Computational Logic

Interactive Theorem Proving

Week 12

Cezary Kaliszyk

June 2, 2015

Summary

So far

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

Today

- Logical Frameworks
- Isabelle
- Exam

Logical Framework

A program (proof assistant)

- To define a logic
- Signature and provability
  - Usually: Objects, Functions, Rules
    - Type, Term
    - Application, Abstraction
    - The rules
- Provability reduced to problems in the program

History

- Rapid prototyping of deductive systems
- Automath, Edinburgh LF, Isabelle

LF

- Meta-language: λ_P (very close to theory)
  - First-order dependent types + Curry Howard gives
  - objects, types and families of types
  - Church-Rosser, Strongly normalizing but no type inference.
- Representing a logic: Judgements as types
- Recent implementations: Twelf, MMT

HOL in LF (1/2)

```
holtype : type
bool : holtype
fun  : holtype → holtype → holtype                    "1 ⇒ 2"

term  : holtype → type
Abs   : {A,B} (term A → term B) → term (A ⇒ B)          "λ 3"
Comb  : {A,B} term (A ⇒ B) → term A → term B          "3 ' 4"
equal : {A} term A ⇒ (A ⇒ bool)                       "2 = 3"

thm   : term bool → type                                "⊢ 1"
REFL  : {A,X:term A} ⊢ X = X
TRANS : {A,X,Y,Z:term A} ⊢ X = Y → ⊢ Y = Z → ⊢ X = Z
MP    : {p,q} ⊢ p = q → ⊢ p → ⊢ q
BETA  : {A,B,F:term A → term B,X:term A} ⊢ (λ F)'X = (F X)
MK_COMB : {A,B, F,G:term A⇒B, X,Y:term A} ⊢ F = G → ⊢ X = Y → ⊢ F'X = G'Y
ABS   : {A,B, S,T:term A → term B}
  ({x: term A} ⊢ (S x) = (T x)) → ⊢ λ S = λ T
DEDUCT_ANTISYM_RULE : {p,q} (⊢ p → ⊢ q) → (⊢ q → ⊢ p) → ⊢ p = q
```

HOL in LF (2/2)

```
extension definition =
   [n: nat] [A: holtype^n → holtype] [a: {T: holtype^n} term (A T)]
      c   : {T} term (A T)
      DEF : {T} ⊢ (c T) = (a T)

extension new_basic_type_definition =
   [n:nat] [A: holtype^n → holtype]
   [P: {T: holtype^n} term (A T) ⇒ bool]
   [w: {T: holtype^n} term (A T)]
   [nonempty: {T: holtype^n} ⊢ (P T) ' (w T)]
      B       : holtype^n → holtype
      abs     : {T} term (B T) ⇒ (A T)
      rep     : {T} term (A T) ⇒ (B T)
      rep_abs : {T} {b: term (B T)} ⊢ (rep T) ' ((abs T) ' b) = b
      abs_rep : {T} {a: term (A T)} ⊢
                        (P T) ' a = ((abs T) ' ((rep T) ' a) = a)
```

Isabelle

Kernel

- Implemented in SML
  - LCF style
  - Makes use of PolyML checkpointing, JIT, nat, ...
- Medium size
  - ($\approx$ 5-10 files)
  - Higher order unification
  - Is code generation part of it?
- Weak type theory
  - Meta-logic Pure
  - Polymorphic types and type classes
  - Explicit connectives: Implication, Equality, Universal quantifier
  - Implicit: Conjunction, Meta-Existence
- Generic theorem prover
  - Larry Paulson: "Isabelle: The Next 700 Theorem Provers"

Object logics

- IFOL
  - FOL
    - ZF
    - LCF (original Edinburgh LCF logic and prover from 1972)
- CTT
  - First version by Martin-Löf in 1971 impredicative
  - After discovery of Girard's paradox later versions predicative
- HOL (minimally different from HOL Light)
- TLA(+) (language for software/hardware specifications)
- ...

Defining a logic

```
typedecl o

axiomatization
  False :: o and
  conj :: "o => o => o" (infixr "&" 35) and
  disj :: "o => o => o" (infixr "|" 30) and
  imp :: "o => o => o" (infixr "-->" 25)
where
  conjI: "P ==> Q ==> P&Q" and
  conjunct1: "P&Q ==> P" and
  conjunct2: "P&Q ==> Q" and
  disjI1: "P ==> P|Q" and
  disjI2: "Q ==> P|Q" and
  disjE: "P|Q ==> (P ==> R) ==> (Q ==> R) ==> R" and
  impI: "(P ==> Q) ==> P-->Q" and
  mp: "P-->Q ==> P ==> Q" and
```

Soundness and Completeness

Is the Isabelle representation of logic correct?

