# Day 10.

## 1. Typing Functions

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

We need to extend our grammar of types:

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

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

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 $\Gamma$, the typing on the right holds. $:$ was originally $\in$.
- $\Gamma : \mathcal{X} \rightharpoonup \mathcal{Y}$ map from variables to their types.
- More about the typing relation... and the significance of our notational choices... to come.

Typing rules:

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

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

- 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:

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

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

- Check typing of functions at *construction*, not at *use*. So: more structure under the typing of a $\lambda$, but less at their uses.

- Same term may have more than one typing derivation: $\lambda a.a$ (up to $\alpha$-equivalence) given both `Int` $\to$ `Int` and (`Int` $\to$ `Int`) $\to$ (`Int` $\to$ `Int`).

## 2. Basic Proof Theory

Historical notes:

- Hilbert's *axiomatic proof theory*: 1. Choose axioms and basic objects 2. Prove consistency 3. Explore independence and completeness 4. Decision procedure
- Aims: geometry, arithmetic, analysis
- Gentzen's development of formal proof theory.
  - Based on work by Frege
  - Starting the above program with logic

Gentzen's observation: rather than starting from *axioms*, most proofs start from a set of *assumptions*. There are then two categories of operations:

- Assumptions are analyzed into parts—*eliminating* them
- Conclusions are analyzed into parts—*introducing* them
- Ideally, you meet in the middle

This means that, to formalize proofs, we want to provide each *logical connective* with a set of introduction rules and a set of elimination rules.

Conjunction:

$$(\wedge\,\mathrm{I})\ \frac{A \quad B}{A \wedge B} \qquad (\wedge\,\mathrm{E}_1)\ \frac{A \wedge B}{A} \qquad (\wedge\,\mathrm{E}_2)\ \frac{A \wedge B}{B}$$

Disjunction:

$$(\vee\,\mathrm{I}_1)\ \frac{A}{A \vee B} \qquad (\vee\,\mathrm{I}_2)\ \frac{B}{A \vee B} \qquad (\vee\,\mathrm{E})\ \frac{A \vee B \quad \begin{matrix}[A]\\ \vdots \\ C\end{matrix} \quad \begin{matrix}[B]\\ \vdots \\ C\end{matrix}}{C}$$

- Bracketed propositions *may* be used in the derivation, as often as needed, but are not required to be.

Implication:

$$(\Rightarrow\mathrm{I})\ \frac{\begin{matrix}[A]\\ \vdots \\ B\end{matrix}}{A \Rightarrow B} \qquad (\Rightarrow\mathrm{E})\ \frac{A \Rightarrow B \quad A}{B}$$

*10.*

Now, we can put together some simple derivations:

$$
(\Rightarrow \mathrm{I})^p\ \dfrac{(\vee \mathrm{E})^{q,r}\ \dfrac{(\wedge \mathrm{E}_2)\ \dfrac{[A \wedge (B \vee C)]^p}{B \vee C} \qquad (\vee \mathrm{I}_1)\ \dfrac{(\wedge \mathrm{I})\ \dfrac{(\wedge \mathrm{E}_1)\ \dfrac{[A \wedge (B \vee C)]^p}{A} \qquad [B]^q}{A \wedge B}}{(A \wedge B) \vee (A \wedge C)} \qquad (\vee \mathrm{I}_2)\ \dfrac{(\wedge \mathrm{I})\ \dfrac{(\wedge \mathrm{E}_1)\ \dfrac{[A \wedge (B \vee C)]^p}{A} \qquad [C]^r}{A \wedge C}}{(A \wedge B) \vee (A \wedge C)}}{(A \wedge B) \vee (A \wedge C)}}{A \wedge (B \vee C) \Rightarrow (A \wedge B) \vee (A \wedge C)}
$$

- We label rules that introduce assumptions and the corresponding uses of those assumptions... for example, the hypothesis introduced at the base of the derivation is used at the points labeled $p$.

We can extend this approach to the logical constants as well:

$$
(\top\,\mathrm{I})\ \dfrac{}{\top} \qquad \text{(No elimination rule for truth)} \qquad (\bot\,\mathrm{E})\ \dfrac{\bot}{A} \qquad \text{(No introduction rule for falsity)}
$$

- We define negation in terms of implication and falsity: $\neg A = A \Rightarrow \bot$. This gives, as we expect, $A \wedge \neg A \implies \bot$.
- Don't actually need $(\bot\,\mathrm{E})$ (also called ECQ). Result is called *minimal* logic.

Key idea: *normalization*

- Eliminate detours (i.e. lemmas) in proofs
- Consistency as a consequence (i.e., because there are no proofs of $\bot$, and normalized proof can only prove $\bot$ if it's assumed it.

Conjunction:

$$
(\wedge \mathrm{E}_1)\ \dfrac{(\wedge \mathrm{I})\ \dfrac{\begin{matrix}\vdots \\ A\end{matrix} \quad \begin{matrix}\vdots \\ B\end{matrix}}{A \wedge B}}{A} \ \leadsto\ \begin{matrix}\vdots \\ A\end{matrix} \qquad\qquad (\wedge \mathrm{E}_2)\ \dfrac{(\wedge \mathrm{I})\ \dfrac{\begin{matrix}\vdots \\ A\end{matrix} \quad \begin{matrix}\vdots \\ B\end{matrix}}{A \wedge B}}{B} \ \leadsto\ \begin{matrix}\vdots \\ B\end{matrix}
$$

Disjunction:

$$
(\vee \mathrm{E})\ \dfrac{(\vee \mathrm{I}_1)\ \dfrac{\begin{matrix}\vdots \\ A\end{matrix}}{A \vee B} \quad \begin{matrix}[A] \\ \vdots \\ C\end{matrix} \quad \begin{matrix}[B] \\ \vdots \\ C\end{matrix}}{C} \ \leadsto\ \begin{matrix}\vdots \\ A \\ \vdots \\ C\end{matrix} \qquad\qquad (\vee \mathrm{E})\ \dfrac{(\vee \mathrm{I}_2)\ \dfrac{\begin{matrix}\vdots \\ B\end{matrix}}{A \vee B} \quad \begin{matrix}[A] \\ \vdots \\ C\end{matrix} \quad \begin{matrix}[B] \\ \vdots \\ C\end{matrix}}{C} \ \leadsto\ \begin{matrix}\vdots \\ B \\ \vdots \\ C\end{matrix}
$$

Implication:

$$
(\Rightarrow \mathrm{E})\ \dfrac{(\Rightarrow \mathrm{I})\ \dfrac{\begin{matrix}[A] \\ \vdots \\ B\end{matrix}}{A \Rightarrow B} \quad \begin{matrix}\vdots \\ A\end{matrix}}{B} \ \leadsto\ \begin{matrix}\vdots \\ A \\ \vdots \\ B\end{matrix}
$$

Key observation: these transformations correspond to evaluation rules for functional languages!