Advanced Progamming Languages	**Lecture 24**
CS 6110 Spring 2015	**Fri. March 20, 2015**

Lecture 24

Topics

1. Review of least fixed points – classical proof and Kleene's constructive proof. The computational significance.
2. Applications of a generalized recursion theorem to *Edinburgh LCF* 1979 grounded in Dana Scott's 1972 paper on *fixed point induction* and lattice theoretic models of type free logics.
3. Issues in reasoning about partial computable functions, experience with Edinburgh LCF and programming logics.
4. Discussion of Voevodsky's Cornell lecture and his *C-systems*.

Recall the definition of $x \sqsubseteq y$ for $x, y : \overline{\mathbb{N}}$ and $f \sqsubseteq g$ for $f, g : \overline{\mathbb{N}} \to \overline{\mathbb{N}},\ f(x) \sqsubseteq g(x)$ for $x : \overline{\mathbb{N}}$. Note $\bot \sqsubseteq n$ for $n \in \mathbb{N}$.

We say that $f : \overline{\mathbb{N}} \to \overline{\mathbb{N}}$ is monotonic iff $x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y)$.

A functional $F : (\overline{\mathbb{N}} \to \overline{\mathbb{N}}) \to (\overline{\mathbb{N}} \to \overline{\mathbb{N}})$ is:

(a) *Monotonic* iff $f \sqsubseteq g \Rightarrow F(f) \sqsubseteq F(g)$.

(b) *Continuous* iff for any chain of partial functions $f_1 \sqsubseteq f_2 \sqsubseteq ... \sqsubseteq f_i,\ F(\text{lub } f_i) = \text{lub } F(f_i)$ where lub f_i is the least $g \in \overline{\mathbb{N}} \to \overline{\mathbb{N}}$ such that $f_i(x) \sqsubseteq g(x)$ for all $x : \overline{\mathbb{N}}$

Theorem *Every partial recursive functional $F : (\overline{\mathbb{N}} \to \overline{\mathbb{N}}) \to (\overline{\mathbb{N}} \to \overline{\mathbb{N}})$ is continuous.*

This is part of Kleene's recursion theorem, the part that is hardest to tease out because of his equational model for general recursive computation. It is easier to see with the lambda calculus.

Theorem (Kleene generalized recursion theorem with explicit continuity, Manna's version) *Given any continuous computable function $F : (\overline{D} \to \overline{D}) \to (\overline{D} \to \overline{D})$, we can find a least fixed point f_ω such that $F(f_\omega) \simeq f_\omega$. In fact, $f_\omega =$ lub $F^i(\bot)$ where $\bot$ is the completely undefined function on $\overline{D} \to \overline{D}$.*

Kleene's Generalized Recursion Theorem à la domain theory, Proof:

1. Define the chain $F(\varphi_0) = \varphi_1,\ F(\varphi_1) = \varphi_2, ..., F(\varphi_i) = \varphi_{i+1}$ where $\varphi_0(x) = \perp$ for all x. We called this $\varphi_0, \varphi_1, \varphi_2, ..., \varphi_\omega$, i.e. $\text{lub}\, \varphi_i = \varphi_\omega$.
2. Since F is continuous we can show that φ_ω is a fixed point, $F(\varphi_\omega) = \varphi_\omega$.

   $F(\varphi_\omega) = F(\text{lub}\ \varphi_i),\ i = 0, 1, 2, ...$
3. For any fixed point ψ of F, $F(\psi),\ \varphi_\omega \sqsubseteq \psi$.
   (a) $F^i(\varphi_0) \sqsubseteq \psi$ for all i by induction on i.

   $F^0(\varphi_0) = \varphi_0 \sqsubseteq \psi$ and if $F^{i-1}(\varphi_0) \sqsubseteq \psi$ then since F is monotonic

   $F^i(\varphi_0) = F(F^{i-1}(\varphi_0)) = F(\psi) = \psi$.

   So $F^i(\varphi_0) \sqsubseteq \psi$ for all i, and thus ψ is an upper bound. But φ_ω is the lub of $F^i(\varphi_0)$ thus $\varphi_\omega \sqsubseteq \psi$.

■

The *Edinburgh LCF* system was an implementation of Dana Scott's classical logic of computable functions based on domain theorey. This theory did not "catch on". We have generalized and improved it as part of constructive type theory, CTT, implemented by the Nuprl proof assitant.

We will look at the LCF *fixed point induction rule.*

$$\begin{array}{rcl} H_1,\ H_2 & \vdash & A(fix(f)/x) \\ H_1 & \vdash & A(\perp /x) \\ H_2,\ x : \overline{D} \to \overline{D},\ A(x) & \vdash & A(f(x)) \end{array}$$

Conditions:

1. No x in H_2
2. A admits induction.

Extending functions to partial functions seems arbitrary.

$\perp + y = ?$

$\perp * 0 = 0\,?$

$0 * \perp = 0\,?$

$0/0 * \perp = ?$