RECITATION 2

1. PROOFS ARE PROGRAMS

As discussed previously in lecture, there is a tight correspondence between the structure of a derivation for a constructive proof and a term in some particular programming language. This leads to the slogans “proofs are programs” and “propositions are types”. The (Curry-Howard-Lambek) correspondence can be fleshed out for the logic we’re studying (intuitionistic propositional logic\(^1\)) by the following table

<table>
<thead>
<tr>
<th>Propositions</th>
<th>Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A \land B)</td>
<td>(A \land B)</td>
</tr>
<tr>
<td>(A \lor B)</td>
<td>(A \lor B)</td>
</tr>
<tr>
<td>(A \supset B)</td>
<td>(A \supset B)</td>
</tr>
<tr>
<td>(\top)</td>
<td>(1) (unit)</td>
</tr>
<tr>
<td>(\bot)</td>
<td>(0) (void)</td>
</tr>
</tbody>
</table>

Based on this we can produce a version of our rules from the previous recitation that annotate each proposition step in the derivation with the program that it constructs. Those rules are

\[
\begin{align*}
M &: A \\
N &: B \\
\langle M, N \rangle &: A \land B \\
\text{inl}(M) &: A \lor B \\
\text{inr}(M) &: A \lor B \\
\text{fst}(M) &: A \\
\text{snd}(M) &: B \\
\text{fn} u &: A \supset B \\
\text{case} M \text{ of \ } \text{inl} c \Rightarrow N | \text{inl} c \Rightarrow R : C \\
\text{fn} u \Rightarrow M &: A \supset B \\
M(N) &: A \\
\end{align*}
\]

2. TRANSLATION

We now turn to the question of translating proofs to programs and back again. In these notes, we present both for the sake of accessibility.

(1) \((A \supset B \supset C) \supset (B \supset A \supset C)\)

Proof:

\[
\begin{align*}
A \supset B \supset C \quad \text{true} f \\
A \quad \text{true} a \\
B \supset C \quad \text{true} b \\
\quad \text{true} E \\
\quad \supset I^a \\
B \supset A \supset C \quad \text{true} b \\
\quad \supset I^b \\
(A \supset B \supset C) \supset (B \supset A \supset C) \quad \text{true} \supset I^f \\
\end{align*}
\]

Program: \(\text{fn} f = \Rightarrow \text{fn} b = \Rightarrow \text{fn} a = \Rightarrow (f \ a) \ b\)

Date: January 21, 2020.

\(^1\)Of course, what makes this correspondence so remarkable is that it extends far beyond this one logic. It is quite robust and extends to almost any well-behaved logic. It also maps between logic and functional programming and lattices which are just closed cartesian categories
(2) \((A \supset B) \lor (A \supset C) \supset A \supset (B \lor C)\)

Proof:

\[
\begin{array}{c}
(A \supset B) \lor (A \supset C) \quad \text{true}^f \\
A \supset B \quad \text{true}^f \\
A \supset C \quad \text{true}^g \\
A \quad \text{true}^a \\
E \\
B \quad \text{true} \\
B \lor C \quad \text{true} \\
C \quad \text{true} \\
E \lor I_1 \\
B \lor C \quad \text{true} \\
A \supset (B \lor C) \quad \text{true} \\
A \supset (B \lor C) \quad \text{true}^{fg} \\
\end{array}
\]

Program: \(\text{fn } x \Rightarrow \text{case } \text{snd } x \text{ of } \text{inl } b \Rightarrow \text{inl } (\text{fst } x, b) \mid \text{inr } c \Rightarrow \text{inr } (\text{fst } x, c)\)

3. Inventing proof terms

Let’s consider a new connective \(\&\). We’ll give the intro and elim rules and try to come up with constructors, destructors and reduction rules that make sense.

\[
\begin{array}{c}
B \quad \text{true}^u \\
A \quad \text{true} \\
A \land B \quad \text{true} \\
\end{array}
\]

\[
\begin{array}{c}
B \quad \text{true}^u \\
A \quad \text{true} \\
A \land B \quad \text{true} \\
\end{array}
\]

\[
\begin{array}{c}
A \supset B \quad \text{true}^a \\
A \supset \neg B \quad \text{true}^v \\
\neg A \quad \text{true}^u \\
B \quad \text{true}^v \\
\neg B \quad \text{true}^v \\
\neg A \quad \text{true}^u \\
\end{array}
\]

\[
\begin{array}{c}
A \land B \quad \text{true} \\
C \quad \text{true} \\
C \quad \text{true} \\
\end{array}
\]

