| **Advanced Progamming Languages** | **Lecture 39** |
|---|---|
| **CS 6110 Spring 2015** | **Fri. May 4, 2015** |

# Lecture 39

We need to discuss material from Lecture 38 that we did not finish due to an extended discussion of the propositions-as-types principle that led to consideration of the propositional interpretation of the following four types:

| | |
|---|---|
| $A \sqsubseteq B$ | *subtyping*, becomes "immediate implication", e.g. the evidence $a$ for $A$ is also evidence for $B$, i.e. $A \Rightarrow B$. |
| $A \cap B$ | *intersection*, becomes a strong &, e.g. the evidence for $A$ and $B$ can be the same, hence $A \cap B \Rightarrow A \& B$. |
| $\bigcap_{A:\mathbb{U}_i} (A \Rightarrow A)$ | *intersection* of an indexed family. The element $\lambda(y.y) \in (A \Rightarrow A)$ shows that $A \Rightarrow A$ is uniformly true. |
| $\bigcap_{x:D} P(x)$ | is $\forall[x:D].P(x)$, and we know that the evidence is uniform in $x$ as in $\forall[x:D].(P(x) \Rightarrow P(x))$ or $\forall[x:D].(P(x) \,\&\, Q(x) \Rightarrow P(x))$. |

In Lecture 38 there is a discussion of the close connection between programs with assertions (*asserted programs*) justified by varieties of programming logics based on Hoare logic and programs that are implicit *constructive proofs* of assertions of the form $\forall x{:}T_1.\exists y{:}T_2.R(x,y)$. This is the $\forall\exists$ pattern.

There are connections between proofs and programs for all forms of assertion, e.g. $\forall x{:}\mathbb{N}(P(x) \vee \sim P(x))$ provides a program for *deciding* the condition $P(x)$.

This lecture reviews both directions of this *correspondence between proofs and programs*, tying it to the details in the lecture and the contents of the attached article by Christoph Kreitz.

**Complete Induction**

We provide further explanation of the instance of complete induction used in the integer square root example by Christoph Kreitz – see attached notes.

He uses the principle $\forall P{:}\mathbb{N} \to \mathbb{U}_i.P(0) \,\&\, \forall i{:}\mathbb{N}.(P(i \div 4) \Rightarrow P(i)) \Rightarrow \forall i{:}\mathbb{N}.P(i)$.

This can be proved by *complete induction*. First let's see intuitively why this is correct.

| | | |
|---|---|---|
| $P(0 \div 4)$ is $P(0)$ | so $P(0 \div 4) \Rightarrow P(0)$ | since we proved $P(0)$. |
| $P(1 \div 4)$ is $P(0)$ | so $P(1 \div 4) \Rightarrow P(1)$ | is $P(0) \Rightarrow P(1)$, we must prove this, call it $P_1$. |
| $P(2 \div 4)$ is $P(0)$ | so $P(2 \div 4) \Rightarrow P(2)$ | is $P(0) \Rightarrow P(2)$, we must prove this, $P_2$. |
| $P(4 \div 4)$ is $P(1)$ | so $P(4 \div 4) \Rightarrow P(4)$ | since we already have $P(0) \Rightarrow P(1)$ |

So we are getting $P(0)\&...\&P(n) \Rightarrow P(n+1)$

This is called by some *course of values induction*. Kleene proves it (p.193) in this form.

$* \; \forall x{:}\mathbb{N}.(\forall y{:}\mathbb{N}.(y < x \Rightarrow P(y)) \Rightarrow P(x)) \Rightarrow \forall x{:}\mathbb{N}.P(x)$

Prove this using

$** \;\; P(0) \,\&\, \forall x.[(\forall y \leq x \Rightarrow P(y)) \Rightarrow P(x+1)] \Rightarrow \forall x{:}\mathbb{N}.P(x).$

Use $*$ to prove $\forall y.(y \leq x \Rightarrow P(y))$ by *ind* on $x$.

**Comments on the Final Exam – Thursday May 14, 2pm in Upson 211**

1. Aim for a 2 hour exam.
2. There will be a short answer part.
3. There will be a Hoare Logic example – while program with proof.
4. There will be a simple proof with code extraction.
5. Definition of the basic types and First-Order Logic types.
6. O'Donnell's critique of Hoare rules, and Winskel's semantics to justify them.
7. Denotational semantics from Winskel.
8. Know the statement of the Blum Size theorem and why it's interesting.
9. Know *Loop Language* significance.
10. Halting problem for partial types, idea of partial types.
11. Kleene's definition of partial recursive functions, the Herbrand-Gödel recursive functions.
12. Significance of Goodstein approach to reasoning about total recursive functions – but no details.
13. Foundations of type theory – computational, minimal knowledge required.
14. Foundations of type theory – conceptual, minimal "conceptual" knowledge required.