

# Midterm 2 Solutions

February 28, 2010

## Problem 1

Using the notation defined in the problem statement:

$$\begin{aligned} f &= \lambda x : \{\text{first} : \text{int}; \text{second} : \text{real}\} . \{\text{first} \leftarrow x.\text{second}; \text{second} \leftarrow x.\text{first}\} \\ g &= \lambda x : \{\text{first} : \text{real}; \text{second} : \text{int}\} . \{\text{first} \leftarrow x.\text{second}; \text{second} \leftarrow x.\text{first}\} \end{aligned}$$

Another acceptable answer would be:

$$\begin{aligned} f &= \lambda x : \{\text{first} : \text{int}; \text{second} : \text{real}\} . \text{the}(y : \{\text{first} : \text{real}; \text{second} : \text{int}\} \text{ s.t. } (y.\text{first} = x.\text{second}) \wedge (y.\text{second} = x.\text{first})) \\ g &= \lambda x : \{\text{first} : \text{real}; \text{second} : \text{int}\} . \text{the}(y : \{\text{first} : \text{int}; \text{second} : \text{real}\} \text{ s.t. } (y.\text{first} = x.\text{second}) \wedge (y.\text{second} = x.\text{first})) \end{aligned}$$

## Problem 2

We can mostly write this as a function in our language, once we assume some notation for the bases:

$$\begin{aligned} f &= \lambda x : (\text{basis for } V) . \text{the}[y : (\text{basis for } V^*) \\ &\quad \forall i : \text{Int}. (((i \geq 1) \wedge (i \leq d)) \Rightarrow [(y_i(x_i) = 1) \wedge \\ &\quad \quad \forall j : \text{Int}. (((j \geq 1) \wedge (j \leq d) \wedge (i \neq j)) \Rightarrow (y_i(x_j) = 0))]])] \\ g &= \lambda x : (\text{basis for } V^*) . \text{the}(y : (\text{basis for } V) \text{ s.t. } f(y) = x) \end{aligned}$$

I've used square brackets in place of some parenthesis, to make it clear what goes where. They should be considered to be equivalent to normal parentheses. All that's happening here is that, given a basis for  $V$ , we choose the unique (where we are given uniqueness by the problem statement) basis for  $V^*$  which satisfies the conditions given in the problem statement.

## Problem 3

Suppose that  $f : \tau \rightarrow \sigma$  and  $g : \sigma \rightarrow \tau$  define an equivalence between  $\tau$  and  $\sigma$ . Then an equivalence between  $\sigma \rightarrow \gamma$  and  $\tau \rightarrow \gamma$  is given by:

$$\begin{aligned} f &= \lambda x : \sigma \rightarrow \gamma . \lambda y : \tau . x(f(y)) \\ g &= \lambda x : \tau \rightarrow \gamma . \lambda y : \sigma . x(g(y)) \end{aligned}$$

Similarly, between  $\gamma \rightarrow \sigma$  and  $\gamma \rightarrow \tau$ :

$$\begin{aligned} f &= \lambda x : \gamma \rightarrow \sigma . \lambda y : \gamma . g(x(y)) \\ g &= \lambda x : \gamma \rightarrow \tau . \lambda y : \gamma . f(x(y)) \end{aligned}$$

## Problem 4

| Context                  | Type 1   | Type 2  | Equivalent? |
|--------------------------|--|---|-------------|
| $\alpha : \text{type}_1$ | $\alpha \rightarrow (\alpha \rightarrow \text{Boole})$                       | $\alpha \times \alpha \rightarrow \text{Boole}$     | Yes         |
| $\alpha : \text{type}_1$ | $(\alpha \rightarrow \alpha) \rightarrow \text{Boole}$                       | $\alpha \times \alpha \rightarrow \text{Boole}$     | No          |
| $\alpha : \text{type}_1$ | $\text{Boole} \rightarrow \alpha$  | $\{\text{first} : \alpha; \text{second} : \alpha\}$ | Yes         |
| $\alpha : \text{type}_1$ | $\{\text{first} : \alpha; \text{second} : \alpha\} \rightarrow \text{Boole}$ | $\alpha \times \alpha \rightarrow \text{Boole}$     | Yes         |

For the first part, a bijection is given by:

$$f = \lambda x : \alpha \rightarrow (\alpha \rightarrow \text{Boole}). \lambda (y, z) : \alpha \times \alpha. f(y)(z)$$

