Benchmarks for mechanized meta-theory
A very personal and partial view

Alberto Momigliano

Dagstuhl, Oct 17, 2016

Loosely based on joint work with
Amy Felty and Brigitte Pientka

Disclaimer

- These are half-baked thoughts cooked up on the plane, offered here only to get a discussion going.
- As Donald Rusmfeld used to say:
  - There are things **I know I** don't know.
  - There are things **I don't know I** don't know .
- So feel free to interject at any time.

Introduction

- Benchmarks in theorem proving are useful, make the state of the art progress or at least take stock
  - automating FOL: TPTP [Sutcliffe, JAR 2009]
  - higher-order extension: THF0 [Benzmüller et al, IJCAR 2008]
  - SATLIB [Hoos & Stützle, SAT 2000]
- The situation is less satisfactory for **proof assistants**, where each system comes with its own set of examples/libraries, some of them, as we well know, gigantic.
- Not surprising, since we're potentially addressing the whole realm of maths
- So, let's pull the breaks and limit ourselves here to the **meta-theory of deductive systems**, such as the ones in Programming Languages theory

Benchmarks for PLT

- Where do we stand?
  - There are obvious differences with say TPLP, qualitative vs quantitative comparisons
- In a dark and stormy night of 2005 the POPLMark Challenge appeared. And then?
  - Lots of clever solutions (mostly incomplete) and to be fair almost all using existing technology
  - What kind of impact on the communities (PL and LF researchers)?

Impact on PL research

- More papers at POPL/ICFP come with at least a partial formalization
  - Sewell POPL 14: "Around 10% of submissions were completely formalised, slightly more partially formalised"
- Formalizations are evaluated (`http://www.artifact-eval.org/`)
- Can I have an hallelujah? Do we feel, as logical framework people vindicated? Does it make a difference for acceptance (or for the betterment of science)?
  - Sewell again "acceptance rates for these were in line with those for submissions as a whole."

Impact on PL research

- Significant success of courses such as “Software Foundations” and to a lesser extent “Concrete Semantics” has been forming new generations of PL people who consider proof assistants as part of the tools belt (OK, it’s mostly Coq but still)
- Huge success of projects where formalization is paramount such as CompCert, Relaxed Concurrency Models, SEL4 etc.
- Significant fundings for new ones (REMS, DeepSpec, SECOMP, RustBelt)

Impact on logical frameworks

- Did they evolve in the last 10 years as a response or on their own accord? Feel free to disagree
  - **HOAS**: Twelf stood still, Abella and Beluga were born out of research (nabla and fixed point logics, CMTT) stemming from the early 2000.
    - to be generous, one could say that Abella 2.0's generalization of SL logic to higher-order owes to Pientka's solution of POPLMark 1.a
    - They still are systems with very limited user-base outside developers
  - **Nominal**: the Isabelle package turned into Nominal2, but more to have a better foundation for equivariance and to handle multiple binders
    - From the traffic on the mailing list, I'd say Nominal Isabelle has gone quiet, although it has some power users not deterred by the amount of low-level proofs one has still to handle
    - Nominal techniques in Coq: I don't follow HTT (too hard for me), so don't know

Coq's masterplan for world domination

- Is a monopoly in PL theory necessarily a bad thing? Dale Miller is proposing a *marketplace* for proofs, is Coq the new Facebook killing it in its infancy?
  - Large user base
  - Very active development
- What about libraries for binders: are they used? Are people happy with that?
  - *Locally nameless* from the "Engineering Formal Metatheory" paper
  - Smolka's *Autosubst*
  - *Hybrid* (no, and I understandably so, until we make it usable by others).

Is my obsession with binders justified?

- Well, not just mine (three words: Harper, Honsell & Milner)
- Binders are ubiquitous in PLT as well as obnoxious and need to be handled simply, correctly and generically.
- Still, a fair amount of formalization stays (apparently successfully so) away from binders
- “LEM does not care about binders, and we’re happy to keep it that way” (Peter Sewell, my recollection 2014)
- Not to name names (got it?), Leroy’s celebrated “Coinductive big-step operational semantics” uses named not α-equivalent syntax, non-capture avoiding substitutions, and so does Software Foundations
- They know what they’re doing. Maybe they’re right?

