# Meta-Logical Frameworks
# A Tutorial

Carsten Schürmann
Carnegie Mellon University
June 2000

## Outline of This Tutorial

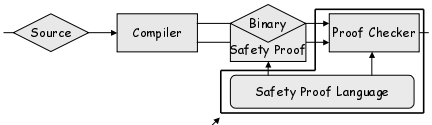
Motivation
- Safety Architectures

Logical Frameworks
- Design, Example
- Implementation

Meta-logical Frameworks
- Design, Example
- Implementation



## Safety Architectures




Trusted Computing Base



## Motivation

Claim: We need tools to
- Engineer trusted bases for computing
- Catch programming language design flaws
- Increase confidence in software

In this tutorial:

Tools to reason about deductive systems



## Safety Architectures

Proof Carrying Code
- Logic: 129 rules [Necula, Lee 97]
  several 100 rules [Appel, Felty 99]
  (Uses a logical framework)
- Proof checker: 206 lines [Schürmann 98]

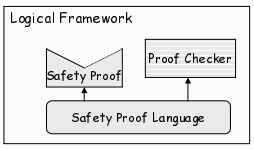
Typed Assembly Language
- Type theory: 31 rules [Morrisett, Crary … 98]
- Proof Checker: approx 4000 lines

Java Bytecode
- Type system: 20 pages prose
- Bytecode verifier



## Logical Frameworks

Uniform representation language
- Storing
- Shipping
- Checking




Logic-independent safety proof checker

## Safety Proof Languages

First-order/higher-order logics
[Gentzen 35]

Temporal logics (CTL, CTL*, LTL)
[Pnueli, Manna, … 84]

Modal and linear logics
[Girard 86]

Type theories

Language and system-specific knowledge

## Good Safety Proof Languages

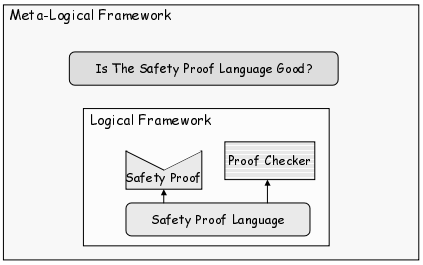
Is The Safety Proof Language Good?

Consistency
- Falsehood should not be derivable

Expressiveness
- Small safety proofs require expressive logics

Extensibility
- Possibility to add new admissible rules

## Meta-Logical Frameworks



## Common Proof Techniques

- Structural induction (over derivations)
- Inversion
- Interpretations
- Logical relations

## Some Meta-Logical Frameworks

Meta-Programming Languages
- Functional (ML, rewriting)
- Relational (λProlog, Twelf)

Theorem Proving Environments
- Tactics and tacticals
- Logic programming and proof checking

## Some Meta-Logical Frameworks

General purpose systems:
(HOL, Coq, LEGO, Nuprl, PVS, Nqthm)
- Deductive systems defined inductively
- Meta-reasoning by structural induction
- Automation by tactics/inductive theorem proving

Maude [Basin, Clavel & Meseguer'99]
- Based on rewriting logic
- Meta-reasoning by reflection and induction
- Automation by rewriting

## Some Meta-Logical Frameworks

FOLDN/Iris [McDowell, Miller 97][Wajs]
- Based on hereditary Harrop formulas (typically)
- Meta-reasoning by definitional reflection
- No automation, interactive

Twelf [Schürmann, Pfenning 98,99]
- Based on logical framework LF
- Meta-reasoning by realizability interpretation
- Automation by inductive theorem proving

## In This Tutorial

### Meta-logical frameworks

that support

the *representation* of logics and type systems
and automates *reasoning about* them

## Overview

| Reasoning | Meta-logical Framework |
|---|---|
| • Consistency arguments<br>• Theorems about logics<br>• Inductive proofs | • Automated proof search<br>• Formulas<br>• Direct encoding as proofs |
| **Deductive System** | **Logical Framework** |
| • Logic<br>• Judgments<br>• Inference rules | • Uniform language<br>• Types<br>• Direct encoding as objects |

## Let's Start

**Deductive System**
- Logic
- Judgments
- Inference rules

## Terminology

Deductive System
- Axioms
- Inference rules
- Derivations

Examples:
- Logics
- Programming Languages
- Operational Semantics
- Compilers

## The Example: Intuitionistic Logic

Abstract syntax
- Terms: $t ::= a \mid x \mid f(t_1,\ldots,t_n)$
- Propositions: $A ::= p(t_1,\ldots,t_n) \mid A_1 \supset A_2 \mid \forall x.A$
- Assumptions: $\Gamma ::= \cdot \mid \Gamma, A$

Sequent calculus
- Judgment: $\Gamma \Rightarrow A$

## Axioms and Rules

[Gentzen 35]

$$\frac{}{\Gamma, A \Rightarrow A}\,axiom$$

$$\frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \supset B}\,impr \qquad \frac{\Gamma, A \supset B \Rightarrow A \quad \Gamma, A \supset B, B \Rightarrow C}{\Gamma, A \supset B \Rightarrow C}\,impl$$

$$\frac{\Gamma \Rightarrow A[a/x]}{\Gamma \Rightarrow \forall x : A}\,allr \qquad \frac{\Gamma, \forall x.A, A[t/x] \Rightarrow B}{\Gamma, \forall x.A \Rightarrow B}\,alll$$

$$\frac{\Gamma \Rightarrow A \quad \Gamma, A \Rightarrow C}{\Gamma \Rightarrow C}\,cut$$



## Next: Logical Framework LF

| Deductive System | Logical Framework |
|---|---|
| • Logic<br>• Judgments<br>• Inference rules | • Uniform language<br>• Types<br>• Direct encoding as objects |



