Logical and Meta-Logical Frameworks

Frank Pfenning

PPDP'99, Paris, France
September 30, 1999

1. Introduction
2. Logical Frameworks
3. Meta-Logical Frameworks
4. Conclusion

(see also: upcoming article in *Handbook of Automated Reasoning*)

Some Terminology

- **Deductive System:** Calculus of axioms and inference rules defining derivable judgments. Used in the presentation of logics and programming languages.

- **Logical Framework:** Meta-language for the formalization of deductive systems.

- **Meta-Logical Framework:** Meta-language for reasoning about deductive systems.

Logical Frameworks

- Factor the effort required to implement various logics.
- Support common concepts in deductive systems.
- Provide generic tools for proof search.
- Characterized by

  1. underlying formalism,
  2. meta-programming language,
  3. theorem proving environment.

Some Logical Frameworks

- Formalisms: hereditary Harrop formulas (HHF), dependently typed λ-calculus (LF), inductive definitions, rewriting logic.
- Meta-Programming Languages: functional (ML, rewriting), relational (λProlog, Elf).
- Theorem Proving Environments: tactics and tacticals, logic programming, proof checking.
- Implementations: Isabelle, λProlog, Elf, linear LF, Maude, Elan, Pi, Agda, ...
- More Information: `http://www.cs.cmu.edu/~fp/lfs.html`
- General Mathematical Reasoning Systems: HOL, Coq, LEGO, Nuprl, PVS, Nqthm, ...

Meta-Logical Frameworks

- Require means for representing logics. (A logical framework!)
- Support common proof techniques in the study of logic and programming languages:
  1. structural induction (over derivations),
  2. inversion,
  3. interpretations,
  4. logical relations.
- Characterized by
  1. underlying logical framework,
  2. formalism for meta-reasoning,
  3. automation techniques.

Some Meta-Logical Frameworks

- General-purpose reasoning systems used as meta-logical frameworks (Nqthm, HOL, Coq, LEGO, Nuprl, Isabelle/HOL, ...):

  1. logics defined inductively,
  2. meta-reasoning by (structural) induction,
  3. automation by tactics (or inductive theorem proving).

- Maude [Basin, Clavel & Meseguer'99]:

  - based on rewriting logic,
  - meta-reasoning by reflection and induction,
  - automation by rewriting.

More Meta-Logical Frameworks

- FOLDN[McDowell & Miller'97] [McDowell'97]:

  1. based on hereditary Harrop formulas (typically),
  2. meta-reasoning by definitional reflection and induction over N,
  3. interactive.

- Twelf [Schürmann & Pf.'98,'99]

  1. based on LF logical framework,
  2. meta-reasoning via total functional programs (termination, coverage),
  3. automation by inductive theorem proving.

This Talk

1. Inductive vs. non-inductive representations of logics.
2. Reasoning about non-inductive definitions.
3. Techniques for automation.
4. Further development (speculative).

The Example: Axiomatic Formulation of Intuitionistic Logic

- Abstract syntax

$$
\begin{array}{rrcl}
\text{Terms} & t & ::= & a \mid x \mid f(t_1, \ldots, t_n) \\
\text{Propositions} & A & ::= & p(t_1, \ldots, t_n) \mid A_1 \supset A_2 \mid \forall x.\, A \\
\text{Assumptions} & \Delta & ::= & \cdot \mid \Delta, A
\end{array}
$$

- Judgments

$$
\begin{array}{lr}
\text{Truth} & \vdash A \\
\text{Entailment} & \Delta \vdash A
\end{array}
$$

- Meta-theorem

If $\Delta, A \vdash C$ then $\Delta \vdash A \supset C$.

Axioms and Inference Rules

- Axioms

$$\vdash A \supset (B \supset A) \qquad (K)$$

$$\vdash (A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C)) \qquad (S)$$

$$\vdash (\forall x.\, A) \supset [t/x]A \qquad (F_1)$$

$$\vdash (\forall x.\, (B \supset A)) \supset (B \supset \forall x.\, A) \qquad (F_2)^*$$

$(F_2)^*$: x not free in B

- Rule of inference

$$\frac{\vdash A \supset B \qquad \vdash A}{\vdash B}\, MP \qquad\qquad \frac{\vdash [a/x]A}{\vdash \forall x.\, A}\, UG^a$$

Higher-Order Abstract Syntax

- Use (simply typed) λ-calculus to represent language.
- Key idea:

  Represent variables in the object language by variables in the meta-language.

- Technique employed in HHF, LF.
- In general not inductive.

Representation Function

$$
\begin{array}{rcllcl}
\ulcorner x \urcorner & = & x & x{:}\mathsf{i} & & \\
\ulcorner f(t_1, \ldots, t_k) \urcorner & = & \mathsf{f}\ \ulcorner t_1 \urcorner \ldots \ulcorner t_k \urcorner & \mathsf{f} & : & \mathsf{i} \rightarrow \cdots \rightarrow \mathsf{i} \\
\\
\ulcorner p(t_1, \ldots, t_k) \urcorner & = & \mathsf{p}\ \ulcorner t_1 \urcorner \ldots \ulcorner t_k \urcorner & \mathsf{p} & : & \mathsf{i} \rightarrow \cdots \rightarrow \mathsf{o} \\
\ulcorner A_1 \supset A_2 \urcorner & = & \mathsf{imp}\ \ulcorner A_1 \urcorner\, \ulcorner A_2 \urcorner & \mathsf{imp} & : & \mathsf{o} \rightarrow \mathsf{o} \rightarrow \mathsf{o} \\
\ulcorner \neg A \urcorner & = & \mathsf{not}\ \ulcorner A \urcorner & \mathsf{not} & : & \mathsf{o} \rightarrow \mathsf{o} \\
\ulcorner \forall x.\, A \urcorner & = & \mathsf{forall}\ (\lambda x{:}\mathsf{i}.\, \ulcorner A \urcorner) & \mathsf{forall} & : & (\mathsf{i} \rightarrow \mathsf{o}) \rightarrow \mathsf{o}
\end{array}
$$

- Constant declarations form *signature* in the framework.
- Variable declarations form *context* in the framework.

