Advanced Progamming Languages	**Lecture 34**
CS 6110 Spring 2015	**Mon. April 20, 2015**

Lecture 34

Topics

1. Finish Loop language

   Bounding lemma.

2. Finish programming logics

   Hoare-procedure rule – flaw, reading.

   Issues with partial correctness, PLCV is total correctness

3. Problem set 4: Problem 5

   (a) Read O'Donnell for second example of a flaw. Describe the core of the issue and options to fix it.

   (b) Consider 91-function, how to prove termination (see end of this lecture).

4. Introduction to type theory

   Issues such as program vs. function.

   Role of type theory in PL and CS.

   Later in the week, CS contribution to foundations of mathematics.

Finishing Programming Logic

Read Michael O'Donnell, *A Critique of the Foundations of Hoare-Style Programming Logic*, CS Tech Reports, Purdue University, 1980.

This report makes interesting observations on logic as well. It is quite an insightful document, from a brilliant Cornell CS graduate.

Mike writes the Hoare logic "triples" as $A\{P\}B$ which Winskell writes as $\{A\}P\{B\}$, calling it (on p. 87) a *partial correctness assertion.*

He notes that the partial correctness interpretation of $\{A\}P\{B\}$ is that if A is true before P executes, and P terminates in state s', then B is true in s'. Winskel notes on p. 79, $\{true\}$ **while** $true$ **do** $skip$ $\{false\}$ is valid in Hoare logic.

Mike uses (on p.10) $A\{$**while** $true$ **do** $x := x$ **end**$\}$ holds for any assertions A and B.

Mike notes that Clint and Hoare in 1972 *Acta Informatica* proposed the *function-rule*:

$$\frac{A\{P\}B}{\forall x.(A \Rightarrow B(f(x)/y))}$$

where f is defined as f : **function**(x); **local** $z_1, ..., z_n$; P; $return(y)$ **end** (and z_i do not occur in A and B).

Consider f : **function**(x); $Fail$; $return(y)$; **end**.

Consider the proof

1. $True\,\{Fail\}\,False$, rule for $Fail$
2. $\forall x.(True \Rightarrow False)$, by function-rule
3. $False$, substitute 0 for x, use basic quantifier logic.

Mike notes that some authors add the proviso that "the function body must halt when A is true inititally". Mike says, "For a logical rule to be useful, we must be able to decide when the rule has been applied correctly" (p.16). We can't decide this condition!

Mike then goes on to note that the function rule essentially leads to *Russell's paradox*. He gives the "Russell" version (p.17).

Next Mike shows that Musser's attempted fix also fails. That was for the programming language *Euclid*.

He comments that in our book, *A Programming Logic*, 1978, we use a total correctness logic to avoid these problems.

The Nuprl type theory deals with partial correctness using *partial types*.We will examine this next week.

PS4

Here is the last problem on PS4, Problem 5.

(a) Consider another way in which the function rule might be "fixed". Consider how partial types are used and whether partial propositions might make sense.

(b) Consider the following definition of a recursive function, clearly in type $\mathbb{Z} \to \overline{\mathbb{Z}}$.

$f(x) =$ **if** $x > 100$ **then** $x - 10$ **else** $f(f(x + 11))$ **fi** show that f is equal to the "91-function", i.e. let $f_{91}(x) =$ **if** $x > 100$ **then** $x - 10$ **else** 91 **fi** and prove that $f(x) = f_{91}(x)$ for all $x \in \mathbb{Z}$.

We can also use the type $\mathbb{N}$ and $\overline{\mathbb{N}}$ if we use $x \dot{-} 10$, e.g. for $x \leq 10$ the value is 0.