Twenty Five Years of LF

Robert Harper
Carnegie Mellon

A Framework for Defining Logics

- H, Honsell, and Plotkin: A Framework for Defining Logics, LICS 1997, JACM 1993.
- Strongly influenced by
  - Martin Löf's type theory
  - de Bruijn's AUTOMATH
  - Avron's consequence relations

A Framework For Defining Logics

- A **language** for defining logical systems.
  - Capture syntactic and deductive regularities.
- A **methodology** for encoding formal systems and assessing their **adequacy**.
  - Compositional bijection between objects and certain canonical forms.

The Judgmental Perspective

- Martin-Löf stressed the **Logic of Judgments** as prior to the **Logic of Propositions.**
  - A prop, A true
  - $J_1,\ldots, J_n \vdash J$ expresses **entailment**
  - $|_{x,y,\ldots}$ J expresses **generality**
- Special emphasis on expressing intuitionistic type theory.

The Logic of Judgments

- Martin-Löf drew the distinction between
  - **Analytic**: self-evident judgments.
  - **Synthetic**: requires evidence.
- A true is synthetic, whereas $M : A$ and $M \equiv N : A$ are analytic

A Synthetic Approach

- The analytic/synthetic distinction depends on the **semantics** of the object logic.

- Our interest was purely in the **syntactics** of logics: represent formal systems in a systematic way, regardless of their meaning.

- Initial focus was on the concept of **natural deduction** introduced by Gentzen.

What Is Natural Deduction?

- AUTOMATH had similar motivations:
  - Content agnosticism: no ideology of math
  - Notation for proofs (“flags” and “brackets”)
  - Many different formalisms studied, esp. by van Daalen, Nederpelt, and Jutting.
  - Did not stress encoding of formalisms, rather just “doing mathematics”.

What is natural Deduction?

- All judgments are treated **synthetically**, with evidence being a **formal derivation.**
  - Atomic judgments are the "observables".
    - Higher-order judgments represent reasoning under hypotheses and generality.
- These are neatly captured using dependently typed λ-terms.

The LF Lambda Calculus

- **Three-level type system:**
  - **Kinds** K: Type, $\prod x{:}A.K$
  - **Families** A: c, $\lambda x{:}A_1.A_2$, AM, $\prod x{:}\ A_1.A_2$
  - **Objects** M : x, c, $\lambda x{:}A.M$, M_1M_2
- **Contexts**: x:A
- **Signatures**: c:K, c:A

Judgments as Types

- The **judgments as types** principle provided the **methodology** of encoding:
  - Atomic judgments are constant familes.
    - Hypothetical and general judgments are expressed using $\prod$ types.
  - **Adequacy** = **compositional** bijection between derivations and **canonical** forms of corresponding type.

Judgments as Types

- Higher-order **abstract syntax**:

```
exp : type.
zero : exp.
prop : type.
eq : exp -> exp -> prop.
imp : prop -> prop -> prop.
all : (exp -> prop) -> prop.
```

- **Adequacy**: α-equivalence, substitution are available "for free."

Judgments as Types

- **Derivations** are higher-order abstract syntax:

```
prop : type.
true : prop -> type.
impI : {a:prop}{b:prop}
(true a->true b) -> true(imp a b).
impE : {a:prop}{b:prop}
true(imp a b) -> true a -> true b.
```

- **Adequacy**: hypothetical and general reasoning is provided "for free".

Judgements as Types

- The LF language enabled expression of unusual **variations** on natural deduction.

- The **Schroeder-Heister** implication elimination rule:

```
shelim :
  ((true A -> true B) -> true C) ->
   true (imp A B) -> true C
```

Two Things That Really Mattered

- The **methodology** of encoding stresses **capturing the consequence relation.**
  - Characterize a class of **contexts** (worlds).
    - Consider the **canonical forms** of each type.
- Synthetic representation allows for **higher-level judgments.**
  - Derivations are objects of the theory.

MetaTheory of LF

- The biggest technical challenge in formulating LF was to develop
  - An algorithm for **type checking**, which reduces to checking definitional equality.
  - A notion of **canonical forms** with which to state adequacy.
- Lots of effort went into figuring this out.

Canonical Forms

- Canonical forms are **long βη-normal forms.**
  - `all([x]eq x x)` represents ∀x.x=x
    - But `all(eq)` is not canonical!
  - βη-equivalence is hard to handle in the presence of dependent types, while retaining decidability of type checking.