The Framework, Simply Typed Fragment

- Simply typed λ-calculus.

$$\begin{array}{llll}
\text{Types} & A & ::= & a \mid A_1 \to A_2 \\
\text{Objects} & M & ::= & c \mid x \mid \lambda x{:}A.\, M \mid M_1\, M_2 \\
\text{Signatures} & \Sigma & ::= & \cdot \mid \Sigma, a{:}\mathsf{type} \mid \Sigma, c{:}A \\
\text{Contexts} & \Gamma & ::= & \cdot \mid \Gamma, x{:}A
\end{array}$$

- Shared by HHF and LF.
- Typing $\Gamma \vdash^{\rightarrow}_{\Sigma} M : A$.
- Canonical (β-normal, η-long) forms $\Gamma \vdash^{\rightarrow}_{\Sigma} M \Uparrow A$.
- Suppress signature Σ.

Adequacy Theorem

- **Theorem** [Adequacy] For $\Gamma_x = x_1{:}\mathsf{i}, \ldots, x_n{:}\mathsf{i}$ and terms t and propositions A with free variables among $x_1, \ldots, x_n$,

  1. $\Gamma_x \vdash^{\rightarrow} M \Uparrow \mathsf{i}$ iff $M = \ulcorner t \urcorner$,

  2. $\Gamma_x \vdash^{\rightarrow} M \Uparrow \mathsf{o}$ iff $M = \ulcorner A \urcorner$.

  3. The representation function $\ulcorner \cdot \urcorner$ is a *compositional bijection*:

$$[\ulcorner t \urcorner / x] \ulcorner s \urcorner = \ulcorner [t/x] s \urcorner \quad \text{and} \quad [\ulcorner t \urcorner / x] \ulcorner A \urcorner = \ulcorner [t/x] A \urcorner.$$

Weak vs. Strong Frameworks

- Recall: $\ulcorner\forall x.\, A\urcorner = \mathsf{forall}\,(\lambda x{:}\mathsf{i}.\, \ulcorner A\urcorner)$ where $\mathsf{forall} : \underbrace{(\mathsf{i} \to \mathsf{o})}_{!!} \to \mathsf{o}$.

- Requires function space to be *parametric*!

- As soon as function space is too rich, either

  - *if* $\Gamma \vdash^{\sqcap} M : \mathsf{o}$ *then* $\Gamma \vdash^{\sqcap} M' \Uparrow \mathsf{o}$ *for* M' *definitionally to* M, or
  - *if* $\Gamma \vdash^{\sqcap} M \Uparrow \mathsf{o}$ *then* $M = \ulcorner A\urcorner$ *for some* A

  will fail. Invalidates or complicates reasoning and meta-reasoning.

- Prohibits case analysis and recursion in LF objects.

- Also: $[\ulcorner t\urcorner/x]\ulcorner A\urcorner = \ulcorner[t/x]A\urcorner$ means variables are "invisible".

Properties of Representations

- Variables by name: inductive, but must axiomatize variable renaming and substitution.
- Variables as de Bruijn indices: inductive, captures variable renaming, must axiomatize substitution.
- Variables as meta-language variables: not inductive, captures variable renaming, substitution.

$$\ulcorner \forall x.\, p(x) \supset q(x) \urcorner = \mathsf{forall}\,(\lambda x{:}\mathsf{i}.\, \mathsf{imp}\,(\mathsf{p}\,x)\,(\mathsf{q}\,x))$$

$$=_\alpha \mathsf{forall}\,(\lambda y{:}\mathsf{i}.\, \mathsf{imp}\,(\mathsf{p}\,y)\,(\mathsf{q}\,y)) = \ulcorner \forall y.\, p(y) \supset q(y) \urcorner$$

$$\ulcorner \forall x.\, A \urcorner = \mathsf{forall}\,(\lambda x{:}\mathsf{i}.\, \ulcorner A \urcorner)$$

$$\ulcorner [t/x]A \urcorner = [\ulcorner t \urcorner / x]\ulcorner A \urcorner =_\beta (\lambda x{:}\mathsf{i}.\, \ulcorner A \urcorner)\, \ulcorner t \urcorner$$

Judgments as Types

- Use (dependently typed) λ-calculus to represent derivations.
- Key idea:

  Represents judgments of the object language by types in the meta-language.

- Technique employed in LF, variations in HHF.
- In general not inductive.

Representation of Axiomatic Derivations I

If $\begin{array}{c}\mathcal{H} \\ \vdash A\end{array}$ then $\ulcorner\mathcal{H}\urcorner : \mathsf{hil}\,\ulcorner A\urcorner$, so $\quad \mathsf{hil} : \mathsf{o} \rightarrow \mathsf{type}$

$$\ulcorner \dfrac{}{\vdash A \supset (B \supset A)}\,K \urcorner = \mathsf{k}\,\ulcorner A\urcorner\,\ulcorner B\urcorner$$

$\mathsf{k} : \Pi A{:}\mathsf{o}.\,\Pi B{:}\mathsf{o}.\,\mathsf{hil}\,(\mathsf{imp}\,A\,(\mathsf{imp}\,B\,A))$

$$\ulcorner \dfrac{\begin{array}{c}\mathcal{G} \\ \vdash A \supset B\end{array} \qquad \begin{array}{c}\mathcal{H} \\ \vdash A\end{array}}{\vdash B}\,MP \urcorner = \mathsf{mp}\,\ulcorner A\urcorner\,\ulcorner B\urcorner\,\ulcorner \mathcal{G}\urcorner\,\ulcorner \mathcal{H}\urcorner$$