## Goals

Factor representation effort
Support common concepts
Provide tools for generic proof search

Characterization:

1. Underlying formalism
2. Meta-programming language
3. Theorem proving environment



## Some Logical Frameworks

Formalisms

- Hereditary Harrop formulas (HHF)
- Dependently Typed λ-calculus
- Inductive definitions
- Rewriting logic

Implementations

- Isabelle, λProlog, Twelf, linear Twelf, Maude, Pi, Elan, Agda …

Survey article [Pfenning 99]



## Higher-Order Abstract Syntax

Represent object-language variables
By meta-language variables

- Use (simply typed) λ-calculus
- Technique employed in HHF, LF
- In general not inductive



## Representation Function

| | | |
|---|---|---|
| $\lceil x \rceil$ | $= x$ | $x : i$ |
| $\lceil f(t_1,\ldots,t_n) \rceil$ | $= f \lceil t_1 \rceil \ldots \lceil t_n \rceil$ | $f : i \to \cdots \to i$ |
| $\lceil p(t_1,\ldots,t_n) \rceil$ | $= p \lceil t_1 \rceil \ldots \lceil t_n \rceil$ | $p : i \to \ldots \to o$ |
| $\lceil A_1 \supset A_2 \rceil$ | $= imp \lceil A_1 \rceil \lceil A_2 \rceil$ | $imp : o \to o \to o$ |
| $\lceil \forall x.A \rceil$ | $= forall\ (\lambda x : i. \lceil A \rceil)$ | $forall : (i \to o) \to o$ |

Constant declarations form *signature*
Variable declarations form *context*

## The Framework (Part 1)

Simply typed fragment

| | | | |
|---|---|---|---|
| Types: | $A$ | ::= | $a \mid A_1 \to A_2$ |
| Objects: | $M$ | ::= | $c \mid x \mid \lambda x{:}A.M \mid M_1\ M_2$ |
| Signatures: | $\Sigma$ | ::= | $\cdot \mid \Sigma, a : \text{type} \mid \Sigma, c : A$ |
| Contexts: | $\Gamma$ | ::= | $\cdot \mid \Gamma, x : A$ |

- Shared by HFF and LF
- Typing Judgment $\Gamma \vdash^{\to} M : A$
- Canonical forms ( ) $\Gamma \vdash^{\to} M \Uparrow A$

## Adequacy Theorem

Theorem: For $\Gamma = x_1 : i, \ldots, x_n : i$ and terms $t$ and propositions $A$ with free variables among $x_1, \ldots, x_n$ .

1. $\Gamma \vdash^{\to} M \Uparrow i$ iff $M = \lceil t \rceil$
2. $\Gamma \vdash^{\to} M \Uparrow o$ iff $M = \lceil A \rceil$
3. Representation function: compositional bijection

$[\lceil t \rceil / x]\lceil s \rceil = \lceil [t/x]s \rceil$ and $[\lceil t \rceil / x]\lceil A \rceil = \lceil [t/x]A \rceil$

## Parametric Functions

Recall:

$\lceil \forall x.A \rceil = \text{forall}\ (\lambda x : \text{i}. \lceil A \rceil)$ $\text{forall} : (\text{i} \to \text{o}) \to \text{o}$

Important: for adequacy theorem to hold

Case Analysis over parametric function $\lambda x : i.A\ x$

$\text{forall}\ (\lambda x : \text{i}.A\ x) : \text{o}$ iff $A = \lambda x : \text{i}.\text{p}\ (t_1\ x) \ldots (t_n\ x)$

$A = \lambda x : \text{i}.\text{imp}\ (A_1\ x)\ (A_n\ x)$

$A = \lambda x : \text{i}.\text{forall}\ (y : i.\ A_1\ x\ y)$

## Variables and Substitutions

Variable by name:

- Inductive
- Axiomatization of renaming
- Axiomatization of substitution

Variable as de Bruijn indices:

- Inductive
- Captures renaming
- Axiomatization of substitution

## Variables and Substitutions

Variable by meta-language variables:

- Not inductive
- Captures renaming

$\lceil \forall x.p(x) \supset q(x) \rceil = \text{forall}\ (\lambda x : \text{i}.\text{imp}\ (p\ x)\ (q\ x))$

$= \text{forall}\ (\lambda y : \text{i}.\text{imp}\ (p\ y)\ (q\ y)) = \lceil \forall y.p(y) \supset q(y) \rceil$

- Captures substitution

$\lceil \forall x.A \rceil = \text{forall}\ (\lambda x : \text{i}. \lceil A \rceil)$

$\lceil [t/x]A \rceil = [\lceil t \rceil / x]\lceil A \rceil =_\beta (\lambda x : \text{i}. \lceil A \rceil)\lceil t \rceil$

## Judgments as Types

Represent object language judgments
By meta-language types

- Dependently typed $\lambda$-calculus
- Technique employed in LF, variations in HHF
- In general not inductive

## Judgments as Types

Logical framework LF [Harper, Honsell, Plotkin 93]

- Dependent types λ-calculus
- Derivations as objects
- Hypothetical judgments

$$\frac{\mathcal{D}}{A_1,\cdots,A_n \Rightarrow A} \quad \Longrightarrow \quad \text{Logical Framework LF}$$

$$\cdot \vdash^{\Pi} D : \text{hyp } A_1 \to \cdots \to \text{hyp } A_n \to \text{conc } A$$

