Advanced Progamming Languages	**Lecture 33**
CS 6110 Spring 2015	**Fri. April 17, 2015**

Lecture 33

Topics

1. PS4 – there will be one more problem assigned on Monday. Recall that the main point of Problem 1 is to show why the LCF fixed point induction principle needs an *admissibility* condition. This condition is similar to Winskel's requirement of an inclusive predicate in his account. You can basically use any cpo D to illustrate the issue. I first suggest $\bar{\mathbb{N}}$.

2. Finishing the rules for First-Order Logic (FOL) in the style of programming logic. This style is essentially "block structured natural deduction". It has been extensively studied in logic. The other styles are *Hilbert style*, just axioms and simple inference rules (used in Kleene), and the *sequent calculus* or *refinement logic*, as used in Coq and Nuprl respectively.

3. Close examination of the program and proof of Gauss's formula, $\sum_{i=0}^{n} = \frac{n\cdot(n+1)}{2}$.

4. Introduction to the Loop language, Meyer and Ritchie 1967 technical report from IBM research.

Introduction to the Loop Language, Meyer and Ritchie 1967

Why are Loop programs interesting?

- An upper bound on the *run time* is determined by program structure in a simple way.
- They are a *natural sublanguage* of all standard procedural languages.
- They compute precisely the primitive recursive functions.
- One of the simplest rich *subrecursive languages*, so they can give insight into the Blum size theorem.

Syntax – register names $X_1, X_2, ..., X_n, ...$

Syntax – instructions

1. $X = Y$
2. $X = X + 1$
3. $X = 0$
4. **LOOP** X P **END**, where P is a loop program.

Definition: A *Loop program* P is any sequence of basic instructions (1 to 3) or a loop instruction of type 4 where P is a Loop program.

The semantics of **LOOP** X P **END** is the same as the PL1 program, (see PLCV p.85),

do $I = X$ to 1 **by** -1

P

end.

If the program P terminates, which it will, it is executed exactly X times $P(X),\ P(X-1),\ P(X-2), ..., P(1)$.

For example **LOOP** $(X = X + 1)$ **END** computes $2 \cdot X$ in $2 \cdot X + 2$ steps.

Sometimes we write $X := X + 1$, $X := 0$ for the assignments so that $X = 0$ etc., can be read as *assertions*.

Program	State (with assertions about X)
$X := 0$	$X = 0$
$X := X + 1$	$X = 1$
$X := X + 1$	$X = 2$
LOOP X	
$X := 1$	$X = 4$
END	

Reading: Please read Meyer & Ritchie pages 1-6 and 12-13.

Properties of Loop programs

Definition of L_n

L_0 no loops
L_n loops nested to depth n.

Definition 2.1. $Loop = \bigcup_{n=0}^{\infty} L_n$, page 2.

Definition 3.1. Iterating g $\quad h(z, y) = g^{(y)}(z)$

Definition 3.2. Bounding functions f_n

$f_0(0) = 1 \quad f_0(1) = 2 \quad f_0(x) = x + 2$ for $x > 1$.
$f_{n+1}(x) = f_n^{(x)}(1)$

Bounding Theorem 3.3. If $P \in L_n$, then we can find $p > 0$ such that $f_n^{(p)}$ bounds the running time of P.