$\mathsf{mp} : \Pi A{:}\mathsf{o}.\,\Pi B{:}\mathsf{o}.\,\mathsf{hil}\,(\mathsf{imp}\,A\,B) \rightarrow \mathsf{hil}\,A \rightarrow \mathsf{hil}\,B$

Representation of Axiomatic Derivations II

$$\ulcorner \frac{}{(\forall x.\, A) \supset [t/x]A} F_1 \urcorner = \mathsf{f}_1\, (\lambda x{:}\mathsf{i}.\, \ulcorner A \urcorner)\, \ulcorner t \urcorner$$

$$\mathsf{f}_1 : \Pi A{:}\mathsf{i} \to \mathsf{o}.\, \Pi t{:}\mathsf{i}.\, \mathsf{hil}\, (\mathsf{imp}\, (\mathsf{forall}\, (\lambda x{:}\mathsf{i}.\, A\, x))\, (A\, t))$$

$$\ulcorner \frac{}{(\forall x.\, (B \supset A)) \supset (B \supset \forall x.\, A)} F_2^* \urcorner = \mathsf{f}_2\, (\lambda x{:}\mathsf{i}.\, \ulcorner A \urcorner)\, \ulcorner B \urcorner$$

$$\begin{aligned} \mathsf{f}_2 : \Pi A{:}\mathsf{i} \to \mathsf{o}.\, \Pi B{:}\mathsf{o}.& \\ & \mathsf{hil}\, (\mathsf{imp}\, (\mathsf{forall}\, (\lambda x{:}\mathsf{i}.\, \mathsf{imp}\, B\, (A\, x)))\, (\mathsf{imp}\, B\, (\mathsf{forall}\, (\lambda x{:}\mathsf{i}.\, A\, x)))) \end{aligned}$$

- (F_2^*) x not free in B.
- Note how side condition is enforced in the representation.

Representation of Axiomatic Derivations III

$$\ulcorner \begin{array}{c} \mathcal{H} \\ \dfrac{\vdash [a/x]A}{\vdash \forall x.\, A}\, UG^a \end{array} \urcorner = \mathsf{ug}\,(\lambda x{:}\mathsf{i}.\, \ulcorner A \urcorner)\,(\lambda a{:}\mathsf{i}.\, \ulcorner \mathcal{H} \urcorner)$$

$$\mathsf{ug} : \Pi A{:}\mathsf{i} \to \mathsf{o}.\, (\Pi a{:}\mathsf{i}.\, \mathsf{hil}\,(A\,a)) \to \mathsf{hil}\,(\mathsf{forall}\,(\lambda x{:}\mathsf{i}.\, A\,x))$$

- Again, side condition enforced in logical framework.

The Framework II: LF Type Theory

- Dependently typed λ-calculus [Harper, Honsell & Plotkin'93].

$$
\begin{array}{llll}
\text{Kinds} & K & ::= & \mathsf{type} \mid \Pi x{:}A.\, K \\
\text{Families} & A & ::= & a \mid A\ M \mid \Pi x{:}A_1.\, A_2 \quad [\mid A_1 \rightarrow A_2] \\
\text{Objects} & M & ::= & c \mid x \mid \lambda x{:}A.\, M \mid M_1\ M_2 \\
\text{Signatures} & \Sigma & ::= & \cdot \mid \Sigma, a{:}K \mid \Sigma, c{:}A \\
\text{Contexts} & \Gamma & ::= & \cdot \mid \Gamma, x{:}A
\end{array}
$$

- Core Judgments (similar ones at other levels):

$$
\begin{array}{ll}
\Gamma \vdash^{\Pi} M : A & M \text{ has type } A \\
\Gamma \vdash^{\Pi} M = N : A & M \text{ is equal to } N \text{ at type } A \\
\Gamma \vdash^{\Pi} M \Uparrow A & M \text{ is canonical of type } A
\end{array}
$$

LF Type Theory

- Characteristic rules:

$$\frac{\Gamma \vdash^{\Pi} M : \Pi x{:}A.\, B \qquad \Gamma \vdash^{\Pi} N : A}{\Gamma \vdash^{\Pi} M\, N : [N/x]B}$$

$$\frac{\Gamma \vdash^{\Pi} M : A \qquad \Gamma \vdash^{\Pi} A = B : \mathsf{type}}{\Gamma \vdash^{\Pi} M : B}$$

- Type checking and conversion is decidable.

- Every well-typed object has a unique canonical form.

Adequacy Theorem

- **Theorem** [Adequacy] For $\Gamma_a = a_1{:}\mathsf{i}, \ldots, a_n{:}\mathsf{i}$ and deductions $\mathcal{H}$ of $\vdash A$ with free parameters among $a_1, \ldots, a_n$,

  1. $\Gamma_a \vdash^{\sqcap} M \Uparrow \mathsf{hil}\ulcorner A \urcorner \quad \text{iff} \quad M = \ulcorner \mathcal{H} \urcorner$

  2. The representation is a compositional bijection:

  $$\ulcorner [t/a]\mathcal{H} \urcorner = [\ulcorner t \urcorner / a]\ulcorner \mathcal{H} \urcorner$$

- Proof-checking in object logic is reduced to type-checking in logical framework.

- Made practical through type reconstruction and redundancy elimination.

Properties of Representations

- Generally not inductive.
- Judgment forms as type families. $\mathsf{hil} : \mathsf{o} \to \mathsf{type}$
- Judgments as types. $\mathsf{hil}\, \ulcorner A \urcorner : \mathsf{type}$
- Deductions as objects. $\ulcorner \mathcal{H} \urcorner : \mathsf{hil}\, \ulcorner A \urcorner$
- Parametric judgments as dependent function types.
  $\Pi a{:}\mathsf{i}.\, \mathsf{hil}\, (\ulcorner [a/x]A \urcorner) : \mathsf{type}$
- Parametric derivations as functions.