Benchmarks, the past

- In the 90's we were happy about proving **type soundness** for PCF and friends – graduated to **memory safety** following PCC
- Then we got to the **Church-Rosser** theorem – at least it deals with *open* not *closed* terms
- POPLMark, we already saw.
- We (Amy Felty, Brigitte Pientka and myself) proposed some (by design) simple benchmarks to stress systems for **reasoning in a context** – see the appendix of this talk
- For example, the equivalence of declarative and algorithmic (α)-equality between lambda terms

Benchmarks, the present (and future?)

- Many reliability and security properties of software depend upon some notion of **program equivalence**:
  - information-flow security (confinement, non-interference). More in general:
  - Bisimulation between (concurrent) systems
  - full abstraction of program transformation within the same language or from high to low level code. More in general:
  - compiler correctness (too many advances to even mention), most of them formalized
- A unifying theme is the emerging, pervasive use of **logical relations**, in particular step-indexed.
- Seems to be prevailing over **coinductive** techniques, perhaps because the latter are less supported in proof assistants (wrt inductive ones, I mean)

Step-indexed logical relations

- People have formalized them quite often by now, so what's the fuss?
- Some do, yes, but it's still challenging to do it right:
  - The logical relation is not immediate to be accepted as an inductive definition
  - It involves a bit of arithmetic reasoning, and this may be annoying for hyper-pure frameworks

    "definitions and proofs have a tendency to become cluttered with extra indices and even arithmetic, which are really playing the role of construction lines" (Benton and Hur, "Step-Indexing: The Good, the Bad and the Ugly")

LR, continued

- An example: "A verified framework for higher-order uncurrying optimization (Dargaye & Leroy, 2010)
  - Interesting as it uses coinduction to encode cyclic closure (the Milner-Tofte trick)
  - But they use closure to avoid substitutions ...
  - it leaves open what to do with divergent source programs
- Still a lot of work for a source language that is the untyped λ-calculus with `letrec`
- A paper such as "Typed Closure Conversion Preserves Observational Equivalence" (Ahmed et al. 2008), where the source language is system F with existential and recursive types is way harder – and I haven't look at the more recent stuff about multi-language semantics

Other directions (please contribute)

- Interesting mix of (co)induction and (co)recursion (e.g. papers by Nakata-Uustalu among others)
- Work on relational reasoning where polymorphism really is a plus (Howe's method in its generality and more in general Soren Lassen's thesis)
- Benchmarks where object logics make heavy use of constraint domains (see the Twelf/clp-examples directory, work by Ivan Scagnetto and al. on LF + oracles)

One slide on ORBI

- Benchmarks to be communicated need a language
- Open challenge problem Repository for systems supporting reasoning with BInders, `https://github.com/pientka/ORBI/`
  - ORBI is designed to be:
    - human-readable
    - easily machine-parsable
    - uniform
    - flexible and extensible
  - Currently oriented toward supporting the following systems:
    - Twelf
    - Beluga
    - Abella
    - Hybrid

    without hopefully precluding other current and future systems supporting HOAS, and eventually supporting other representation techniques such as nominal.

Related Work

- The above libraries for ATP
- Very little about **inductive** problems
- Ott and LEM as front-ends
- LF as a common ground, e.g.,
  - Logosphere (`http://www.logosphere.org`)
  - SASyLF [Aldrich et al, 2008]
  - Modularity in LF specifications [Rabe & Schürmann]
- Why3 (`http://why3.lri.fr`), a software verification platform providing a front-end to third-party theorem provers.
- Environments for programming language descriptions
  - PLT-Redex [Felleisen et al, 2009]
  - The K framework [Roșu & Șerbănuță, JLAP, 2010]
- Handling and sharing of mathematical content
- ...

