Computational Logic

Interactive Theorem Proving

Week 13

Cezary Kaliszyk

June 9, 2015

Summary

So far

- Proof Assistants, HOL Light, Coq, Isabelle/HOL, LF $\lambda_{\rightarrow}$, λ_P, λ_2, Curry-Howard, Declarative Proof, Mizar, Proofs about Programs

Today

- Some more $\lambda_?$
  - PTS, Naive TT, CC^{∞}
- Automation
- Exam

Quiz: $\lambda_?$

- The set of sorts: $\{Prop, Type, Type'\}$
- Pseudo-terms: $T ::= Prop | Type | Type' | Var | (\Pi V : T.T) | (\lambda V : T.T) | T\ T$
- Axiom, Axiom, Var, Weak

$$\frac{}{\vdash Prop : Type} \qquad \frac{}{\vdash Type : Type'} \qquad \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash A : B \qquad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}$$

- Π: if $(s_1, s_2) \in \{(Type, Type), (Prop, Prop), (Type, Prop)\}$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A.\, B : s_2}$$

- λ, app, conv

$$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x : A.\, B : s}{\Gamma \vdash \lambda x : A.\, M : \Pi x : A.\, B} \qquad \frac{\Gamma \vdash M : \Pi x : A.\, B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x := N]}$$

$$\frac{\Gamma \vdash A : B \qquad \Gamma \vdash B' : s}{\Gamma \vdash A : B'} \quad \text{where } B =_\beta B'$$

Quiz: $\lambda_?$

- The combination (Type,Type) forms the function types $A \to B$ for A, B:Type.
  This comprises the unary predicate types and binary relations types: $A \to \mathit{Prop}$ and $A \to A \to \mathit{Prop}$.
  - Also: higher order predicate types like $(A \to A \to \mathit{Prop}) \to \mathit{Prop}$.
  - A Π-type formed by (Type,Type) is always an $\to$-type.
- (Prop,Prop) forms the propositional types $\varphi \to \psi$ for $\varphi, \psi : \mathit{Prop}$; implicational formulas.
  - A Π-type formed by (Type,Type) is always an $\to$-type.
- (Type,Prop) forms the dependent propositional type $\Pi x : A.\varphi$ for A:Type, φ:Prop; universally quantified formulas.

Properties of λ_{HOL} (1/2)

Question: is the type theory λHOL really isomorphic with HOL?
Yes: it is possible to disambiguate the syntax.

- Uniqueness of types
  If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$, then $A =_\beta B$.
- Subject Reduction
  If $\Gamma \vdash M : A$ and $M \to_\beta N$, then $\Gamma \vdash N : A$.
- Strong Normalization
  If $\Gamma \vdash M : A$, then all β-reductions from M terminate.

Properties of λ_{HOL} (2/2)

Decidability Questions:

$$\Gamma \vdash M : \sigma ? \, TCP$$

$$\Gamma \vdash M : ? \, TSP$$

$$\Gamma \vdash ? : \sigma \, TIP$$

For λHOL:

- TIP is undecidable
- TCP/TSP
  - Formally: we introduce a judgement of correct context, algorithm close to λP

λ_{HOL} contains $\lambda 2$ and $\lambda \rightarrow$

if $(s_1, s_2) \in \{(\mathit{Type}, \mathit{Type}), (\mathit{Prop}, \mathit{Prop}), (\mathit{Type}, \mathit{Prop})\}$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A.\, B : s_2}$$

This rule allows to form

- $\rightarrow$-types on the Type-level (first copy of $\lambda\rightarrow$)
- $\rightarrow$-types on the Prop-level (second copy of $\lambda\rightarrow$)
- $\Pi\alpha$:Prop.$\alpha\rightarrow\alpha$: polymorphic types on the Prop-level (copy of $\lambda 2$)

Why not extend HOL further?

if $(s_1, s_2) \in \{(\mathit{Type}, \mathit{Type}), (\mathit{Prop}, \mathit{Prop}), (\mathit{Type}, \mathit{Prop})\}$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A.\, B : s_2}$$

We could include:

- Higher order logic over polymorphic domains?
  like ΠA : Type.A→A
- Quantification over all domains? like ΠA : Type.ΠP :A→Prop.Πx:A.P x→P x

This is just a small extension of the Π rule:

- Add $(s_1, s_2) = (\mathit{Type}', \mathit{Type})$ to get
  system λU^-: higher-order logic over polymorphic domains
- Add $(s_1, s_2) = (\mathit{Type}', \mathit{Prop})$ to get
  system λU: HOL with quantification over all domains

Problem

- λU (λHOL + (Type',Type) and (Type',Prop)) is inconsistent (Girard)
- λU^- (λHOL + (Type',Type)) is inconsistent (Coquand, Hurkens)
- (λHOL + (Type',Prop) is consistent)

Consequences:

- In λU^-, there is a closed term M with $\vdash M : \bot$
- This M can not be in normal form (just because of the syntax)
- So, λU^- is not SN

So λU^- can not be used as a logic. However:

- All terms of type Prop, Type or Type' are strongly normalizing
- Type Checking in λU^- is still decidable
  - The algorithm for checking an application only checks type equality

Lambda cube again

if $(s_1, s_2) \in \{(\mathit{Type}, \mathit{Type}), (\mathit{Prop}, \mathit{Prop}), (\mathit{Type}, \mathit{Prop})\}$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A.\, B : s_2}$$

$\lambda\!\rightarrow$	(Prop, Prop)			
$\lambda 2$ (system F)	(Prop, Prop)	(Type, Prop)		
λP (LF)	(Prop, Prop)		(Prop, Type)	
$\lambda\underline{\omega}$	(Prop, Prop)			(Type, Type)
λP2	(Prop, Prop)	(Type, Prop)	(Prop, Type)	
$\lambda\omega$ (system Fω)	(Prop, Prop)	(Type, Prop)		(Type, Type)
λP ω	(Prop, Prop)		(Prop, Type)	(Type, Type)
λPω (CC)	(Prop, Prop)	(Type, Prop)	(Prop, Type)	(Type, Type)

Can we do HOL in CC?

