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:
  \[ J_1, \ldots, J_n \vdash J \]
  \[ \Gamma = J_1, \ldots, J_n \text{ are hypotheses} \]

- 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

⇒ Terms and Extensionality

• Proofs and Irrelevance

• Expressions and Intensionality

• Conclusion
Terms and Types

• Basic judgments:
  
  – \( A : \text{type} \) — \( A \) is a type
  
  – \( A = B : \text{type} \) — types \( A \) and \( B \) are equal
  
  – \( M : A \) — object \( M \) has type \( A \)
  
  – \( M = N : A \) — object \( M \) equals \( N \) at type \( A \)
  
  – \( M \rightarrow M' \) — object \( M \) reduces to \( M' \)

• Hypotheses \( \Gamma = x_1:A_1, \ldots, x_n:A_n \)

• All judgments except reduction are hypothetical in \( \Gamma \)

• Presupposition: all \( x_i \) distinct and
  
  \( x_1:A_1, \ldots, x_i:A_i \vdash A_{i+1} : \text{type} \)
Role of Definitional Equality

- Related to operational semantics:
  - If $M \rightarrow M'$ 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:

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

- Functional programming: congruence based on $\beta$-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
  
  $\Gamma \vdash A : type \quad \Gamma, x: A \vdash M(x) : B(x)$

  $\Gamma \vdash \lambda x: A. M(x) : \Pi x: A. B(x)$

- Elimination rule
  
  $\Gamma \vdash M : \Pi x: A. B(x) \quad \Gamma \vdash N : A$

  $\Gamma \vdash MN : B(N)$

- Reduction $(\lambda x: A. M(x)) N \rightarrow M(N)$
Extensional Definitional Equality

- $\Gamma \vdash M = N : A$

- Reflexive, transitive, congruent

- Computational equality ($\beta$)
  \[
  \Gamma \vdash A_1 : type \quad \Gamma, x:A_1 \vdash M_2(x) : A_2(x) \quad \Gamma \vdash M_1 : A_1 \\
  \Gamma \vdash (\lambda x:A_1. M_2(x)) M_1 = M_2(M_1) : A_2(M_1)
  \]

- Extensionality (equivalent to $\eta$)
  \[
  \Gamma \vdash A : type \quad \Gamma, x:A \vdash M x = N x : B(x) \\
  \Gamma \vdash M = N : \prod x:A. B(x)
  \]

- Equality is decidable
Outline

- Introduction
- Judgmental Analysis
- Terms and Extensionality

⇒ Proofs and Irrelevance
  - Expressions and Intensionality
  - Conclusion
A Puzzle: Subset Types

- Illustrates computational irrelevance of proofs

- Introduction rule

\[
\begin{align*}
\Gamma \vdash M : A & \quad \Gamma \vdash D : B(M) \\
\hline
\Gamma \vdash M : \{x:A \mid B(x)\}
\end{align*}
\]

- Second elimination rule

\[
\begin{align*}
\Gamma \vdash M : \{x:A \mid B(x)\} & \quad \Gamma, u : B(M) \vdash N : C \\
\hline
\Gamma \vdash N : C
\end{align*}
\]

provided \(u\) not free in \(N\)

- \(u\) can still be used for proofs in second premise

- Type-checking undecidable
Example: Using a Proof Judgment

- $D \vdash P$ — $D$ is proof of proposition $P$

- Introduction rule

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

- Second elimination rule

$$
\Gamma \vdash M : \{x : A \mid B(x)\} \quad \Gamma, u \vdash B(\pi_1 M) \vdash N : C \\
\Gamma \vdash \text{let } u = \pi_2 M \text{ in } N : C
$$

- No side condition

- Proofs are dead code (erase before computation)

