Assignment 6
Parametricity and Data Abstraction

15-814: Types and Programming Languages
Frank Pfenning

Due Tuesday, November 17, 2020

You should hand in two files

- `hw06.pdf` with your written solutions to the questions. You may exclude those questions whose answer is just code, but please leave a pointer to the code.
- `hw06.cbv` with the code, where the solutions to the questions are clearly marked and auxiliary code (either from lecture or your own) is included so it passes the LAMBDA implementation.

1 Programs as Proofs

Task 1 (L15.1, 15 points) One proposition is *more general* than another if we can instantiate the propositional variables in the first to obtain the second. For example, $A \supset (B \supset A)$ is more general than $A \supset (\bot \supset A)$ (with $[\bot/B]$), $(C \wedge D) \supset (B \supset (C \wedge D))$ (with $[C \wedge D/A]$, but not more general than $C \supset (D \supset E)$.

For each of the following proof terms, give the most general proposition proved by it. (We are justified in saying "*the most general*" because the most general proposition is unique up to the names of the propositional variables.)

1. $\lambda u.\, \lambda w.\, \lambda k.\, w\,(u\,k)$
2. $\lambda w.\, \langle\!|(\lambda u.\, w\,(\mathbf{l} \cdot u)), (\lambda k.\, w\,(\mathbf{r} \cdot k))|\!\rangle$
3. $\lambda x.\, (\mathsf{fst}\, x)\,(\mathsf{snd}\, x)\,(\mathsf{snd}\, x)$
4. $\lambda x.\, \lambda y.\, \lambda z.\, (x\,z)\,(y\,z)$

Partially verify your answer by providing code

```
1  decl p_i = % your answer as a polymorphic type
2  defn p_i = % the term in Lambda syntax
```

in `hw06.cbv` for each each question. In the absence of lazy pairs in LAMBDA, you may use eager pairs and define suitable polymorphic functions *fst* and *snd*.

Task 2 (L14.2, 15 points) Write out a proof term for each of the following propositions. As you know from lecture, this is the same as writing a program of the translated type in our program language without the use of fixed points.

1. $(A \wedge (A \supset \bot)) \supset B$
2. $(A \vee (A \supset \bot)) \supset (((A \supset \bot) \supset \bot) \supset A)$

Verify your answer by providing code

```
1  decl q_i = % the proposition as a polymorphic type
2  defn q_i = % your answer in Lambda syntax
```

in `hw06.cbv` for each question.

2 Parametricity

Task 3 (L16.1, 15 points) Prove that $\forall\alpha.\, \alpha \to \alpha \cong 1$.

Task 4 (L16.3, 15 points) Prove, using parametricity, that if we have $f : \forall\alpha.\, \alpha \to \alpha \to \alpha$ for a value f then either $f \sim \Lambda\alpha.\, \lambda x.\, \lambda y.\, x \in [\forall\alpha.\, \alpha \to \alpha \to \alpha]$ or $f \sim \Lambda\alpha.\, \lambda x.\, \lambda y.\, y \in [\forall\alpha.\, \alpha \to \alpha \to \alpha]$.

3 Representation Independence

Task 5 (L18.1, 15 points) We can represent integers a as pairs $\langle x, y\rangle$ of natural numbers where $a = x - y$. We call this the *difference representation* and call the representation type *diff*.

$$\begin{array}{lcl} \mathit{nat} & = & \rho\alpha.\, (\mathbf{zero} : 1) + (\mathbf{succ} : \alpha) \\ \mathit{diff} & = & \mathit{nat} \times \mathit{nat} \end{array}$$

If you need auxiliary functions on natural numbers, you should define them. Your answers should be included in the file `hw06.cbv` together with several test cases.

