CS6180: Lecture 5

Robert Constable

September 5, 2017

1 Lecture Summary

We will look at how to define number theoretic functions and reason about them using mathematical induction. Kleene does this without introducing the notion of types. On the other hand types are just below the surface, and it is easy to make them explicit. Doing this will introduce the notion of types and type theory in a natural and gradual way. We can see from his notation that he intends to reason about the natural numbers. He adopts a very simple notation for the number "constants," namely 0, 0', 0", ... Another common notation for the successor is $S(x)$. In this case the natural numbers are $0, S(S(0)), S(S(S(0))), ...$ We give this type the name $\mathbb{N}$. It has a simple inductive definition which we can write as

$$0 \in \mathbb{N}, \qquad \frac{n \in \mathbb{N}}{S(n) \in \mathbb{N}}.$$

Number Theory Axioms

13. $A(0)\&\forall x.(A(x) \Rightarrow A(x')) \Rightarrow A(x)$.
14. $a' = b' \Rightarrow a = b$.
15. $\sim (a' = 0)$.
16. $a = b \Rightarrow (a = c) \Rightarrow b = c$.
17. $a = b \Rightarrow a' = b'$.
18. $a + 0 = a$.
19. $a + b' = (a + b)'$.
20. $a \times 0 = 0$.
21. $a \times b' = (a \times b) + a$.

We single out axiom 13, the *principle of mathematical induction*. We can also write it as

$$A(0)\&\forall x.(A(x) \Rightarrow A(x')) \Rightarrow \forall x.A(x).$$

It implicitly conveys typing information which we can make explicit by using the type of natural

numbers $\mathbb{N}$ and using this typed version of the axiom. This will lead us to talking explicitly about *types*.

$$A(0)\&\forall x : \mathbb{N}.(A(x) \Rightarrow A(x')) \Rightarrow \forall x : \mathbb{N}.A(x).$$

The type of natural numbers also has the rule that $0 \in \mathbb{N}$ and that $\dfrac{n \in \mathbb{N}}{S(n) \in \mathbb{N}}$.

We will examine an example from Kleene showing how to structure proofs as trees using Frege's turnstile symbol, $\vdash$, separating hypotheses from the goal to be proved, $H \vdash G$ where the hypotheses H is a *list* of labeled formulas and variable declarations, and the goal is a single formula. Such expressions are called *sequents*. In the Nuprl book they are written as $H >> G$. In these notes we use $x_1 : A_1, ..., x_n : A_n \vdash G$. Some formalisms, such as tableaux, allow multiple goals, say $G_1, ..., G_m$, but Nuprl does not.

2 Intuitionistic First-Order Logic, iFOL

In his 1908 PhD thesis [1], Brouwer proposed a new interpretation of first-order logic. His idea is that we come to know mathematical truths based on our intuitive experience of the *continuum of time* and how the mind breaks this continuum into the experience of now, before, and after. There is no ideal *Platonic world* in which mathematical statements are either true or false, and that we come to know these truths based on a conception of logic in which every meaningful mathematical statement is either true in the Platonic world or false there. This Platonic view was the prevailing understanding of truth before Brouwer proposed a compelling alternative. He justified his alternative understanding based on how we come to experience mathematical ideas and recognize those constructions which the human mind can grasp and explore. We are not able to experience this Platonic world, yet we can recognize simple logical truths as intuitively true and computationally meaningful in the sense of our mental constructions.

Brouwer gave the interpretation of mathematical truth that Per Martin-Löf writes about in several papers [6, 5]. These ideas are discussed well in our on-line textbook [7]. We will also study the presentation in *On the Meaning of the Logical Constants and the Justification of the Logical Laws* [6]. This semantics is also given in the Nuprl book [4] and in numerous other articles and books [3, 2].

Here is a brief summary of how we covered this topic in Lecture 4 stressing the notion of evidence and presenting the *evidence semantics* from the readings associated with this lecture. We go over the evidence semantics for the first-order logical operators from Kleene and give simple examples.

- the semantics of *conjunction*: to have evidence for $A\&B$ is an ordered pair *pair(a;b)* where a is evidence for A and b is evidence for B.

