CS6180: Lecture 7

Robert Constable

September 12, 2017

1 Lecture Summary

In this lecture we will expand our coverage of the basic types of constructive type theory. In due course we will add types that are characteristic of Brouwer type theory, the most recent theory implemented by Nuprl. Recall that we have previously stressed in Lecture 6 that to define a type, we need to give it a name, then specify its canonical members and the notion of equality among them. One journal source of information about these types is the article from 2006 [1] and these articles [3, 2].

We will use the refinement style to present the rules of type theory. Recall that we use Frege's turnstile symbol, $\vdash$, separating the list of hypotheses H from the single goal G to be proved. Sequents, $H \vdash G$, are used to express rules. The goal is a single formula. Some sequent calculi allow multiple goals, but the Nuprl style does not. In the Nuprl book sequents are written as $H >> G$. We use $x_1 : A_1, ..., x_n : A_n \vdash G$.

1.1 Plans for the week

This Lecture 7 will introduce more types: *products* (x,&), *function* ($\rightarrow$,$\forall$), *disjoint union* ($+$,$\vee$,$\cup$), *universes* (U_i), *quotient* (A/=), *set types* $\{x : A|B(x)\}$, *intersection* ($\cap$, $A \cap B$), *dependent intersections* ($x : A \cap B(x)$) or $x : A \cap B(x)$.

partial types $\bar{A}$, old (impredicative) *recursive types* and *dependent records*. We will probably study this in Lecture 8.

Lecture 8 will continue the discussion of the new types and also introduce *Euclidean Geometry* (**EG**) as manifest in CTT with proofs of three or four propositions, discussion of constructivity and Euclid Prop 2. We might try to introduce a synthetic approach to the real numbers by introducing $\mathbb{N}$ into EG.

1.2 Suggested problems and questions

It would be a good exercise for students to answer the following two questions and write a one page answer for me to read.

Problem 1: What were Brouwer's reasons for changing the meaning of the logical operations and denying the law of excluded middle?

Problem 2: Write an informal constructive proof of some simple theorem about integers or natural numbers and *extract* a functional program from the proof of the theorem, e.g. every positive natural number has a least prime factor. Another good example is from PLCV, if a prime p divides the product of two positive natural numbers, say a and b, then it divides either a or b.

Reading: Please look over the article posted with this lecture: *The Structure of Nuprl's Type Theory.*

2 Current Types in Constructive Type Theory (CTT)

Here we reiterate with slightly different language what we said informally in Lecture 6 about the nature of types. To define a type, we are required to first choose a standard name for it, e.g. $\mathbb{N}$ for the natural numbers. Then we must provide standard names for the constants, also called the *canonical elements* of the type. It is also essential to say when two canonical elements are considered equal. The canonical names with a precise equality provide the basis for computation on the type. The data format must support the operations that characterize the type, such as addition for natural numbers. These operations give rise to *non canonical expressions*, such as 12 + 5.

There must be computation rules showing how to reduce non canonical expressions to canonical data, e.g. rules for reducing 12 + 5 to 17. It would be a good exercise to analyze Kleene's account of the natural numbers and relate it to the standard type-theoretic account. We will do this together in class.

Consider next the type of *ordered pairs* of natural numbers. We name the type of pairs of natural numbers this way, $\mathbb{N} \times \mathbb{N}$. The canonical pairs are usually denoted either as $pair(n; m)$ or as $< n, m >$ for n and m natural numbers. We also need operations for selecting the components of ordered pairs. Standard names for such operators are fst and snd for *first* and *second*. Alternative operator names are $1of$ and $2of$ which select the first and second components respectively. These operators are defined by *reduction rules* or *computation rules*. Here are the rules for pairs:

- $fst(pair(l; r)) \ = \ l.$
- $snd(pair(l; r)) \ = \ r.$

As we introduce new types, we need to provide the components illustrated above: canonical type name, canonical elements, definition of equality on the canonical elements, primitive operations on the canonical elements given by computational reduction rules. We now carry this out for the types mentioned above.

2.1 Ordered pairs

Type constructor: If A and B are types, then $A \times B$ is a type.

Canonical elements: If $a \in A$ and $b \in B$, then $pair(a; b)$ is a canonical element of $A \times B$.

Equality: If $(a = a') \in A$ and $(b = b') \in B$, then $(pair(a, b) = pair(a', b')) \in (A \times B)$.

Reduction rules:

$((fst\ pair(a; b)) \ = \ a) \in A$.

$((snd\ pair(a; b)) \ = \ b) \in B$.

2.2 Disjoint union

Type constructor: If A and B are types, then $A + B$ is a type.

Canonical elements: If $a \in A$ then $inl(a) \in A + B$, and if $b \in B$ then $inr(b) \in A + B$.

Reduction rule: If $d \in (A + B)$ and if $u \in A$ implies $lft(u) \in G$ and $v \in B$ implies $rt(v) \in G$, then $decide(d; u.lft(u); v.rt(v)) \in G$.

2.3 Function space

Type constructor: If A and B are types, then $A \to B$ is a type.

Canonical elements: If for any $x \in A$, $b(x) \in B$, then $\lambda(x.b(x)) \in A \to B$.

Reduction rule: If $f \in (A \to B)$ and $a \in A$, then $ap(f; a) \in B$.

References

[1] Stuart Allen, Mark Bickford, Robert Constable, Richard Eaton, Christoph Kreitz, Lori Lorigo, and Evan Moran. Innovations in computational type theory using Nuprl. *Journal of Applied Logic*, 4(4):428–469, 2006.

[2] R. L. Constable. The structure of Nuprl's type theory. In Helmut Schwichtenberg, editor, *Logic of Computation*, volume 157 of *Series F: Computer and Systems Sciences*, pages 123–156, Berlin, 1997. NATO Advanced Study Institute, International Summer School held in Marktoberdorf, Germany, July 25–August 6, 1995, Springer.

[3] Robert L. Constable and Jason Hickey. Nuprl's class theory and its applications. In Friedrich L. Bauer and Ralf Steinbrueggen, editors, *Foundations of Secure Computation*, NATO ASI Series, Series F: Computer & System Sciences, pages 91–116. IOS Press, 2000.