## Judgments as Types

Alternatively:

$$\frac{\mathcal{D}}{A_1,\cdots,A_n \Rightarrow A} \quad \Longrightarrow \quad \text{Logical Framework}$$

$$u_1 : \text{hyp } A_1, \cdots, u_n : \text{hyp } A_n \vdash^{\Pi} D\, u_1 \ldots u_n : \text{conc } A$$

In LF: $\text{hyp} : \text{o} \to \text{type}$

$\text{conc} : \text{o} \to \text{type}$

Remark: substitution, weakening, exchange, contraction inherited from LF

## Derivations as objects

$$\left\lceil \frac{}{\Gamma, A \Rightarrow A}\, axiom \right\rceil = \text{axiom} \lceil A \rceil : \text{hyp} \lceil A \rceil \to \text{conc} \lceil A \rceil$$

$$\text{axiom} : \Pi A : \text{o}.\text{hyp}\, A \to \text{conc}\, A$$

$$\left\lceil \frac{\mathcal{D}}{\Gamma, A \Rightarrow B}\, impr \right\rceil = \text{impr} \lceil A \rceil \lceil B \rceil \lceil D \rceil : \text{conc} (\lceil A \rceil \text{imp} \lceil B \rceil)$$

$$\text{impr} : \Pi A : \text{o}.\Pi B : \text{o}.\ (\text{hyp}\, A \to \text{conc}\, B) \to \text{conc}\, (A \text{ imp } B)$$

## Derivations as objects

$$\left\lceil \frac{\mathcal{D}_1 \qquad \mathcal{D}_1}{\Gamma, A \supset B \Rightarrow A \quad \Gamma, A \supset B, B \Rightarrow C}\, impl \right\rceil =$$

$$\text{impl} \lceil A \rceil \lceil B \rceil \lceil C \rceil \lceil D_1 \rceil \lceil D_2 \rceil : \text{hyp} (\lceil A \rceil \text{imp} \lceil B \rceil) \to \text{conc} \lceil C \rceil$$

$$\text{impl} : \Pi A : \text{o}.\Pi B : \text{o}.\Pi C : \text{o}.\ \text{conc}\, A \to (\text{hyp}\, B \to \text{conc}\, C) \to (\text{hyp}\, (A \text{ imp } B) \to \text{conc}\, C)$$

## Derivations as objects

$$\left\lceil \frac{\mathcal{D}}{\Gamma \Rightarrow A[a/x]}\, allr^{*} \right\rceil = \text{allr} \lceil A \rceil \lceil D \rceil : \text{conc} (\text{forall} (\lambda x : i.\lceil A \rceil))$$

$$\text{allr} : \Pi A : \text{i} \to \text{o}.\ (\Pi a : i.\text{conc}\, (A\, a)) \to \text{conc}\, (\text{forall}\, (\lambda x : i.\lceil A \rceil))$$

* $a$ must be new

Side condition enforced by LF

## Derivations as objects

$$\left\lceil \frac{\mathcal{D}}{\Gamma, \forall x.A, A[t/x] \Rightarrow B}\, alll \right\rceil = \text{alll} \lceil A \rceil \lceil B \rceil \lceil D \rceil : \text{hyp} (\text{forall} (\lambda x : i.\lceil A \rceil)) \to \text{conc}\, B$$

$$\text{alll} : \Pi A : \text{i} \to \text{o}.\Pi B : \text{o}.\Pi t : \text{i}.\ (\text{hyp}\, (A\, t) \to \text{conc}\, B) \to (\text{hyp}\, (\text{forall}\, (\lambda x : i.A\, x)) \to \text{conc}\, B)$$

## Derivations as objects

$$\left[\frac{\begin{array}{cc} D_1 & D_2 \\ \Gamma \Rightarrow A & \Gamma, A \Rightarrow C \end{array}}{\Gamma \Rightarrow C}\, cut\right] = \text{cut}\lceil A\rceil\lceil C\rceil\lceil D_1\rceil\lceil D_2\rceil : \text{conc}\lceil C\rceil$$

$$\begin{array}{l} \text{cut} : \Pi A : \text{o}.\Pi C : \text{o}. \\ \quad \text{conc}\ A \\ \quad \rightarrow (\text{hyp}\ A \rightarrow \text{conc}\ C) \\ \quad \rightarrow \text{conc}\ C \end{array}$$

## Some Alternatives

Hereditary Harrop formulas ( Prolog, 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 representation techniques
- Derivations as objects (sometimes)

## The Framework (Part 2)

Logical Framework LF

[Harper, Honsell, Plotkin 93]

- Dependently typed -calculus
- No polymorphism
- No type constructors

## Logical Framework LF

- Syntax

| | | | |
|---|---|---|---|
| Kinds: | $K$ | ::= | $\text{type} \mid \Pi x : A.K$ |
| Types: | $A$ | ::= | $a \mid A\,M \mid \Pi x : A_1.A_2 \mid A_1 \rightarrow A_2]$ |
| Objects: | $M$ | ::= | $c \mid x \mid \lambda x : A.M \mid M_1\,M_2$ |
| Signatures: | $\Sigma$ | ::= | $\cdot \mid \Sigma, a : K \mid \Sigma, c : A$ |
| Contexts: | $\Gamma$ | ::= | $\cdot \mid \Gamma, x : A$ |

- Typing judgment $\Gamma \vdash^{\Pi} M : A$
- Equality judgment $\Gamma \vdash^{\Pi} M = N : A$
- Canonical forms ( ) $\Gamma \vdash^{\Pi} M \Uparrow A$

## Logical Framework LF

Characteristic rules:

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

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

- Calculus is small
- Type checking and conversion decidable
- Well-typed objects have unique canonical form