Some Alternative Representations

- Hereditary Harrop formulas (λProlog, Isabelle)
  - Judgments as propositions in meta-logic.
  - No internal notion of deduction (logic vs. type theory)
- Inductive definitions (FS_0, many others)
  - No higher-order abstract syntax.
  - Deductions as objects (sometimes).

Hypothetical Judgments

- Would like to prove Hilbert's deduction theorem:

  If $\Delta, A \vdash C$ then $\Delta \vdash A \supset C$.

  (Note: Δ considered modulo permutations)

- Relies on *hypothetical judgment* $\Delta \vdash C$.

- Systematically extend rules. For example:

$$\frac{}{\Delta, A \vdash A}\,\mathsf{hyp} \qquad\qquad \frac{\Delta \vdash [a/x]A}{\Delta \vdash \forall x.\, A}\,UG^a$$

  where a does not occur in Δ or A.

Characteristic Properties of Hypothetical Judgments

- Substitution property:

  If $\Delta \vdash A$ and $\Delta', A \vdash C$ then $\Delta', \Delta \vdash C$.

  Realized by substitution into derivations.

- Exchange.
- Weakening.
- Contraction.

Extending the Representation

- Modeling assumptions in object language by hypotheses in meta-language.

- **Theorem** [Extended Adequacy] For $\Gamma_a = a_1{:}\mathsf{i}, \ldots, a_n{:}\mathsf{i}$, $\Gamma_H = u_1{:}\mathsf{hil}\,\ulcorner A_1 \urcorner, \ldots, u_k{:}\mathsf{hil}\,\ulcorner A_k \urcorner$, and deductions $\mathcal{H}$ of $A_1, \ldots, A_k \vdash A$ with free parameters among $a_1, \ldots, a_n$,

  1. $\Gamma_a, \Gamma_H \vdash^{\sqcap} M \Uparrow \mathsf{hil}\,\ulcorner A \urcorner$ iff $\ulcorner \mathcal{H} \urcorner = M$.

  2. The representation is a compositional bijection:

$$\ulcorner [t/a]\mathcal{H} \urcorner = [\ulcorner t \urcorner / a] \ulcorner \mathcal{H} \urcorner \quad \text{and} \quad \ulcorner [\mathcal{G}/u]\mathcal{H} \urcorner = [\ulcorner \mathcal{G} \urcorner / u] \ulcorner \mathcal{H} \urcorner$$

- $\ulcorner \dfrac{}{\Delta, A \vdash A}\,\mathsf{hyp} \urcorner = u$ where $u{:}\mathsf{hil}\,\ulcorner A \urcorner$ in context.

Logical Frameworks Assessment

- Concise, elegant, effective in many cases.
- Examples in programming languages and logics.
- Meta-programming via ML (Isabelle).
- Constraint logic programming (λProlog, Elf).
- Applications with *state* much easier in *linear* frameworks.
- Some applications call for general constraint systems.
- Proof-carrying code and theorem proving
  On the value of proof terms! (LF, but not HHF.)

Logical Frameworks Summary

- Higher-order abstract syntax captures variable renaming, substitution, occurrence conditions.
- Contextual representation of parametric and hypothetical judgments captures substitution, exchange, weakening, and contraction.
- Representation of judgments as types allows deductions as objects with internal validity conditions.
- Inductive representations allow induction.

Meta-Logical Frameworks

- Can we take advantage of the immediate nature of logical framework encodings to automate meta-theory?
- Four approaches:

  1. Relational meta-theory, plus schema-checking [Rohwedder & Pf'96]
  2. Reflection, consistent via modality [Despeyroux, Schürmann & Pf'97]
  3. (Meta-)meta-logic constructed over logical framework.
  4. Definitional reflection and induction over logical framework.

- Alternative: use inductive encodings and develop theories of substitution, assumptions, etc.
- Alternative: reflection in rewriting logic.

Reasoning About Deductive Systems Represented in LF

- Design a logic or type theory to reason about deductive systems encoded in LF.
- Impractical to simply define LF as an inductive theory.
  (LF is simple, but its theory is complex!)
- Goals:
  - Conservative — preserve present LF representation techniques.
  - Natural — informal proofs can be expressed directly.
  - General — allow many meta-theorems and meta-proof techniques.
  - Automatable — support efficient automation of meta-proof search.
- Inherit as much as possible from the logical framework!

Example: The Deduction Theorem, Propositional Case

- **Theorem:** If $\begin{array}{c}\mathcal{H}\\ A \vdash C\end{array}$ then $\begin{array}{c}\mathcal{D}\\ \vdash A \supset C\end{array}$.
- **Proof:** By induction on $\mathcal{H}$ (see following slides).
- Recall: Hypothetical judgments as functions types.

$$\ulcorner \begin{array}{c}\mathcal{H}\\ A \vdash C\end{array} \urcorner = \lambda u{:}\mathsf{hil}\ulcorner A\urcorner.\ulcorner\mathcal{H}\urcorner\, u : \mathsf{hil}\ulcorner A\urcorner \to \mathsf{hil}\ulcorner C\urcorner$$

- Representation, Hypothesis:

$$\ulcorner \overline{A \vdash A}\,\mathsf{hyp} \urcorner = \lambda u{:}\mathsf{hil}\ulcorner A\urcorner.\, u : \mathsf{hil}\ulcorner A\urcorner \to \mathsf{hil}\ulcorner A\urcorner$$

