Mechanizing the Metatheory of LF in Twelf

Chris Martens Karl Crary

Carnegie Mellon University

August 11, 2011

Presented at ITU Denmark

Twelf and LF

Twelf is a proof assistant.

- Proof-carrying code [1]
- SeLF/Gray (distributed security) [2]
- Metatheory of ML [3]

LF: Twelf's underlying (dependent) type theory

[1] George C. Necula. Compiling with Proofs. PhD thesis, School of Computer Science, Carnegie Mellon Univ., Sept. 1998.

[2] http://www.cs.cmu.edu/ self/

[3] http://www.cs.cmu.edu/ dklee/tslf/

Mechanizing the Metatheory of LF in Twelf

A proof about a proof assistant: LF is “correct”.

This project is about...

- Reasoning **about** dependent types: syntactic (as opposed to LR) approach
- Reasoning **with** dependent types: proof engineering

LF Methodology: Syntax as Terms

Metalanguage and object language
Example OL: STLC

$$
\begin{array}{lcl}
e & ::= & \lambda x{:}\tau.e \mid (e\ e) \mid x \\
\tau & ::= & o \mid \tau \rightarrow \tau
\end{array}
$$

```
exp : type.  tp : type.
lam : tp -> (exp -> exp) -> exp.
app : exp -> exp -> exp.
o : tp.
arr : tp -> tp -> tp.
```

LF Methodology: Judgments as Types

$$\boxed{\Gamma \vdash e : \tau}$$

```
of : exp -> tp -> type.
```

$$\frac{e_1 : \tau' \rightarrow \tau \quad e_2 : \tau'}{(e_1\ e_2) : \tau}\ \mathit{of/app}$$

```
of/app : of E1 (arrow T' T)
          -> of E2 T'
          -> of (app E1 E2) T.
```

Twelf Methodology

Theorems = **total** relations over **derivations**

```
preservation : of E T -> step E E' -> of E' T -> type.
%mode preservation +X1 +X2 -X3.

%% proof goes here...

%worlds () (preservation _ _ _).
%total D (preservation _ D _).
```

Relation on a typing derivation, a stepping derivation, and another typing derivation.
Twelf directives: **%mode**, **%worlds**, **%total**

LF

Dependent types!

$$\frac{x : A \vdash M : B}{\lambda x{:}A.\, M : \Pi x{:}A.\, B} \qquad \frac{M : \Pi x{:}A.\, B \qquad N : A}{(M\ N) : [N/x]B}$$

```
of : tm -> tp -> type
of (lam o [x] x) (arr o o)
```

type family indexed by a tm and tp

dependent type inhabited by derivations

LF

$$
\begin{array}{rcl}
\text{Objects } M & ::= & c \mid x \mid \lambda x{:}A.\, M \mid (M\ M) \\
\text{Families } A & ::= & a \mid \Pi x{:}A.\, A \mid \lambda x{:}A.\, B \mid (A\ M) \\
\text{Kinds } K & ::= & \mathsf{type} \mid \Pi x{:}A.\, K \\
\text{Contexts } \Gamma & ::= & \cdot \mid \Gamma, x : A \\
\text{Signatures } \Sigma & ::= & \cdot \mid \Sigma, c : A \mid \Sigma, a : K
\end{array}
$$

Using LF

The user populates a signature Σ, which is checked:

$$\frac{\Sigma\ \mathsf{ok} \qquad \Sigma; \cdot \vdash A : \mathsf{type}}{(\Sigma, c : A)\ \mathsf{ok}}$$

$$\frac{\Sigma\ \mathsf{ok} \qquad \Sigma; \cdot \vdash K\ \mathsf{wf}}{(\Sigma, a : K)\ \mathsf{ok}}$$

LF typechecking

Type formation depends on term typing.
Key rule:

$$\frac{M : A \qquad A \equiv B : \mathsf{type}}{M : B}$$

LF typechecking

$\boxed{\Gamma \vdash M \equiv N : A}$

$\beta - \eta$ equivalence:

$$\frac{}{((\lambda x{:}A.\ M)\ N) \equiv [N/x]M : [N/x]B}\ beta$$

$$\frac{x : A \vdash (M\ x) \equiv (N\ x) : B}{M \equiv N : \Pi x{:}A.\ B}\ ext$$

Adequacy

LF's notion of *correctness of an encoding*:

Terms of the OL are in bijective, compositional correspondence with **canonical** LF terms of type `tm`.