## Adequacy theorem

Theorem: For $\Gamma_a = a_1 : \text{i}, \ldots, a_n : \text{i}$

$\Gamma_H = u_1 : \text{hyp}\lceil A_1\rceil, \ldots, u_k : \text{hyp}\lceil A_k\rceil$

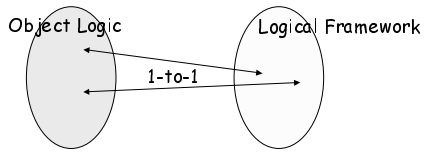
Deductions $\begin{array}{c} D \\ A_1, \ldots, A_n \Rightarrow A \end{array}$ with parameters

among $a_1, \ldots, a_n$ the following holds:

1. $\Gamma_a, \Gamma_H \vdash^{\Pi} M \Uparrow \text{conc}\lceil A\rceil$ iff $M = \lceil D\rceil$
2. $\lceil [t/a]D\rceil = [\lceil t\rceil/a]\lceil D\rceil$ and $\lceil [H/u]D\rceil = [\lceil H\rceil/u]\lceil D\rceil$

## Assessment

Reasoning about the real world
is as good as the encoding is






## Assessment

Proof-checking in object logic
=
Type-checking in LF

It's practical because of type reconstruction
It's elegant, concise, and effective in many cases

Application: Proof-carrying code [Necula 97]



## Our Advantage

Commonly used properties

- Exchange
- Weakening
- Contraction
- Substitution

Directly supported by the
Logical Framework



## For Example: Exchange

Theorem [Exchange]:

If $\ldots, A_i, \ldots, A_j, \ldots \Rightarrow A$ then $\ldots, A_j, \ldots, A_i, \ldots \Rightarrow A$

Proof: by induction.

In LF:

If $\ldots, u_i : \text{hyp } A_i, \ldots, u_j : \text{hyp } A_j, \ldots \vdash^{\Pi} D : \text{conc } A$

then $\ldots, u_j : \text{hyp } A_j, \ldots, u_i : \text{hyp } A_i, \ldots \vdash^{\Pi} D : \text{conc } A$



## For Example: Weakening

Theorem: [Weakening]

If $\ldots \Rightarrow A$ then $\ldots, A' \Rightarrow A$

Proof: by induction.

In LF:

If $\ldots \vdash^{\Pi} D : \text{conc } A$

then $\ldots, u' : \text{hyp } A' \vdash^{\Pi} D : \text{conc } A$



## For Example: Contraction

Theorem: [Contraction]

If $\ldots, A', A', \ldots \Rightarrow A$ then $\ldots, A', \ldots \Rightarrow A$

Proof: by induction.

In LF:

If $\ldots, u_1 : \text{hyp } A', u_2 : \text{hyp } A', \ldots \vdash^{\Pi} D : \text{conc } A$

then $\ldots, u_1 : \text{hyp } A', \ldots \vdash^{\Pi} [u_1 / u_2] D : [u_1 / u_2](\text{conc } A)$

## In General

Use LF substitution property
freely available
and applicable

If $\Gamma \vdash^{\Pi} M : A$

and $\Delta \vdash^{\Pi} \sigma : \Gamma$

then $\Delta \vdash^{\Pi} M[\sigma] : A[\sigma]$

## Applications

Meta-programming via ML (Isabelle)
Constraint logic programming ( Prolog, Elf)
Applications with state: linear framework
Some applications: General constraint solving
Proof carrying code and theorem proving:

- Value of proof terms
- Exist in LF but not HHF

## Summary of Example

$$\frac{}{\Gamma, A \Rightarrow A}\,axiom \qquad \frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \supset B}\,impr$$

$$\frac{\Gamma \Rightarrow A \quad \Gamma, A \supset B, B \Rightarrow C}{\Gamma, A \supset B \Rightarrow C}\,impl \qquad \frac{\Gamma \Rightarrow A[a/x]}{\Gamma \Rightarrow \forall x : A}\,allr$$

$$\frac{\Gamma, \forall x.A, A[t/x] \Rightarrow B}{\Gamma, \forall x.A \Rightarrow B}\,alll \qquad \frac{\Gamma \Rightarrow A \quad \Gamma, A \Rightarrow C}{\Gamma \Rightarrow C}\,cut$$

```
axiom : (hyp A -> conc A).

impr  : (hyp A -> conc B)
          -> conc (A imp B).

impl  : conc A -> (hyp B -> conc C)
          -> (hyp (A imp B) -> conc C).

forallr : ({a:i} conc (A a))
            -> conc (forall A).

foralll : {T:i} (hyp (A T) -> conc C)
            -> (hyp (forall A) -> conc C).

cut : conc A -> (hyp A -> conc C)
        -> conc C.
```

## Summary of Example

- Judgment forms as type families
  $\mathsf{hyp} : o \to \mathsf{type}, \quad \mathsf{conc} : o \to \mathsf{type}$
- Judgments as types
  $\mathsf{hyp}\lceil A \rceil : \mathsf{type}, \quad \mathsf{conc}\lceil A \rceil : \mathsf{type}$
- Derivations as objects
  $\lceil D \rceil : \mathsf{conc}\lceil A \rceil$
- Parametric derivations as function types
  $\lceil D \rceil : \mathsf{hyp}\lceil A \rceil \to \mathsf{conc}\lceil A \rceil$

## Summary of Example

Higher-order abstract syntax:

- Variable renaming
- Substitution
- Occurrence conditions

Representation of hypothetical judgments:

- Substitution properties
- Exchange properties
- Weakening properties
- Contraction properties

## Summary of Example

> We can look at the current field of problem solving by computers as a series of ideas about how to present a problem. If a problem can be cast into one of these representations in a natural way, then it is possible to manipulate it and stand some chance of solving it.
>
> [Allen Newell]

Elegance

- Higher-order representation techniques
- Dependent types

Benefit for this work:

- Variables and substitutions come for free!

