Refining Mechanized Metatheory: Subtyping for LF

William Lovas
(with Frank Pfenning)

LF: a framework for defining logics

(Harper, Honsell, and Plotkin, 1987, 1993)

- Dependently-typed lambda calculus
- Encode deductive systems and metatheory, in a machine-checkable way
  - e.g. a programming language and its type safety theorem
- Guiding principle: “judgements as types”

Judgements as types

On paper:

- Syntax
  - e ::= …
- Judgement
  - $\Gamma \vdash e : \tau$
- Deduction
  - $\mathcal{D} :: \Gamma \vdash e : \tau$
- Proof checking

In LF:

- Simple type
  - exp : **type.**
- Type family
  - of : exp → tp → **type.**
- Well-typed term
  - M : of E T
- Type checking

Inclusion as subtyping

- Some judgements have a natural notion of inclusion
  - all *values* are *expressions*
  - all *odd natural numbers* are *positive*
- More interesting types means more interesting judgements!

Example: natural numbers

nat : type.
z : nat.
s : nat $\rightarrow$ nat.

double : nat $\rightarrow$ nat $\rightarrow$ type. *% plus rules*

even : nat $\rightarrow$ type. *% plus some rules…*
odd : nat $\rightarrow$ type. *% plus some rules…*

dbl−even : ΠX:nat. ΠY:nat. *% plus cases*
double X Y $\rightarrow$ even Y $\rightarrow$ type.

Example: nats using refinements

nat : type. $\quad$ even $\leq$ nat. $\quad$ odd $\leq$ nat.
z : even.
s : even $\rightarrow$ odd $\wedge$ odd $\rightarrow$ even.

double : nat $\rightarrow$ even $\rightarrow$ type. *% plus rules*

even : nat $\rightarrow$ type. *% plus some rules…*
odd : nat $\rightarrow$ type. *% plus some rules…*

dbl−even : ΠX:nat. ΠY:nat. *% plus cases*
double X Y $\rightarrow$ even Y $\rightarrow$ type.

metatheorem checking problem
$\rightsquigarrow$ typechecking problem!

Example: nats using refinements

nat : type. even $\leq$ nat. odd $\leq$ nat.

z : even.

s : even $\longrightarrow$ odd $\wedge$ odd $\longrightarrow$ even.

double : nat $\longrightarrow$ even $\longrightarrow$ type.

dbl–z : double z z.

dbl–s : ΠX:nat. ΠY:even.
    double X Y $\longrightarrow$ double (s X) (s (s Y)).

Example: the lambda calculus

- Intrinsic values encoding:

  exp : type.

  val : type. val ≤ exp.

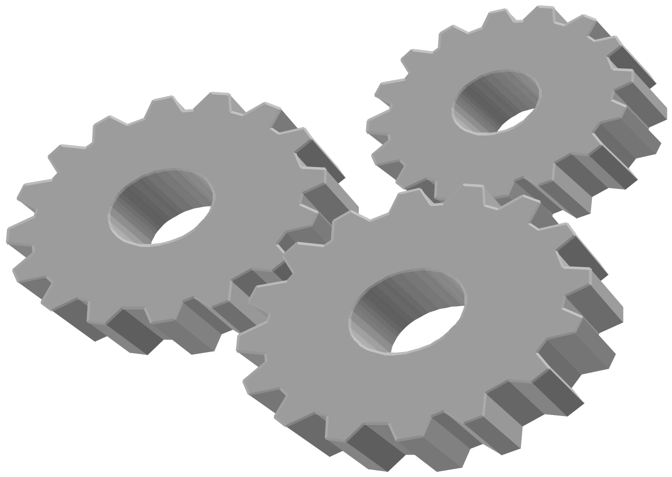
  lam : (val → exp) → val.

  app : exp → exp → exp.

  $: val → exp. *% not needed*

  lam (λx. app x x) : val

Technology



Adequacy

- Does this really mean what I think it means?

- Strategy: exhibit a *compositional bijection* between *mathematical objects* and *canonical forms* following *judgements as types*.
  - *"canonical forms"* are β-normal and η-long.