Let’s come up with constructors that make sense for \(\&\).

\[
\begin{array}{c}
M : A \\
N : \bot \\
\text{inl}(M, u.N) : A \land B \\
\end{array}
\]

\[
\begin{array}{c}
M : \bot \\
N : B \\
\text{inr}(u.M, N) : A \land B \\
\end{array}
\]

And the destructor...

\[
\begin{array}{c}
E : A \land B \\
M : C \\
N : C \\
\end{array}
\]

Now we still need to define a reduction rule for \(\land\). Reduction rules are applied when the destructor is applied to a constructor.

\[
\text{case } E \text{ of } \text{inl}(u, v) \Rightarrow M \mid \text{inr}(u, v) \Rightarrow N \Rightarrow^* \text{ [fn } u' \Rightarrow M' \text{/ } v]\text{M}
\]

\[
\text{case } \text{inr}(u'.N', M') \text{ of } \text{inl}(u, v) \Rightarrow M \mid \text{inr}(u, v) \Rightarrow N \Rightarrow^* \text{ [fn } u' \Rightarrow N' \text{/ } u, M' \text{/ } v]N
\]
4. Reductions

Let’s try reducing a term until we can no longer apply reduction rules.

\[(\text{fn} \ a \Rightarrow \text{fn} \ b \Rightarrow (\text{fn} \ f \Rightarrow \text{fn} \ p \Rightarrow (\text{fn} \ (f \Rightarrow \text{fst}(p) \Rightarrow \text{snd}(f \Rightarrow \text{snd}(p)) \Rightarrow (\text{fn} \ u \Rightarrow a, \text{fn} \ u \Rightarrow b) \Rightarrow (\text{b}, \text{a})))) \]

Notice at this point we have a few options on how to proceed. It’s actually the case that there is a term that we will reach no matter which order we apply reduction rules. It’s generally known as the Church Rosser theorem that if a term finishes reducing in two ways, then they arrive at the same place. With our system we’ll always reach a “normal” form after a finite number of reductions, so we can apply rules in whatever order we wish.

\[(\text{fn} \ a \Rightarrow \text{fn} \ b \Rightarrow (\text{fn} \ p \Rightarrow (\text{fn} \ u \Rightarrow a, \text{fn} \ u \Rightarrow b) \Rightarrow (\text{fn} \ u \Rightarrow a, \text{fn} \ u \Rightarrow b) \Rightarrow (\text{snd}(p)) \Rightarrow (\text{b}, \text{a}))) \]

\[(\text{fn} \ a \Rightarrow \text{fn} \ b \Rightarrow (\text{fn} \ p \Rightarrow (\text{fn} \ u \Rightarrow a, \text{fn} \ u \Rightarrow b) \Rightarrow (\text{snd}(p)) \Rightarrow (\text{b}, \text{a}))) \]

\[(\text{fn} \ a \Rightarrow \text{fn} \ b \Rightarrow (\text{fn} \ p \Rightarrow (a, \text{fn} \ u \Rightarrow b) \Rightarrow (\text{snd}(p)) \Rightarrow (\text{b}, \text{a}))) \]

\[(\text{fn} \ a \Rightarrow \text{fn} \ b \Rightarrow (\text{fn} \ p \Rightarrow (a, \text{fn} \ u \Rightarrow b) \Rightarrow (\text{snd}(p)) \Rightarrow (\text{b}, \text{a}))) \]

\[(\text{fn} \ a \Rightarrow \text{fn} \ b \Rightarrow (\text{fn} \ p \Rightarrow (a, b) \Rightarrow (\text{b}, \text{a}))) \]

\[(\text{fn} \ a \Rightarrow \text{fn} \ b \Rightarrow (\text{fn} \ p \Rightarrow (a, b) \Rightarrow (\text{b}, \text{a}))) \]

\[(\text{fn} \ a \Rightarrow \text{fn} \ b \Rightarrow (a, b) \Rightarrow (\text{b}, \text{a}))) \]

\[(\text{fn} \ a \Rightarrow \text{fn} \ b \Rightarrow (a, b)) \]