Advanced Progamming Languages	**Lecture 36**
CS 6110 Spring 2015	**Fri. April 24, 2015**

Lecture 36

Topics

1. Comments on the "frog and mouse war" between L.E.J Brouwer and David Hilbert culminating in 1928. W.P van Stigt *Brouwer's Intuitionism*, N-H, 1990, page 100.

   Also Xanda Schofield looked up the phrase "frog and mouse wars". It comes from a parody of the Illiad, as a Battle of Frogs and Mice.

2. The type theory we are studying will turn out to give us much more than a type theory for programming languages. It is closely related to the type theory in Chapter 4 of Thompson. I will try to be clear about just what you should know from this very rich subject. One thing is the Bishop definition of a set which became Martin-Löf's basis for his type theory – M-L 82.

   The subject is very "hot" right now and is undergoing a lot of changes, but there is a *common core* that I will stress.

The elements of $\mathbb{Z}$ are the positive and negative numbers as imported from Lisp's BigNum package. We just need for now the *constants* or *canonical integers* as taught in school, that is 0, 1, -1, 2, -2, 3, -3,..., 15720731, -15720731,... There is a precise rule for generating these canonical decimal numbers. They are of "unbounded" or "infinite" precicision. We use infinite in this setting as a synonym for *unbounded*.

For A and B types, $A \to B$ is a type. To be precise about universe levels, if $A \in \mathbb{U}_i$ and $B \in \mathbb{U}_j$ then $A \to B \in \mathbb{U}_{max(i,j)}$.

The elements of $A \to B$ are *functions*, specifically $\lambda(x.b) \in A \to B$ iff for all $a \in A,\ b(a) \in B$ and $a = a' \in A \Rightarrow b(a) = b(a') \in B$. This is called extensional equality.

We say that $\lambda(x.b) = \lambda(y.b')$ iff for all $a \in A,\ b(a/x) = b'(a/y) \in B$. *Thus the elements are functions, not programs.*

For $B(x)$ a *family of types* indexed by A, the members of $x : A \to B(x)$ are λ-terms $\lambda(x.b(x))$ such that if $a \in A$, then $b(a) \in B(a)$.

A family of types $B(x)$ must have the property that if $a_1 = a_2$ in A then $B(a_1) = B(a_2)$. To understand this equality, we must first define equality on types (see Nuprl book page 139). Canonical types are equal iff they are α-equal. This is a very strong (tight) equality. For example $x : \mathbb{Z} \to \mathbb{Z} = y : \mathbb{Z} \to \mathbb{Z}$.

Summary of basic CTT types

$\mathbb{U}_1$, $\mathbb{U}_2$,...

$\mathbb{Z}$

$Void$, $Atom$

$a = b \in A$

$u < v$

$x : A \to B(x)$

$x : A \times B(x)$

$A + B$

Two of these seem out place, what are they? We mentioned other types in Lecture 35 that we will discuss later.

Additional types

$\{x : A|B(x)\}$, the *set types.*

$A//E$, *quotient types.*

This is just the syntax of canonical terms for types. All types will be considered to be in a universe, e.g. $\mathbb{Z} \in \mathbb{U}_1$, $\mathbb{U}_1 \in \mathbb{U}_2$, $\mathbb{U}_2 \in \mathbb{U}_3$, ... these 1, 2, 3, are *universe levels*, not numbers.

We now want to know the elements of those types. This is the interesting part of their definition, their meaning beyond pure sytanx.

$Void$ has no elements.

One method of defining types comes from Bishop 1967. He used it to define constructive sets. See Lecture 35, and learn his definition "by heart".