Canonical forms method

- Represent *only* the canonical forms.
  - β-normal: syntactically
  - η-long: through typing
  - Hereditary substitutions contract redexes
- Simplifies metatheory, emphasizes adequacy
- Concurrent LF (Watkins, et al, 2003)

LF typing

- Bidirectional typing

- Synthesis: $\Gamma \vdash R \Rightarrow A$
  - elims: $R ::= x \mid c \mid R\ N$
- Checking: $\Gamma \vdash N \Leftarrow A$
  - intros: $N ::= R \mid \lambda x.\ N$

Checking

$$\boxed{\Gamma \vdash N \Leftarrow A}$$

- Key rule:
  - base type, so atoms fully applied
  - the only appeal to type equality

$$\frac{\Gamma \vdash R \Rightarrow P' \qquad P' = P}{\Gamma \vdash R \Leftarrow P}$$

Checking… with subtyping!

$$\boxed{\Gamma \vdash N \Leftarrow A}$$

- Easy to adapt!
  - just change equality to subtyping
  - subtyping… only at base type?

$$\frac{\Gamma \vdash R \Rightarrow P' \qquad P' \leq P}{\Gamma \vdash R \Leftarrow P}$$

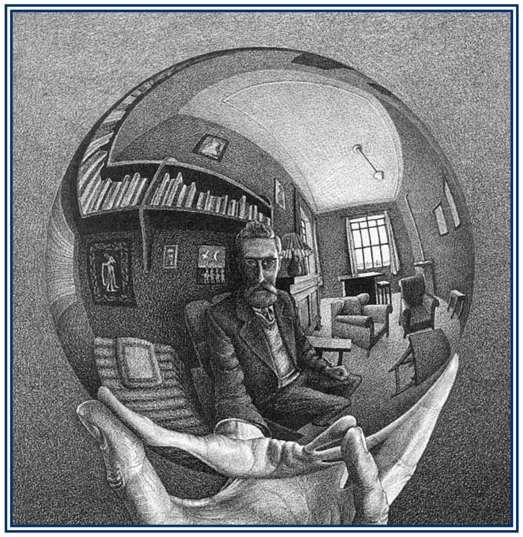
Intersections

- Kind of like pairs, but the terms don't change

$$\frac{\Gamma \vdash N \Leftarrow A_1 \qquad \Gamma \vdash N \Leftarrow A_2}{\Gamma \vdash N \Leftarrow A_1 \wedge A_2} \qquad \frac{}{\Gamma \vdash N \Leftarrow \top}$$

$$\frac{\Gamma \vdash R \Rightarrow A_1 \wedge A_2}{\Gamma \vdash R \Rightarrow A_1} \qquad \frac{\Gamma \vdash R \Rightarrow A_1 \wedge A_2}{\Gamma \vdash R \Rightarrow A_2}$$

Metatheory



LF(R) as a logic

- Entailment should be reflexive:
  - $A \vdash A$
- and transitive:
  - if $A \vdash B$ and $B \vdash C$, then $A \vdash C$

LF(R) as a logic

- Assume x is a proof of A. Is x a proof of A?
  - not necessarily!

    ✗ $x : A_1 \to A_2 \nvdash x \Leftarrow A_1 \to A_2$
  - have to η-expand:

    ✓ $x : A_1 \to A_2 \vdash \lambda y.\ x\ y \Leftarrow A_1 \to A_2$

LF(R) as a logic

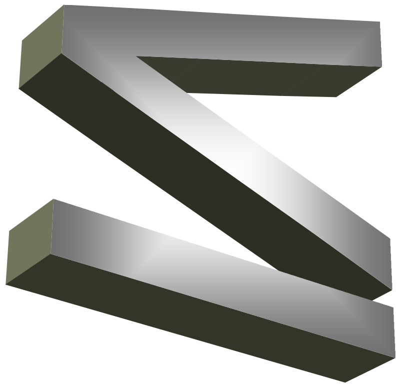
- Assume x is a proof of A. Can $M \Leftarrow A$ stand in for x?
  - if substitution is hereditary?
    - ? $[M/x]_A\ N$ not obviously defined…

Important principles

- **Substitution**
  if $\Gamma, x{:}A \vdash N \Leftarrow B$ and $\Gamma \vdash M \Leftarrow A$,
  then $\Gamma \vdash [M/x]_A\, N \Leftarrow [M/x]_A\, B$.
- **Identity**
  for all A, $\Gamma, x{:}A \vdash \eta_A(x) \Leftarrow A$.

- “Substitution” morally a normalization proof

More about subtyping



Subtyping

$\boxed{\Gamma \vdash N \Leftarrow A}$

- Key rule:

$$\frac{\Gamma \vdash R \Rightarrow P' \qquad P' \leq P}{\Gamma \vdash R \Leftarrow P}$$

- Bidirectional: subtyping only at mode switch
- Canonical: mode switch only at base type

Subtyping at higher types?

- What happened to the structural rules? E.g.,

$$\frac{A_2 \leq A_1 \qquad B_1 \leq B_2}{A_1 \to B_1 \;\leq\; A_2 \to B_2}$$

- Distributivity?

$$\frac{}{(A \to B_1) \wedge (A \to B_2) \;\leq\; A \to (B_1 \wedge B_2)}$$

Subtyping at higher types!

