

[*8 marks*]

(c) Prove the following.

$$\texttt{Theorem pal\_rev} : \forall (X : \texttt{Type})\ (xs : \texttt{list}\ X),$$
$$\texttt{pal}\ xs \to xs = \texttt{rev}\ xs$$

[*9 marks*]

2. ANSWER EITHER THIS QUESTION OR QUESTION 3

You will be provided with a definition of a simple imperative language in Coq. Mark any changes you make to the definition with `(* !!! *)`.

Consider a guarded command construct satisfying the following rules:

Evaluation:

$$\texttt{E\_GuardEnd}\;\frac{\begin{array}{c}\texttt{beval}\ st\ b_1 = \texttt{false}\\ \texttt{beval}\ st\ b_2 = \texttt{false}\end{array}}{\texttt{DO}\ b_1\ \texttt{THEN}\ c_1\ \texttt{OR}\ b_2\ \texttt{THEN}\ c_2\ \texttt{OD}/st \Downarrow st}$$

$$\texttt{E\_GuardLoop1}\;\frac{\begin{array}{c}\texttt{beval}\ st\ b_1 = \texttt{true}\\ c_1/st \Downarrow st'\\ \texttt{DO}\ b_1\ \texttt{THEN}\ c_1\ \texttt{OR}\ b_2\ \texttt{THEN}\ c_2\ \texttt{OD}/st' \Downarrow st''\end{array}}{\texttt{DO}\ b_1\ \texttt{THEN}\ c_1\ \texttt{OR}\ b_2\ \texttt{THEN}\ c_2\ \texttt{OD}/st \Downarrow st''}$$

$$\texttt{E\_GuardLoop2}\;\frac{\begin{array}{c}\texttt{beval}\ st\ b_2 = \texttt{true}\\ c_2/st \Downarrow st'\\ \texttt{DO}\ b_1\ \texttt{THEN}\ c_1\ \texttt{OR}\ b_2\ \texttt{THEN}\ c_2\ \texttt{OD}/st' \Downarrow st''\end{array}}{\texttt{DO}\ b_1\ \texttt{THEN}\ c_1\ \texttt{OR}\ b_2\ \texttt{THEN}\ c_2\ \texttt{OD}/st \Downarrow st''}$$

Hoare logic:

$$\texttt{hoare\_guarded}\;\frac{\begin{array}{c}\{\{P \wedge b_1\}\}\ c_1\ \{\{P\}\}\\ \{\{P \wedge b_2\}\}\ c_2\ \{\{P\}\}\end{array}}{\{\{P\}\}\ \texttt{DO}\ b_1\ \texttt{THEN}\ c_1\ \texttt{OR}\ b_2\ \texttt{THEN}\ c_2\ \texttt{OD}\ \{\{P \wedge \neg b_1 \wedge \neg b_2\}\}}$$

Note that guarded commands are non-deterministic in the case that both b_1 and b_2 are true.

(a) Extend the given definition to formalise the evaluation rules. *[12 marks]*

(b) Prove the Hoare rule. You will be provided with proofs of Hoare rules for the simple imperative language that you may modify. *[13 marks]*

3. ANSWER EITHER THIS QUESTION OR QUESTION 2
You will be provided with a definition of simply-typed lambda calculus in Coq. Mark any changes you make to the definition with `(* !!! *)`.

Consider constructs satisfying the following rules:

Values:

$$\texttt{v\_delay}\,\frac{}{\texttt{value}\ (\texttt{tdelay}\ t)}$$

Evaluation:

$$\texttt{ST\_ForceDelay}\,\frac{}{\texttt{tforce}\ (\texttt{tdelay}\ t) \Longrightarrow t} \qquad \texttt{ST\_Force}\,\frac{t \Longrightarrow t'}{\texttt{tforce}\ t \Longrightarrow \texttt{tforce}\ t'}$$

Typing:

$$\texttt{T\_Delay}\,\frac{\Gamma \vdash t \in T}{\Gamma \vdash \texttt{tdelay}\ t \in \texttt{TLift}\ T}$$

$$\texttt{T\_Force}\,\frac{\Gamma \vdash t \in \texttt{TLift}\ T}{\Gamma \vdash \texttt{tforce}\ t \in T}$$

(a) Extend the given definition to formalise the evaluation and typing rules. [*12 marks*]

(b) Prove progress. You will be provided with a proof of progress for the simply-typed lambda calculus that you may extend. [*13 marks*]