Homework 3

Due: Monday, Feb 26, 11:59pm
Extension: Wednesday, Feb 28, 11:59pm

Preface

Discussing high-level approaches to homework problems with your peers is encouraged. You must include at the top of all of your assignment documents a Collaboration Statement which declares any other people with whom you discussed homework problems. For example:

> *Collaboration Statement: I discussed problems 1 and 3 with Jamie Smith. I discussed problem 2 with one of the TAs. I discussed problem 4 with a personal tutor.*

If you did not discuss the assignment with anyone, you still must declare:

> *Collaboration Statement: I did not discuss homework problems with anyone.*

Copying answers or doing the work for another student is not allowed. If there are multiple documents to submit (e.g., a written portion and programming portion), you must write a collaboration statement at the top of each portion. Please do not write your Collaboration Statement in the text of an email; you must include it directly it in your assignment.

For the written portion of the assignment, you must write your name at the top of your assignment. For the programming portion, you must write your name at the top *and the bottom* of your submitted *.ml file in an OCaml comment. E.g.:

```
<beginning of file>
(* Name: Jamie Smith *)
... your solutions to the assignment ...
(* Name: Jamie Smith *)
<end of the file>
```

Submitting

Written Portion

Prepare the written portion of your assignment as either handwritten or as a pdf exported from typesetting software like LaTeX or Word. *I will not accept scanned pdfs of handwritten work.*

Submit the written portion of your assignment by either (1) emailing your exported pdf to me: David.Darais@uvm.edu with "CS 225 HW3" in the subject line, or (2) placed under my office door (Votey 319) before the deadline.

Programming Portion

Prepare the programming portion of your assignment in a text editor of your choice. You will be working from a provided file named `hw3.ml`. Do not change the file name when you submit your assignment.

Submit the programming portion of your assignment by emailing your `hw3.ml` file to me: David.Darais@uvm.edu with "CS 225 HW3" in the subject line.

Written Portion (25 Points)

Consider the language of arithmetic expressions, described in the book in chapters 3 and 8, as well as at the end of this assignment in the section titled "Reference: Arithmetic Expressions".

Also consider the following expression definitions:

$$e_1 \coloneqq \texttt{0} \qquad e_2 \coloneqq \texttt{F} \qquad e_3 \coloneqq \texttt{pred T} \qquad e_4 \coloneqq \texttt{iszero (pred 0)}$$

$$e_5 \coloneqq \texttt{if T then (succ 0) else 0} \qquad e_6 \coloneqq \texttt{if T then F else 0}$$

Problem 1 (15 Points)

For each of the expressions e_i, either:

(1) Identify a type τ such that $e_i : \tau$ and give the full typing derivation of $e_i : \tau$

OR

(2) Write "untypeable" if there is no τ such that $e_i : \tau$

Problem 2 (10 Points)

Recall the definitions of *unsafe* and *safe* expressions:

Definition 1. *An expression e is unsafe if and only if there exists some e' such that: (1) $e \longrightarrow^* e'$, (2) e' is not a value, and (3) there does not exist any e'' such that $e' \longrightarrow e''$.*

Definition 2. *An expression e is safe if and only if it is not unsafe.*

Write an expression e such that e is *safe*, but where e is not typeable, that is, there does not exist any τ such that $e : \tau$.

Programming Portion (75 Points)

Follow the directions posted on the course website for installing OCaml and required packages.

Download hw3.zip from the course webpage and extract it to a folder on your machine. Ensure that you are able to run the following commands from the directory without error:

```
> make
> make hw3
```

Running `make` should print out a log of compiling various OCaml files. Running `make hw3` should run a bunch of tests, and report that 5 passed, 0 failed, and 15 are still "todo". In order for the `merlin` plugin to work, you must first run `make`.

Your assignment is to complete the definitions of `free_vars` and `infer` to handle the following extensions to the simply typed lambda calculus: (1) the empty type, (2) the unit type, (3) the sum type, and (4) the product type. These extensions are described in the book in Chapter 11, and are also included for reference at the end of this assignment in the section titled "Reference: Extended Simply Typed Lambda Calculus".

