Day 4

1. Evaluation May Be Partial

Let's extend our language a little:

$$\mathcal{T} \ni t ::= z \mid t_1 + t_2 \mid t_1 \times t_2 \mid t_1 - t_2 \mid t_1 \div t_2$$

and correspondingly, extend our evaluation relation:

$$\cdots \qquad \frac{t_1 \Downarrow z_1 \quad t_2 \Downarrow z_2}{t_1 - t_2 \Downarrow z_1 - z_2} \qquad \frac{t_1 \Downarrow z_1 \quad t_2 \Downarrow z_2}{t_1 - t_2 \Downarrow \lfloor {}^{t_1}/_{t_2} \rfloor}\ (z_2 \neq 0)$$

Is our evaluation relation still total? Deterministic?

No! a deterministic algorithm is an algorithm which, given a particular input, will always produce the same output, with the underlying machine always passing through the same sequence of states.

2. Characterizing Partiality

We could attempt to characterize when our evaluation relation does hold. We'll begin by extending the $\pm$ semantics to incorporate the new cases. Again, we need some lookup tables.

$\hat{-}$	$-$	0	$+$
$-$	$\{-,0,+\}$	$\{-\}$	$\{-\}$
0	$\{-\}$	$\{0\}$	$\{+\}$
$+$	$\{+\}$	$\{+\}$	$\{-,0,+\}$

$\hat{\div}$	$-$	0	$+$
$-$	$\{+\}$	$\emptyset$	$\{-\}$
0	$\{0\}$	$\emptyset$	$\{0\}$
$+$	$\{-\}$	$\emptyset$	$\{+\}$

Using them, we can define new inference rules for evaluation (or *evaluation rules*)

$$\cdots \qquad \frac{t_1 \Downarrow_{\pm} S_1 \quad t_2 \Downarrow_{\pm} S_2}{t_1 - t_2 \Downarrow_{\pm} \bigcup\{s_1 \hat{-} s_2 \mid s_1 \in S_1, s_2 \in S_2\}} \qquad \frac{t_1 \Downarrow_{\pm} S_1 \quad t_2 \Downarrow_{\pm} S_2}{t_1 \div t_2 \Downarrow_{\pm} \bigcup\{s_1 \hat{\div} s_2 \mid s_1 \in S_1, s_2 \in S_2\}}$$

We already have some neat results. Consider:

$$\frac{\overline{6 \Downarrow_{\pm} \{+\}} \quad \overline{0 \Downarrow_{\pm} \{0\}}}{6 \div 0 \Downarrow_{\pm} \emptyset} \qquad \frac{\overline{6 \Downarrow_{\pm} \{+\}} \quad \dfrac{\overline{0 \Downarrow_{\pm} \{0\}} \quad \overline{0 \Downarrow_{\pm} \{0\}}}{0 + 0 \Downarrow_{\pm} \{0\}}}{6 \div (0 + 0) \Downarrow_{\pm} \emptyset}$$

集合{+} {0} ==>{empty}

but unfortunately:

$$\frac{\overline{6 \Downarrow_{\pm} \{+\}} \quad \dfrac{\overline{6 \Downarrow_{\pm} \{+\}} \quad \overline{6 \Downarrow_{\pm} \{+\}}}{6 - 6 \Downarrow_{\pm} \{-,0,+\}}}{6 \div (6 - 6) \Downarrow_{\pm} \{-,+\}}$$

但集合多的可能性情况下，有多个结果，不符合determinstic

过多的预测结果

Key idea: $\Downarrow_{\pm}$ *over-approximates* the behavior of $\Downarrow$. So while we have a guarantee one direction:

$$t \Downarrow z \implies t \Downarrow_{\pm} S \land \mathsf{signum}(z) \in S$$

we do *not* have a guarantee the other direction:

$$t \Downarrow_{\pm} S \land s \in S \;\not\Longrightarrow\; t \Downarrow z \land \mathsf{signum}(z) = s$$

given

$$\mathsf{signum}(z) = \begin{cases} - & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ + & \text{otherwise} \end{cases}$$

So $\Downarrow_{\pm}$ doesn't fully characterize when terms evaluate, although it gets us much of the way there. To get the rest of the way, I'll define a relation called "safety":

$$\frac{}{n\ \mathsf{safe}} \qquad \frac{t_1\ \mathsf{safe} \quad t_2\ \mathsf{safe}}{t_1 + t_2\ \mathsf{safe}} \qquad \frac{t_1\ \mathsf{safe} \quad t_2\ \mathsf{safe}}{t_1 - t_2\ \mathsf{safe}} \qquad \frac{t_1\ \mathsf{safe} \quad t_2\ \mathsf{safe}}{t_1 \times t_2\ \mathsf{safe}} \qquad \frac{t_1\ \mathsf{safe} \quad t_2\ \mathsf{safe} \quad t_2 \Downarrow_{\pm} S \quad 0 \notin S}{t_1 \div t_2\ \mathsf{safe}}$$

Why not just $\dfrac{}{t_1 + t_2\ \mathsf{safe}}$? Addition never "goes wrong".

Is safety sufficient (*sound*)? Want: $t\ \mathsf{safe} \implies \exists z. t \Downarrow z$.

Is safety necessary (*complete*)? Want: $\exists z. t \Downarrow z \implies t\ \mathsf{safe}$.

Summary: Just as $\Downarrow_{\pm}$ *over*-approximates the behavior of $\Downarrow$, safety *under*-approximates the behavior of $\Downarrow$.

少预测，多限制条件

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.