Mixed Worlds

Nico

January 6, 2015

1 Goals

The goal of this document is to motivate a three-world interpretation of two-staged languages. I'll be using some labels in a way that is completely opposite from previous formulations, so reader beware.

2 Two Independent Languages

As our baseline, we start with two independent languages, called L$\mathbb{1}$ and L$\mathbb{2}$. These languages can in general have any desired set of constructs. In this document, we'll assume that both languages have products, sums, functions, and some base types.

The valid types of L$\mathbb{1}$ and L$\mathbb{2}$ are given by the judgements A type @ $\mathbb{1}$ and A type @ $\mathbb{2}$, respectively. Likewise, the typing judgements are given by $e : A$ @ $\mathbb{1}$ and $e : A$ @ $\mathbb{2}$. As it stands, these languages are completely independent. That is, the rules for the @ $\mathbb{1}$ judgements and @ $\mathbb{2}$ judgements never depend on each other.

We call the thing after the @ sign a *world*. So far, this has only been either $\mathbb{1}$ or $\mathbb{2}$. We can save space in our presentation by abstracting judgements over worlds, conventionally using the metavariable w. Thus, the valid type judgement becomes A type @ w and the typing judgement becomes $e : A$ @ w. We can also define the rules parametrically over world, where appropriate.

3 Bridging the Languages: A Roadmap

Our goal in this project is to define some sort of linguistic superstructure that bridges L$\mathbb{1}$ and L$\mathbb{2}$. In particular, we'll be looking for a way to do this that admits a *temporal interpretation*, wherein the language L$\mathbb{1}$ is thought to operate at one point in time, the language L$\mathbb{2}$ is thought to operate at a later point in time, and information cannot pass from L$\mathbb{2}$ to L$\mathbb{1}$, lest there be a violation of causality. Strictly speaking, we should probably call this *directionality*, but we go with a temporal metaphor, because time is the most accessable example of a directional process.

We'll start by adding a new world $\mathbb{M}$ (for *mixed*) which encapsulates and coordiantes the other two languages. We'll then proceed by adding features, one at a time, to the $\mathbb{M}$-level language, starting with structure constructs, then moving on to products, functions, and sums.

All along the way, we'll be keeping track of how well the language admits a temporal interpretation. In particular, we do this by defining a partial evaluation semantics and a stage-splitter [pretend I explain what these mean].

4 Structural Constructs

We start with a few structural constructs to get us off the ground, the most important being *encapsulation* rules at the type and term level. These rules allow us to encapsulate computations of world $\mathbb{1}$ or $\mathbb{2}$, and pass around their result at the mixed level.

$$\frac{A \text{ type} \mathbin{@} \mathbb{1}}{\nabla A \text{ type} \mathbin{@} \mathbb{M}} \qquad \frac{A \text{ type} \mathbin{@} \mathbb{2}}{\bigcirc A \text{ type} \mathbin{@} \mathbb{M}}$$

$$\frac{\Gamma \vdash e : A \mathbin{@} \mathbb{1}}{\Gamma \vdash \texttt{mono}\ e : \nabla A \mathbin{@} \mathbb{M}} \qquad \frac{\Gamma \vdash e : A \mathbin{@} \mathbb{2}}{\Gamma \vdash \texttt{next}\ e : \bigcirc A \mathbin{@} \mathbb{M}}$$

These are essentially the only base types at $\mathbb{M}$. We'll also add a general way to form let bindings at stage two:

$$\frac{\Gamma \vdash e_1 : A \mathbin{@} \mathbb{M} \quad \Gamma, x : A \mathbin{@} \mathbb{M} \vdash e_2 : B \mathbin{@} \mathbb{M}}{\Gamma \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : B \mathbin{@} \mathbb{M}}$$

