Intensionality, Extensionality, and Proof Irrelevance in Modal Type Theory

Frank Pfenning

LICS'01
Boston, Massachusetts
June 17, 2001

Acknowledgments: Alberto Momigliano, ...

Outline

1. Introduction
2. Judgmental Analysis
3. Programs and Extensionality
4. Proofs and Irrelevance
5. Expressions and Intensionality
6. Conclusion

Objective and Approach

- Study fundamental notions in logic and computer science
  - Formal expressions, intensionality
  - Programs, types, computations, extensionality
  - Proofs, propositions, truth, proof-irrelevance
- How are they related?
- How can they co-exist?
- Analysis via judgments in the style of Martin-Löf

Motivation

- Formal expressions, intensionality:
  - reflection, meta-programming
  - run-time code generation
- Programs, types, computations, extensionality:
  - (functional) programming
  - logical frameworks
- Proofs, propositions, truth, proof-irrelevance:
  - (constructive) logic, reasoning about programs
  - computational contents, dead code elimination

Preview of Results

- Judgmental analysis of expressions, programs, proofs
- Definitional equality is intensional, extensional, irrelevant
- Smooth integration in a single modal type theory
- Presently restricted to dependent functions only
- Type theory is decidable
- Canonical forms exist
- Sufficient for logical framework applications
- More work needed for functional programming

Judgments

- Judgment — object of knowledge
- Evident judgment — something we know
- Derivation — evidence for a judgment
- Basic judgments, for example
  - P is a proposition
  - P is true
  - D is a proof of P
  - A is a type
  - M is a term of type A
  - M and N are equal terms of type A
- Following Martin-Löf ['83,'94,'96]

Judgmental Analysis

- Minimal conceptual machinery
  - Basic judgments
  - Parametric and hypothetical judgments
- Extends to richer type theories
  - Categorical judgments (modal logic)
  - Linear hypothetical judgments (linear logic)
  - Ordered hypothetical judgments
    (Lambek calculus and ordered logic)
- Orthogonality and open-endedness
- Constructive

Hypothetical Judgments

- General form of hypothetical judgment:

$$J_1, \ldots, J_n \vdash J$$

  $\Gamma = J_1, \ldots, J_n$ are *hypotheses*

- General form of hypothesis rule: $J_1, \ldots, J_n \vdash J_i$

- Substitution property:

  If $\Gamma \vdash J$
  and $\Gamma, J \vdash J'$
  then $\Gamma \vdash J'$

  Substitute the derivation of J for uses of the hypothesis J.

- Also satisfies weakening and contraction.

Outline

- Introduction
- Judgmental Analysis

$\Rightarrow$ **Terms and Extensionality**

- Proofs and Irrelevance
- Expressions and Intensionality
- Conclusion

Terms and Types

- Basic judgments:
  - A : *type* — A is a type
  - $A = B$: *type* — types A and B are equal
  - $M : A$ — object M has type A
  - $M = N : A$ — object M equals N at type A
  - $M \longrightarrow M'$ — object M reduces to M'
- Hypotheses $\Gamma = x_1{:}A_1, \ldots, x_n{:}A_n$
- All judgments except reduction are hypothetical in Γ
- Presupposition: all x_i distinct and $x_1{:}A_1, \ldots, x_i{:}A_i \vdash A_{i+1}$: *type*

Role of Definitional Equality

- Related to operational semantics:
  - If $M \longrightarrow M'$ and $M : A$ then $M = M' : A$
  - If $M = M' : A$ then M and M' are observably equivalent
  - $A = B$: *type* if embedded terms are equal
- Necessary for type conversion:

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash A = B : \mathit{type}}{\Gamma \vdash M : B}$$

- Functional programming: congruence based on β-reduction
- Logical framework: congruence based on $\beta\eta$-conversion
- Should be decidable for decidable type-checking

Functions

- Reflect hypothetical judgment as type $\Pi x{:}A.\, B(x)$
- Introduction rule

$$\frac{\Gamma \vdash A : \mathit{type} \qquad \Gamma, x{:}A \vdash M(x) : B(x)}{\Gamma \vdash \lambda x{:}A.\, M(x) : \Pi x{:}A.\, B(x)}$$

- Elimination rule

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

- Reduction $(\lambda x{:}A.\, M(x))\, N \longrightarrow M(N)$

Extensional Definitional Equality

- $\Gamma \vdash M = N : A$
- Reflexive, transitive, congruent
- Computational equality (β)

$$\frac{\Gamma \vdash A_1 : \mathit{type} \quad \Gamma, x{:}A_1 \vdash M_2(x) : A_2(x) \quad \Gamma \vdash M_1 : A_1}{\Gamma \vdash (\lambda x{:}A_1.\, M_2(x))\, M_1 = M_2(M_1) : A_2(M_1)}$$