- Intrinsic subtyping: if $A \leq B$ and $\Gamma \vdash N \Leftarrow A$, then $\Gamma \vdash N \Leftarrow B$.
- Equivalently: if $A \leq B$ then $x{:}A \vdash \eta_A(x) \Leftarrow B$.
  - Just like the Identity principle!
  - … also the Substitution principle…
- Usual rules are all *sound* in this sense.

Subtyping at higher types!?

- … and also *complete!*
- **Theorem**: if $x{:}\mathrm{A} \vdash \eta_{\mathrm{A}}(x) \Leftarrow \mathrm{B}$ then $\mathrm{A} \leq \mathrm{B}$.
- **Also**: if $\Gamma \vdash \mathrm{N} \Leftarrow \mathrm{A}$ implies $\Gamma \vdash \mathrm{N} \Leftarrow \mathrm{B}$, then $\mathrm{A} \leq \mathrm{B}$.
- There are no new subtyping principles.

Future work

Two directions:

1. Extend LFR with Twelf stuff
    - type reconstruction
    - unification
    - proof search
2. Extend LFR with CLF stuff
    - more type constructors
    - subtyping with linearity?

Summary

- Refinement types are a useful addition to LF.
- Canonical forms method is up to the task.
- Concentrating only on canonical forms and bidirectional typing yields new insights into subtyping.

secret slides

Related work

- Refinement types
  - Tim Freeman, Rowan Davies, Joshua Dunfield
- Logical frameworks
  - Robert Harper, Furio Honsell, Gordon Plotkin
  - Frank Pfenning
- Subtyping and dependent types
  - David Aspinall, Adriana Compagnoni

LF syntax

- Terms

no redexes

$R ::= c \mid x \mid R\ N$	atomic
$N ::= R \mid \lambda x.\ N$	normal

- Types

$P ::= a \mid P\ N$	atomic
$A, B ::= P \mid \Pi x{:}A.B$	normal

Hereditary substitution

- Substitution *must* contract redexes
- Example:

  $[(\lambda x.\ \mathrm{d}\ x\ x)\ /\ y]\ y\ z = \mathrm{d}\ z\ z$
- Indexed by type subscript for termination

  $[\mathrm{M}/x]_{\mathrm{A}}\ \mathrm{N}$
- Sometimes undefined:

  $[(\lambda x.\ x\ x)\ /\ y]_{\mathrm{A}}\ y\ y$ fails by induction on A

Synthesis

$$\boxed{\Gamma \vdash R \Rightarrow A}$$

$$\frac{x{:}A \in \Gamma}{\Gamma \vdash x \Rightarrow A} \qquad \frac{c{:}A \in \Sigma}{\Gamma \vdash c \Rightarrow A}$$

$$\frac{\Gamma \vdash R \Rightarrow \Pi x{:}A.B \qquad \Gamma \vdash N \Leftarrow A}{\Gamma \vdash R\ N \Rightarrow [N/x]_A\ B}$$

hereditary substitution

Checking

$$\boxed{\Gamma \vdash N \Leftarrow A}$$

$$\frac{\Gamma, x{:}A \vdash N \Leftarrow B}{\Gamma \vdash \lambda x.N \Leftarrow \Pi x{:}A.B}$$

$$\frac{\Gamma \vdash R \Rightarrow P' \qquad P' = P}{\Gamma \vdash R \Leftarrow P}$$

Example: the lambda calculus

exp : type.
val ⊏ exp.

lam : (exp → exp) → exp.
lam : (val → exp) → val.
app : exp → exp → exp.

value : exp → type.

- $s \sqsubset a :: L$

$$\frac{\Gamma \vdash R \Rightarrow P' \qquad P' = P}{\Gamma \vdash R \Leftarrow P}$$

$$\frac{\Gamma \vdash R \Rightarrow P' \qquad P' \leq P}{\Gamma \vdash R \Leftarrow P}$$

Subtyping

- Key rule:

$$\frac{\Gamma \vdash R \Rightarrow Q' \qquad Q' \leq Q}{\Gamma \vdash R \Leftarrow Q}\ \textbf{(switch)}$$

- Bidirectional at mode switch
- Canonical: base type

subtyping only at base type!

- $\Leftarrow\Gamma, x{:}A \vdash M \Leftarrow B \leq e$
- abcdefghijklmnopqrstuvwxyz
- ABCDEFGHIJKLMNOPQRSTUVWXYZ
- abcdefghijklmnopqrstuvwxyz
- ABCDEFGHIJKLMNOPQRSTUVWXYZ

- $\Gamma \vdash R \Rightarrow A$