Review: A simple functional language and its semantics

On Wednesday, September 17, we designed a little language of expressions with this syntax:

\[ e \Rightarrow \lambda x.e \mid e_1 e_2 \mid x \mid \text{if } e_1 e_2 e_3 \]

We decided to endow our language with a rich collection of values, including booleans, lists, primitive functions, closures, and several kinds of numbers.

We gave the language an operational semantics using the judgment form \( \langle e, \rho \rangle \Downarrow v \), with these rules:

\[
\begin{align*}
\langle v, \rho \rangle &\Downarrow v \\
\langle x, \rho \rangle &\Downarrow \rho(x) \\
\langle \lambda x.e, \rho \rangle &\Downarrow \{ \lambda x.e, \rho \} \\
\langle e_1, \rho \rangle &\Downarrow \{ \lambda x.e, \rho \} \\
\langle e_2, \rho \rangle &\Downarrow v \\
\langle e, \rho \{ x \mapsto v \} \rangle &\Downarrow v' \\
\langle e_1 e_2, \rho \rangle &\Downarrow v' \\
\langle \text{if } e_1 e_2 e_3, v \rangle &\Downarrow \\
\langle \text{if } e_1 e_2 e_3, v \rangle &\Downarrow
\end{align*}
\]

Finally, we agreed that our language would include an initial basis containing a rich set of arithmetic and relational primitives.

Bonus question

This one is intended for you to think about at home, although if you get stuck writing semantics, it may help you get unstuck:

3. Design a type system for our language. To keep things simple, write nondeterministic typing rules, so that we can imagine type inference without requiring any annotations.

A simple probabilistic language

Please answer the following two questions:

1. Make our language probabilistic by adding syntax, primitive functions, or both.

2. Think about what could reasonably be considered an operational semantics for a probabilistic language. (Based on what we’ve done in class so far, I could imagine two reasonable alternatives.)

Then, write the evaluation judgment and rules of such a semantics.