- Extensionality (equivalent to η)

$$\frac{\Gamma \vdash A : \mathit{type} \quad \Gamma, x{:}A \vdash M\, x = N\, x : B(x)}{\Gamma \vdash M = N : \Pi x{:}A.\, B(x)}$$

- Equality is decidable

Outline

- Introduction
- Judgmental Analysis
- Terms and Extensionality

$\Rightarrow$ **Proofs and Irrelevance**

- Expressions and Intensionality
- Conclusion

A Puzzle: Subset Types

- Illustrates computational irrelevance of proofs
- Introduction rule

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash D : B(M)}{\Gamma \vdash M : \{x{:}A \mid B(x)\}}$$

- Second elimination rule

$$\frac{\Gamma \vdash M : \{x{:}A \mid B(x)\} \qquad \Gamma, u{:}B(M) \vdash N : C}{\Gamma \vdash N : C}$$

  provided u **not free in** N

- u can still be used for *proofs* in second premise
- Type-checking undecidable

Example: Using a Proof Judgment

- $D \div P$ — D is proof of proposition P
- Introduction rule

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash D \div B(M)}{\Gamma \vdash \langle M, D \rangle : \{x{:}A \mid B(x)\}}$$

- Second elimination rule

$$\frac{\Gamma \vdash M : \{x{:}A \mid B(x)\} \qquad \Gamma, u \div B(\pi_1\, M) \vdash N : C}{\Gamma \vdash \mathsf{let}\ u = \pi_2\, M\ \mathsf{in}\ N : C}$$

- No side condition
- Proofs are dead code (erase before computation)
- Type-checking decidable

Proofs and Propositions

- $\Gamma \vdash A \div \mathit{type}$ — type A is a proposition
- $\Gamma \vdash M \div A$ — object M is a proof of proposition A
- New judgment on same language of objects and types!
- Defined by only one rule (terms are proofs)

$$\frac{\Gamma^{\oplus} \vdash M : A}{\Gamma \vdash M \div A}$$

- Proofs are not terms ($M \div A \not\supset M : A$)
- $\Gamma^{\oplus}$ allows proof variables as term variables:

$$\begin{aligned} (\cdot)^{\oplus} &= \cdot \\ (\Gamma, x{:}A)^{\oplus} &= \Gamma^{\oplus}, x{:}A \\ (\Gamma, x{\div}A)^{\oplus} &= \Gamma^{\oplus}, x{:}A \end{aligned}$$

Proof Irrelevance

- $\Gamma \vdash M = N \div A$ — M and N are equal proofs
- Do not care about the proofs, only their existence
- Defined by only one rule (all valid proofs are equal)

$$\frac{\Gamma^{\oplus} \vdash M : A \qquad \Gamma^{\oplus} \vdash N : A}{\Gamma \vdash M = N \div A}$$

- Proofs are not observable
- Erase dead code before computation

Irrelevant Function Types

- $\Pi x \div A.\, B(x)$ — a function that ignores its argument
- May use x in correctness proof
- Introduction rule (boring)

$$\frac{\Gamma \vdash A \div \mathit{type} \qquad \Gamma, x \div A \vdash M(x) : B(x)}{\Gamma \vdash \lambda x \div A.\, M(x) : \Pi x \div A.\, B(x)}$$

- Elimination rule (boring)

$$\frac{\Gamma \vdash M : \Pi x \div A.\, B(x) \qquad \Gamma \vdash N \div A}{\Gamma \vdash M \circ N : B(N)}$$

- Reduction $(\lambda x \div A.\, M(x)) \circ N \longrightarrow M(N)$

Logical Framework Application

- Proofs of decidable propositions can be erased
- Decidability can be established syntactically
- Replace proofs of undecidable propositions by oracle strings [Necula & Rahul'01]
  - Enough information to reconstruct a proof
  - All proofs are equal, so any proof will do!
  - Consistent integration of oracle strings in LF
  - Important for compact certificates in proof-carrying code

Proof Irrelevance as a Modality

- Internalize proof irrelevance as a modal operator
- Introduction rule

$$\frac{\Gamma \vdash M \mathbin{\div} A}{\Gamma \vdash \mathsf{tri}\ M : \triangle A}$$

- Elimination rule

$$\frac{\Gamma \vdash M : \triangle A \qquad \Gamma, x \mathbin{\div} A \vdash N(x) : C}{\Gamma \vdash \mathsf{let\ tri}\ x = M\ \mathsf{in}\ N(x) : C}$$

- Proof theory not yet fully investigated
- Commuting conversions and dependent types?
- Categorical analysis [Awodey & Bauer'01]