Hypothetical Judgments, Revisited

- Representation, Modus Ponens:

$$
\ulcorner \begin{array}{c} \dfrac{\begin{array}{c}\mathcal{H}_1 \\ A \vdash B \supset C\end{array} \qquad \begin{array}{c}\mathcal{H}_2 \\ A \vdash B\end{array}}{A \vdash C} MP \end{array} \urcorner
$$

$$
= \quad \lambda u{:}\mathsf{hil}\,\ulcorner A \urcorner.\ \mathsf{mp}\,\ulcorner B \urcorner \ulcorner C \urcorner\,(\ulcorner \mathcal{H}_1 \urcorner\, u)(\ulcorner \mathcal{H}_2 \urcorner\, u) : \mathsf{hil}\,\ulcorner A \urcorner \to \mathsf{hil}\,\ulcorner C \urcorner
$$

Representation of Meta-Proofs

- Attempt: Represent proof of deduction theorem as function

$$\mathsf{ded} : \Pi A{:}\mathsf{o}.\, \Pi C{:}\mathsf{o}.\ \underbrace{(\mathsf{hil}\, A \to \mathsf{hil}\, C)}_{\text{in LF}} \quad \underbrace{\quad\to\quad}_{\text{outside LF}} \quad \mathsf{hil}\,(\mathsf{imp}\, A\, C)$$

  fails, because outer function is defined inductively.

- Solution: introduce separate level with $\forall$, $\exists$, $\top$.

$$\mathsf{ded} \in \forall A{:}\mathsf{o}.\, \forall C{:}\mathsf{o}.\, \forall H{:}(\mathsf{hil}\, A \to \mathsf{hil}\, C).\, \exists D{:}\mathsf{hil}\,(\mathsf{imp}\, A\, C).\, \top$$

- Suppress propositional arguments for deductions.

$$\mathsf{ded} \in \forall H{:}(\mathsf{hil}\, A \to \mathsf{hil}\, C).\, \exists D{:}\mathsf{hil}\,(\mathsf{imp}\, A\, C).\, \top$$

Proof of Deduction Theorem, Hypothesis

- **Case:**

$$\mathcal{H} = \frac{}{A \vdash A}\,\mathsf{hyp}$$

1	$(A \supset ((A \supset A) \supset A)) \supset ((A \supset (A \supset A)) \supset (A \supset A))$	S
2	$(A \supset ((A \supset A) \supset A))$	K
3	$(A \supset (A \supset A)) \supset (A \supset A)$	MP 1 2
4	$A \supset (A \supset A)$	K
5	$A \supset A$	MP 3 4

$$\mathsf{ded}\,(\lambda u{:}\mathsf{hil}\,A.\,u) = \mathsf{mp}\,(\mathsf{mp}\,\mathsf{s}\,\mathsf{k})\,\mathsf{k}$$

Proof of Deduction Theorem, Axiom

- **Case:**

$$\mathcal{H} = \dfrac{}{A \vdash C_1 \supset (C_2 \supset C_1)}\,K$$

1	$\vdash (C_1 \supset (C_2 \supset C_1)) \supset (A \supset (C_1 \supset (C_2 \supset C_1)))$	K
2	$\vdash C_1 \supset (C_2 \supset C_1)$	K
3	$\vdash A \supset (C_1 \supset (C_2 \supset C_1))$	$MP\,1\,2$

$$\mathsf{ded}\,(\lambda u{:}\mathsf{hil}A.\,\mathsf{k}) = \mathsf{mp}\,\mathsf{k}\,\mathsf{k}$$

- **Case:** Axiom (S) similar.

Proof of Deduction Theorem, Modus Ponens

- **Case:**

$$\mathcal{H} = \dfrac{\begin{array}{c}\mathcal{H}_1 \\ A \vdash C_1 \supset C_2\end{array} \qquad \begin{array}{c}\mathcal{H}_2 \\ A \vdash C_1\end{array}}{A \vdash C_2}\, MP$$

1	$\vdash A \supset (C_1 \supset C_2)$	Ind. hyp. on $\mathcal{H}_1$
2	$\vdash (A \supset (C_1 \supset C_2)) \supset ((A \supset C_1) \supset (A \supset C_2))$	S
3	$\vdash (A \supset C_1) \supset (A \supset C_2)$	MP 2 1
4	$\vdash A \supset C_1$	Ind. hyp. on $\mathcal{H}_2$
5	$\vdash A \supset C_2$	MP 3 4

$$\mathsf{ded}\,(\lambda u{:}\mathsf{hil}\,A.\,\mathsf{mp}\,(H_1\,u)\,(H_2\,u)) = \mathsf{mp}(\mathsf{mp}\,\mathsf{s}\,(\mathsf{ded}\,H_1))\,(\mathsf{ded}\,H_2)$$

Proof of Deduction Theorem, Summary

- Implicational fragment

$$\begin{array}{lcl}[\ \mathsf{ded} & : & (\mathsf{hil}\,A \to \mathsf{hil}\,C) \Rightarrow \mathsf{hil}\,(\mathsf{imp}\,A\,C)\ ] \\ \mathsf{ded} & \in & \forall H{:}(\mathsf{hil}\,A \to \mathsf{hil}\,C).\,\exists D{:}\mathsf{hil}\,(\mathsf{imp}\,A\,C).\,\top\end{array}$$

$$\begin{array}{lcl}\mathsf{ded}\,(\lambda u{:}\mathsf{hil}\,A.\,u) & = & \mathsf{mp}\,(\mathsf{mp}\,\mathsf{s}\,\mathsf{k})\,\mathsf{k} \\ \mathsf{ded}\,(\lambda u{:}\mathsf{hil}A.\,\mathsf{k}) & = & \mathsf{mp}\,\mathsf{k}\,\mathsf{k} \\ \mathsf{ded}\,(\lambda u{:}\mathsf{hil}A.\,\mathsf{s}) & = & \mathsf{mp}\,\mathsf{k}\,\mathsf{s} \\ \mathsf{ded}\,(\lambda u{:}\mathsf{hil}\,A.\,\mathsf{mp}\,(H_1\,u)\,(H_2\,u)) & = & \mathsf{mp}\,(\mathsf{mp}\,\mathsf{s}\,(\mathsf{ded}\,H_1))\,(\mathsf{ded}\,H_2)\end{array}$$

- ded is a total function of the given type, therefore represents meta-theoretic proof.

