Computational Logic

# Interactive Theorem Proving

Week 11

Cezary Kaliszyk


June 6, 2013

# Summary

## So far

Proof Assistants, HOL Light, $\lambda_{\rightarrow}$, $\lambda_P$, $\lambda_2$, 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: $\lambda\Pi$ (very close to $\lambda P$)
  - 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 implementation: Twelf

```
plus : nat -> nat -> nat -> type.

plus_zero : {M:nat} plus M z M.

plus_succ : {M:nat} {N:nat} {P:nat}
              plus M (s N) (s P)
            <- plus M N P.
```

# 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"

Isabelle formalized provability, whereas LF formalized the proof objects!

# 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 over 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

# Why Isabelle

## 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
- Isabelle: Pure, HOL, ZF, CTT, ...

## Next time

- Exam