POPLMark Reloaded

Andreas Abel
Department of Computer Science and Engineering, Gothenburg University / Chalmers, Sweden
andreas.abel@gu.se

Alberto Momigliano
DI, Università degli Studi di Milano, Italy
momigliano@di.unimi.it

Brigitte Pientka
School of Computer Science, McGill University, Montreal, Canada
bpientka@cs.mcgill.ca

ABSTRACT

As a follow-up to the POPLMark Challenge, we propose a new benchmark for machine-checked metatheory of programming languages: establishing strong normalization of a simply-typed lambda-calculus with a proof by Kripke-style logical relations. We believe that this case-study overcomes some of the limitations of the original challenge and highlights, among others, the need of native support for context reasoning and simultaneous substitutions.

1 INTRODUCTION

The usefulness of sets of benchmarks has been recognized in many areas of computer science, and in particular in the theorem proving community, for stimulating progress or at least taking stocks of what the state of the art is — *TPTP* [Sutcliffe 2009] is one shining example. The situation is less satisfactory for proof assistants, where each system comes with its own set of examples/libraries, some of them gigantic; this is not surprisingly, since we are potentially addressing the whole realm of mathematics.

In a more limited setting, some 12 years ago, a group of renowned programming language theorists came together and issued the so-called *POPLMark Challenge* [Aydemir et al. 2005] (PC, in short), with the aim of fostering the collaboration between the PL community and researchers in proofs assistants/logical frameworks to bring about:

> "[…] a future where the papers in conferences such as POPL and ICFP are routinely accompanied by mechanically checkable proofs of the theorems they claim" (page 51 op. cit.)

As we know, the challenge revolved around the meta-theory of $\mathsf{F}_{<:}$, which, requiring induction over *open* terms, was an improvement over the gold standard of mechanized meta-theory in the nineties: type soundness. Yet, the spotlight of the PC was still on

> "type preservation and soundness theorems, unique decomposition properties of operational semantics, proofs of equivalence between algorithmic and declarative versions of type systems, etc." (*ibidem*)

Further, the authors made paramount "the problem of representing and reasoning about inductively-defined structure with *binders*" (our emphasis), while providing a balanced criticism of de Bruijn indexes as an encoding technique. That focus was understandable, since at that time the only alternative to concrete representations was higher-order abstract syntax (HOAS), mostly in the rather peculiar Twelf setting, the implementation of nominal logic being in its infancy.

LFMTP 2017, September 2017, Oxford, UK

While the response of the theorem-proving community was impressive with more than 15 (partial) solutions submitted (https://www.seas.upenn.edu/~plclub/poplmark/), one can argue whether the envisioned future has became our present — according to Sewell's POPL 2014 Program Chair's Report (https://www.cl.cam.ac.uk/~pes20/popl2014-pc-chair-report.pdf) "Around 10% of submissions were completely formalised, slightly more partially formalised". It is also debatable whether the challenge had a direct impact on the development of proof assistants and logical frameworks: specialized systems such as Abella [Baelde et al. 2014] and Beluga [Pientka and Cave 2015] were born out of independent research of the early 2000. To be generous, we could impute Abella's generalization of its specification logic to higher-order [Wang et al. 2013] to this Twelf POPLMark solution [Pientka 2007], but development in mainstream systems such as Coq, Agda, and (Nominal) Isabelle were largely driven by other (internal) considerations.

In a much more modest setting, but in tune with the goal of the PC, [Felty et al. 2017] recently presented some benchmarks with the intention of going beyond the issue of representing binders, whose pro and cons they consider well-understood. Rather, the emphasis was on the all important and often neglected issue of reasoning within a *context of assumptions*, and the role that properties such as weakening, ordering, subsumption play in formal proofs. These are more or less supported in systems featuring some form of hypothetical and parametric reasoning, but the same issues occur in first-order representation as well; in this setting, typically, they are not recognized as crucial, rather they are considered one of the prices one has to pay when reasoning over open terms. This set of benchmarks was accompanied by a preliminary design of a common language and open repository [Felty et al. 2015b], which is fair to say did not have a resounding impact so far.

In the mean time, the PL world did not stand still, obviously. One element that we have picked on is the multiplication of the use of proofs by *logical relations* [Statman 1985] — not coincidentally, those featured in [Aydemir et al. 2005]'s section "Beyond the challenge". From the go-to technique to prove normalization of certain calculi, proofs by logical relations are now used to attack problems in the theory of complex languages models, with applications to issues in equivalence of programs, compiler correctness, representation independence and even more intensional properties such as non-interference, differential privacy and secure multi-language interoperability, to cite just a few [Ahmed 2015; Bowman and Ahmed 2015; Neis et al. 2015].

Picking up on PC's final remark "We will issue a small number of further challenges […]", we propose, as we detail in Section 2.3 a new challenge that we hope it will move the bar a bit forward. We suggest Strong Normalization (SN) for the simply-typed lambda-calculus proven via logical relations in the Kripke style formulation,

see [Coquand 1991] for an early use. We discuss the rationale in the next Section.

2 THE CHALLENGE

2.1 Problem Selection

(Strong) normalization by Tait's method is a well-understood and reasonably circumscribed problem that has been a cornerstone of mechanized PL theory, starting from [Altenkirch 1993]. There are of course many alternative ways to prove SN for a lambda-calculus, see for example the inductive approach of [Joachimski and Matthes 2003], partially formalized in [Abel 2008], or by reduction from strong to weak normalization [Sorensen 1997]. For that matter, a SN proof via logical relations for the simply-typed lambda calculus can be carried out (see for a classic example [Girard et al. 1990]) without appealing to a Kripke definition of reducibility, at the cost, though, of a rather cavalier approach to "free" variables. However, the Kripke technique is handy in establishing SN for richer theories such as dependently typed ones, as well as for proving stronger results, for example about equivalence checking [Crary 2005; Harper and Pfenning 2005].

