Day 12.

1. State

Next, we'll consider a simple kind of state—a single accumulator value. As before, we extend our language with new terms to retrieve and update the value of the accumulator:

$$e ::= z \mid e \odot e \mid x \mid \lambda x.e \mid e\ e \mid \mathsf{get} \mid \mathsf{put}\ e$$

We need to extend our evaluation relation to incorporate the accumulator. As with a reader, terms will need access to the initial value of the accumulator; unlike the reader, they'll also need to produce a new value of the accumulator at the end of the computation. We'll represent this with the judgment $t \mid s_0 \Downarrow v \mid s_1$, indicating that term t in state s_0 evaluates to value v with final state s_1. (Of course, this is just notation; the crux is that $\Downarrow \in \mathcal{E} \times \mathcal{V} \rightharpoonup \mathcal{V} \times V$.)

We start with the evaluation rules for the state operations:

$$\frac{}{\mathsf{get} \mid s \Downarrow s \mid s} \qquad \frac{e \mid s_0 \Downarrow v \mid s_1}{\mathsf{put}\ e \mid s_0 \Downarrow 0 \mid v}$$

The rule for get says that it returns the current state, without changing it. The rule for put t replaces the state (after computing t) with a new state; the return value is here essentially arbitrary. Of course, just because put discards the initial state doesn't mean that it can't be used in computing the new state. The evaluation rules for values can be extended to incorporate state in the "obvious" fashion:

$$\frac{}{z \mid s \Downarrow z \mid s} \qquad \frac{}{\lambda x.e \mid s \Downarrow \lambda x.e \mid s}$$

However, when we come to the other operations things become a little more interesting.

2. Order of Evaluation

Consider the rule for binary operators $t_1 \odot t_2$. The evaluation of each of t_1 and t_2 will itself have an initial state and a final state, and so we have two ways that we can thread the state through the operations:

$$\frac{e_1 \mid s_1 \Downarrow_{\mathsf{ltr}} z_1 \mid s_2 \quad e_2 \mid s_2 \Downarrow_{\mathsf{ltr}} z_2 \mid s_3}{e_1 \odot e_2 \mid s_1 \Downarrow_{\mathsf{ltr}} z_1 \odot z_2 \mid s_3} \qquad \frac{e_1 \mid s_2 \Downarrow_{\mathsf{rtl}} z_1 \mid s_3 \quad e_2 \mid s_1 \Downarrow_{\mathsf{rtl}} z_2 \mid s_2}{e_1 \odot e_2 \mid s_1 \Downarrow_{\mathsf{rtl}} z_1 \odot z_2 \mid s_3}$$

With left-to-right evaluation (on the left), the initial state is used in evaluating the left-hand argument, with the resulting state used in evaluating the right-hand argument. With right-to-left evaluation, the state is threaded the opposite way. Note that otherwise the operations are the

same—in each case, we combine the values of t_1 and t_2. We can observe the difference if we consider a simple term combining get and put:

$$\dfrac{\dfrac{\overline{2 \mid 0 \Downarrow_{\mathsf{ltr}} 2 \mid 0}}{\mathsf{put}\,2 \mid 0 \Downarrow_{\mathsf{ltr}} 0 \mid 2} \quad \dfrac{}{\mathsf{get} \mid 2 \Downarrow_{\mathsf{ltr}} 2 \mid 2}}{\mathsf{put}\,2 + \mathsf{get} \mid 0 \Downarrow_{\mathsf{ltr}} 2 \mid 2} \qquad \dfrac{\dfrac{\overline{2 \mid 0 \Downarrow_{\mathsf{rtl}} 2 \mid 0}}{\mathsf{put}\,2 \mid 0 \Downarrow_{\mathsf{rtl}} 0 \mid 2} \quad \dfrac{}{\mathsf{get} \mid 0 \Downarrow_{\mathsf{rtl}} 0 \mid 0}}{\mathsf{put}\,2 + \mathsf{get} \mid 0 \Downarrow_{\mathsf{rtl}} 0 \mid 2}$$

With left-to-right evaluation, the effect of put 2 is observed by get, while with right-to-left evaluation it is not.