## But

LF signatures not inductive

55

## Intermission

| Deductive System | Logical Framework |
|---|---|
| • Logic<br>• Judgments<br>• Inference rules | • Uniform language<br>• Types<br>• Direct encoding as objects |

56

## Goal

Meta-theory of deductive systems
without
inductive definitions

- Representation
- Automation
- Soundness of the design

57

## Four Direct Approaches

Relational meta-theory plus schema-checking
[Rohwedder & Pfenning 96]

Reflection via modality
[Despeyroux, Schürmann, Pfenning 97]

Meta logical framework over LF
[Schürmann 00]

Definitional reflection and induction over logical framework
[McDowell, Miller 97]

58

## Guiding Design Principles

Conservative:
Preservation of LF representation techniques

Natural:
Expressiveness of informal proofs

General:
Many meta-theorems and meta-proof techniques

Automatable:
Efficient automation of meta-proof search

59

## Next: Reasoning

| Reasoning | |
|---|---|
| • Consistency arguments<br>• Theorems about logics<br>• Inductive proofs | |
| **Deductive System** | **Logical Framework** |
| • Logic<br>• Judgments<br>• Inference rules | • Uniform language<br>• Types<br>• Direct encoding as objects |

60

## Axioms and Rules

$$\frac{}{\Gamma, A \Rightarrow^- A}\,axiom$$ [Gentzen 35]

$$\frac{\Gamma, A \Rightarrow^- B}{\Gamma \Rightarrow^- A \supset B}\,impr \qquad \frac{\Gamma, A \supset B \Rightarrow^- A \quad \Gamma, A \supset B, B \Rightarrow^- C}{\Gamma, A \supset B \Rightarrow^- C}\,impl$$

$$\frac{\Gamma \Rightarrow^- A[a/x]}{\Gamma \Rightarrow^- \forall x{:}A}\,allr \qquad \frac{\Gamma, \forall x.A, A[t/x] \Rightarrow^- B}{\Gamma, \forall x.A \Rightarrow^- B}\,alll$$

no cut rule



## A (Not So) Simple Argument

Theorem [Admissibility of Cut]:
If $\mathcal{D} :: \Gamma \Rightarrow^- A$ and $\mathcal{E} :: \Gamma, A \Rightarrow^- C$ then $\mathcal{F} :: \Gamma \Rightarrow^- C$

- Fundamental theorem in Logic [Gentzen 35]
- Consistency of first-order logic
- Structural proof [Pfenning 95]
- Twelf can prove it automatically



## A (Not So) Simple Argument

Proof: by induction on A,(D,E).
Case: $\mathcal{D} :: \Gamma \Rightarrow^- A$ $\quad \mathcal{E} = \dfrac{\mathcal{E}'}{\Gamma, A, B' \Rightarrow^- C'}\,impr$

$\mathcal{D}' :: \Gamma, B' \Rightarrow^- A$ by weakening on $\mathcal{D}$
$\mathcal{F}' :: \Gamma, B' \Rightarrow^- C'$ by induction hypothesis on $\mathcal{D}'$, $\mathcal{E}'$
$\mathcal{F} :: \Gamma \Rightarrow^- B' \supset C'$ by application of *impr* to $\mathcal{F}'$



## Another Case

$\mathcal{D} = \dfrac{\mathcal{D}_1}{\Gamma \Rightarrow^- A' \supset B'}\,impr$ $\quad \mathcal{E} = \dfrac{\mathcal{E}_1 \quad \mathcal{E}_2}{\Gamma, A' \supset B' \Rightarrow^- C'}\,impl$

$\mathcal{E}_1' :: \Gamma \Rightarrow^- A'$ by i.h. on $A'$ $B$, $\mathcal{D}$, $\mathcal{E}_1$
$\mathcal{D}' :: \Gamma, B' \Rightarrow^- A' \supset B'$ by weakening on $\mathcal{D}$
$\mathcal{E}_2' :: \Gamma, B' \Rightarrow^- C'$ by i.h. on $A'$ $B$, $\mathcal{D}'$, $\mathcal{E}_2$
$\mathcal{D}_1' :: \Gamma \Rightarrow^- B'$ by i.h. on $A'$, $\mathcal{E}_1'$, $\mathcal{D}_1$
$\mathcal{F} :: \Gamma \Rightarrow^- C'$ by i.h. on $B'$, $\mathcal{D}_1'$, $\mathcal{E}_2'$



## Next: Meta-logical Framework

| Reasoning | Meta-logical Framework |
|---|---|
| • Consistency arguments<br>• Theorems about logics<br>• Inductive proofs | • Automated proof search<br>• Formulas<br>• Direct encoding as proofs |
| **Deductive System** | **Logical Framework** |
| • Logic<br>• Judgments<br>• Inference rules | • Uniform language<br>• Types<br>• Direct encoding as objects |



## Outline

- Reasoning about non-inductive definitions
- The meta-logical framework Twelf
- Techniques for automation
- Further development (Speculation)

## Attempt 1

Standard Induction Principles
In general not applicable
Because:

LF signatures not inductive

67

## Why?

Closed World Assumption

- Fixed set of constructors
- Existence of induction principles

Example: Natural number induction

$$\frac{\cdot \vdash P(\mathsf{zero}) \quad \cdot \vdash P(\mathsf{succ}\, n)}{\cdot \vdash \forall x : \mathsf{nat}.P(x)} \qquad \frac{\overline{\cdot \vdash P(n)}}{\vdots}$$

