Advanced Progamming Languages	**Lecture 40**
CS 6110 Spring 2015	**Weds. May 6, 2015**

Lecture 40

Topics

1. We will not cover the logic of events nor monads. There is not enough time for the former, and I was talked out of the latter by my colleagues. Let me just note that we covered the key application well when we showed how to define state and give semantics to imperative programs.

   I recommend Spring 2014 Lecture 41 by Kozen on the algebraic structure of monads.

2. I need to modify my comment on the final exam. Nine CS PhD students will join our exam. That makes it easier for you all since I need to cover the basics as well; thus the basic λ-calculus, typed λ-calculus, partial types, fixed points. I'll say more in lecture.

3. First, let's finish up propositions-as-types and cover the new results on classical logic I mentioned at the end of Lecture 39.

Propositions-as-types

How do we know that "types capture all of constructive logic"? What does that mean? Typically "logic" means first-order logic, FOL. Sometimes it includes second-order logic or even all of higher-order logic, HOL. Here is the difference:

FOL: &, $\vee$, $\Rightarrow$, $\sim$, $\forall x\!:\!D.P(x)$, $\exists x\!:\!D.P(x)$, $False$.

HOL: all of FOL and $\forall P\!:\!Prop.F(P)$, $\exists P\!:\!Prop.F(P)$.

To say that the *evidence semantics* of types captures FOL is to say that by treating propositions as types and defining the constructive evidence for knowing them, *we can find evidence for all provable propositions in iFOL.*

We don't want to spend a great deal of time on proof systems, e.g. natural deduction, sequents (refinement), but we can sketch the ideas for how proofs build evidence.

The sequent rules are very intuitive, especially in the refinement (top down) style. We'll post with this lecture a complete set of rules for iFOL. Note, we can think of a proving task or a programming task in the form

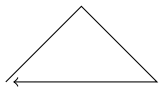
$$H_1, ..., H_n \vdash G$$

where the H_i are the assumptions under which we are proving the goal G or solving the programming problem posed by the proposition (type) G.

Implication – Introduction

$H \vdash A \Rightarrow B$ by $\lambda(x.\_\_)$

$H,\ x\!:\!A \vdash B$ by $b(x)$



Implication – Elimination

$H,\ f\!:\!A \Rightarrow B,\ H' \vdash G$ by $ap(f;\_\_;\ v.\_\_)$

$\vdash A\ a$

$x\!:\!B \vdash G$ by $g(v)$

v is $f(a)$

And – Intro

$H \vdash A \,\&\, B$

$H \vdash A$

$H \vdash B$

And – Elim

$H,\ p\!:\!A \,\&\, B,\ H' \vdash G$ by $spread(e; x, y.\_\_)$

$H,\ x\!:\!h,\ y\!:\!B,\ H' \vdash G$ by $g(x, y)$

See the attached notes for all of the iFOL rules.

Here is a solution to the task of programming the evidence for $\sim\sim\ (P \vee\ \sim P)$. It was gratifying that students solved this. Here is a solution using proof rules (those that underpin Coq and Nuprl).

Recall that $\sim\sim\ (P \vee\ \sim P)$ is $((P \vee\ \sim P) \Rightarrow\bot) \Rightarrow\bot)$, where for this example $\bot$ is $False$, the $Void$ type.

$\vdash ((P \vee\ \sim P) \Rightarrow\bot) \Rightarrow\bot$ by $\lambda(f.\_\_\_)$

$f : (P \vee\ \sim P) \Rightarrow \bot \vdash \bot$ by $apseq(f; \_\_\_; v.\_\_\_)$

$\vdash (P \vee\ \sim P)$ by $inr(\_\_\_)$

$\vdash \sim P$ by $\lambda(p.\_\_\_)$

$p : P \vdash \bot$ by $apseq(f; \_\_\_; w.\_\_\_)$

$\vdash P \vee\ \sim P$ $inl(\_\_\_)$

$\vdash P$ by p

$w :\bot \vdash \bot$ by w

$v :\bot \vdash \bot$ by v

We compute $apseq(f; inl(p); v)$ as $v = f(inl(p))$, then compute v, $apseq(f; inr(\lambda(p.inl(p))); v)$. So $v = f(inr(\lambda(p.f(inl(p)))))$. The realizer, or program is

$$\lambda(f.f(inr(\lambda(p.f(inl(p)))))).$$

Whenever we have $apseq(f; a; v.v)$ the result is $f(a)$. If we have $apseq(f; a; v.exp(v))$ the result is $exp(f(a))$.

How to explain classical logic?

We can imagine "oracular powers", and use "magic".

Oracle powers, $H \vdash P \vee \sim P$ by $magic(P)$.

Note, the evidence mentions the proposition P, and this is the only rule that mentions a proposition.

By virtual evidence – to be explained.

One of the axioms that suffices to give classical logic from iFOL is $\sim\sim P \Rightarrow P$. This was favored by Kolmogorov. At age 20 he made a compelling case that this is the more primitive insight. Especially in light of this interesting *constructive fact*.

$\vdash \sim\sim (P \vee \sim P)$

Can we just "see the evidence"? Brouwer could. It is just this: $\lambda(f.f(inr(\lambda(p.f(inl(p))))))$.

$\lambda(f^{(P \vee \sim P) \Rightarrow \bot}.f^{(P \vee \sim P) \Rightarrow \bot}(inr(\lambda(p^p.f^{(P \vee \sim P) \Rightarrow \bot}(inl(p))))))$.

Now for the "virtual evidence" idea. Suppose we just ask to know the Nuprl set type $\{Unit \,|\, P \vee \sim P\}$.

This type either has $*$ in it or nothing. To be empty we need to know that $(P \vee \sim P)$ is empty, i.e. $\sim (P \vee \sim P)$.

But we know that being empty is not possible since we know $\sim\sim (P \vee \sim P)$.

Thus, it is consistent to add the rule $ClassicalIntro$, $\sim\sim (P \vee \sim P) \Rightarrow \{P \vee \sim P\}$, where $\{P \vee \sim P\}$ is an abbreviation for $\{Unit \,|\, P \vee \sim P\}$, and in general $\{Q\}$ for any proposition Q is $\{Unit \,|\, Q\}$.

When we invoke this rule, we are giving up on evidence for $P \vee \sim P$. We say that we can use $*$ as a maker for the fact that $\{P \vee \sim P\}$ cannot be empty, and since it can only have $*$ in it, it must have $*$ – a marker for *virtual evidence*.

We get all of classical logic inside the curly bracket as $\{A\}$. We need to allow nested curly braces as in $\{A \Rightarrow \{B\}\}$ or $\{\forall x : D.\{A(x)\}\}$, etc., nesting as deeply as necessary. The depth of nesting might tell us *how classical* the proposition is. Sarah Sernaker is formalizing the classical propositions posed in Kleene's *Introduction to Metamathematics*. Of 99 propositional calculus and first-order logic propostions presented by Kleene, about 20 are classical and thought not to be constructively solvable. However with virtual evidence and rules based on this idea, these classical propositions can be constructively proved in Nuprl.