Day 8.

1. Functions and Environments

To develop our approximation of local variables, we needed to move from a substution-based view of evaluation to an environment-based view. We'll have to do something similar for functions. So, let's get started!

$$\frac{}{\mathrm{H} \vdash_{\mathsf{cbv}} x \Downarrow \mathrm{H}(x)} \qquad \frac{}{\mathrm{H} \vdash_{\mathsf{cbv}} \lambda x.t \Downarrow \lambda x.t} \qquad \frac{\mathrm{H} \vdash_{\mathsf{cbv}} t_1 \Downarrow \lambda x.t \quad \mathrm{H} \vdash_{\mathsf{cbv}} t_2 \Downarrow w \quad \mathrm{H}[x \mapsto w] \vdash_{\mathsf{cbv}} t \Downarrow v}{\mathrm{H} \vdash_{\mathsf{cbv}} t_1\, t_2 \Downarrow v}$$

- Omitted rules for numeric constants, because they don't behave any different than they did in the last version
- Again, reusing syntax for 2-place and 3-place evaluation relations

We should confirm that it works. Let's try some simple reductions:

$$\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \lambda a.\lambda b.b \Downarrow \lambda a.\lambda b.b} \quad \dfrac{}{\emptyset \vdash 3 \Downarrow 3} \quad \dfrac{}{\{a \mapsto 3\} \vdash \lambda b.b \Downarrow \lambda b.b}}{\emptyset \vdash (\lambda a.\lambda b.b)3 \Downarrow \lambda b.b} \quad \dfrac{}{\emptyset \vdash 2 \Downarrow 2} \quad \dfrac{}{\{b \mapsto 2\} \vdash b \Downarrow 2}}{\emptyset \vdash_{\mathsf{cbv}} (\lambda a.\lambda b.b)\, 3\, 2 \Downarrow 2}$$

Looks good so far!

$$\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \lambda a.\lambda b.a \Downarrow \lambda a.\lambda b.a} \quad \dfrac{}{\emptyset \vdash 3 \Downarrow 3} \quad \dfrac{}{\{a \mapsto 3\} \vdash \lambda b.a \Downarrow \lambda b.a}}{\emptyset \vdash (\lambda a.\lambda b.a)\, 3 \Downarrow \lambda b.a} \quad \dfrac{}{\emptyset \vdash 2 \Downarrow 2} \quad \begin{array}{c} \lightning \\ \{b \mapsto 2\} \vdash a \Downarrow 3 \end{array}}{\emptyset \vdash_{\mathsf{cbv}} (\lambda a.\lambda b.a)\, 3\, 2 \Downarrow 3}$$

What's gone wrong?

- We're trying to use variable a when it's not apparently in scope. Fair enough—this *shouldn't* be derivable.
- Variable a should have gotten its meaning in reducing the left-hand argument, but it didn't. This is the real problem.
- Missing one aspect of substitution—although evaluation doesn't touch λs, substitution does!

Solution: λ terms need to carry their defining environments with them!

- Means we don't have to reintroduce substitution
- Combination of a function and its environment called a *closure*.

2. Closures

Let's recap our language:

$$
\begin{aligned}
\mathcal{X} &\ni x \\
\mathcal{V} &\ni v ::= z \mid \lambda^{\mathrm{H}} x.t \\
\mathcal{E} &\ni t ::= z \mid t_1 \odot t_2 \mid x \mid \lambda x.t \mid t_1\, t_2
\end{aligned}
$$

- New value form: closures. Package environment with function
- Values no longer subset of terms... but can think of $\lambda^{\mathrm{H}} x.t$ as being syntax for $(\lambda x.t)[v_i/y_i]$ where $\mathrm{H} = \{y_i \mapsto v_i\}$.

Now we can adjust evaluation rules to construct and use closures.

$$
\frac{}{\mathrm{H} \vdash \lambda x.t \Downarrow \lambda^{\mathrm{H}} x.t}
\qquad
\frac{\mathrm{H} \vdash t_1 \Downarrow \lambda^{\mathrm{H}'} x.t \quad \mathrm{H} \vdash t_2 \Downarrow w \quad \mathrm{H}'[x \mapsto w] \vdash t \Downarrow v}{\mathrm{H} \vdash_{\mathsf{cbv}} t_1\, t_2 \Downarrow v}
$$

Does this work?

$$
\frac{\dfrac{\dfrac{}{\emptyset \vdash \lambda a.\lambda b.b \Downarrow \lambda a.\lambda b.b} \quad \dfrac{}{\emptyset \vdash 3 \Downarrow 3} \quad \dfrac{}{\{a \mapsto 3\} \vdash \lambda b.b \Downarrow \lambda^{\{a \mapsto 3\}} b.b}}{\emptyset \vdash (\lambda a.\lambda b.b)\, 3 \Downarrow \lambda^{\{a \mapsto 3\}} b.b} \quad \dfrac{}{\emptyset \vdash 2 \Downarrow 2} \quad \dfrac{}{\{a \mapsto 3, b \mapsto 2\} \vdash b \Downarrow 2}}{\emptyset \vdash_{\mathsf{cbv}} (\lambda a.\lambda b.b)\, 3\, 2 \Downarrow 2}
$$

Looks promising.

$$
\frac{\dfrac{\dfrac{}{\emptyset \vdash (\lambda a.\lambda b.a) \Downarrow \lambda^{\emptyset} a.\lambda b.a} \quad \dfrac{}{\emptyset \vdash 3 \Downarrow 3} \quad \dfrac{}{\{a \mapsto 3\} \vdash \lambda b.a \Downarrow \lambda^{\{a \mapsto 3\}} b.a}}{\emptyset \vdash (\lambda a.\lambda b.a)\, 3 \Downarrow \lambda^{\{a \mapsto 3\}} b.a} \quad \dfrac{}{\emptyset \vdash 2 \Downarrow 2} \quad \dfrac{}{\{a \mapsto 3, b \mapsto 2\} \vdash a \Downarrow 3}}{\emptyset \vdash_{\mathsf{cbv}} (\lambda a.\lambda b.a)\, 3\, 2 \Downarrow 3}
$$

Seems to work!

