MPRI 2-7-2: Proof Assistants

Bruno Barras, Matthieu Sozeau

Dec 15, 2016

Last week recap...

- Use of typed λ-calculus as a logical formalism:
  Curry-Howard isomorphism
  - Propositions as types
  - Proofs as inhabitants
  - Deduction rules as typing rules
- Simple types correspond to propositional logic
- Predicate logic requires dependent products and pairs (Π and Σ types).
- Terms and types share the same syntax.
- Types are terms whose type is one of the special constants called sorts.

Plan

Martin-Löf's Type Theory

System F, Polymorphism

Calculus of Constructions

Pure Type Systems

Metatheory: consistency, strong normalization, canonicity

Martin-Löf's Type Theory

Judgments:

- $\Gamma \vdash A \ \mathrm{type}$ (types have a specific judgment)
- $\Gamma \vdash M : A$
- $\Gamma \vdash A = B$ (equality only on well-typed terms)
- $\Gamma \vdash M = N : A$

Organized as:

- formation rules (rule for Π)
- introduction rules (rule for λ)
- elimination rules (rule for application)
- computation rules (β-reduction)

MLTT: History

Versions:

- 1971: Type:Type (inconsistent impredicativity)
- 1973: Intensional Type Theory (predicative)
- 1979: Extensional Type Theory
  - Types are sets with a specific equality (setoids)
  - Reflection rule: conversion and propositional equality are identified

Features:

- Logical connectives
- Universes
- Equality
- Inductive definitions (or W-types)

System F

System F (J.-Y. Girard (72), Reynolds (74)) extends the simply typed λ-calculus with a new type former (polymorphism):

$$\forall\alpha.\tau$$

Inhabitants of this type are terms that have type τ for all possible substitution of a type for α.

Ex: $(\lambda x.\, x) : \forall\alpha.\, \alpha \to \alpha$

$$\frac{\Gamma \vdash M : \tau \quad \alpha \text{ not free in } \Gamma}{\Gamma \vdash M : \forall\alpha.\, \tau} \qquad \frac{\Gamma \vdash M : \forall\alpha.\, \tau}{\Gamma \vdash M : \tau[\tau'/\alpha]}$$

Explicit version (needed when λ carries the domain type):

$$\frac{\Gamma \vdash M : \tau \quad \alpha \text{ not free in } \Gamma}{\Gamma \vdash \Lambda\alpha.M : \forall\alpha.\, \tau} \qquad \frac{\Gamma \vdash M : \forall\alpha.\, \tau}{\Gamma \vdash M\,\tau' : \tau[\tau'/\alpha]}$$

System F and arithmetic

System F can encode datatypes and a wide range of functions over them, by functional encodings.

Ex: arithmetic

$$\begin{aligned} \mathbb{N} &= \forall\alpha.\alpha \to (\alpha \to \alpha) \to \alpha \\ [n] &= \lambda x.\lambda f.\, f^n\, x \quad (0 = \lambda x.\lambda f.x,\;\; 2 = \lambda x.\lambda f.f\ (f\ x)) \end{aligned}$$

Encodes all functions of second order arithmetic (quantifiers can range over predicates a.k.a. sets of natural numbers)

Limitations as a logical formalism:

- Cannot encode equality via the Curry-Howard isomorphism

System F: an impredicative theory

Polymorphism allows to define a type by quantification over *all* types, including itself.

Allows for self-application!

- id$= \lambda x.x : \forall\alpha.\alpha \to \alpha$
- id id is well-typed (α instantited with $\forall\alpha.\alpha \to \alpha$)

"System F is not set-theoretical" (Reynolds)

Calculus of Constructions: History

Coquand and Huet (85)

Merges ideas from:

- System F (polymorphism)
- Automath (related to Martin Löf's Type Theory)

Calculus of Constructions (CC)

2 sorts: **Prop** and **Type** (literature: **Type**/**Kind** or $*/\Box$)

$$\frac{}{[\,]\vdash} \qquad \frac{\Gamma\vdash T : s}{\Gamma; x : T\vdash} \qquad \frac{\Gamma\vdash \quad (x : T)\in\Gamma}{\Gamma\vdash x : T} \qquad \frac{\Gamma\vdash}{\Gamma\vdash \mathbf{Prop} : \mathbf{Type}}$$

$$\frac{\Gamma\vdash A : s_1 \quad \Gamma; x : A\vdash B : s_2}{\Gamma\vdash \Pi x : A.\, B \ :\ s_2} \qquad \frac{\Gamma\vdash \Pi x : A.\, B \ :\ s \quad \Gamma; x : A\vdash M : B}{\Gamma\vdash \lambda x : T.\, M \ :\ \Pi x : A.\, B}$$

$$\frac{\Gamma\vdash M : \Pi x : A.\, B \quad \Gamma\vdash N : A}{\Gamma\vdash M\, N \ :\ B[N/x]} \qquad \frac{\Gamma\vdash M : T \quad T =_\beta T' \quad \Gamma\vdash T' : s}{\Gamma\vdash M : T'}$$

Conversion rule ($=_\beta$ includes β-reduction/expansion + congruence rules): 2 convertible types have the same inhabitants/proofs. Necessary for good metatheoretical properties.