Appendix

Some benchmarks from:

- Amy Felty, A.M. and Brigitte Pientka. Benchmarks for Reasoning with Higher-Order Abstract Syntax Representations. To appern in MSCS

Untyped Lambda Terms

$$\begin{array}{llll} \text{Types} & A, B & ::= & \alpha \mid \mathsf{arr} A\, B \\ \text{Terms} & M & ::= & x \mathsf{lam}\, x.\, M \mid \mathsf{app}\ M_1\ M_2 \end{array}$$

Well-Formed Terms: (is_tm M)

$$\frac{\text{is\_tm } x \in \Gamma}{\Gamma \vdash \text{is\_tm } x}\ tm_v \qquad \frac{\Gamma, \text{is\_tm } x \vdash \text{is\_tm } M}{\Gamma \vdash \text{is\_tm } (\text{lam}\, x.\, M)}\ tm_l$$

$$\frac{\Gamma \vdash \text{is\_tm } M_1 \qquad \Gamma \vdash \text{is\_tm } M_2}{\Gamma \vdash \text{is\_tm } (\text{app } M_1\ M_2)}\ tm_a$$

Equality of Lambda Terms

Algorithmic Equality: (aeq M N)

$$\frac{\text{aeq } x\ x \in \Gamma}{\Gamma \vdash \text{aeq } x\ x}\ ae_v \qquad \frac{\Gamma, \text{is\_tm } x; \text{aeq } x\ x \vdash \text{aeq } M\ N}{\Gamma \vdash \text{aeq } (\text{lam}\, x.\, M)\ (\text{lam}\, x.\, N)}\ ae_l$$

$$\frac{\Gamma \vdash \text{aeq } M_1\ N_1 \quad \Gamma \vdash \text{aeq } M_2\ N_2}{\Gamma \vdash \text{aeq } (\text{app } M_1\ M_2)\ (\text{app } N_1\ N_2)}\ ae_a$$

Contexts

$S_x := \text{is\_tm } x \qquad S_{xa} := \text{is\_tm } x; \text{aeq } x\ x$

Note: S_{xa} is a basic linear extension of S_x.

Benchmark: Basic Linear Context Extension

Theorem (Admissibility of Reflexivity)
If $\Phi_{xa} \vdash \text{is\_tm}\ M$ then $\Phi_{xa} \vdash \text{aeq}\ M\ M$.

Theorem (Admissibility of Symmetry and Transitivity)

1. If $\Phi_{xa} \vdash \text{aeq}\ M\ N$ then $\Phi_{xa} \vdash \text{aeq}\ N\ M$.
2. If $\Phi_{xa} \vdash \text{aeq}\ M\ L$ and $\Phi_{xa} \vdash \text{aeq}\ L\ N$ then $\Phi_{xa} \vdash \text{aeq}\ M\ N$.

The Polymorphic Lambda Calculus

Types	A, B	$::=$	$\alpha \mid \mathsf{arr} A\, B \mid \mathsf{all}\, \alpha.\, A$
Terms	M	$::=$	$x \mid \mathsf{lam}\, x.\, M \mid \mathsf{app}\ M_1\ M_2 \mid$
			$\mathsf{tlam}\, \alpha.\, M \mid \mathsf{tapp}\ M\ A$

Rules	Well-formedness of Types	$(\mathsf{is\_tp}\ A)$
	Well-formedness of Terms	$(\mathsf{is\_tm}\ M)$
	Equality of Types	$(\mathsf{atp}\ A\ B)$
	Equality of Terms	$(\mathsf{aeq}\ M\ N)$

Context Schemas	S_α	$::=$	$\mathsf{is\_tp}\ \alpha$
	$S_{\alpha x}$	$::=$	$\mathsf{is\_tp}\ \alpha + \mathsf{is\_tm}\ x$
	S_{atp}	$::=$	$\mathsf{is\_tp}\ \alpha; \mathsf{atp}\ \alpha\ \alpha$
	S_{aeq}	$::=$	$\mathsf{is\_tp}\ \alpha; \mathsf{atp}\ \alpha\ \alpha + \mathsf{is\_tm}\ x; \mathsf{aeq}\ x\ x$

Linear Context Extensions with Alternatives

Theorem (Admissibility of Reflexivity for Types)
If $\Phi_{atp} \vdash \text{is\_tp}\ A$ then $\Phi_{atp} \vdash \text{atp}\ A\ A$.