Look in the assignment file for `raise TODO` expressions and replace them with your solution. To help you, the corresponding mathematics for each of these definitions are shown in comments.

At the end of `hw3.ml` is a test suite which will test your functions on a few test cases. Feel free to add your own tests. Just because you pass the provided tests doesn't guarantee your solution is correct.

Reference: Arithmetic Expressions

Syntax for simple arithmetic expressions, types and values:

$$
\begin{array}{rcl}
e & ::= & \texttt{T} \mid \texttt{F} \mid \texttt{if } e \texttt{ then } e \texttt{ else } e \\
& \mid & 0 \mid \texttt{succ } e \mid \texttt{pred } e \mid \texttt{iszero } e \\
\\
\tau & ::= & \texttt{bool} \mid \texttt{nat} \\
\\
nv & ::= & 0 \mid \texttt{succ } nv \\
v & ::= & \texttt{T} \mid \texttt{F} \mid nv
\end{array}
$$

Small-step semantics:

$\boxed{e \longrightarrow e}$

$$\frac{}{\texttt{if T then } e_2 \texttt{ else } e_3 \longrightarrow e_2} \qquad \frac{}{\texttt{if F then } e_2 \texttt{ else } e_3 \longrightarrow e_3}$$

$$\frac{e_1 \longrightarrow e_1'}{\texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 \longrightarrow \texttt{if } e_1' \texttt{ then } e_2 \texttt{ else } e_3} \qquad \frac{e \longrightarrow e'}{\texttt{succ } e \longrightarrow \texttt{succ } e'} \qquad \frac{}{\texttt{pred } 0 \longrightarrow 0}$$

$$\frac{}{\texttt{pred (succ } nv\texttt{)} \longrightarrow nv} \qquad \frac{e \longrightarrow e'}{\texttt{pred } e \longrightarrow \texttt{pred } e'} \qquad \frac{}{\texttt{iszero } 0 \longrightarrow \texttt{T}} \qquad \frac{}{\texttt{iszero (succ } nv\texttt{)} \longrightarrow \texttt{F}}$$

$$\frac{e \longrightarrow e'}{\texttt{iszero } e \longrightarrow \texttt{iszero } e'}$$

Type system:

$\boxed{e : \tau}$

$$\frac{}{\texttt{T} : \texttt{bool}} \qquad \frac{}{\texttt{F} : \texttt{bool}} \qquad \frac{e_1 : \texttt{bool} \qquad e_2 : \tau \qquad e_2 : \tau}{\texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 : \tau} \qquad \frac{}{0 : \texttt{nat}} \qquad \frac{e : \texttt{nat}}{\texttt{succ } e : \texttt{nat}} \qquad \frac{e : \texttt{nat}}{\texttt{pred } e : \texttt{nat}}$$

$$\frac{e : \texttt{nat}}{\texttt{iszero } e : \texttt{bool}}$$

Reference: Extended Simply Typed Lambda Calculus

Syntax for extended simply typed lambda calculus expressions, types and values:

$$
\begin{aligned}
e ::= {} & \texttt{T} \mid \texttt{F} \mid \texttt{if}(e)\{e\}\{e\} \\
\mid {} & x \mid \lambda x : \tau . e \mid e_1\ e_2 \\
\mid {} & \texttt{absurd}(e)\ \texttt{as}\ \tau \\
\mid {} & \bullet \\
\mid {} & \texttt{inl}(e)\ \texttt{as}\ \tau \mid \texttt{inr}(e)\ \texttt{as}\ \tau \mid \texttt{case}(e)\{x.e\}\{x.e\} \\
\mid {} & \langle e, e \rangle \mid \texttt{projl}(e) \mid \texttt{projr}(e)
\end{aligned}
$$

$$
\tau ::= \texttt{bool} \mid \tau \Rightarrow \tau \mid \texttt{empty} \mid \texttt{unit} \mid \tau + \tau \mid \tau \times \tau
$$