And some simple ways to eliminate ∇ and $\bigcirc$:

$$\frac{\Gamma \vdash e_1 : \nabla A \mathbin{@} \mathbb{M} \quad \Gamma, x : A \mathbin{@} \mathbb{1} \vdash e_2 : B \mathbin{@} \mathbb{M}}{\Gamma \vdash \texttt{let mono}\{x\} = e_1\ \texttt{in}\ e_2 : B \mathbin{@} \mathbb{M}}$$

$$\frac{\Gamma \vdash e_1 : \bigcirc A \mathbin{@} \mathbb{M} \quad \Gamma, x : A \mathbin{@} \mathbb{2} \vdash e_2 : B \mathbin{@} \mathbb{M}}{\Gamma \vdash \texttt{let next}\{x\} = e_1\ \texttt{in}\ e_2 : B \mathbin{@} \mathbb{M}}$$

Finally, we need a way to move base types from $\mathbb{1}$ to $\mathbb{2}$:

$$\frac{\Gamma \vdash e : \nabla \text{int} \mathbin{@} \mathbb{M}}{\Gamma \vdash \texttt{hold}\ e : \bigcirc \text{int} \mathbin{@} \mathbb{M}}$$

So far we have nothing that breaks a temporal interpretation, and our splitting rules can be given by:

$$\frac{e \overset{\mathbb{1}}{\rightsquigarrow} [c, l.r]}{\texttt{hold}\ e \overset{\mathbb{1}}{\rightsquigarrow} [((), c), (y, l).(r; y)]} \qquad \frac{.}{\texttt{mono}\ e \overset{\mathbb{1}}{\rightsquigarrow} [(e, ()), \_.()]}$$

$$\frac{.}{\texttt{next}\ e \overset{\mathbb{1}}{\rightsquigarrow} [((), ()), \_.e]}$$

$$\frac{e_1 \overset{\mathbb{1}}{\rightsquigarrow} [c_1, l_1.r_1] \quad e_2 \overset{\mathbb{1}}{\rightsquigarrow} [c_2, l_2.r_2]}{\texttt{let}\ x = e_1\ \texttt{in}\ e_2 \overset{\mathbb{1}}{\rightsquigarrow} \left[\begin{pmatrix}\texttt{let}\ (x, z_1) = c_1\ \texttt{in} \\ \texttt{let}\ (y, z_2) = c_2\ \texttt{in} \\ (y, (z_1, z_2))\end{pmatrix}, (l_1, l_2).\texttt{let}\ x = r_1\ \texttt{in}\ r_2\right]}$$

$$\frac{e_1 \overset{\mathbb{1}}{\rightsquigarrow} [c_1, l_1.r_1] \quad e_2 \overset{\mathbb{1}}{\rightsquigarrow} [c_2, l_2.r_2]}{\texttt{let mono}\{x\} = e_1\ \texttt{in}\ e_2 \overset{\mathbb{1}}{\rightsquigarrow} \left[\begin{pmatrix}\texttt{let}\ (x, z_1) = c_1\ \texttt{in} \\ \texttt{let}\ (y, z_2) = c_2\ \texttt{in} \\ (y, (z_1, z_2))\end{pmatrix}, (l_1, l_2).(r_1; r_2)\right]}$$

$$\frac{e_1 \overset{\mathbb{1}}{\rightsquigarrow} [c_1, l_1.r_1] \quad e_2 \overset{\mathbb{1}}{\rightsquigarrow} [c_2, l_2.r_2]}{\texttt{let next}\{x\} = e_1\ \texttt{in}\ e_2 \overset{\mathbb{1}}{\rightsquigarrow} \left[\begin{pmatrix}\texttt{let}\ (y, z) = c_2\ \texttt{in} \\ (y, (\texttt{pi2}\ c_1, z_2))\end{pmatrix}, (l_1, l_2).\texttt{let}\ x = r_1\ \texttt{in}\ r_2\right]}$$

5 Adding Product and Functions

Adding mixed products and functions is straight-forward.