68

## But…

*Standard induction principles do not exist*

```
axiom : (hyp A -> conc A).
impr  : (hyp A -> conc B)
          -> conc (A imp B).
impl  : conc A -> (hyp B -> conc C)
          -> (hyp (A imp B) -> conc C).
forallr : ({a:i} conc (A a))
            -> conc (forall A).
forallI : {T:i} (hyp (A T) -> conc C)
            -> (hyp (forall A) -> conc C).
```

Negative occurrences

69

## Thus…

Positivity side condition:

- In many meta-logical frameworks
  - Coq
  - Isabelle
  - PVS
  - …
- Prevents higher-order representation techniques

70

## Attempt 2

Open World Assumption

No induction principles

- Type definitions are open-ended
- New types, new inference rules may be added

Example: Admissibility theorem

- Not stable under extensions of the world

71

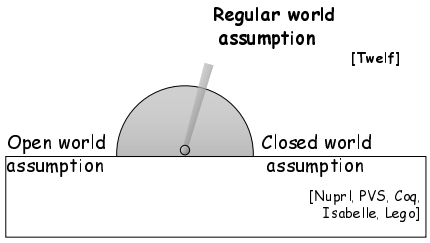
## Attempt 2 (cont'd)

No guarantee that all cases are covered

$$\frac{\Phi \vdash P(\mathsf{axiom}\, H) \quad \Phi \vdash P(\mathsf{impr}\, (\lambda u : \mathsf{hyp}\, B.Du)) \quad \cdots}{\Phi \vdash \forall d : \mathsf{conc}\, A.P(d)} \qquad \frac{\overline{\Phi, \underline{u} : \mathsf{hyp}\, B \vdash P(D\underline{u})}}{\vdots}$$

72

## Solution






## Regular World Assumption

Extensions to the world are predictable!

- Sound induction principle exist
- But it is not standard!

$$\Phi ::= \cdot \mid \Phi, u : \text{hyp } A \mid \Phi, a : i$$

```
axiom : (hyp A -> conc A).
impr  : (hyp A -> conc B)
          -> conc (A imp B).
impl  : conc A -> (hyp B -> conc C)
          -> (hyp (A imp B) -> conc C).
```

```
forallr : ({a:i} conc (A a))
            -> conc (forall A).
foralll : {T:i} (hyp (A T) -> conc C)
            -> (hyp (forall A) -> conc C).
```



## But …

Induction principles difficult to formulate

- Proofs invariant under properties of $\Phi$
- World $\Phi ::= \cdot \mid \Phi, u : \text{hyp } A \mid \Phi, a : i$ satisfies
  - Exchange
  - Weakening
  - Contraction
  - Substitution

Induction principles are not standard



## Attempt 3:

Represent proof as a realizer
Under the
Regular world assumption

- Realizer are total functions
- Mapping input arguments to output arguments
- Construct witnesses for existentials



## Natural number induction

Theorem [Half]: Forall natural number N
there exists a M such that N div 2 = M.

Proof:

```
fun half zero = zero
  | half (succ zero) = zero
  | half (succ (succ N)) =
      let
        M' = half N
      in
        (succ M')
      end
```

half is a realizer:

- Type: nat → nat
- Terminating
- Covers all cases



## Induction over Deductions

Partial Correctness

- How to recurse under a Π-binder?
- Functions are natural!

Termination

- Does a computation terminate?
- What termination orderings to consider?

Coverage

- All cases considered
- How to treat dependent types?

## Type Theory $M_2^+$

Paradigm: Theorems-as-types

Theorem [Admissibility]:

If $\mathcal{D} :: \ldots \Rightarrow^- A$ and $\mathcal{E} :: \ldots, A \Rightarrow^- C$ then $\mathcal{F} :: \ldots \Rightarrow^- C$

$\forall A : o. \forall C : o.$

$\forall D : \text{conc}^- A. \forall E : \text{hyp}\, A \to \text{conc}^- C. \exists F : \text{conc}^- C.\, \text{true}$

## Type Theory $M_2^+$ (cont'd)

Paradigm: Proofs-as-functions

Proof: by induction on A,(D,E).

Case:

$$\mathcal{D} :: \Gamma \Rightarrow^- A \qquad \mathcal{E} = \frac{\mathcal{E}'}{\Gamma, A, B' \Rightarrow^- C'}\, impr \qquad \Phi ::= \cdot \mid \Phi, \underline{u} : \text{hyp}\, A \mid \Phi, \underline{a} : i$$

$\mathcal{D}' :: \Gamma, B' \Rightarrow^- A$

$\mathcal{F}' :: \Gamma, B' \Rightarrow^- C'$

$\mathcal{F} :: \Gamma \Rightarrow^- B' \supset C'$

```
fun ca D ( u:hyp A. impr ( v:hyp B'. E' u v)) =
      let
            new v:hyp B'
            (F' v) = ca D ( u:hyp A. E' u v)
      in
            (impr ( v:hyp B'. F' v))
      end
  | ...
```

## Type Theory $M_2^+$ (cont'd)

$$\mathcal{D} = \frac{\mathcal{D}_1}{\Gamma \Rightarrow^- A} \qquad \mathcal{E}_i \qquad \Phi ::= \cdot \mid \Phi, \underline{u} : \text{hyp}\, A \mid \Phi, \underline{a} : i$$