CC extends System F

Prop (a.k.a. $*$) is a sort of types that includes the types of System F:

- $(*, *, *)$ governs arrow types
  $\mathbb{N} : * \implies \mathbb{N} \to \mathbb{N} : *$
- $(\Box, *, *)$ governs polymorphism
  e.g. $\forall \alpha.\tau \implies \Pi\alpha : *.\tau$

Explicit polymorphism

- Generalization rule : $\Lambda\alpha.t \implies \lambda\alpha : *.t$
- Instantiation rule : $t\ \tau \implies t\ \tau$

CC: a powerful system

CC extends System F (and Fω):

- Functions of higher-order arithmetic
- Propositional connectives ($\wedge$, $\vee$,...)

CC extends $\lambda\Pi$:

- Predicate calculus: existential quantifier, equality

CC is a higher-order logic:

- In MLTT, predicative rule $(\Box, *, \Box)$ prevents quantifications to always be a proposition
  No type of all propositions
- In CC, $*$ is the type of all propositions of higher-order logic

Calculus of Constructions with Universes (CC_ω)

A hierarchy of predicative universes is added (Coquand, 1986).

$$\mathbf{Prop} : \mathbf{Type}_1 : \mathbf{Type}_2 : \mathbf{Type}_3 \ldots$$

Logical strength:

- CC with 2 universes can model Zermelo set theory (Miquel)
  (Uses predicative polymorphic encodings)
- CC_ω can be proved consistent in ZF (Luo).

Limitations of polymorphics encodings

- *Case of impredicative encoding*
  - $0 \neq 1$ is not provable (by erasability of dependencies)
  - induction is not "directly" provable (only the recursor is available)
- *Case of predicative encoding in the calculus with universes*
  - OK for expressivity (we have $0 \neq 1$ and an "indirect" induction)
  - *But* no predecessor in 1 step
  - not "natural", introduces universe issues
  - difficult to write automated tools that can distinguish between inductive types constructors and arbitrary terms
- Primitive inductive types "a la Martin-Löf" have been added.

Pure Type Systems

Pure Type Systems (PTS) are a way to factorize the syntax of many formalisms of type theory. Many metatheoretical results can be established for large classes of PTS.

Definition of Pure Type Systems (PTS)

- Sorts (types of types), organised in axioms $\mathcal{A}$ and rules for product $\mathcal{R}$.

Rules

$$\frac{\Gamma \vdash \quad (s_1, s_2) \in \mathcal{A}}{\Gamma \vdash s_1 : s_2} \qquad \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash} \qquad \frac{\Gamma \vdash \quad (x, A) \in \Gamma}{\Gamma \vdash x : A}$$

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2 \quad (s_1, s_2, s_3) \in \mathcal{R}}{\Gamma \vdash \Pi x : A.B : s_3}$$

$$\frac{\Gamma, x : A \vdash t : B \quad \Gamma \vdash \Pi x : A.B : s}{\Gamma \vdash \lambda x : A.t : \Pi x : A.B} \qquad \frac{\Gamma \vdash t : \Pi x : A.B \quad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B[x \leftarrow u]}$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash B : s \quad A =_\beta B}{\Gamma \vdash t : B} \qquad \lambda x : A.t\, u =_\beta t[x \leftarrow u]$$

(Predicativity: when s_3 is not lower than s_1 or s_2)

PTS instances: Barendregt's cube

$\mathcal{S} = \{*, \Box\}$, $\mathcal{A} = (*, \Box)$ Rules:

- $(*, *, *)$ simple types
- $(*, \Box, \Box)$ dependent types ($\lambda\Pi$)
- $(\Box, *, *)$ polymorphism (system F)
- $(\Box, \Box, \Box)$ higher-order (system Fω)

Metatheory

For a logical formalism, the main metatheoretical property is consistency (the existence of a non-provable proposition).

But other properties are of interest:

- Strong normalization (SN)
- Canonicity / Constructivity

Actually, SN is the strongest property, the others can be seen as corollaries (within arithmetic).

Establishing SN requires preliminary results:

- confluence of $\rightarrow_\beta$
- substitution lemma
- subject-reduction (soundness of typing)

Strong Normalization as the strongest property

Strong Normalization property (SN):

$$\Gamma \vdash M : T \;\Rightarrow\; \neg\exists (M_i)_{i\in\mathbb{N}}. M = M_0 \rightarrow_\beta M_1 \rightarrow_\beta \cdots$$

Gödel's 2nd incompleteness theorem: a formal system as strong as arithmetic cannot prove its own consistency, unless it is inconsistent.