- Type-checking decidable
Proofs and Propositions

- $\Gamma \vdash A \div type$ — 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 ($M \div A \not\vdash M : A$)
- $\Gamma \oplus$ allows proof variables as term variables:

$$\cdot \oplus = \cdot$$

$$\frac{}{\Gamma, x : A \oplus = \Gamma \oplus, x : A}$$

$$\frac{}{(\Gamma, x \div A) \oplus = \Gamma \oplus, x : A}$$
Proof Irrelevance

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

\[
\begin{align*}
\Gamma^\oplus \vdash M : A & \quad \Gamma^\oplus \vdash N : A \\
\Gamma \vdash M = N \vdash A
\end{align*}
\]

- Proofs are not observable
- Erase dead code before computation
Irrelevant Function Types

- $\prod x : A. B(x)$ — a function that ignores its argument
- May use $x$ in correctness proof
- Introduction rule (boring)
  \[
  \frac{
    \Gamma \vdash A \div type \quad \Gamma, x : A \vdash M(x) : B(x)
  }{
    \Gamma \vdash \lambda x : A. M(x) : \prod x : A. B(x)
  }
  \]
- Elimination rule (boring)
  \[
  \frac{
    \Gamma \vdash M : \prod x : A. B(x) \quad \Gamma \vdash N \div A
  }{
    \Gamma \vdash M \circ N : B(N)
  }
  \]
- Reduction ($\lambda x : A. M(x)) \circ N \rightarrow M(N)$
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

\[ \Gamma \vdash M : A \]
\[ \Gamma \vdash \text{tri } M : \Delta A \]

• Elimination rule

\[ \Gamma \vdash M : \Delta A \quad \Gamma, x : A \vdash N(x) : C \]
\[ \Gamma \vdash \text{let tri } x = M \text{ in } N(x) : C \]

• Proof theory not yet fully investigated

• Commuting conversions and dependent types?

• Categorical analysis [Awodey & Bauer’01]
Modal Logic

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

\[ \vdash A \supset \Box A \]
\[ \vdash \Box \Box A \supset \Box A \]
\[ \vdash \Box (A \supset B) \supset \Box A \supset \Box B \]

• $\forall x: A. B(x)$ quantifies over ephemeral objects
  Need exist only in “present” worlds

• $\forall x: \Diamond A. B(x)$ 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)
  $$\Gamma, x::A, \Gamma' \vdash x : A$$

- Some terms are expressions
  $$\Gamma^\Theta \vdash M : A$$
  $$\Gamma \vdash M :: A$$

- $\Gamma^\Theta$ prohibits term variables and proof variables in expressions except inside proofs

\[
\begin{align*}
(\cdot)^\Theta &= \cdot \\
(\Gamma, x::A)^\Theta &= \Gamma^\Theta, x::A \\
(\Gamma, x:A)^\Theta &= \Gamma^\Theta, x\vdash A \\
(\Gamma, x\vdash A)^\Theta &= \Gamma^\Theta, x\vdash A
\end{align*}
\]
Intensionality (modulo Proofs)

• $\Gamma \vdash M = N :: A$ — objects $M$ and $N$ are intensionally equal

• Defined as $\alpha$-conversion

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

• However, embedded proofs are still identified

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

• 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

\[
\frac{\Gamma \vdash M :: A}{\Gamma \vdash \text{box } M : \Box A}
\]

- Elimination rule

\[
\frac{\Gamma \vdash M : \Box A \quad \Gamma, x::A \vdash N(x) : C}{\Gamma \vdash \text{let box } x = M \text{ in } N(x) : C}
\]
Modal Logic Revisited

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

\[\vdash \Box A \supset A\]
\[\vdash \Box A \supset \Box \Box A\]
\[\vdash \Box (A \supset B) \supset \Box A \supset \Box B\]

\[\vdash A \quad \vdash \Box A\]

- 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 □A
- Dependent type theory to reason about run-time code generating programs.
- Requires both □A and △A
Some Theorems

- LF is based on type theory with $\Pi x:A. B(x)$
- LF extended with $\Pi x:\vdash A. B(x)$, 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]
  \( P : \text{prop}, \ A : \text{type} \)

• Program extraction [Berger et al.’01] (many others)

• Extensional concepts in non-extensional type theory [Hofmann’95]
Conclusions

- Intensionality ($\alpha$-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!