Assignment 4
Lazy Pairs and Type Isomorphisms

15-814: Types and Programming Languages
Frank Pfenning

Due Tuesday, October 6, 2020

1 Lazy Pairs

Task 1 (L8.6, 20 points) *Lazy pairs*, constructed as $\langle\!|e_1, e_2|\!\rangle$, are an alternative to the eager pairs $\langle e_1, e_2\rangle$. Lazy pairs are typically available in "lazy" languages such as Haskell. The key differences are that a lazy pair $\langle\!|e_1, e_2|\!\rangle$ is always a value, whether its components are or not. In that way, it is like a λ-expression, since $\lambda x.\, e$ is always a value. The second difference is that its destructors are $\mathsf{fst}\ e$ and $\mathsf{snd}\ e$ rather than a new form of case expression.

We write the type of lazy pairs as $\tau_1 \,\&\, \tau_2$. In this exercise you are asked to design the rules for lazy pairs and check their correctness.

1. Write out the new rule(s) for e *value*.
2. State the typing rules for new expressions $\langle\!|e_1, e_2|\!\rangle$, $\mathsf{fst}\ e$, and $\mathsf{snd}\ e$.
3. Give evaluation rules for the new forms of expressions.

Instead of giving the complete set of new proof cases for the additional constructs, we only ask you to explicate a few items. Nevertheless, you need to make sure that the progress and preservation continue to hold.

4. State the new clause in the canonical forms theorem.
5. Show one case in the proof of the preservation theorem where a destructor is applied to a constructor.
6. Show the case in the proof of the progress theorem analyzing the typing rule for $\mathsf{fst}\ e$.

Task 2 (L8.8, 5 points) It is often stated that lazy pairs are not necessary in an eager language, because we can already define $\tau_1 \,\&\, \tau_2$ and the corresponding constructors and destructors. Fill in this table.

$\tau_1 \,\&\, \tau_2$	$\triangleq$	$(1 \to \tau_1) \times (1 \to \tau_2)$
$\langle\!|e_1, e_2|\!\rangle$	$\triangleq$	
$\mathsf{fst}\ e$	$\triangleq$	
$\mathsf{snd}\ e$	$\triangleq$	

2 Type Isomorphisms

Task 3 (L9.1, 25 points) Exhibit the functions *Forth* and *Back* witnessing the following isomorphisms. You do not need to prove that they constitute an ismorphism, just show the functions. We remain here in the pure language of Lecture 9 where every function is terminating.

(i) $\tau \times (\sigma + \rho) \cong (\tau \times \sigma) + (\tau \times \rho)$

(ii) $2 \to \tau \cong \tau \times \tau$

(iii) $1 \to \tau \cong \tau$

(iv) $0 \to \tau \cong 1$

(v) $(\sigma + \rho) \to \tau \cong (\sigma \to \tau) \times (\rho \to \tau)$

Task 4 (L9.3, 10 points) Verify that the composition $\mathit{Forth} \circ \mathit{Back} = \lambda g.\, g$ where *Forth* and *Back* coerce from a curried function to its tupled counterpart.

$$\begin{array}{lcl} \mathit{Forth} & : & ((\tau \times \sigma) \to \rho) \to (\tau \to (\sigma \to \rho)) \\ \mathit{Forth} & = & \lambda f.\, \lambda x.\, \lambda y.\, f\, \langle x, y \rangle \end{array}$$

$$\begin{array}{lcl} \mathit{Back} & : & (\tau \to (\sigma \to \rho)) \to ((\tau \times \sigma) \to \rho) \\ \mathit{Back} & = & \lambda g.\, \lambda p.\, \mathsf{case}\ p\ (\langle x, y \rangle \Rightarrow g\, x\, y) \end{array}$$

For equality of functions, use the simple call-by-value extensionality principle that $f = g : \tau_1 \to \tau_2$ if for every value $v : \tau_1$ we have $f\, v = g\, v : \tau_2$.