$\mathcal{E}_1' :: \Gamma \Rightarrow^-$

$\mathcal{D}' :: \Gamma, B' \Rightarrow$

$\mathcal{E}_2' :: \Gamma, B' \Rightarrow$

$\mathcal{D}_1' :: \Gamma \Rightarrow^-$

$\mathcal{F} :: \Gamma \Rightarrow^-$

```
| ca (impr ( u:hyp A'. D1 u))
     ( v:hyp A'  B'. impl (Ei v) ( w:hyp B'. E2 v w)) =
       let
            E1'=ca (impr D1) Ei
            new w:hyp B'
              (E2' w) = ca (impr D1) (v:hyp A'  B'. E2 v w))
            D1'=ca Ei' ( u:hyp A'. D1 u)
            F=ca D1' ( w:hyp B'. E2' w)
       in
            F
       end
  | ...
```

## Type Theory $M_2^+$ (cont'd)

Direct respresentation of

- Case analysis
- Recursion
- Elementary rule construction

Not necessary to represent

- Exchange
- Weakening
- Substitution

## Meta Logic $M_2^+$

Functions are realizers:

- Termination: Computation must terminate
- Coverage: Computation must not get stuck

Realizer allow logical interpretation

Theorems-as-formulas
Proofs-as-realizers

## Meta Logic $M_2^+$ (cont'd)

Design decisions.

- Trade off generality vs. automation.
- Exploit LF as much as possible.
- Restrict meta logic to the simplest fragment.

## Meta Logic $M_2^+$ (cont'd)

Formulas: ( $_2$-fragment)

$$F ::= \forall x : A.F \mid \exists x : A.F \mid F_1 \wedge F_2 \mid \text{true}$$

Semantics:

$$\begin{array}{lll} \Phi \models \forall x : A.F & \text{iff} & \Phi \models F[M/x] \text{ for all } M, \text{ s.t. } \Phi \vdash_{LF} M : A \\ \Phi \models \exists x : A.F & \text{iff} & \Phi \models F[M/x] \text{ for some } M, \text{ s.t. } \Phi \vdash_{LF} M : A \\ \Phi \models F_1 \wedge F_2 & \text{iff} & \Phi \models F_1 \text{ and } \Phi \models F_2 \\ \Phi \models \text{true} & & \end{array}$$

## Meta Logic $M_2^+$ (cont'd)

- Falsehood
  $\text{void} : \text{type}$
- Negation of A(x)
  $A(x) \to \text{void}$
- Disjunction A(x) or A(x)

$$\text{disj} : i \to \text{type}$$
$$\text{injl} : \Pi x{:}i.A(x) \to \text{disj}\ x$$
$$\text{injr} : \Pi x{:}i.(A(x) \to \text{void}) \to \text{disj}\ x$$

## Meta Logic $M_2^+$ (cont'd)

Meta-programs:

$$P ::= \text{fn x} \Rightarrow P \mid \text{inx } P \mid \text{unit} \mid \text{let } D \text{ in } P \mid \text{rec } x \in F.P \mid \text{case } \psi \text{ of } \Omega \mid \ldots$$

Meta-declarations:

$$D ::= \cdot \mid \text{new } (\underline{x_1}{:}A_1,\ldots,\underline{x_n}{:}A_n)^L, D \mid (x = P) \text{ in } D \mid \boldsymbol{x} = \boldsymbol{y}\ M, D \mid \ldots$$

Meta-cases:

$$\Omega ::= \cdot \mid (\Psi \bullet \psi \Rightarrow P \mid \Omega)$$

## Meta Logic $M_2^+$ (cont'd)

Proof calculus for $M_2^+$:

Judgments:

Provability (programs): $\Psi \vdash P \in F$

Provability (declarations): $\Psi \vdash D \in \Psi'$

Empty — Assumptions

$$\Psi ::= \cdot \mid \Psi, (\underline{x_1} : A_1,\ldots,\underline{x_n} : A_n)^L \mid \Psi, x : A \mid \Psi, \boldsymbol{x} \in F$$

Regular world extensions — Meta assumptions

## Meta Logic $M_2^+$ (cont'd)

$$\frac{\Psi, x : A \vdash P \in F}{\Psi \vdash (\text{fn x} \Rightarrow P) \in \forall x : A.F} \qquad \frac{\Psi \vdash M : A \quad \Psi \vdash P \in F[M/x]}{\Psi \vdash (\text{inx } M\ P) \in \exists x : A.F}$$

$$\frac{}{\Psi \vdash \text{unit} \in \text{true}} \qquad \frac{\Psi \vdash D : \Psi' \quad \Psi, \Psi' \vdash P \in F}{\Psi \vdash (\text{let } D \text{ in } P) \in F}$$

$$\frac{}{\Psi \vdash \ldots} \qquad \frac{\Psi \vdash P \in \exists x : A.F \quad \Psi \vdash D \in \Psi'}{\Psi \vdash \ldots}$$

$$\frac{}{\Psi, (\underline{x_1} : A_1,\ldots,\underline{x_n} : A_n)^L \vdash D \in \Psi'} \qquad \text{where}$$

$$\Psi \vdash \text{new } (\underline{x_1} : A_1,\ldots,\underline{x_n} : A_n)^L, D \in \Psi'' \qquad \Psi'' = \text{abstract}(\underline{x_1} : A_1,\ldots,\underline{x_n} : A_n, \Psi')$$



And others...

## Meta Logic $M_2^+$ (cont'd)

Recursion:

$$\frac{}{\Psi, \boldsymbol{x} \in F \vdash P \in F}$$

P terminates in x

Case Analysis:

$$\frac{}{\Psi \vdash \psi \in \Psi' \quad \Psi' \vdash \Omega \in F}$$

covers all cases

## Meta Logic $M_2^+$ (cont'd)

Definition: Operational semantics: $\Phi \vdash P \Rightarrow V$

Property 1: Type preservation
- Return values have expected type

Property 2: Termination
- Simultaneous, lexicographic extensions of higher-order subterm ordering

Property 3: Coverage
- Higher-order pattern unification

Corollary: All proof terms are total



## Meta Logic $M_2^+$ (cont'd)

Theorem [Soundness of $M_2^+$]

If $\Phi \vdash P \in F$ then $\Phi \models F$

Proof: via a realizability interpretation.