To see that a well-typed bijection does not exist for the second part, we may apply a counting argument. In general, there are  $|\beta|^{|\alpha|}$  functions of type  $\alpha \rightarrow \beta$  (because for each of the  $|\alpha|$  possible inputs, there are  $|\beta|$  possible outputs). Also, there are  $|\alpha|^2$  values of type  $\alpha \times \alpha$ . Hence, there are  $2^{|\alpha|^{|\alpha|}}$  values of type  $(\alpha \rightarrow \alpha) \rightarrow \text{Boole}$ , and  $2^{|\alpha|^2}$  values of type  $\alpha \times \alpha \rightarrow \alpha$ . For  $|\alpha| \geq 3$ , these sets have different sizes, and since there can be no bijection between differently-sized sets, there certainly can't be a well-typed bijection.

Note that the converse of this argument is not valid—simply because two sets have the same size, it is not necessarily the case that they are equivalent (because it may not be possible to write down a well-typed expression for any bijection). On homework 3, we saw that this is the case between the types  $V$  and  $V^*$ , when the dimension of  $V$  is at least 2.

A bijection for the third part is:

$$f = \lambda x : \text{Boole} \rightarrow \alpha. \{\text{first} \leftarrow x(\text{True}); \text{second} \leftarrow x(\text{False})\}$$

And for the fourth part:

$$f = \lambda x : \{\text{first} : \alpha; \text{second} : \alpha\} \rightarrow \text{Boole}. \lambda (y, z) : \alpha \times \alpha. x(\{\text{first} \leftarrow y; \text{second} \leftarrow z\})$$

## Problem 5

| Context   | Type   | Expression Exists? |
|---|--|--------------------|
| $\alpha : \text{type}_1$  | $\alpha$   | No                 |
| $\alpha : \text{type}_1; P : \alpha \rightarrow \text{Boole}$         | $\alpha$   | No                 |
| $\alpha : \text{type}_1; f : \alpha \times \alpha \rightarrow \alpha$ | $\alpha \rightarrow (\alpha \rightarrow \alpha)$ | Yes                |
| $\alpha : \text{type}_1; f : \alpha \rightarrow \text{Int}$           | $\alpha$   | No                 |
| $\alpha : \text{type}_1; f : \alpha \rightarrow \text{Int}$           | $\alpha \rightarrow \text{Boole}$                | Yes                |

The question, which may have been a bit confusing, was whether you could write down a well-typed *value* of the given type, in the given context. So in the first part, the question is whether, given that you only know that  $\alpha$  is a type, you could write down a value of type  $\alpha$ . The answer is no, since we are given nothing whatsoever to distinguish between different values of type  $\alpha$ .

The answers to the second and fourth parts are also no, since while one might be tempted to write something like the  $(x : \alpha \text{ s.t. } P(x) = \text{True})$ , we have no reason to think that there is any value—let alone a unique value—which makes  $P$  true.

In the third and fifth parts, interestingly, not only can we write expressions of the given type in the provided contexts, we can in fact do so in the smaller context “ $\alpha : \text{type}_1$ ” (although all contexts implicitly contain  $\text{Int}$  and  $\text{Boole}$ , and the various values of/operations on these types,  $\text{Boole}$  and  $\text{True}$  are explicitly included in the context below, for clarity):

$$\begin{aligned} \alpha : \text{type}_1 &\vdash (\lambda x : \alpha. \lambda y : \alpha. x) : \alpha \rightarrow (\alpha \rightarrow \alpha) \\ \alpha : \text{type}_1, \text{Boole} : \text{type}_0, \text{True} : \text{Boole} &\vdash (\lambda x : \alpha. \text{True}) : \alpha \rightarrow \text{Boole} \end{aligned}$$

although the following are also correct:

$$\begin{aligned} \alpha : \text{type}_1, f : \alpha \times \alpha \rightarrow \alpha &\vdash (\lambda x : \alpha. \lambda y : \alpha. f(x, y)) : \alpha \rightarrow (\alpha \rightarrow \alpha) \\ \alpha : \text{type}_1, f : \alpha \rightarrow \text{Int} &\vdash (\lambda x : \alpha. (f(x) = 0)) : \alpha \rightarrow \text{Boole} \end{aligned}$$