Computational Logic

Interactive Theorem Proving

Week 5

Cezary Kaliszyk

April 19, 2013

Summary

So far

Proof Assistants, HOL Light, $\lambda_{\rightarrow}$, Gentzen-style, Tactics

- Properties of $\lambda_{\rightarrow}$
- BHK interpretation and λ-cube again
- Dependent types
- λ_P

Today

- Properties of λ_P
- Curry Howard for λ_P
- HOL Light more advanced tactics

Examples

- $$X : *,\ x : X \vdash x : X$$
- $$X : * \vdash (X \to X) : *$$
- $$A : *,\ P : A \to *,\ a : A \vdash (P\,a) \to * : \Box$$
- $$\alpha : 0 \to \star, \beta : 0 \to \star \quad \vdash \quad \vdash \quad \lambda y : (\forall x : 0.\alpha x \to \beta x).\lambda z : (\forall x : 0.\alpha x).\lambda x : 0.yx(zx) \quad : \quad : \quad (\forall x : 0.\alpha x \to \beta x) \to (\forall x : 0.\alpha x) \to \forall x : 0.\beta x$$
- $$\alpha : 0 \Rightarrow \star \vdash (\Pi y : 0)(\alpha y \Rightarrow \star) : \Box$$

Properties of λ_P

- Possible to extend by an existential quantifier
  - Disjoint union (coproduct) of types
- Strong Normalization (using forgetting map)
- Church-Rosser (corollary)
- Subject Reduction
- Type reconstruction is decidable in PTIME.
  - Type checking is undecidable!
- Type inhabitation in λ_P is undecidable
  - ?

Properties of λ_P

- Possible to extend by an existential quantifier
  - Disjoint union (coproduct) of types
- Strong Normalization (using forgetting map)
- Church-Rosser (corollary)
- Subject Reduction
- Type reconstruction is decidable in PTIME.
  - Type checking is undecidable!
- Type inhabitation in λ_P is undecidable

  First order intuitionistic logic is undecidable

Correspondence with first order logic

FOL corresponds to a fairly weak fragment of λ_P

- Only one type variable 0 (constant, type of individuals)
- All kinds are of the form $0 \Rightarrow \ldots \Rightarrow 0 \Rightarrow \star$
- Function symbols are distinguished object variables $0 \to \ldots \to 0$
- Constants are distinguished variables of type 0
- Other declarations may only be of the form $x : 0$

Proof correspondence

- Proof by generalization
  - abstraction $\lambda x : 0.M^{\varphi}$
- Proof by MP
  - application $M^{\forall x:0\varphi} N^0$

More: Proseminar.

Rules, Conversions, Tactics

- Rule: `thm -> thm`
  - Examples?
- Conversion: `term -> thm`
  - Examples?
- Tactic: goalstate refinement
  - Type?
- Thm_tactical: `thm_tactic -> thm_tactic`
- How to combine them?
  - THEN...
  - DEPTH_CONV
- How to transform one to other?
  - CONV_TAC, CONV_RULE

Rules, Conversions, Tactics

- Rule: `thm -> thm`
  - Examples?
- Conversion: `term -> thm`
  - Examples? `BETA_CONV`
- Tactic: goalstate refinement
  - Type?
- Thm_tactical: `thm_tactic -> thm_tactic`
- How to combine them?
  - THEN...
  - DEPTH_CONV
- How to transform one to other?
  - CONV_TAC, CONV_RULE

Rules, Conversions, Tactics

- Rule: `thm -> thm`
  - Examples?
- Conversion: `term -> thm`
  - Examples? `BETA_CONV`
- Tactic: goalstate refinement
  - Type? `goal -> goalstate`
- Thm_tactical: `thm_tactic -> thm_tactic`
- How to combine them?
  - THEN...
  - DEPTH_CONV
- How to transform one to other?
  - CONV_TAC, CONV_RULE

More advanced tactics

Tautologies

```
ITAUT `(A ==> B) ==> A ==> B`;;
TAUT `(p <=> (q <=> r)) <=> ((p <=> q) <=> r)`;;
```

Rewriting

- Matching
  - Slightly more general than first-order
- REWR_CONV does rewriting on top level

```
ADD_ASSOC;;
⊢ ∀mnp.m+n+p=(m+n)+p
REWR_CONV    ADD_ASSOC    `x+y+z`;;
⊢ x+y+z=(x+y)+z
```

- REWRITE_RULE, REWRITE_CONV, REWRITE_TAC
- Conditional rewriting: SIMP...

Arithmetic

Definition of Natural Numbers

From the axiom of infinity

Rewriting

- `REWRITE_TAC [ARITH]`
- What rules are being used?
- Is it complete?

Other domains

- Real, Complex, Integer, $\mathbb{R}^n$ (vectors)
- Bootstrapped decision procedures

MESON

- Model-Elimination
  - Loveland 1968
- How it works
  - Given helper theorems (possibly polymorphic) assume them with appropriate types
  - Try to remove occurrences of the Hilbert operator
  - Eliminate trivial assumptions
  - Beta-reduce
  - Eliminate remaining abstractions (using λ-lifting)
  - 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
  - Translate to FOL and execute model elimination

MESON export — monomorphisation

- Simple, but effective procedure
  - Find all polymorphic constants in the goal and the first assumption
  - For every occurrence of a constant in the goal and in the assumptions find a type instantiation
  - Apply the instantiation to the assumption and include its new instantiated constants in the goal constants
  - Repeat for all other assumptions
- May produce very big goals for set constants
- Considering all constants repeatedly can be very slow

MESON export — first order

- Given a term like:

  ```
  MAP f [a] = [f a]
  ```

  we have the symbol `f` sometimes applied to zero sometimes one argument
- Can be encoded in FO logic like:

  ```
  MAP f [a] = [I f a]
  ```

  If we assume that identity `I` is always applied to two arguments
- For every constant or free variable we find the minimum number of arguments it is applied to
- An application of a function `F` that needs two arguments to 4 arguments is now encoded as:

  ```
  I (I (F(a1, a2), a3), a4)
  ```

Looking at the code

- Unit type
- Quotient Package
- Pairs
- Natural numbers
- Inductive types
- Arithmetic
- Lists
- Reals
- Integers
- Sets

Summary

Today

- Properties of λ_P
- Curry Howard for λ_P
- More on HOL Light

Next time

- Polymorphism
- Set Theory Introduction