Canonical $=$ β short, η long

The theorems

- Decidability of type checking
- Existence of canonical forms

Prior work

Most relevant:

- Detailed paper proof using logical relations [4]
- Formalization of that proof in Isabelle [5]
- Crary's proof about the singleton calculus [6]

[4] Robert Harper and Frank Pfenning. On equivalence and canonical forms in the LF type theory. TOCL 6:61-101, January 2005.

[5] Christian Urban et. al. Mechanizing the Metatheory of LF. TOCL 12.2, January 2011.

[6] Karl Crary. A syntactic account of singleton types via hereditary substitution. LFMTP '09.

Our approach

Proof sketch:

- Define a system in which equivalence is syntactic
- Define a translation to that system
- Show the translation sound and complete

The Canonical Forms Presentation

Syntactic separation of terms (due to Felty [7]):

$$
\begin{array}{rcl}
\text{Terms } M & ::= & \lambda x.\, M \mid R \\
\text{Atoms } R & ::= & c \mid x \mid (R\ M) \\
\text{Families } A & ::= & \Pi x{:}A.\, A \mid \lambda x{:}A.\, B \mid P \\
\text{Atomic Families } P & ::= & a \mid (P\ M)
\end{array}
$$

[7] Amy Felty. Encoding dependent types in intuitionistic logic. In Gerard Huet and Gordon D. Plotkin, editors, Logical Frameworks, pages 214-251. Cambridge University Press, 1991.

The Canonical Forms Presentation

Bidirectional typechecking:

Canonical Typing

$\Gamma^+ \vdash R^+ \Rightarrow A^-$	R synthesizes type A
$\Gamma^+ \vdash M^+ \Leftarrow A^+$	M checks at type A

β-shortness: syntactic
η-length: enforced by typing

$$\frac{R \Rightarrow P}{R \Leftarrow P}$$

(P is a base type)

Hereditary Substitution

Hereditary substitution [8] maintains canonical forms:
To substitute N into $(R\ M)$:

- Let $M' = [N/x]M$ and $R' = [N/x]R$
- If $R' = \lambda x.\, O$, recursively do $[M'/x]O$
- Otherwise, $(R'\ M')$

[8] Watkins, Pfenning, and Walker. A concurrent logical framework I: Judgments and properties. Tech. rept. CMU-CS- 02-101. Department of Computer Science, Carnegie Mellon University. Revised May 2003.

Expansion

To turn an arbitrary atom into a term, expand at a simple type.

$$\begin{aligned} \eta_o(R) &= R \\ \eta_{S \to T}(R) &= \lambda x.\, \eta_T(R\ \eta_S(x)) \end{aligned}$$

Canonical LF Metatheory

- **Substitution**: Given $x : A \vdash M \Leftarrow B$ and $N \Leftarrow A$, the hereditary substitution $[N/x]M$ exists and $[N/x]M \Leftarrow B$
- **Identity**: Given $R \Rightarrow A$, the expansion $\eta_{simp(A)}(R)$ exists and $\eta_{simp(A)}(R) \Leftarrow A$.

Canonical LF Metatheory

Substitution proof

Key lemma: permuting substitutions
$[N/x][M/y]O = [[N/x]M/y][N/x]O$
Termination metric: The **simple type** of the variable and the **typing derivation** of the open term.
The type in the metric is needed for the hereditary case.

Identity proof

Straightforward induction over the expansion derivation.

Stop here?

Why not stop here?
Hereditary substitution eliminates all need for noncanonical forms...

Translation

"EL" = LF with definitional equivalence
"IL" = Canonical forms LF

$$\boxed{\Gamma \vdash M \rightsquigarrow \overline{M} : A}$$

- η-expands constants and variables
- Translates each piece of application and uses hsub
- Transliterates the rest (type output needed for λ)

Proof engineering sidebar

$x \leadsto x$ on paper, but in LF we have to hypothesize EL var, IL var, and a translation between them.
Maintained in *blocks*, which further grow to carry type translations.

```
%block cbind
  : some {EA:etp} {A:tp} {d_tptrans:tptrans EA A ktype}
    block {x:atm} {d:atof x A}
          {ex:etm} {ed:eof ex EA}
          {xt:vtrans ex x}.
```

Translation and Family-level lambda

The target of term translation is canonical terms.
How should a family $A : \Pi x{:}B.K$ translate?
Our approach: add λ at the family level; treat objects and families uniformly.
Perhaps a simpler avenue: translate to a disjunctive class ($P + A$).