A similar possibility occurs in function evaluation:

$$\dfrac{e_1 \mid s_1 \Downarrow_{\mathsf{ltr}} \lambda x.e \mid s_2 \quad e_2 \mid s_2 \Downarrow_{\mathsf{ltr}} w \mid s_3 \quad e[w/x] \mid s_3 \Downarrow_{\mathsf{ltr}} v \mid s_4}{e_1\ e_2 \mid s_1 \Downarrow_{\mathsf{ltr}} v \mid s_4}$$

$$\dfrac{e_1 \mid s_2 \Downarrow_{\mathsf{rtl}} \lambda x.e \mid s_3 \quad e_2 \mid s_1 \Downarrow_{\mathsf{rtl}} w \mid s_2 \quad e[w/x] \mid s_3 \Downarrow_{\mathsf{rtl}} v \mid s_4}{e_1\ e_2 \mid s_1 \Downarrow_{\mathsf{rtl}} v \mid s_4}$$

We can reuse the previous example to see the difference here:

$$\dfrac{\dfrac{\Delta}{(\lambda a.\lambda b.a + b)\,\mathsf{get} \mid 0 \Downarrow_{\mathsf{ltr}} \lambda b.0 + b \mid 0} \quad \dfrac{\overline{3 \mid 0 \Downarrow_{\mathsf{ltr}} 3 \mid 0}}{\mathsf{put}\,3 \mid 0 \Downarrow_{\mathsf{ltr}} 0 \mid 3} \quad \dfrac{\overline{0 \mid 3 \Downarrow_{\mathsf{ltr}} 0 \mid 3} \quad \overline{0 \mid 3 \Downarrow_{\mathsf{ltr}} 0 \mid 3}}{(0 + b)[0/b] \mid 3 \Downarrow_{\mathsf{ltr}} 0 \mid 3}}{(\lambda a.\lambda b.a + b)\,\mathsf{get}\,(\mathsf{put}\,3) \mid 0 \Downarrow_{\mathsf{ltr}} 0 \mid 3}$$

where $\Delta = \dfrac{\overline{\lambda a.\lambda b.a + b \mid 0 \Downarrow_{\mathsf{ltr}} \lambda a.\lambda b.a + b \mid 0} \quad \overline{\mathsf{get} \mid 0 \Downarrow_{\mathsf{ltr}} 0 \mid 0} \quad \overline{(\lambda b.a + b)[0/a] \mid 0 \Downarrow_{\mathsf{ltr}} \lambda b.0 + b \mid 0}}{(\lambda a.\lambda b.a + b)\,\mathsf{get} \mid 0 \Downarrow_{\mathsf{ltr}} \lambda b.0 + b \mid 0}$

vs

$$\dfrac{\dfrac{\Delta}{(\lambda a.\lambda b.a + b)\,\mathsf{get} \mid 3 \Downarrow_{\mathsf{ltr}} \lambda b.3 + b \mid 3} \quad \dfrac{\overline{3 \mid 0 \Downarrow_{\mathsf{ltr}} 3 \mid 0}}{\mathsf{put}\,3 \mid 0 \Downarrow_{\mathsf{ltr}} 0 \mid 3} \quad \dfrac{\overline{3 \mid 3 \Downarrow_{\mathsf{ltr}} 3 \mid 3} \quad \overline{0 \mid 3 \Downarrow_{\mathsf{ltr}} 0 \mid 3}}{(3 + b)[0/b] \mid 3 \Downarrow_{\mathsf{ltr}} 3 \mid 3}}{(\lambda a.\lambda b.a + b)\,\mathsf{get}\,(\mathsf{put}\,3) \mid 0 \Downarrow_{\mathsf{ltr}} 3 \mid 3}$$

where $\Delta = \dfrac{\overline{\lambda a.\lambda b.a + b \mid 3 \Downarrow_{\mathsf{ltr}} \lambda a.\lambda b.a + b \mid 3} \quad \overline{\mathsf{get} \mid 3 \Downarrow_{\mathsf{ltr}} 3 \mid 3} \quad \overline{(\lambda b.a + b)[0/a] \mid 3 \Downarrow_{\mathsf{ltr}} \lambda b.0 + b \mid 3}}{(\lambda a.\lambda b.a + b)\,\mathsf{get} \mid 3 \Downarrow_{\mathsf{ltr}} \lambda b.3 + b \mid 3}$