1. Define a function *nat2diff* : $\mathit{nat} \to \mathit{diff}$ that, when given a representation of the natural number n returns an integer representing n.
2. Define a constant *d_zero* : *diff* representing the integer 0 as well as functions *d_plus* : $\mathit{diff} \to \mathit{diff} \to \mathit{diff}$ and *d_minus* : $\mathit{diff} \to \mathit{diff} \to \mathit{diff}$ representing addition and subtraction on integers, respectively.
3. Consider the type
$$\mathit{ord} = (\mathbf{lt} : 1) + (\mathbf{eq} : 1) + (\mathbf{gt} : 1)$$
that represents the outcome of a comparison (**lt** = "less than", **eq** = "equal", **gt** = "greater than"). Define a function *dcompare* : $\mathit{diff} \to \mathit{diff} \to \mathit{ord}$ to compare the two integer arguments. Again, you may use *lt*, *eq* and *gt* as constructors.

Task 6 (L18.2, 15 points) We consider an alternative *signed representation* of integers where

$$\mathit{sign} = (\mathbf{pos} : \mathit{nat}) + (\mathbf{neg} : \mathit{nat})$$

where $\mathbf{pos} \cdot x$ represents the integer x and $\mathbf{neg} \cdot x$ represents the integer $-x$. In your answers below you may use *pos* and *neg* as data constructors, to construct elements of type *sign*. Define the following functions in analogy with Task 5:

1. $\mathit{nat2sign} : \mathit{nat} \to \mathit{sign}$
2. $\mathit{s\_zero} : \mathit{sign}$
3. $\mathit{s\_plus} : \mathit{sign} \to \mathit{sign} \to \mathit{sign}$
4. $\mathit{s\_minus} : \mathit{sign} \to \mathit{sign} \to \mathit{sign}$
5. $\mathit{s\_compare} : \mathit{sign} \to \mathit{sign} \to \mathit{ord}$

Your answers should be included in the file `hw06.cbv` together with several test cases.

Task 7 (L18.3, 30 points) In this task we pursue two different implementations of an integer counter, which can become negative (unlike the natural number counter in this lecture). The functions are simpler than the ones in Tasks 5 and 6 so that the logical equality argument is more manageable. We specify a signature

$$\begin{array}{l}\mathit{INTCTR} = \{\\ \quad \mathbf{type}\ \mathit{ictr}\\ \quad \mathit{new} : \mathit{ictr}\\ \quad \mathit{inc} : \mathit{ictr} \to \mathit{ictr}\\ \quad \mathit{dec} : \mathit{ictr} \to \mathit{ictr}\\ \quad \mathit{is0} : \mathit{ictr} \to \mathit{bool}\\ \}\end{array}$$

where *new*, *inc*, *dec* and *is0* have their obvious specification with respect to integers, generalizing the *CTR* type defined in the last lecture and used in this one.

1. Write out the definition of *INTCTR* as an existential type.
2. Define the constants and functions *d_zero*, *d_inc*, *d_dec* and *d_is0* for the implementation where type *ictr* = *diff* from Task 5.
3. Define the constants and functions *szero*, *s_inc*, *s_dec* and *s_is0* for the implementation where type *ictr* = *sign* from Task 6.

Definitions from questions 1–3 should be included in the file `hw06.cbv`. Now consider the two definitions

$$\begin{array}{l}\mathit{DiffCtr} : \mathit{INTCTR} = \langle \mathit{diff}, \langle \mathit{d\_zero}, \mathit{d\_inc}, \mathit{d\_dec}, \mathit{d\_is0}\rangle\rangle\\ \mathit{SignCtr} : \mathit{INTCTR} = \langle \mathit{sign}, \langle \mathit{s\_zero}, \mathit{s\_inc}, \mathit{s\_dec}, \mathit{s\_is0}\rangle\rangle\end{array}$$

4. Prove that $\mathit{DiffCtr} \sim \mathit{SignCtr} \in [\mathit{INTCTR}]$ by defining a suitable relation $R : \mathit{diff} \leftrightarrow \mathit{sign}$ and proving that

$$\begin{array}{c}\langle \mathit{d\_zero}, \mathit{d\_inc}, \mathit{d\_dec}, \mathit{d\_is0}\rangle \sim \langle \mathit{s\_zero}, \mathit{s\_inc}, \mathit{s\_dec}, \mathit{s\_is0}\rangle\\ \in [R \times (R \to R) \times (R \to R) \times (R \to \mathit{bool})]\end{array}$$