Correctness of the Translation

Completeness

If $M \equiv N : A$, then
$M \rightsquigarrow Q : \overline{A}$ and $N \rightsquigarrow Q : \overline{A}$

Soundness

If $M \rightsquigarrow Q : A$, $N \rightsquigarrow Q : A$ and $\widehat{A} \rightsquigarrow A : \mathrm{type}$
Then $M \equiv N : \widehat{A}$

Completeness

Completeness

If $M \equiv N : A$, then
$M \rightsquigarrow Q : \overline{A}$ and $N \rightsquigarrow Q : \overline{A}$

Proof: By induction over the structure of $M \equiv N : A$.
Hard cases: beta and ext (as expected)
Key lemma: Permutability of translation and substitution.

Soundness

Soundness

If $M \leadsto Q : A$, $N \leadsto Q : A$ and $\widehat{A} \leadsto A : \text{type}$
Then $M \equiv N : \widehat{A}$

Proven via "transliteration": $\boxed{\Gamma \vdash M : A \mapsto \widehat{M}}$
Approximately id; needs the type for λ
Theorem: the transliterated translation of M is $\equiv M$.

Soundness

Key lemma:

Transliteration "almost" permutes with substitution

If $x : A \vdash (M : B) \mapsto \widehat{M}$
and $(N : A) \mapsto \widehat{N}$
then $[\widehat{N}/x]\widehat{M} \equiv \widehat{[N/x]M : A}$

This held us up for about a year.

Solution: syntactic reduction approach.

$$\boxed{\Gamma \vdash M \longrightarrow N}$$

Transliteration "almost" permutes with substitution

If $x : A \vdash (M : B) \mapsto \widehat{M}$
and $(N : A) \mapsto \widehat{N}$
then $[\widehat{N}/x]\widehat{M} \longrightarrow^* \widehat{[N/x]M}$

Relating reduction and equivalence

If $M \longrightarrow M'$ and $M : A$ then $M \equiv M' : A$.

Relating reduction and equivalence

Relating reduction and equivalence

If $M \longrightarrow M'$ and $M : A$ then $M \equiv M' : A$.

Must be proved simultaneously with inversion of λ typing

- reduce-equiv uses λ inversion in the β case
- λ inversion uses reduce-equiv in the ext case

$$\frac{x : A' \vdash ((\lambda x{:}A.\, M)\ x) : B}{\lambda x{:}A.\, M : \Pi x{:}A'.B}\ \mathit{ext}$$

- Both cases need Π **injectivity**

Relating reduction and equivalence

Π injectivity

If $\Pi x{:}A.\, B \equiv \Pi x{:}A'.\, B'$: type then
$A \equiv A'$: type and $x : A \vdash B \equiv B'$: type

Prior approaches [9] based on logical relations.
Our approach:

Generalized Π injectivity

If $A \equiv B : K$ and $A \downarrow A'$,
then $B \downarrow B'$ and $A' \sim B' : K$.

[9] Harper-Pfenning;
Karl Crary and Joe Vanderwaart. A Simplified Account of the Metatheory of Linear LF. April 17, 2002. CMU-CS-01-154.

Pi Injectivity

Generalized Π injectivity

If $A \equiv B : K$ and $A \downarrow A'$,
then $B \downarrow B'$ and $A' \sim B' : K$.