Definitional Equality

- Earliest versions used **type conversion**.
  - If $M : A$ and $A \equiv B$, then $M : B$
  - $A \equiv B$ is **untyped** $\beta\eta$-conversion.
- Untyped $\beta\eta$-conversion is not CR.
  - $\lambda x{:}A.(\lambda y{:}B.M)(x)$ critical pair
- **Strengthening** is very difficult, solved by Anna Salvesen.

Definitional Equality

- Pfenning and H developed a **typed** algorithm with **label-free** canonical forms, no family λ's.
  - Typed phase: $\Gamma \vdash M \Longleftrightarrow N \downarrow A$.
  - Structural phase: $\Gamma \vdash M \longleftrightarrow N \uparrow A$.
- Coquand developed a method based on the **shapes** of terms.
  - Handles family λ's, but not other types.

Definitional Equality

- The critical insight came from Watkins:
  - Define **canonical LF** consisting of long βη-normal forms **only**.
    - Define **hereditary substitution** by a clever inductive argument over types and terms.
  - See H + Licata JFP 2007 for canonical LF with subordination.

Definitional Equality

- Key idea: **β-reduce** during substitution, preserving η-long form.
  - Defined by a simultaneous induction on type and structure of term.
    - Substitution of canonical into canonical may induce further reductions.
- Derived from Pfenning's structural cut elimination, itself proved using LF.

Higher-level Judgments

- It quickly became apparent that **judgments about derivations** can express impurities.
  - Avron, Honsell, Mason used "D **closed**" to capture **validity** consequence relation.
    - Led to **judgmental reconstruction** of modal logic by Pfenning and Davies.
- Similar methods made it possible to **formalize the metatheory** of logical systems.

Representing Metatheory

- Type preservation is a three-place relation:

```
pres : red M N -> of M A -> of N A.
```

- Populate with derivations of the lemma:

```
_ : (pres Dred DofM DofM′) ->
    (pres (red/apl Dred)
          (of/ap DofM DofN)
          (of/ap DofM′ DofN)).
```

Mechanizing Metatheory

- The **Twelf** implementation made LF relevant and useful!
  - Frank Pfenning and Carsten Schürmann, starting from Eliot and Pf's Elf and H and Pf's LF implementation.
  - Contributions by dozens: see `twelf.org`.
- Robust, industrial-strength proof system for mechanized meta-reasoning.

Mechanizing Metatheory

- For me the crucial features of Twelf are:
  - **Unification** and type inference / argument synthesis (Eliot, Pym).
    - **Coverage** and **totality** checking for mechanization (Schürmann).
  - Regular worlds classify **adequacy contexts**, hence provide **induction on canonical forms**.

Mechanizing Metatheory

- Many meta-theorems are ∀∃ statements over **derivations** (canonical forms).
  - eg, for every reduction and every typing there is another typing
  - for every typing, there is either a reduction or a canonicity derivation.
- Schürmann developed a logic and prover for checking exhaustiveness and termination.

Mechanizing Metatheory

- Define a relation with **input** (∀) and **output** (∃) modes for arguments.
  - Use pattern-matching a la ML to express the inductive steps of the proof.
- Specify the **worlds** (contexts) over which to induct (determines the canonical forms).
- Check **coverage** and **totality** for the relation over the canonical forms in that proof.