Free variables metafunction:

$$
\begin{aligned}
\mathrm{FV} &\in \mathrm{exp} \to \mathcal{P}(\mathrm{var}) \\
\mathrm{FV}(\texttt{T}) &:= \{\} \\
\mathrm{FV}(\texttt{F}) &:= \{\} \\
\mathrm{FV}(\texttt{if}(e_1)\{e_2\}\{e_3\}) &:= \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \cup \mathrm{FV}(e_3) \\
\mathrm{FV}(x) &:= \{x\} \\
\mathrm{FV}(\lambda x : \tau . e) &:= \mathrm{FV}(e) \setminus \{x\} \\
\mathrm{FV}(e_1\ e_2) &:= \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \\
\mathrm{FV}(\texttt{absurd}(e)\ \texttt{as}\ \tau) &:= \mathrm{FV}(e) \\
\mathrm{FV}(\bullet) &:= \{\} \\
\mathrm{FV}(\texttt{inl}(e)\ \texttt{as}\ \tau) &:= \mathrm{FV}(e) \\
\mathrm{FV}(\texttt{inr}(e)\ \texttt{as}\ \tau) &:= \mathrm{FV}(e) \\
\mathrm{FV}(\texttt{case}(e_1)\{x_2.e_2\}\{x_3.e_3\}) &:= \mathrm{FV}(e_1) \cup (\mathrm{FV}(e_2) \setminus \{x_2\}) \cup (\mathrm{FV}(e_3) \setminus \{x_3\}) \\
\mathrm{FV}(\langle e_1, e_2 \rangle) &:= \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \\
\mathrm{FV}(\texttt{projl}(e)) &:= \mathrm{FV}(e) \\
\mathrm{FV}(\texttt{projr}(e)) &:= \mathrm{FV}(e)
\end{aligned}
$$

Type system:

$\boxed{\Gamma \vdash e : \tau}$

$$\frac{}{\Gamma \vdash \texttt{T} : \texttt{bool}} \qquad \frac{}{\Gamma \vdash \texttt{F} : \texttt{bool}} \qquad \frac{\Gamma \vdash e_1 : \texttt{bool} \qquad \Gamma \vdash e_2 : \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \texttt{if}(e_1)\{e_2\}\{e_3\} : \tau} \qquad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}$$

$$\frac{x : \tau_1, \Gamma \vdash e : \tau_2}{\Gamma \vdash (\lambda x : \tau_1 . e) : (\tau_1 \Rightarrow \tau_2)} \qquad \frac{\Gamma \vdash e_1 : (\tau_1 \Rightarrow \tau_2) \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash (e_1\ e_2) : \tau_2} \qquad \frac{\Gamma \vdash e : \texttt{empty}}{\Gamma \vdash \texttt{absurd}(e) : \tau} \qquad \frac{}{\Gamma \vdash \bullet : \texttt{unit}}$$

$$\frac{\Gamma \vdash e : \tau_1}{\Gamma \vdash \texttt{inl}(e)\ \texttt{as}\ (\tau_1 + \tau_2) : (\tau_1 + \tau_2)} \qquad \frac{\Gamma \vdash e : \tau_2}{\Gamma \vdash \texttt{inr}(e)\ \texttt{as}\ (\tau_1 + \tau_2) : (\tau_1 + \tau_2)}$$

$$\frac{\Gamma \vdash e_1 : (\tau_2 + \tau_3) \qquad x_2 : \tau_2, \Gamma \vdash e_2 : \tau \qquad x_3 : \tau_3, \Gamma \vdash e_3 : \tau}{\Gamma \vdash \texttt{case}(e_1)\{x_2.e_2\}\{x_3.e_3\} : \tau} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash \langle e_1, e_2 \rangle : (\tau_1 \times \tau_2)}$$

$$\frac{\Gamma \vdash e : (\tau_1 \times \tau_2)}{\Gamma \vdash \texttt{projl}(e) : \tau_1} \qquad \frac{\Gamma \vdash e : (\tau_1 \times \tau_2)}{\Gamma \vdash \texttt{projr}(e) : \tau_2}$$