The Meta-Logic

- Trade off generality vs. automation.
- Exploit LF as much as possible.

$$\text{Formulas} \quad F \quad ::= \quad \forall x{:}A.\, F \mid \exists x{:}A.\, F \mid \top$$

  At present, restricted to $\forall \ldots \forall \exists \ldots \exists$ prefix.

- Validity of meta-logic.

$$\begin{array}{lll} \models \forall x{:}A.\, F & \text{iff} \quad \models [M/x]F & \text{for all } M \text{ s.t. } \cdot \vdash^{\sqcap} M : A \\ \models \exists x{:}A.\, F & \text{iff} \quad \models [M/x]F & \text{for some } M \text{ s.t. } \cdot \vdash^{\sqcap} M : A \\ \models \top & & \end{array}$$

- Implementation is, of course, incomplete.

Excursion: How to Exploit LF

- Represent falsehood by

$$\mathsf{void} : \mathsf{type}$$

  and no constructors.

- Represent $\neg A(x)$ by $A(x) \to \mathsf{void}$.

- Represent disjunction of types $A(x)$ and $\neg A(x)$ by

$$\mathsf{disj} : \mathsf{i} \to \mathsf{type}$$

$$\mathsf{injl} : \Pi x{:}\mathsf{i}.\, A\,x \to \mathsf{disj}\,x$$

$$\mathsf{injr} : \Pi x{:}\mathsf{i}.\, (A\,x \to \mathsf{void}) \to \mathsf{disj}\,x$$

- Prove decidability as $\forall x.\, A(x) \vee \neg A(x)$:

$$\forall x{:}\mathsf{i}.\, \exists D{:}\mathsf{disj}\,x.\, \top$$

Meta-Logic via Realizability

- $\models F$ if there is a total function P such that $P \in F$.

$$
\begin{array}{llll}
\text{Meta-Functions} \quad P & ::= & \Lambda x{:}A.\, P \mid P\, M & (\forall x{:}A.\, F) \\
 & \mid & \langle M, P \rangle \mid \mathbf{let}\ \langle x, p \rangle = P_1\ \mathbf{in}\ P_2 & (\exists x{:}A.\, F) \\
 & \mid & \langle\,\rangle \mid \mathbf{let}\ \langle\,\rangle = P_1\ \mathbf{in}\ P_2 & (\top) \\
 & \mid & \mu p{:}F.\, P \mid p & (\text{recursion}) \\
 & \mid & \mathbf{case}\ x\ \mathbf{of}\ \Omega & (\text{cases}) \\
 & & & \\
\Omega & ::= & \cdot & \\
 & \mid & (M \Rightarrow P \mid \Omega) & (\text{case})
\end{array}
$$

- Standard (non-deterministic) operational semantics.

Checking Realizors

- Check three conditions.

  1. **Type Correctness:** Returned values have expected type.
     Standard dependent type checking.
  2. **Termination:** Function always succeeds with a value or fails finitely along each computation branch.
     Currently, simultaneous and lexicographic extensions of higher-order subterm ordering, given explicitly [Rohwedder & Pf '96].
  3. **Coverage:** Functions never fail.
     Using higher-order pattern unification, check that all cases for closed terms of a given type a covered *syntactically*.

- Functions may be don't-care non-deterministic. (Many proofs are.)

- Both termination and coverage are in principle undecidable.

- Termination is open-ended in practice.

Generating Realizors = Meta-Theorem Proving

- Filling ($\Longrightarrow$ Type-checking):
  Simple CLP-style iterative deepening search in LF.
  Uses signature and results of induction hypothesis.

- Recursion ($\Longrightarrow$ Termination-checking):
  Generate legal appeals to induction hypothesis given a termination order.

- Splitting ($\Longrightarrow$ Coverage):
  Generate possible cases for variable by unification (one step backward search).

Objects With Parameters

- Above relies crucially on *closed* terms.

$$\models \forall x{:}A.\, F \quad \text{iff} \quad \models [M/x]F \quad \text{for all } M \text{ s.t. } \cdot \vdash^{\Pi} M : A$$

$$\models \exists x{:}A.\, F \quad \text{iff} \quad \models [M/x]F \quad \text{for some } M \text{ s.t. } \cdot \vdash^{\Pi} M : A$$

$$\models \top$$

- Must generalize to allow proof of deduction theorem in first-order logic (parameters) or with arbitrary assumptions (hypotheses).

  If $\Delta, A \vdash C$ then $\Delta \vdash A \supset C$.

Informal Proof

- **New Case:**

$$\mathcal{H} = \frac{}{\underbrace{\Delta', C}_{=\Delta}, A \vdash C}\ \mathsf{hyp}$$

$$\begin{array}{llr}
1 & \Delta', C \vdash C \supset (A \supset C) & (K) \\
2 & \Delta', C \vdash C & (\mathsf{hyp}) \\
3 & \Delta', C \vdash A \supset C & MP\ 1\ 2
\end{array}$$

$$\mathsf{ded}\,(\lambda u{:}\mathsf{hil}\,A.\,\underline{w}) = \mathsf{mp}\,\mathsf{k}\,\underline{w}$$

where $\underline{w}{:}\mathsf{hil}\,\ulcorner C \urcorner$ in context?