Mechanizing MetaTheory

```
preserv : step E E' -> of E T -> of E' T -> type.
%mode preserv +Dstep +Dof -Dof'.

preserv-app-1     : preserv
                      (step-app-1 (DstepE1 : step E1 E1'))
                      (of-app (DofE2 : of E2 T2)
                              (DofE1 : of E1 (arrow T2 T)))
                      (of-app DofE2 DofE1')
                     <- preserv DstepE1 DofE1 (DofE1' : of E1' (arrow T2 T)).

preserv-app-beta : preserv
                      (step-app-beta (Dval : value E2))
                      (of-app (DofE2 : of E2 T2)
                              (of-lam (([x] [dx] DofE x dx)
                                       : {x : tm} {dx : of x T2} of (E x) T)))
                      (DofE E2 DofE2).

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

Mechanizing Metatheory

- D. Lee, K. Crary, and H (POPL 07): A mechanically checked proof of safety for Standard ML.

  - About 80K lines of Twelf.

  - Done by "pair programming", once per week over a semester.

- Important: we were not able to use The Definition as-is!

Semantics of Standard ML

- LF imposes **hygiene** on logical systems.
  - Must be precise about **everything**.
    - Intolerant of “side conditions”.
- The Definition was no exception.
  - van Inwegen uncovered many issues.
  - Scoping rules, informal side conditions.

The Re-Definition of Standard ML

- First, we had to **re-define** Standard ML.
  - **Internal** type theory with well-behaved notions of binding and scope.
  - **Dynamics** defined for the internal language using Plotkin's SOS, not ES.
  - **Statics** elaborates Standard ML into the internal language.

The Re-Definition of Standard ML

- Second, we used Twelf to formalize the **internal language** (2K loc).
  - See Lee, Crary, and H, POPL 2007.
  - Borrowing from Dreyer, Stone PhD's.
- **Progress** and **preservation** proved as outlined earlier by totality checking.
  - about 30K loc

The Re-Definition of Standard ML

- Third, we defined an **elaboration** of Standard ML into the internal language (3k loc).
  - Similar to "static semantics" of The Definition, but with typed internals.
- Fourth, we proved the **static correctness** of the elaboration (45k loc).
  - Result is always well-typed, hence safe.

Trouble Spots

- Two problems with representing PL's and their metatheory in LF:
  - State requires "manual" encoding of memory allocation and lookup.
  - Must carry along all aspects of machine state at each transition.
- Threatens adequacy and complicates specifications.

Substructural Frameworks

- LF has been extended to **substructural** frameworks by Pfenning, et al:
  - Linear LF (Cervesato)
  - Ordered LF (Polakow)
  - Hybrid LF (Reed)
  - Concurrent LF (Watkins, Simmons)
- Celf provides **substructural operational semantics.**

References in Celf (Simmons)

```
_ :
  eval (newref E1) D
   -o {Exists d1:dest A. eval E1 d1 * fnewref d1 D}.
_ :
  return V1 D1 * fnewref D1 D
   -o { Exists c:dest A. @contains c V1 * return (cell c) D }.
```

```
_ :
  eval (deref E1) D
   -o {Exists d1 : dest (ref A). eval E1 d1 * fderef d1 D}.
_ :
  return (cell C1) D1 * contains C1 V1 * fderef D1 D
   -o { @contains C1 V1 * return V1 D }.
```

References in Celf

- The Celf specification works beautifully for state, concurrency, continuations, ...
  - Purely **local** specifications.
    - Induces a **transition system** on contexts.
  - But the worlds cannot be characterized as simply as in Twelf.
    - eg, “no variable declared twice”

References in LF and Celf

- Neither LF nor Celf can handle **disequality** of references (locations).
  - Mechanizing metatheory, e.g. prog+pres.
    - Languages with equality of references.
- Celf cannot handle **locally scoped** (stack-allocated) assignables.
  - eg, Modernized Algol in H's PFPL book.

Handling references

- J. Cheney's Nominal LF adds **symbols** to LF.
  - Admit disequality.
  - Models references, channels, etc.
- See also Tiu and Miller's **generic** quantifiers in λ-Prolog.
- Ideally, we'd like to have a **Nominal Celf**, with a meta-reasoning prover a la Twelf.

Further Developments

- The development of λ-Prolog by Miller, et al. has been enormously influential.
  - Pattern unification, efficient implementation.
  - Generic quantifiers, logical account of modularity.
- Cannot express higher-level judgments, which have proved very useful in LF.

Further Developments

- LF has influenced **functional programming**.
  - Delphin (Schürmann, et al.)
  - Beluga (Pientka, Dunfield)
  - Polarization (Licata, Zeilberger, and H)
- Rabe has built a **module system** for Twelf.
- Pitts, Urban, et al.: **Nominal Logic**.

Some PhD's On LF

- Carnegie Mellon
  - Conal Elliott
  - Penny Anderson
  - Iliano Cervesato
  - Robert Virga
  - Carsten Schürmann
  - Alberto Momigliano
  - Jeff Polakow
  - Brigitte Pientka
  - Jason Reed
  - William Lovas
  - Rob Simmons
- Udine
  - Merino Miculan
- Edinburgh
  - David Pym
  - Philippa Gardner
  - Anna Salvesen
- Penn (λ-Prolog)
  - Gopalan Nadathur
  - Amy Felty
  - John Hannan
  - Josh Hodas
  - Ray MacDowell
- Cornell
  - James Cheney

Conclusion

- I feel fortunate to have been at the right place at the right time to help develop LF.

  - Thanks especially to my co-authors, and to Avron, Mason, and Pfenning, from whom I have learned a great deal.

- I am astonished by how quickly 25 years has passed, and how much has been done in that seemingly short amount of time!