- Each axiom is sound with respect to the truth-table semantics
- Syntactic rule-by-rule translation
  - For each meta-proof there is a corresponding object proof

Completeness

- Object proofs are translated to meta-proofs
  - By induction on the size of the proof-object

Intuitionistic logic with natural deduction

Theory NJ

- First order logic (Prawitz, 1965)
- Combination of forward and backward reasoning

Rules

```
[| P |] ==> [| Q |] ==> [| P&Q |]
[| P&Q |] ==> [| P |]       [| P&Q |] ==> [| Q |]
[| P |] ==> [| P|Q |]       [| Q |] ==> [| P|Q |]
[| P|Q |] ==> ([| P |] ==> [| R |]) ==>
              ([| Q |] ==> [| R |]) ==> [| R |]
([| P |] ==> [| Q |]) ==> [| P-->Q |]
[| P-->Q |] ==> [| P |] ==> [| Q |]
```

Quantifiers

```
(!(y) [| P(y) |]) ==> [| ALL x.P(x) |]
[| ALL x.P(x) |] ==> [| P(a) |]
[| P(a) |] ==> [| EXISTS x.P(x) |]
[| EXISTS x.P(x) |] ==> (!(y)[| P(y) |] ==> [| R |]) ==> [| R |]
```

Constructive Type Theory

Theory CTT

- Extensional version of Martin-Löf Type Theory
- Normally: Typing judgements ($a(...) \in A(...)$)
- But also: being a family of types over A (B(x) type)

Rules

```
[| A type |] ==> [| B type |] ==> [| A+B type |]
[| a: A |] ==> [| B type |] ==> [| inl(a): A+B |]
[| p: A+B |] ==> (!(x)[| x: A |] ==> [| c(x): C(inl(x)) |]) ==>
                 (!(y)[| y: B |] ==> [| d(y): C(inr(y)) |]) ==>
                 [| when(p,c,d): C(p) |]
[| a: A |] ==> (!(x)[| x: A |] ==> [| c(x): C(inl(x)) |]) ==>
               (!(y)[| y: B |] ==> [| d(y): C(inr(y)) |]) ==>
               [| when(inl(a),c,d) = c(a): C(inl(a)) |]
```

Intuitionistic and Classical FOL

Theories IFOL and FOL

- Sequent calculus with sequent variables

Sequents, Thinning, Cut

```
[| $H, P, $G |- $E, P, $F |]
[| $H |- $E, $F |] ==> [| $H |- $E, P, $F |]
[| $H |- $E, P |] ==> [| $H, P |- $E |] ==> [| $H |- $E |]
```

Conjunction and Negation

```
[| $H |- $E, P, $F |] ==> [| $H |- $E, Q, $F |] ==>
                          [| $H |- $E, P&Q, $F |]
[| $H, P, Q, $G |- $E |] ==> [| $H, P&Q, $G |- $E |]
[| $H, P |- $E, $F |] ==> [| $H |- $E, ~P, $F |]
[| $H, $G |- $E, P |] ==> [| $H, ~P, $G |- $E |]
```

Zermelo-Fraenkel set theory

Theory ZF

- Based on FOL
  - Axioms of ZF are complex
- Extensionality is defined using subsets
- Power-set axiom states that $A \in Pow(B) \iff A \subseteq B$
- Limited comprehention using `Collect`
- Replacement axiom separate `Replace`
- Isabelle formalizes schemes using function variables

```
Collect(A,P)== a:A & P(a)
Replace(f,B) == EXISTS a. a:B & c=f(a)
```

Looking at the theory

Isabelle: Contesting for the biggest formal library

HOL + Other Logics

ZF, CTT, TLA+

Efficiency

Object-Logic, but PolyML+JIT, Checkpointing, Limited proof objects

Code Generation

Number of supported languages, JIT, External compilation

HOL: Big Library + AFP

Math, Protocols, Algorithms, Programs

Isar

Declarative proof, Tactics, Access to ML

Summary

Today

- Logical Frameworks: LF, MMT
- Isabelle: Pure, HOL, ZF, CTT, ...

Next time

- Outlook on special features in Proof Assistants
- Exam