Theorem (Admissibility of Reflexivity for Terms)
If $\Phi_{aeq} \vdash \text{is\_tm}\ M$ then $\Phi_{aeq} \vdash \text{aeq}\ M\ M$.

Non-linear Context Extensions

Declarative Equality of the Untyped Lambda Calculus

$$\ldots$$

$$\frac{}{\Gamma \vdash \text{deq}\ M\ M}\ de_r$$

$$\frac{\Gamma \vdash \text{deq}\ M\ L \quad \Gamma \vdash \text{deq}\ L\ N}{\Gamma \vdash \text{deq}\ M\ N}\ de_t \qquad \frac{\Gamma \vdash \text{deq}\ N\ M}{\Gamma \vdash \text{deq}\ M\ N}\ de_s$$

$$\text{Context Schemas} \quad \begin{array}{lcl} S_{xd} & ::= & \text{is\_tm}\ x; \text{deq}\ x\ x \\ S_{da} & ::= & \text{is\_tm}\ x; \text{deq}\ x\ x; \text{aeq}\ x\ x \end{array}$$

Theorem (Completeness)
If $\Phi_{da} \vdash \text{deq}\ M\ N$ then $\Phi_{da} \vdash \text{aeq}\ M\ N$.

Order

Theorem (Pairwise Substitution)
If $\Phi_{xa}, \text{is\_tm } x; \text{aeq } x\ x \vdash \text{aeq } M_1\ M_2$ and $\Phi_{xa} \vdash \text{aeq } N_1\ N_2$, then $\Phi_{xa} \vdash \text{aeq } ([N_1/x]M_1)\ ([N_2/x]M_2)$.

Uniqueness

Terms	M	$::=$	$x \mid \text{lam}\, x.\, M \mid \text{app } M_1\ M_2$
Types	A	$::=$	$\text{i} \mid \text{arr } A\, B$
Context Schema	S_t	$:=$	$\text{is\_tm } x; x{:}A$

Theorem (Type Uniqueness)
If $\Phi_t \vdash M : A$ and $\Phi_t \vdash M : B$ then $A = B$.

Substitution

Parallel Reduction

$$\frac{x \leadsto x \in \Gamma}{\Gamma \vdash x \leadsto x}\, pr_v \qquad \frac{\Gamma, \text{is\_tm } x; x \leadsto x \vdash M \leadsto N}{\Gamma \vdash \text{lam}\, x.\, M \leadsto \text{lam}\, x.\, N}\, pr_l$$

$$\frac{\Gamma, \text{is\_tm } x; x \leadsto x \vdash M \leadsto M' \qquad \Gamma \vdash N \leadsto N'}{\Gamma \vdash (\text{app } (\text{lam}\, x.\, M)\ N) \leadsto [N'/x]M'}\, pr_\beta$$

$$\frac{\Gamma \vdash M \leadsto M' \qquad \Gamma \vdash N \leadsto N'}{\Gamma \vdash (\text{app } M\ N) \leadsto (\text{app } M'\ N')}\, pr_a$$

Context Schemas $\quad S_r := \text{is\_tm } x; x \leadsto x$

$\qquad S_{rt} := \text{is\_tm } x; x \leadsto x; x{:}A$

Substitution (Continued)

Lemma (Substitution)
If $\Phi_t, \text{is\_tm } x; x{:}A \vdash M : B$ and $\Phi_t \vdash N : A$, then $\Phi_t \vdash [N/x]M : B$.

Theorem (Type Preservation for Parallel Reduction)
If $\Phi_{rt} \vdash M \rightsquigarrow N$ and $\Phi_{rt} \vdash M : A$, then $\Phi_{rt} \vdash N : A$.

$$\text{Context Schema} \quad S_{\alpha t} \quad ::= \quad \text{is\_tp } \alpha + \text{is\_tm } x; x{:}A$$

Lemma (Substitution)
If $\Phi_{\alpha t}, \text{is\_tp } \alpha \vdash M : B$ and $\Phi_{\alpha t} \vdash \text{is\_tp } A$, then
$\Phi_{\alpha t} \vdash [A/\alpha]M : [A/\alpha]B$.