$$\frac{A \text{ type @ } \mathbb{M} \quad B \text{ type @ } \mathbb{M}}{A \times B \text{ type @ } \mathbb{M}} \qquad \frac{A \text{ type @ } \mathbb{M} \quad B \text{ type @ } \mathbb{M}}{A \to B \text{ type @ } \mathbb{M}}$$

$$\frac{\Gamma \vdash e_1 : A \text{ @ } \mathbb{M} \quad \Gamma \vdash e_2 : B \text{ @ } \mathbb{M}}{\Gamma \vdash (e_1, e_2) : A \times B \text{ @ } \mathbb{M}} \times I \qquad \frac{\Gamma \vdash e : A \times B \text{ @ } \mathbb{M}}{\Gamma \vdash \texttt{pi1}\ e : A \text{ @ } \mathbb{M}} \times E_1$$

$$\frac{A \text{ type @ } \mathbb{M} \quad \Gamma, x : A \text{ @ } \mathbb{M} \vdash e : B \text{ @ } \mathbb{M}}{\Gamma \vdash \lambda x{:}A.e : A \to B \text{ @ } \mathbb{M}} \to I$$

$$\frac{\Gamma \vdash e_1 : A \to B \text{ @ } \mathbb{M} \quad \Gamma \vdash e_2 : A \text{ @ } \mathbb{M}}{\Gamma \vdash e_1\ e_2 : B \text{ @ } \mathbb{M}} \to E$$

And the splitting rules:

$$\frac{e \overset{\mathbb{1}}{\rightsquigarrow} [c, l.r]}{\lambda x{:}A.e \overset{\mathbb{1}}{\rightsquigarrow} [(\lambda x.c, ()), \_.(\lambda(x, l).r)]}$$

$$\frac{e_1 \overset{\mathbb{1}}{\rightsquigarrow} [c_1, l_1.r_1] \quad e_2 \overset{\mathbb{1}}{\rightsquigarrow} [c_2, l_2.r_2]}{e_1\ e_2 \overset{\mathbb{1}}{\rightsquigarrow} \left[\left(\begin{array}{l}\texttt{let}\ (y_1, z_1) = c_1\ \texttt{in}\\ \texttt{let}\ (y_2, z_2) = c_2\ \texttt{in}\\ \texttt{let}\ (y_3, z_3) = y_1\ y_2\ \texttt{in}\\ (y_3, (z_1, z_2, z_3))\end{array}\right), \begin{array}{l}(l_1, l_2, l_3).\\ \qquad (r_1\ (r_2, l_3))\end{array}\right]}$$

$$\frac{e_1 \overset{\mathbb{1}}{\rightsquigarrow} [c_1, l_1.r_1] \quad e_2 \overset{\mathbb{1}}{\rightsquigarrow} [c_2, l_2.r_2]}{(e_1, e_2) \overset{\mathbb{1}}{\rightsquigarrow} \left[\left(\begin{array}{l}\texttt{let}\ (y_1, z_1) = c_1\ \texttt{in}\\ \texttt{let}\ (y_2, z_2) = c_2\ \texttt{in}\\ ((y_1, y_2), (z_1, z_2))\end{array}\right), (l_1, l_2).(r_1, r_2)\right]}$$

$$\frac{e \overset{\mathbb{1}}{\rightsquigarrow} [c, l.r]}{\texttt{pi1}\ e \overset{\mathbb{1}}{\rightsquigarrow} \left[\begin{array}{l}\texttt{let}\ (y, z) = c\ \texttt{in}\\ (\texttt{pi1}\ y, z)\end{array}, l.\texttt{pi1}\ r\right]}$$

6 Adding Sums

So adding mixed sums is actually a bit more difficult. It forces us to answer the question: when is the tag known?

7 Adding `prev`

We can also add `prev`, which is more expressive than the current $\bigcirc$ elim form, but this requires changing all our rules.