Lecture 8

Intuitionistic Logic and Constructive Logic

Logic is perhaps the oldest "academic discipline", dating back to Aristotle. The *Organon* is Aristotle's treatise on logic, written in approximately 350 BCE. Modern logic includes Boole's writing on propositional logic and its algebraic treatment. On this view, propositions have *truth values*, and propositional logic is concerned with the logical connectives &, $\vee$, and $\sim$, for not. From these, implication can be defined, namely $A \Rightarrow B$ iff $\sim A \vee B$. This logic is widely taught and we assume it is known to those who read these notes. We call it *classical Boolean logic*.

Constructive logic is a less precise term, and its history is less clear. It can be associated with constructive mathematics, but the connection is not that informative because logic and language were not critical to constructive mathematics until Brouwer made it critical in his *intuitionistic mathematics*. In this realm the role of logic was critical, and there is a very precise notion of *intuitionistic logic* as distinct from classical logic. This subject was initiated by L.E.J. Brouwer already in 1907 and progressively elaborated by him and his students and followers up until this very day. Indeed for this very course we have made more precise the intuitionistic rule of logic called *ex falso quodlibet*.

The notes for this lecture include Per-Martin Löf's notes on the meaning of the logical constants. This is an explanation of intuitionistic logic. We will explore this logic in detail. The article by van Atten in the on-line *Stanford Encyclopedia of Philosophy* (SEP) is highly recommended (`http://plato.stanford.edu/`).

Another interesting source book is: *One Hundred Years of Intuitionism (1907-2007)*, Birkhäuser Basel, Boston, Berlin, 2008.

Computer Science and Intuitionism

It is remarkable how well Brouwer's views on logic are appropriate for computer science and how computer science validates them. For Brouwer, to *know* a mathematical proposition is to have *experienced* (judged or grasped) that a certain mental construction *realizes* its computational meaning. Before we attempt to prove a proposition, we must grasp its sense, i.e. its computational meaning. That is the first *logical act*– a point made by Frege. Understanding the computational meaning of a proposition enables us to know that a purported mental construction realizes that meaning. This is a kind of type checking that is not mechanical or decidable. It requires structured judgements that create evidence.

The most basic judgements are given by the *computation rules*. They tell us that one expression reduces to another or computes to another. For example, $\mathrm{ap}(\lambda(x.x); a)$ *reduces to* a by the rule of β-reduction for the lambda calculus. These rules are ingredients of proofs.

Proofs are, in general, ways of searching for the computational content of a proposition. They organize the claim that their computational content is that of the type needed to know the proposition we are thinking about. The proof is a specific piece of evidence that causes us to know the judgement. We also say that we know or believe a proposition if we have understood evidence for it. A proof provides this evidence.