Lecture 4

Topics

1. The *propositions-as-types* semantics is also called the “Curry-Howard Isomorphism” and sometimes “proofs-as-programs”. This “logic” deals with how we know, how we process evidence and how most of (all of?) logic can be explained in terms of computation.

   Constructive type theory is the branch of computer science that studies these concepts. I think it illustrates why many people say that “computer science is about understanding and automating intellectual processes”.

2. We will do more complicated examples and come to see the need for organizing the *construction of evidence as proofs*. This is the proofs-as-programs idea from my 1971 article on constructive math as a programming language and the 1985 article with Joe Bates, “Proofs as Programs”.

3. The examples

   1. $\sim P \lor Q \Rightarrow (P \Rightarrow Q)$  
      (What about $\Leftarrow$?)
   2. $\sim (P \Rightarrow Q) \Rightarrow (P \Rightarrow \sim Q)$
   3. $\sim (P \lor Q) \Rightarrow \sim P \& \sim Q$
   4. $(P \lor \sim P) \Rightarrow \sim P \Rightarrow P$
   5. $(P \lor \sim P) \Rightarrow (P \Rightarrow Q) \Rightarrow \sim P \lor Q$

   Exercise for Monday: Try to prove $\sim \sim (P \lor \sim P)$ both informally and by our proof method. This example shows the need for a systematic method to search for proofs.

6. $(P \Rightarrow Q) \Rightarrow \sim \sim (\sim P \lor Q)$

   If we just start with $\lambda(f.\lambda(g.\text{?}))$ it’s hard to see the next step, even if we annotate the variables with types, e.g. $\lambda(f^{P\Rightarrow Q}.\lambda(g^{\sim(P\lor Q)}\text{?}))$. We need to study $g : (\sim P \lor Q) \Rightarrow \text{False}$, how can we use this? Try to prove $\sim P$, that will allow us to use $g$ to prove $\text{False}$, $g(\text{inl}(np))$.

How do we prove $\sim P$? We assume we have evidence $p$ for $P$ and try to get $\text{False}$. We see that using $f$ we can get $Q$ as $f(p)$. Now we get an *insight*. We still have $g$, so we get $\text{False}$ from $g(\text{inr}(f(p)))$! We seem to be done, but how can we organize all this into a “proof” or into evidence?
We organize this “top down” search for evidence into an object that can count as a proof. It is just the organization of a systematic search for the necessary evidence. It is built around the idea that we keep track of what we know on one hand and what we seek to know on the other. We use the sign ⊢, called a turnstyle, to separate them.

We write **assumptions ⊢ goal**

Then we build a “search tree” to achieve the goal using the assumptions. The “proof” ends up being a tree grown top down, roughly like this:

$$\vdash G$$

$$\vdash H_1 \vdash G_1 \quad \vdash H_2 \vdash G_2$$

$$\vdash H_{1,1} \vdash G_{1,1} \quad \vdash H_{1,2} \vdash G_{1,2}$$

(This is *not* the actual tree of the next search)

$$\vdash (P \Rightarrow Q) \Rightarrow \neg\neg (\neg P \lor Q)$$

build a $\lambda(f.\_\_)$

$$f : (P \Rightarrow Q) \vdash \neg\neg (\neg P \lor Q)$$

build a $\lambda(g.\_\_)$

$$f : (P \Rightarrow Q), g : (\neg P \lor Q) \Rightarrow False \vdash False \quad \text{use } g \text{ get } v, (v \text{ will be value of } g)$$

$\text{inl}(g)$

$$\vdash \neg P \lor Q$$

use $\text{inl}$

$$\vdash \neg P$$

build $\lambda(p.\_\_)$

$$p : P \vdash False$$

use $f$ get $w$

$$w : Q \vdash False$$

use $g$ get $u$

$\text{inr}(g)$

$$\vdash \neg P \lor Q$$

use $\text{inr}$

$$\vdash Q$$

by $w$

$$u : False \vdash False$$

by $u$

$$v : False \vdash False$$

by $v$

$$\lambda(f.\lambda(g.g(inl(\lambda(p.g(inr(f(p))))))))$$

2
To see how this value is built, look at the value of $v$, which is $g$ applied to $\text{inl}$ at the $\vdash P \lor Q$ goal. Then look at the value of $w$ which is $f$ applied to $p$. Note, $u$ is from an application of $g$ to an $\text{inr}$ of a value for $Q$, in particular $\text{inr}(w)$. 