Formulation in LF

- Recall: Contexts are mapped to contexts.
- For $\Delta = A_1, \ldots, A_n$,

$$\ulcorner \Delta \urcorner = u_1{:}\ulcorner A_1 \urcorner, \ldots, u_n{:}\ulcorner A_n \urcorner$$

- If $\begin{array}{c} \mathcal{H} \\ \Delta, A \vdash C \end{array}$ then $\ulcorner \Delta \urcorner \vdash^{\Pi} \ulcorner \mathcal{H} \urcorner : \mathsf{hil}\, \ulcorner A \urcorner \to \mathsf{hil}\, \ulcorner C \urcorner$.
- As a statement about encoding:

  For all Γ of the form $u_1{:}\mathsf{hil}\, A_1, \ldots, u_n{:}\mathsf{hil}\, A_n$ and objects H such that $\Gamma \vdash^{\Pi} H : \mathsf{hil}\, A \to \mathsf{hil}\, C$, there exists an object D such that $\Gamma \vdash^{\Pi} D : \mathsf{hil}\, (\mathsf{imp}\, A\, C)$.

Extension of Meta-Logic

- Inductive description of classes of contexts.
- Example: $\Gamma_H = \cdot \mid \Gamma_H, \underline{u}{:}\mathsf{hil}\, A$ (for some A). Write $\Gamma \in \Gamma_H$.
- Allow outermost quantification over inductively defined contexts.
- Fix signature Σ and context class Γ_X.

$$
\begin{array}{lll}
\models F & \text{iff} & \Gamma \models F \quad \text{for all } \Gamma \in \Gamma_X \\
\Gamma \models \forall x{:}A.\, F & \text{iff} & \Gamma \models [M/x]F \quad \text{for all } M \text{ s.t. } \Gamma \vdash^{\Pi} M : A \\
\Gamma \models \exists x{:}A.\, F & \text{iff} & \Gamma \models [M/x]F \quad \text{for some } M \text{ s.t. } \Gamma \vdash^{\Pi} M : A \\
\Gamma \models \top & &
\end{array}
$$

Extension of Realizors

- Cannot use parameters $\underline{x}$ at the top-level, because the context may be empty!
- Two ways to introduce parameters

  1. In the case that a given term is a parameter.
  2. Explicit $\nu\underline{x}{:}A$.

Proof of Deduction Theorem Revisited

- $\Gamma_H ::= \cdot \mid \Gamma_H, \underline{w}{:}A$

$$
\begin{aligned}
\mathsf{ded}\,(\lambda u{:}\mathsf{hil}\,A.\,\underline{w}) &= \mathsf{mp}\,\mathsf{k}\,\underline{w}\\
\mathsf{ded}\,(\lambda u{:}\mathsf{hil}\,A.\,u) &= \mathsf{mp}\,(\mathsf{mp}\,\mathsf{s}\,\mathsf{k})\,\mathsf{k}\\
\mathsf{ded}\,(\lambda u{:}\mathsf{hil}A.\,\mathsf{k}) &= \mathsf{mp}\,\mathsf{k}\,\mathsf{k}\\
\mathsf{ded}\,(\lambda u{:}\mathsf{hil}A.\,\mathsf{s}) &= \mathsf{mp}\,\mathsf{k}\,\mathsf{s}\\
\mathsf{ded}\,(\lambda u{:}\mathsf{hil}\,A.\,\mathsf{mp}\,(H_1\,u)\,(H_2\,u)) &= \mathsf{mp}\,(\mathsf{mp}\,\mathsf{s}\,(\mathsf{ded}\,H_1))\,(\mathsf{ded}\,H_2)
\end{aligned}
$$

- At run-time, $\underline{w}$ will match one of many possible parameters w_i.
- Need to cross-reference parameters with context-class definition.

Impact on Verification

- **Type-checking.** Verify that any parameters introduced lie within the specified context class. Context class inclusion when using lemmas.
- **Termination.** Not affected.
- **Coverage.** Verify that in addition to signature elements, all possible parameter cases are covered.
- Treats only context properties stable under exchange, weakening, contraction (hypothetical judgments).

Deduction Theorem in First-Order Logic

- **Case:**

$$\mathcal{H} = \frac{\begin{array}{c}\mathcal{H}_1\\ \Delta, A \vdash [a/x]C_1\end{array}}{\Delta, A \vdash \forall x.\, C_1}\, UG^a$$

where a not in Δ, A, or C.

1	$\Delta \vdash A \supset [a/x]C_1$	Ind. hyp. on $\mathcal{H}_1$
2	$\Delta \vdash \forall x.\, A \supset C_1$	UG^a 1
3	$\Delta \vdash (\forall x.\, A \supset C_1) \supset (A \supset \forall x.\, C_1)$	F_2^*
4	$\Delta \vdash A \supset \forall x.\, C_1$	MP 3 2

Representation in Meta-Logic

- Recall: $\mathsf{ug} : \Pi C_1{:}\mathsf{i} \to \mathsf{o}.\, (\Pi a{:}\mathsf{i}.\, \mathsf{hil}\,(C_1\, a)) \to \mathsf{hil}\,(\mathsf{forall}\,(\lambda x{:}\mathsf{i}.\, C_1\, x))$.
- Declare

$$\Gamma_D = \cdot \mid \Gamma_D, \underline{w}{:}\mathsf{hil}\, A \mid \Gamma_D, \underline{a}{:}\mathsf{i}$$

- New case:

$$\mathsf{ded}\,(\lambda u{:}\mathsf{hil}\, A.\, \mathsf{ug}\,(\lambda a{:}\mathsf{i}.\, C_1\, u\, a)) = \mathsf{ug}\,(\nu \underline{a}{:}\mathsf{i}.\, \mathsf{mp}\ \mathsf{f}_2\,(\mathsf{ded}\,(\lambda u{:}\mathsf{hil}\, A.\, C_1\, u\, \underline{a})))$$

- Evaluation of $\nu \underline{x}{:}A.\, P$

1. creates a new actual parameter x for $\underline{x}$,
2. evaluates $[x/\underline{x}]P$ to V,
3. returns the abstraction $\lambda x{:}A.\, V$.