$\boxed{A \downarrow A'}$: Syntactic normalization of **family-level** reduction (maintains terms)

$\boxed{A \sim B : K}$: A stronger notion of equivalence; implies injectivity

Family normalization

$$\frac{}{\Pi x{:}A.B \downarrow \Pi x{:}A.B}\ \textit{norm/pi}$$

$$\frac{x \vdash B \downarrow B'}{\lambda x{:}A.B \downarrow \lambda x{:}A.B'}\ \textit{norm/lam}$$

$$\frac{B \downarrow \lambda x{:}A.C}{(B\ M) \downarrow [M/x]C}\ \textit{norm/app}$$

Similarity

$$\frac{A \equiv A' : \mathsf{type} \qquad x : A \vdash B \equiv B' : \mathsf{type}}{\Pi x{:}A.B \sim \Pi x{:}A'.B' : \mathsf{type}}\ \mathit{sim/pi}$$

$$\frac{x : C \vdash B \sim B' : K}{\lambda x{:}A.B \sim \lambda x{:}A'.B' : \Pi x{:}C.K}\ \mathit{sim/lam}$$

Pi Injectivity

Generalized Π injectivity

If $A \equiv B : K$ and $A \downarrow A'$,
then $B \downarrow B'$ and $A' \sim B' : K$.

Key: Separating family-level from term-level computation

Recap

- Syntactic proof of LF's metatheory using hereditary substitution
- Fully mechanized
- Novel (AFAWK) approach to Π injectivity for LF with η expansion and family-level λ

Proof engineering challenges

Explicit Contexts [10]

Need to define a copy of the language in certain cases to avoid dependencies on the ambient context (example follows)

[10] Karl Crary. Explicit Contexts in LF. LFMTP 2009.

Explicit contexts

λ case of substitution with dependent types:

$$\frac{y \vdash [N/x]M_{xy} = M'_y}{[N/x](\lambda y.M_{xy}) = \lambda y.M'_y}\ \mathit{sub/lam}$$

$$\frac{y : A \vdash M_y : B_y}{\lambda y.M_y : \Pi y{:}A.B_y}\ \mathit{of/lam}$$

Explicit contexts

λ case of substitution with dependent types:

$$\frac{y \vdash [N/x]M_{xy} = M'_y}{[N/x](\lambda y.M_{xy}) = \lambda y.M'_y}\ sub/lam$$

$$\frac{x : C, y : A_x \vdash M_{xy} : B_{xy}}{x : C \vdash \lambda y.M_{xy} : \Pi y{:}A_x.B_{xy}}\ of/lam$$

By induction we want $y : A_x \vdash M'_y : B'_y$

In Twelf...

```
- : subst
      (sub/lam ([y] DsubM y : sub ([x] M x y) N (M' y)))
      ([x] [d:of x C]
        of/lam
        ([y] [e:of y (A x)]
          DofM x d y e : of (M x y) (B x y)))
      [...]
      XXX
   <- ({y} {d: of y (A x)}  % x not bound!
        subst (DsubM y) (DofM y) [...] (DofM' y))
```

Proof engineering challenges

Explicit Contexts

```
ctx : type. [...]
ofe : ctx -> tm -> tp -> type.
```

Can shift between `ofe nil M A` and `of M A`.
Unfortunately, significant overhead, and no good way to "librarify".

Proof engineering challenges

Manual equality reasoning

```
tm-eq : tm -> tm -> type.
tm-eq/i : tm-eq M M.
```

For every type family (no polymorphism)...

- plus congruence, compatibility, and respects lemmas
- Library?
- Automation?

Proof engineering challenges

No internalization of functionality

Effectiveness and **uniqueness** lemmas abound

Proof Engineering

Joys:

- Type reconstruction: thinking is plan B
- Visible derivation manipulation
- HOAS and blocks

Regrets:

- Well-formedness checks in inference rules
- Adding family-level λ (mixed blessing)

Proof Engineering

- 36503 lines of (heavily spaced, annotated) Twelf
  - about 14 KLOC for Canonical LF
  - about 13 KLOC just on explicit contexts
- Checks in a few seconds
- On my webpage
  `http://www.cs.cmu.edu/~cmartens`

Future Work

- Formalize lack of need for (or do away with) family λ
- Intrinsic encoding[11]
- Polymorphism; other extensions to LF

[11] Frank Pfenning. Church and Curry: Combining intrinsic and extrinsic typing. In C.Benzmller, C.Brown, J.Siekmann, and R.Statman, editors, Reasoning in Simple Type Theory: Festschrift in Honor of Peter B. Andrews on His 70th Birthday, Studies in Logic 17, pages 303-338. College Publications, 2008

Current interests

- Logic programming and dependent types in other settings
- Two-level logics e.g. Abella
- Substructural/polarized logic programming