

Note that as a matter of convention, I write $(s \text{ } \text{⊞})$ to mean “delete this tube if the resulting instantiation mentions ⊞ ” (it’s how I implement \forall).

CoeFcom/o

$$\begin{array}{c}
 (s \text{ } r') = (s' \text{ } r') \quad (s \text{ } x) \neq (s' \text{ } x) \quad (s_i \text{ } x) \neq (s'_i \text{ } x) \\
 \hline
 (\text{coe}^{r \rightsquigarrow r'}_{[x] (\text{fcom}^{(s \text{ } x) \rightsquigarrow (s' \text{ } x)} (A \text{ } x))} \xrightarrow{[(s_i \text{ } x) = (s'_i \text{ } x) \hookrightarrow [z] (B_i \text{ } x \text{ } z)]} M) \\
 \mapsto \\
 (\text{gcom}^{r \rightsquigarrow r'}_{[x] (A \text{ } x)} M [(s_i \text{ } \text{⊞}) = (s'_i \text{ } \text{⊞}) \hookrightarrow [x] (\text{coe}^{(s' \text{ } x) \rightsquigarrow (s \text{ } x)}_{[z] (B_i \text{ } x \text{ } z)} (\text{coe}^{r \rightsquigarrow x}_{[x] (B_i \text{ } x (s \text{ } x))} M))] [(s \text{ } \text{⊞}) = (s' \text{ } \text{⊞}) \hookrightarrow [x] (\text{coe}^{r \rightsquigarrow x}_{[x] (A \text{ } x)} M)])
 \end{array}$$

It is not at all clear how to implement the evaluation semantics for such a rule, at least in a naïve way. That is, in the presentation above we maintain the fiction that we can “see” into the binder-closures well enough to take out the $(A \text{ } x)$ (or even the $(s \text{ } x)!$), etc. and use it elsewhere. We do not have this luxury, however, in the semantic domain; generally, the closures need to be thought of as black boxes.

One idea that doesn’t work is to add a new kind of stack frame to the binder closure, that says *During instantiation, if this becomes an fcom, project the cap!* But this doesn’t work, since depending on with what dimension the closure is instantiated, the fcom may disappear, and (for instance) be replaced by one of its tubes. Then, we have lost track of the cap forever.

However, I think there is a way out. During evaluation, when we encounter a non-rigid composition, we simply project out the appropriate part (either the cap or one of the tubes), and lose the rest of the information. We can keep ahold of that information instead, by adding new value constructors to our semantic domain, like:

$$(\text{hcom/cap}_A^{r \rightsquigarrow r'} M \dots) \quad (\text{hcom/tube}/i_A^{r \rightsquigarrow r'} M \dots [s_i = s'_i \hookrightarrow \dots] \dots)$$

Now, meta-operations like function application, etc. must respond to these new kinds of values; they would simply project out the cap or the appropriate tube, and continue. But what this gives us the ability to do is see the cap of an fcom even when the composition has been replaced by one of its tubes, etc.; this would allow us to implement a continuation which grabs the appropriate part of the composition for use in the resulting gcom.