Modal Logic

- Axiomatic characterization of $\triangle$ (non-dependent fragment)

$$\vdash A \supset \triangle A$$

$$\vdash \triangle\triangle A \supset \triangle A$$

$$\vdash \triangle(A \supset B) \supset \triangle A \supset \triangle B$$

- $\forall x{:}A.\, B(x)$ quantifies over ephemeral objects
  Need exist only in "present" worlds

- $\forall x \div A.\, B(x)$ quantifies over "real" objects
  Existed in some "past" or "present" world

Outline

- Introduction
- Judgmental Analysis
- Terms and Extensionality
- Proofs and Irrelevance

⇒ **Expressions and Intensionality**

- Conclusion

Intensional Expressions

- $\Gamma \vdash M :: A$ — object M is an expression of type A
- All expressions are terms (via evaluation)

$$\frac{}{\Gamma, x{::}A, \Gamma' \vdash x : A}$$

- Some terms are expressions

$$\frac{\Gamma^{\ominus} \vdash M : A}{\Gamma \vdash M :: A}$$

- $\Gamma^{\ominus}$ prohibits term variables and proof variables in expressions *except inside proofs*

$$\begin{aligned}
(\cdot)^{\ominus} &= \cdot \\
(\Gamma, x{::}A)^{\ominus} &= \Gamma^{\ominus}, x{::}A \\
(\Gamma, x{:}A)^{\ominus} &= \Gamma^{\ominus}, x \div A \\
(\Gamma, x \div A)^{\ominus} &= \Gamma^{\ominus}, x \div A
\end{aligned}$$

Intensionality (modulo Proofs)

- $\Gamma \vdash M = N :: A$ — objects M and N are intensionally equal
- Defined as α-conversion

$$\frac{\Gamma^{\ominus} \vdash M : A}{\Gamma \vdash M = M :: A}$$

- However, embedded proofs are still identified

$$M(D) = (\lambda x \div A.\, M(x)) \circ D = (\lambda x \div A.\, M(x)) \circ E = M(E)$$

- Caveat: type labels (see paper)

Internalizing Expressions

- Intensional function type $\Pi x{::}A.\, B$
- Rules completely analogous to terms and proofs
- Internalizing expressions as a modal operator $\Box A$
- Introduction rule

$$\frac{\Gamma \vdash M :: A}{\Gamma \vdash \mathsf{box}\, M : \Box A}$$

- Elimination rule

$$\frac{\Gamma \vdash M : \Box A \qquad \Gamma, x{::}A \vdash N(x) : C}{\Gamma \vdash \mathsf{let\ box}\ x = M\ \mathsf{in}\ N(x) : C}$$

Modal Logic Revisited

- Axiomatic characterization of $\Box$ (non-dependent fragment)

$$\vdash \Box A \supset A$$

$$\vdash \Box A \supset \Box\Box A$$

$$\vdash \Box(A \supset B) \supset \Box A \supset \Box B$$

$$\frac{\vdash A}{\vdash \Box A}$$

- Interaction with $\triangle$

$$\vdash A \supset \Box\triangle A$$

- $\forall x{::}A.\, B(x)$ quantifies over persistent objects
  Must exist in all "future" worlds

Application: Run-Time Code Generation

- To generate (optimized) code at run-time, we need (at least conceptually) the source expression.
- Property guaranteed by $\Box A$
- Dependent type theory to reason about run-time code generating programs.
- Requires both $\Box A$ and $\triangle A$

Some Theorems

- LF is based on type theory with $\Pi x{:}A.\, B(x)$
- LF extended with $\Pi x \div A.\, B(x)$, and $\Pi x{::}A.\, B(x)$ satisfies:
  - decidability of definitional equality
  - decidability of type-checking
  - existence of canonical forms
  - conservativity over LF
- Approximately typed algorithm for equality [Harper & Pf.'00]
- See paper and technical report for details

Related Work

- Non-dependent modal type theory [Davies & Pf'96,'01]
- Program extraction [Constable'86]
  $M : \#A$ (smash type)
- Program extraction [Paulin-Mohring'89]
  P : *prop*, A : *type*
- Program extraction [Berger et al.'01] (many others)
- Extensional concepts in non-extensional type theory [Hofmann'95]

Conclusions

- Intensionality (α-conversion)
- Extensionality ($\beta\eta$-conversion)
- Proof irrelevance (all proofs equal)
- Co-exist easily in judgmental framework
- Applications of proof irrelevance:
  proof compression (PCC), dead code elimination
- Applications of intensionality:
  run-time code generation, reflection(?)
- Basic study of fundamental notions
- The framework makes a difference!