Consider extensionality of propositions:

$$EXT := \forall \alpha, \beta : prop.(\alpha \Leftrightarrow \beta) \Rightarrow (\alpha =_{prop} \beta)$$

In CC this becomes:

$$\Pi \alpha, \beta : Prop.(\alpha \leftrightarrow \beta) \rightarrow (\alpha =_{Prop} \beta)$$

Suppose two base domains A and B and constants a : A, b : B.
In HOL, the following formulas are consistent.

$$\varphi := \forall x : A.x = a \qquad \psi := \forall x : B.\exists y : B.x \neq y$$

In CC, EXT also applies to A and B: A $\leftrightarrow$ B (both are non-empty)
then A $=_{Prop}$ B
also property ψ (of B) also applies to A
thus $\forall$x:A.$\exists$y:A.x $\neq$ y !

Either be careful, or take Set and Prop apart!

PTSs and Extended CC

Pure Type Systems

Determined by a triple (S, A, R) with

- S the set of sorts
- A the set of axioms, $A \subseteq S \times S$
- R the set of rules, $R \subseteq S \times S \times S$

If $s_2 = s_3$ in $(s_1 , s_2 , s_3) \in R$, we write $(s_1 , s_2) \in R$.

Examples?

PTSs and Extended CC

Pure Type Systems

Determined by a triple (S, A, R) with

- S the set of sorts
- A the set of axioms, $A \subseteq S \times S$
- R the set of rules, $R \subseteq S \times S \times S$

If $s_2 = s_3$ in $(s_1 , s_2 , s_3) \in R$, we write $(s_1 , s_2) \in R$.

Examples?

CC^{∞}

S: Prop, $\{Type_i\}$ where $i \in \mathbb{N}$
A: Prop : Type, Type i : Type i+1
R: (Prop, Prop), (Prop, $Type_i$), ($Type_i$, Prop) ($Type_i$, $Type_j$, $Type_{max(i,j)}$)

Recall that ($Type_1$, $Type_0$, $Type_0$) is inconsistent in λU, similarily for $Type_i$.

Intensionality vs Extensionality (1/2)

The equality in the side condition in the conv rule is intensional and decidable.
It can also be extensional.

- Intensional equality of functions
  - equality of algorithms
  - the way the function is presented to us (syntax)
- Extensional equality of functions
  - equality of graphs
  - the (set-theoretic) meaning of the function (semantics)

Extensionality amounts to two rules: ext and conv

$$\frac{\Gamma \vdash M, N : A \to B \qquad \Gamma \vdash p : \Pi x : A.(M\ x = N\ x)}{\Gamma \vdash M = N : A \to B}$$

$$\frac{\Gamma \vdash P : A \qquad \Gamma \vdash A = B : s}{\Gamma \vdash P : B}$$

Intensionality vs Extensionality (2/2)

Adding the rule (ext) renders TCP undecidable:
Suppose H : (A→B)→Prop and x : (H f); then

$$x : (H\ g) \text{ iff there is a } p : \Pi x : A.f\ x = g\ x$$

So, to solve TCP, we need to solve TIP.
The interactive theorem prover Nuprl is based on extensional type theory.

Automation for Interactive Proof

- ML level, Isar level, LTac
  - Repeat, Try, Orelse, Every
  - Matching

```
Ltac user_tauto :=
  repeat match goal with
           | [ H : ?P |- ?P ] => exact H

           | [ |- True ] => constructor
           | [ |- _ /\ _ ] => constructor
           | [ |- _ -> _ ] => intro

           | [ H : False |- _ ] => destruct H
           | [ H : _ /\ _ |- _ ] => destruct H
           | [ H : _ \/ _ |- _ ] => destruct H
           | [ H1 : ?P -> ?Q, H2 : ?P |- _ ] => specialize (H1 H2)
         end.
```

Isabelle automation

```
val apply_tac =
  let
    val intros = [@{thm conjI}, @{thm disjI1}, @{thm disjI2},
      @{thm impI}, @{thm iffI}]
    val elims = [@{thm FalseE}, @{thm conjE}, @{thm disjE},
      @{thm iffE}, @{thm impE2}, @{thm impE3},
      @{thm impE4}, @{thm impE5}, @{thm impE1}]
    in
      atac ORELSE' resolve_tac intros ORELSE' eresolve_tac elims
  end

lemma "((((P ⟶ Q) ⟶ P) ⟶ P) ⟶ Q) ⟶ Q"
by(tactic {* (DEPTH_SOLVE o apply_tac) 1 *})
```

Integration of Tableaux proof search

- Copy polymorphic theorems with monomorphic instances
- Try to remove occurrences of the Hilbert operator
- Eliminate trivial assumptions
- Beta-reduce
- Eliminate remaining abstractions (using combinators)
- Replace `if..then..else` expressions using disjunctions
- For quantification expressions over booleans, consider all cases
- Transform to NNF and Skolemize
- Make all applications first-order
- Add equality axioms (for each instance of equality!)
- CNF
- Run actual first-order proof search

MESON, Itaut, Tauto, Blast, ...

Summary

Today

- More (and less) exotic lambda calculi
- Automation

Next time

- Presentations
- EXAM