Day 10.

1. Typing Functions

What can go wrong? \(12, (\lambda c.c) + 1\).

We need to extend our grammar of types:

\[ \mathcal{Y} \ni T ::= \text{Int} \mid T_1 \rightarrow T_2 \]

- 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.

- Why don’t we have to represent the closure in the application rule?

Let’s look at some simple derivations:

\[
\{a \mapsto \text{Int}, b \mapsto \text{Int} \rightarrow \text{Int}\} \vdash a : \text{Int} \\
\{a \mapsto \text{Int}\} \vdash \lambda b.a : (\text{Int} \rightarrow \text{Int}) \rightarrow \text{Int} \\
\emptyset \vdash (\lambda a.\lambda b.a) : \text{Int} \rightarrow (\text{Int} \rightarrow \text{Int}) \rightarrow \text{Int} \\
\emptyset \vdash 3 : \text{Int} \\
\{c \mapsto \text{Int}\} \vdash c : \text{Int} \\
\emptyset \vdash (\lambda a.\lambda b.a) \ 3 \ (\lambda c.c) : \text{Int} \\
\emptyset \vdash (\lambda a.\lambda b.a) \ (\lambda b.b) : \text{Int} \rightarrow \text{Int} \\
\emptyset \vdash (\lambda a.a) \ (\lambda b.b) : \text{Int} \rightarrow \text{Int}
\]

- 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 \( \text{Int} \rightarrow \text{Int} \) and \( (\text{Int} \rightarrow \text{Int}) \rightarrow (\text{Int} \rightarrow \text{Int}) \).

2. Basic Proof Theory

Historical notes:
• 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:

\[
(\wedge I) \quad \frac{A \quad B}{A \wedge B} \quad (\wedge E_1) \quad \frac{A \wedge B}{A} \quad (\wedge E_2) \quad \frac{A \wedge B}{B}
\]

Disjunction:

\[
(\vee I_1) \quad \frac{A}{A \vee B} \quad (\vee I_2) \quad \frac{B}{A \vee B} \quad (\vee E) \quad \frac{A \vee B \quad C \quad C}{C}
\]

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

Implication:

\[
(\Rightarrow I) \quad \frac{B}{A \Rightarrow B} \quad (\Rightarrow E) \quad \frac{A \Rightarrow B \quad A}{B}
\]
Now, we can put together some simple derivations:

\[
\begin{align*}
(\land E_2) & \quad [A \land (B \lor C)]^p \\
(\lor E)^{q,r} & \quad B \lor C
\end{align*}
\]

\[
\begin{align*}
(\land I) & \quad \frac{A}{A \land B} \\
(\land E_1) & \quad \frac{[B]^q}{[A \land (B \lor C)]^p}
\end{align*}
\]

\[
\begin{align*}
(\lor I) & \quad \frac{A \land C}{(A \land B) \lor (A \land C)} \\
(\lor I_2) & \quad \frac{[C]^r}{[A \land (B \lor C)]^p}
\end{align*}
\]

\[
\begin{align*}
(\Rightarrow I)^p & \quad \frac{(A \land B) \lor (A \land C)}{A \land (B \lor C) \Rightarrow (A \land B) \lor (A \land C)}
\end{align*}
\]

- 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:

\[
\begin{align*}
(\top I) & \quad \frac{\top}{\top} & (\bot E) & \quad \frac{\bot}{A}
\end{align*}
\]

- No elimination rule for truth
- No introduction rule for falsity

- We define negation in terms of implication and falsity: \( \neg A = A \Rightarrow \bot \). This gives, as we expect, \( A \land \neg A \Rightarrow \bot \).
- Don’t actually need \( (\bot E) \) (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:

\[
\begin{align*}
(\land I) & \quad \frac{A \land B}{A \land B} \\
(\land E_1) & \quad \frac{A^p}{A}
\end{align*}
\]

Disjunction:

\[
\begin{align*}
(\lor I_1) & \quad \frac{A}{A \lor B} \\
(\lor E) & \quad \frac{B}{C}
\end{align*}
\]

Implication:

\[
\begin{align*}
(\Rightarrow I) & \quad \frac{A \Rightarrow B}{B} \\
(\Rightarrow E) & \quad \frac{A}{B}
\end{align*}
\]

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