Day 3.

1. Evaluation May Be Partial

Let's extend our language a little:

$$z \in \mathbb{Z} \qquad b \in \mathbb{B} \qquad v \in \mathbb{Z} \cup \mathbb{B}$$
$$\mathcal{E} \ni e ::= z \mid e_1 + e_2 \mid e_1 \times e_2 \mid b \mid \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3$$

and correspondingly, extend our evaluation relation:

$$\cdots \qquad \frac{}{b \Downarrow b} \qquad \frac{e_1 \Downarrow \mathsf{T} \quad e_2 \Downarrow v}{\mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \Downarrow v} \qquad \frac{e_1 \Downarrow \mathsf{F} \quad e_3 \Downarrow v}{\mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \Downarrow v}$$

Note that we use meta-variables to stand in for side-conditions: addition is only defined for expression that return integers, while conditionals are only defined when the condition expression is Boolean.

Is our evaluation relation still total? Deterministic?

2. Characterizing Partiality

We could attempt to characterize when our evaluation relation does hold. We'll introduce an approximation of evaluation, called *typing*, which distinguishes been Boolean-producing expressions and integer-producing expressions. We begin by introducing a grammar for our approximations.

$$\mathcal{T} \ni t ::= \texttt{Int} \mid \texttt{Bool}$$

We can then describe our approximation, building from our evaluation rules.

$$\frac{}{z : \texttt{Int}} \qquad \frac{e_1 : \texttt{Int} \quad e_2 : \texttt{Int}}{e_1 + e_2 : \texttt{Int}} \qquad \frac{e_1 : \texttt{Int} \quad e_2 : \texttt{Int}}{e_1 \times e_2 : \texttt{Int}}$$

$$\frac{}{b : \texttt{Bool}} \qquad \frac{e_1 : \texttt{Bool} \quad e_2 : t \quad e_3 : t}{\mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 : t}$$

- The : symbol was originally $\in$, motivated by the idea that types described sets of expressions
- Our rule for if has collapsed two rules in the evaluation relation, because we cannot tell from the approximation of a Boolean value whether it is true or false. What are the consequences of this approximation?

How can we relate the type system (our approximation) to the original system? We identify two properties.

Definition 3.1. Our type system is *sound* if its claims are all borne out by evalution. In particular:

3.

- If $e : \texttt{Int}$, then there is some $z \in \mathbb{Z}$ such that $e \Downarrow z$; and,
- If $e : \texttt{Bool}$, then there is some $b \in \mathbb{B}$ such that $e \Downarrow b$.

Definition 3.2. Our type system is *complete* if everything observable with evaluation is claimed by the type system. In particular:

- If there is some $z \in \mathbb{Z}$ such that $e \Downarrow z$, then $e : \texttt{Int}$; and,
- If there is some $b \in \mathbb{B}$ such that $e \Downarrow b$, then $e : \texttt{Bool}$

Our type system is sound, roughly by construction. To show this formally, we would do a proof by structural induction on typing derivations.

Our type system is not complete, however. For example:

$$\frac{\dfrac{}{\mathsf{T} \Downarrow \mathsf{T}} \quad \dfrac{}{1 \Downarrow 1}}{\mathsf{if}\ \mathsf{T}\ \mathsf{then}\ 1\ \mathsf{else}\ \mathsf{F} \Downarrow 1}$$

but there is no corresponding typing derivation:

$$\frac{\dfrac{}{\mathsf{T} : \texttt{Bool}} \quad \dfrac{}{1 : \texttt{Int}} \quad \overset{\lightning}{\mathsf{F} : \texttt{Int}}}{\mathsf{if}\ \mathsf{T}\ \mathsf{then}\ 1\ \mathsf{else}\ \mathsf{F} : \texttt{Int}}$$

3. The Goal

In general, can we have a sound and complete characterization of a property like safety?

No! Rice's theorem says that any non-trivial property of the partial computable functions is itself undecidable.

(Reduction to halting problem: given program p input x, is the function $y \mapsto p(x); y$ the identity function?)

But does this mean that we can't prove anything about programs? No! We certainly can prove that $y \mapsto y$ is the identity function.

The goal: identify *useful* subsets of programs for which *desirable* properties are provable.