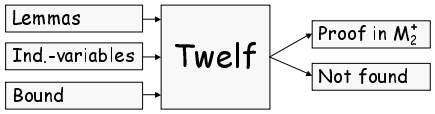
## Twelf Implementation

Implements a theorem prover for $M_2^+$

Success due to *regular world assumption*

Automated proof search

No tactics (yet)






## Twelf Implementation (cont'd)

Theorem [Admissibility]:

If $\ldots \Rightarrow^- A$ (D) and $\ldots, A \Rightarrow^- C$ (E) then $\ldots \Rightarrow^- C$ (F)

```
%theorem ca :  forallG (some {A:o} pi {H:hyp A} | pi {a:I})
    forall* {C:o}
    forall {A:o} {D:conc- A} {E: (hyp A -> conc- C)}
    exists {F: conc- C}
    true.

%prove 4 {A [D E]} (ca A D E _).
```



## Twelf Implementation (cont'd)

Splitting (Coverage)
- Case analysis over LF objects
- *Regular world assumption*

Recursion (Termination)
- Induction hypotheses
- *Regular world assumption*

Filling (Type-checking)
- Applies LF theorem prover
- *Or theorem prover for underlying logic*



## Experiments

| Problem | | Total time | Reason |
|---|---|---|---|
| Cut-Elimination IL | Admissibility of Cut | 6min 35sec | |
| | Cut-elimination | 0.28sec | |
| ND - Sequent | ND -> Sequent | 0.11sec | |
| | Sequent -> ND | 0.12sec | |
| ND - Hilbert | Deduction theorem | 0.12sec | |
| | Translation theorem | 0.37sec | |
| | | | |
| | | | |
| | | | |

Machine: Pentium II, 400Mhz, 192MB RAM, Linux 2.0

## Experiments (cont'd)

| Problem | | Total time | Reason |
|---|---|---|---|
| Mini-ML | Value-soundness | 0.13sec | |
| | Type preservation | 0.42sec | |
| | Reduction theorem | 0.66sec | |
| (app/lam) | Uniqueness of typing | 0.25sec | |
| Compiler (CPM) | Soundness | not yet. | Compl. ind. |
| | Completeness | 0.31sec | |
| (both directions) | Proof equivalence | 0.46sec | |
| CCC | Translation lambda | 3.39sec | |
| | Distributivity | not yet. | LF Prover |

Machine: Pentium II, 400Mhz, 192MB RAM, Linux 2.0

## Experiments (cont'd)

| Problem | | Total time | Reason |
|---|---|---|---|
| Church-Rosser | Append lemma | 0.08sec | |
| | Substitution lemma | 0.18sec | |
| | Diamond lemma | 5.6sec | |
| | Strip lemma | 3min 58sec | |
| | Confluence lemma | 28.52sec | |
| | Church-Rosser thm | 2.05sec | |
| | | | |
| | | | |

Machine: Pentium II, 400Mhz, 192MB RAM, Linux 2.0

## Experiments (cont'd)

| Problem | | Total time | Reason |
|---|---|---|---|
| LP (Harrop) | Soundness (Uni) | 0.31sec | |
| | Canonical forms | 0.34sec | |
| | Completeness (Uni) | 0.28sec | |
| | Soundness (Res) | 1.05sec | |
| | Completeness (Res) | 0.52sec | |
| Kolmogorov CL->IL | Soundness | 9.55.sec | |
| | Completeness | not yet | LF Prover |
| Rippling | Equivalence lemma | 0.65sec | |
| | Skeleton preservatio | 0.94sec | |

Machine: Pentium II, 400Mhz, 192MB RAM, Linux 2.0

## Assessment

Requires
- Induction order
- Search depth

Plus
- Derives its power from two level system
- Excellent, if proof is known ahead of time

## Assessment (cont'd)

Minus
- Not robust with respect to failure
- Naïve strategy
- Filling bottle neck
- Not enough termination/reduction orders

## Future work

- Constraints domains: integer, boolean, rational. (Currently only in operational semantics of Twelf) [Virga '99]
- Linearity (reasoning about state) [Cervesato & Pfenning '97]
- Order (reasoning about sequencing) [Polakow and Pfenning '99]
- Proof compression [Necula '98]
- Compilation [Nadathur '99]

## Related Work: FOLDN

[McDowell, Miller 97]

- Logical framework flexible (HHF, linear HHF)
- Meta-logic richer, less automation (present)
- Induction over natural numbers
- Definitional reflection
- Does not inherit HHF theorem proving
- Does not inherit reasoning about worlds

## Related Work: 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 logical framework.
- Only inductive encodings and reasoning
  - No higher-order abstract syntax
  - No hypothetical or parametric meta-reasoning
- Less automation, different "look & feel"
- Numerous experiments

## Related Work: Modality

Modal $\mu$-calculus [Schürmann, Despeyroux, Pfenning 97]

Categorical model [Hofmann 99]

- Collapse two level design into one level
- Case analysis permitted only over closed terms
- Closed terms are of modal type
- Few experiments

## Summary

Presented principles underlying LF
- Higher-order abstract syntax
- Jugdments as types
- Hypothetical and parametric judgments

Explored design of Twelf
- Regular world assumption
- Proofs as realizers

Sketched automation techniques in Twelf

## Conclusion

For
more information on the
Twelf project
visit our website

http://www.twelf.org