The ν Operator

- Slightly more complicated when several LF objects are returned ($\forall \ldots \forall \exists \ldots \exists$)
- Final complication: *subordination*.
- We do not abstract, if the ν-bound variable cannot occur in the result.

Example: Counting Axiom Occurrences

$$\mathsf{nat} : \mathsf{type}. \qquad \mathsf{zero} : \mathsf{nat}. \qquad \mathsf{one} : \mathsf{nat}.$$

$$\mathsf{plus} : \mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}$$

- $\Gamma_a ::= \cdot \mid \Gamma_a, \underline{a}{:}\mathsf{i}$ $\qquad$ [or $\Gamma_D ::= \cdot \mid \Gamma_D, \underline{a}{:}\mathsf{i} \mid \Gamma_D, \underline{w}{:}\mathsf{hil}\, A$].
- $\mathsf{cnt} \in \forall H{:}\mathsf{hil}\, A.\, \exists N{:}\mathsf{nat}.\, \top.$

$$\begin{array}{lcl}
\mathsf{cnt}\,\mathsf{k} & = & \mathsf{one} \\
\ldots \quad \text{(other axioms)} & & \ldots \\
\mathsf{cnt}\,(\mathsf{mp}\, H_1\, H_2) & = & \mathsf{plus}\,(\mathsf{cnt}\, H_1)\,(\mathsf{cnt}\, H_2) \\
\mathsf{cnt}\,(\mathsf{ug}\,(\lambda a{:}\mathsf{i}.\, H\, a)) & = & \nu \underline{a}{:}\mathsf{i}.\, \mathsf{cnt}\,(H\, \underline{a}) \\
[\ \mathsf{cnt}\, \underline{w} & = & \mathsf{zero}\ ]
\end{array}$$

Subordination

- If $A \lhd B$ then a term of type B can not occur as a subterm of a canonical term of type A.
- Extracted statically from signature.
- Determines if λ-abstraction is constructed from ν.
- Also important in termination checking [Rohwedder & Pf'96]
  $[t/x]A < \forall x{:}\mathsf{i}.\, A$ in first-order logic ($\mathsf{i} \lhd \mathsf{o}$)
  $[B/p]A \not< \forall p{:}\mathsf{i}.\, A$ in higher-order logic ($\mathsf{o} \not\lhd \mathsf{o}$)
- Also needed for equational reasoning in LF [Virga'99].
- In (predicative) inductive theories, given by the order of definition.

Status and Implementation

- Theory of meta-logic recently completed [Schürmann '00].
- Prototype of theorem prover exists (not yet released, but available).
- No tactics(!), development in definition/lemma/theorem style.
- Work on various extensions in progress.

Some Twelf Experiments

Experiment	**Time**
CCC to λ-calculus	1.099
CPM completeness	1.134
(Horn) LP soundness	4.501
(Horn) LP completeness	0.195
Mini-ML type preservation	0.799
Mini-ML evaluation/reduction	25.546
Deduction theorem	0.322
*Axiomatic to natural deductions	
*Natural to axiomatic deductions	
*Intuitionistic cut elimination	
*Classical cut elimination	
*Sequent calculus to natural deduction	
*Natural deduction to sequent calculus	
*Church-Rosser theorem	

Linux 2.30, SML/NJ 110, Twelf 1.2 (*Twelf 1.5) on Pentium II (300 Mhz)

Assessment I

- Must provide: induction order, search depth.
- Derives its power from dependent types and separation of powers.
- Excellent, if you know the proof ahead of time.
- More Information: `http://www.cs.cmu.edu/~twelf/`

Assessment II

- Not robust with respect to failure.
- Naive strategy (filling $\rightarrow$ splitting $\rightarrow$ recursion).
- Filling sometimes a bottle-neck (anticipate lemmas).
- Too dependent on number of expression constructors (orthogonality?).
- Inefficient implementation (bottom-up vs. top-down).
- More termination orders and reduction properties.
- Proof terms (separate checking, interactive vs. automatic).

Other Future Work

- Constraints (currently, only in operational semantics of Twelf). [Virga'99]
- Linearity (reason about state). [Cervesato & Pf.'96]
- Order (reasoning about sequencing). [Polakow & Pf.'99]
- Proof compression. [Necula'98] [Schürmann & Pf.'98]
- Compilation. [Nadathur'99]

Related Work: FOLDN

- FOLDN [McDowell & Miller'97] [McDowell'97]
- Logical framework flexible (HHF, linear HHF).
- Meta-logic richer, less automation (at present).
- Induction over natural numbers (rather than termination).
- Definitional reflection (rather than splitting).
- Does not inherit HHF theorem proving.
- Does not inherit reasoning about hypotheses (modeled as lists).
- Does not inherit reasoning about parameters (nested abstractions?)

Related Work: Maude

- Maude [Basin, Clavel & Meseguer'99].
- Logical framework based on rewriting logic.
- Encodings are first-order.
- Use as meta-logical framework from
  - reflection (representation of system in itself),
  - initiality (satisfies inductive properties).
- Not yet as deeply explored.

Related Work: Inductive Encodings

- FS_0, Isabelle/HOL, Coq, LEGO, HOL, Nuprl, Agda.
- Except for FS_0, not explicitly designed as meta-logical framework.
- Only inductive encodings and reasoning
  - No higher-order abstract syntax.
  - No hypothetical or parametric meta-reasoning.
- Theorem proving generally based on tactics.
- Less automation, different "look & feel".
- Numerous experiments.

Summary

- Presented principles underlying LF and similar logical frameworks.
  - Higher-order abstract syntax.
  - Judgments as types.
  - Hypothetical and parametric judgments.
- Explored design of meta-logical framework of LF encodings.
  - Reasoning about closed objects.
  - Reasoning about hypotheses and parameters.
- Sketched automation techniques in Twelf.