- the semantics of *implication*: to have evidence for $A \Rightarrow B$ is to have a computable function *fun(x.b(x))* such that if x is evidence for A, then b(x) is evidence for B.
- the semantics of *disjunction* $A \vee B$ is either *inl(a)* where a is evidence for A or *inr(b)* where b is evidence for B. The operator *inl* is mnemonic for *inject left* and *inr* is mnemonic for *inject right*. So we are thinking of injecting the evidence into the type of evidence for either the left disjunct or the right disjunct.
- the semantics *false* is the empty type, say *void*. This says simply that there is no evidence for false, so we can never "know it" independent of some assumptions. This leads to a natural computational semantics for negation.
- the semantics of $\neg A$: to have *evidence for the negation* of A is to have a function that can convert any evidence a for A into evidence for the empty type, *void*. Since there is no evidence for void, this function tells us that there can be no evidence for A either, thus we can never know A to be true.
- the semantics of $\exists x : D.A(x)$: to have evidence that there is an object of the *domain of discourse* D, say d such that we have evidence a for $A(d)$.
- the semantics of universal statements $\forall x : D.A(x)$: to have evidence that for all elements d of the domain of discourse D there is evidence for $A(d)$, we need a function *fun(x.a(x))* such that for any element d of D, $a(d)$ is evidence for the assertion $A(d)$.

We will know in detail what the evidence is for these statements once we investigate Kleene's axioms for arithmetic in more detail. We will have evidence for all of these claims.

$$0 < 1 \& 1 < 2,\ (x > 0) \Rightarrow ((x \times x) > 0),\ (0 = 0) \vee (0 = 1),\ \neg(0 = 1),\ \forall x : D.(x = 0) \vee (\neg(x = 0))$$

We now look at the evidence for the *induction axiom*. Here is the axiom with explicit type information and using $S(x)$ for the successor operation instead of Kleene's notation x'.

$$A(0) \& \forall x : \mathbb{N}.(A(x) \Rightarrow A(S(x))) \Rightarrow \forall x : \mathbb{N}.A(x).$$

The intuitive computation behind the induction axiom is that we start with evidence for $A(0)$ which is explicitly given to us in the very first clause, the *base case*. Call this $a0$. The function for extending this evidence is the computational content of the inductive clause $\forall x : \mathbb{N}.(A(x) \Rightarrow A(S(x)))$. Let us abbreviate the body of this function as $ind(x)$. So the induction function is $fun(x.ind(x))$. We know from the axiom that $ind(a0)$ is evidence for $A(1)$, that is $ind(0)$ is evidence for $A(S(0))$. Then we see that $ind(ind(0))$ is evidence for $A(S(0))$, that is for $A(1)$, so we write $a1 = ind(a0)$. The evidence for $A(S(S(0))$, i.e. for $A(2)$ is $ind^2(a0)$. We now "see" that the nth iteration of $ind(x)$ starting at $a0$ is evidence for $A(n)$, that is $an = ind^{(n)}(a0)$.

We will see the induction axiom at work in the derivation of the *integer square root function*. Here is a specification of the task.

Theorem/Task: $\forall n : \mathbb{N}.\exists r : \mathbb{N}.(r^2 \leq n \ \& \ x < (r+1)^2)$.

The inductive proof is given in the notes posted with this lecture, and the realizer or program that is extracted from the proof is also given directly from a Nuprl proof. Here is another presentation of the what we call the *extract* or the program generated from the proof.

$\lambda\, n.letrec\ sqrt(n) = if\ n = 0\ then\ 0\ else\ let\ r2 = sqrt(x{-}1) in\ letr3 = r2{+}1\ in\ if x < (r3)^2 then r2\ else\ r3.$

References

[1] L.E.J. Brouwer. *Over de grondslagen der wiskunde (On the foundations of mathematics).* PhD thesis, Amsterdam and Leipzig, 1907.

[2] Robert L. Constable. Naïve computational type theory. In H. Schwichtenberg and R. Steinbrüggen, editors, *Proof and System-Reliability, Proceedings of International Summer School Marktoberdorf, July 24 to August 5, 2001*, volume 62 of *NATO Science Series III*, pages 213–260, Amsterdam, 2002. Kluwer Academic Publishers.

[3] Robert L. Constable. Computational type theory. *Scholarpedia*, 4(2):7618, 2009.

[4] Robert L. Constable, Stuart F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, Douglas J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, James T. Sasaki, and Scott F. Smith. *Implementing Mathematics with the Nuprl Proof Development System.* Prentice-Hall, NJ, 1986.

[5] Per Martin-Löf. *Notes on Constructive Mathematics.* Almqvist & Wiksell, Uppsala, 1970.

[6] Per Martin-Löf. On the meaning of the logical constants and the justification of the logical laws. Lectures in Siena, 1983.

[7] Simon Thompson. *Type Theory and Functional Programming.* Addison-Wesley, 1991.