Call by name variation: just replace $H \in \mathcal{X} \rightharpoonup \mathcal{V}$ with $H \in \mathcal{X} \rightharpoonup \mathcal{E}$ and:

$$
\frac{\mathrm{H} \vdash_{\mathsf{cbn}} \mathrm{H}(x) \Downarrow v}{\mathrm{H} \vdash_{\mathsf{cbn}} x \Downarrow v}
\qquad
\frac{\mathrm{H} \vdash_{\mathsf{cbn}} t \Downarrow_1 \lambda^{\mathrm{H}'} x.t \quad \mathrm{H}'[x \mapsto t_2] \vdash_{\mathsf{cbn}} t \Downarrow v}{\mathrm{H} \vdash_{\mathsf{cbn}} t_1\, t_2 \Downarrow v}
$$

Historical note. Early implementations of LISP, including some still in use (ELISP), got closures wrong. Some people like to present this as a design choice; they call it "dynamic scope" or similar euphemisms. This is not a design choice, any more than $2 + 2 = 5$ would be a design choice for addition. It is a system that fails to match the semantics of the λ-calculus.

3. Typing Functions

What can go wrong? $1\,2$, $(\lambda c.c) + 1$.

We need to extend our grammar of types:

$$
\mathcal{T} \ni T ::= \texttt{Int} \mid T_1 \to T_2
$$

- Why don't closures need to be reflected in the types of functions?

8.

As before, we define a variation of the evaluation relation that characterizes the types of values: $\Gamma \vdash t : T$.

- Syntax: $\vdash$ denotes *consequence*—under the assumptions in Γ, the typing on the right holds. $:$ was originally $\in$.
- $\Gamma : \mathcal{X} \rightharpoonup \mathcal{T}$ map from variables to their types.
- More about the typing relation... and the significance of our notational choices... to come.

Typing rules:

$$\dfrac{}{\Gamma \vdash z : \texttt{Int}} \qquad \dfrac{\Gamma \vdash t_1 : \texttt{Int} \quad \Gamma \vdash t_2 : \texttt{Int}}{\Gamma \vdash t_1 + t_2 : \texttt{Int}} \qquad \cdots$$

$$\dfrac{}{\Gamma \vdash x : \Gamma(x)} \qquad \dfrac{\Gamma[x \mapsto T_1] \vdash t : T_2}{\Gamma \vdash \lambda x.t : T_1 \to T_2} \qquad \dfrac{\Gamma \vdash t_1 : T_1 \to T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1\, t_2 : T_2}$$

- Common notation for $\Gamma[x \mapsto T_1]$ is $\Gamma, x{:}T_1$. May fall into this later, but not yet.
- Why don't we have to represent the closure in the application rule?

Let's look at some simple derivations:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{a \mapsto \texttt{Int}, b \mapsto \texttt{Int} \to \texttt{Int}\} \vdash a : \texttt{Int}}}{\{a \mapsto \texttt{Int}\} \vdash \lambda b.a : (\texttt{Int} \to \texttt{Int}) \to \texttt{Int}}}{\emptyset \vdash (\lambda a.\lambda b.a) : \texttt{Int} \to (\texttt{Int} \to \texttt{Int}) \to \texttt{Int}} \quad \dfrac{}{\emptyset \vdash 3 : \texttt{Int}}}{\emptyset \vdash (\lambda a.\lambda b.a)\, 3 : (\texttt{Int} \to \texttt{Int}) \to \texttt{Int}} \quad \dfrac{\dfrac{}{\{c \mapsto \texttt{Int}\} \vdash c : \texttt{Int}}}{\emptyset \vdash \lambda c.c : \texttt{Int} \to \texttt{Int}}}{\emptyset \vdash (\lambda a.\lambda b.a)\, 3\, (\lambda c.c) : \texttt{Int}}$$

$$\dfrac{\dfrac{\dfrac{}{\{a \mapsto \texttt{Int} \to \texttt{Int}\} \vdash a : \texttt{Int} \to \texttt{Int}}}{\emptyset \vdash (\lambda a.a) : (\texttt{Int} \to \texttt{Int}) \to (\texttt{Int} \to \texttt{Int})} \quad \dfrac{\dfrac{}{\{b \mapsto \texttt{Int}\} \vdash b : \texttt{Int}}}{\emptyset \vdash (\lambda b.b) : \texttt{Int} \to \texttt{Int}}}{\emptyset \vdash (\lambda a.a)\, (\lambda b.b) : \texttt{Int} \to \texttt{Int}}$$

- Check typing of functions at *construction*, not at *use*. So: more structure under the typing of a λ, but less at their uses.
- Same term may have more than one typing derivation: $\lambda a.a$ (up to α-equivalence) given both $\texttt{Int} \to \texttt{Int}$ and $(\texttt{Int} \to \texttt{Int}) \to (\texttt{Int} \to \texttt{Int})$.