$\Rightarrow$ Consistency and SN cannot be proved in arithmetic. Need a stronger formalism, e.g. set theory.

Strong Normalization proofs

Milestone: Girard's reducibility candidates (CR)

CR are sets of SN λ-terms with well-chosen closure properties.

$$X \to Y = \{t \mid \forall u \in X. t\ u \in Y\}$$

models arrow types and intersection of a family of CR is a CR, so Girard could show SN for System F:

$$\Gamma \vdash M : \tau \ \Rightarrow\ \forall \sigma \in [\![\Gamma]\!]. M[\sigma] \in [\![\tau]\!]$$

Proof simplified by Mitchell and Tait.

Adapts to theories with dependent types (Altenkirch's Λ-sets), but may require a model.

Metatheory: conversion

Conversion:

- Confluence:

$$A \to_\beta^* B \;\wedge A \to_\beta^* C \;\Rightarrow\; \exists D.\, B \to_\beta^* D \wedge C \to_\beta^* D$$

- Corollary: inversion of products

$$\Pi x : A.B =_\beta \Pi x : A'.B' \;\Rightarrow\; A =_\beta A' \;\wedge\; B =_\beta B'$$

Metatheory: typing

Typing:

- Substitution lemma:

$$\frac{\Gamma; x : A; \Delta \vdash M : T \quad \Gamma \vdash N : A}{\Gamma; \Delta[N/x] \vdash M[N/x] : T[N/x]}$$

- Inversion lemmas (one for each term constructor):
  $\Gamma \vdash \lambda x : A.M : C$
  $\Rightarrow \ \exists B\, s.\ C =_\beta \Pi x : A.B \ \wedge \ \Gamma; x : A \vdash M : B \ \wedge \ \Gamma \vdash \Pi x : A.B \ : \ s$
- Subject Reduction:

$$\Gamma \vdash M : T \ \wedge \ M \to_\beta M' \ \Rightarrow \ \Gamma \vdash M' : T$$

Note: if $N \to_\beta N'$, refl $N : N = N$ but refl $N' : N = N$ requires the conversion rule

Canonicity

Characterization of inhabitants (in normal form) of type constructors

Using inversion lemmas, if M in normal form (atomic terms: $x\ t_1 \cdots t_n$):

- $\Gamma \vdash M : \Pi x : A.B$ implies M is either a λ or an atomic term.
- $\Gamma \vdash M : s$ implies M is either a sort, a Π or an atomic term.

Note: when $\Gamma = [\,]$, the atomic case does not apply

If the formalism encodes arithmetic, we expect:

- $\Gamma \vdash M : \mathbb{N}$ implies M is either 0 or a successor, or an atomic term.

Canonicty + SN: Constructivity

Using SN:

- $\vdash M : \Pi x : A.B$ then M reduces to a λ.
- $\vdash M : s$ then M reduces to a sort or a Π.
- $\vdash M : \mathbb{N}$ then M reduces to a numeral.

Constructivity and Consistency

Constructivitiy: canonicity applied to connectives (cut elimination)

- $\vdash M : A \vee B$ implies M reduces to an introduction rule, thus we get either a proof of A or a proof of B.
- $\vdash M : \exists x : A.B$ implies M reduces to a pair (a, b) where a is a witness.
- $\vdash M : \perp$ is impossible: consistency.

Note: non-normalization of a type theory often (not always!) lead to inconsistency.

Towards the formalism of Coq

Recap on CC:

- Encodes correctly higher-order logic.
- Encodes (using polymorphism) datatypes and functions on them.
- Does not encode correctly the equational theory of those datatypes

Calculus of Inductive Constructions (CIC)

- Extends CC with universes and primitive (co-)inductive types (a la Martin-Löf, but impredicativity allowed)
- Enjoys the expected canonicity results

Coquand, Paulin-Mohring (90).

CIC: sort setup

Universes:

- An impredicative sort **Prop**:
- A hierarchy of predicative sorts $\mathbf{Type}_i$

$$\mathbf{Prop} : \mathbf{Type}_1 : \mathbf{Type}_2 : \mathbf{Type}_3 \ldots$$

$$\mathbf{Prop} \subset \mathbf{Type}_1 \subset \mathbf{Type}_2 \subset \mathbf{Type}_3 \ldots$$

Proof-irrelevance ($\forall P : \mathbf{Prop}. \forall pq : P.\, p = q$):

- Admissible.
- Not provable: axioms discriminating proofs are consistent (but the interpretation of functions have to be restricted to computable ones)

Classical logic

- **Prop** can be interpreted as a boolean type (implies proof-irrelevance)

Exercises

TP 2 on my webpage
`http://www.lix.polytechnique.fr/~barras/mpri/`

Or `http://www.lix.polytechnique.fr/~barras/mpri/2016/tp2.pdf`