We claim that mechanizing such a proof is indeed challenging since:

- It focus on reasoning on *open* terms and on relating different contexts or *worlds*, taking seriously the Kripke analogy. The quantification over *all* extensions of the given world may be problematic for frameworks where contexts are only implicitly represented, or, on the flip side, may require several boring weakening lemmas in first-order representations.
- The definition of *reducibility* requires a sophisticated notion of inductive definition, which must be compatible with the binding structures, but also be able to take into account *stratification*, to tame the negative occurrence of the defined notion.
- Simultaneous substitutions and their equational theory (composition, commutation etc.) are central in formulating and proving the main result. For example, in the proof of the Fundamental Theorem 2.8, we need to push substitutions through (binding) constructs.

In this sense, this challenge goes well beyond the original PC, where the emphasis was on binder representations, proofs by structural induction and operational semantics animation.

Previous formalizations of strong normalization usually follows Girard's approach, see for example [Donnelly and Xi 2007] carried out in ATS/LF, or the one available in the Abella repository (abella-prover.org/~normalization/). Less frequent are formalizations following the Kripke discipline: both [Cave and Pientka 2015] and [Narboux and Urban 2008] encode [Crary 2005]'s account of decision procedures for term equivalence in the STLC, in Beluga and Nominal Isabelle respectively; the latter was then extended in [Urban et al. 2011] to formalize the analogous result for LF [Harper and Pfenning 2005]. See [Abel and Vezzosi 2014] for a SN Kripke-style proof for a more complex calculus and [Rabe and Sojakova 2013] for another take to handling dependent types — this paper also contains many more references to the literature.

The choice of a Kripke-style proof of SN for the STLC may sound contentious on several grounds and hence we will try to motivate it further:

- We acknowledge that SN is not the most exciting application of logical relations, some of which we have mentioned in the previous Section. Still, it is an important topic in type theory, in particular w.r.t. logical frameworks' meta-theory, see for example [Altenkirch and Kaposi 2016], and in this sense dear to our hearts. It is a well-known textbook example, which uses techniques that should be familiar to the community of interest in the simplest possible setting.
- Yes, the STLC is the prototypical toy language, while a POPL paper will address richer PL theory aspects. For one, adding more constructs, say in the PCF direction, perhaps with an iterator, would make the proof of the fundamental theorem longer, but not more interesting. Secondly, we think that a good benchmark should be simple enough that it could be tried out almost immediately if one is acquainted with proof-assistants. Conversely, it should encourage a PL theorist to start playing with proof assistants. Finally, we do suggest extensions of our challenge in the next Section.
- The requirement of the "Kripke-style" may seem overly constrictive, especially since this may not be strictly needed for the STLC. However, as we have argued before, this is meant as a springboard for more complex case studies, where this technique is forced on us. Remember that we are interested in *comparing* solutions. A more ambitious challenge may not solicit enough solutions, if the problem is too exotic or simply too lengthy.

2.2 Evaluation Criteria

One of the limitations of the PC experiment was in the *evaluation* of the solutions, although it is not easy to avoid the "trip to the zoo" effect, well-known from trying to comparing programming languages: there is no theory underlying the evaluation; criteria tend to be rather qualitative, and finally, the comparison itself may be lengthy [Felty et al. 2015a]. Within these limitations, of the proposed solutions we will take into consideration the:

- Size of the necessary infrastructure for defining the base language: binding, substitutions, renamings, contexts, together with substitution and other infrastructural lemmas.
- Size of the main development versus the main theorems in the on-paper proof, in particular, number of technical lemmas not having a direct counterpart in the on-paper proof.

More qualitatively, we will try to assess the:

- Ease of using the infrastructure for supporting binding, contexts, etc. How easy is it to apply the appropriate lemmas in the main proof? For example, does applying the equational theory of substitutions require low-level rewriting, or is it automatic?
- Ease of development of the overall proof; what support is present for proof construction, when not for proof and counterexample search?

2.3 The Challenge, Explained

Let us recall the definition of the STLC, starting with the grammar of terms, types, contexts and substitutions:

$$\begin{array}{lll} \text{Terms} & M, N & ::= x \mid \lambda x{:}T.M \mid M\ N \\ \text{Types} & T, S & ::= B \mid T \to S \\ \text{Context} & \Gamma & ::= \cdot \mid \Gamma, x{:}T \\ \text{Subs} & \sigma & ::= \epsilon \mid \sigma, N/x \end{array}$$

The static and dynamic semantics are standard and are depicted in Figure 1. Since we want to be very upfront about the fact that evaluation goes under a lambda and thus involves open terms, we make the context explicit even in the reduction rules, contrary to what, say, Barendregt would do. Note that, because of rule E-Abs, we do not need to assume that the base type is inhabited by a constant. We denote with $[\sigma]M$ the application of the simultaneous substitution σ to M and with $[\sigma_1]\sigma_2$ their composition.

We now define the set of *strongly-normalizing* terms as pioneered by [Altenkirch 1993] and by now usual:

$$\frac{\forall M'.\ \Gamma \vdash M \longrightarrow M' \quad \Gamma \vdash M' \in \mathsf{SN}}{\Gamma \vdash M \in \mathsf{SN}}\ SN - WF$$

expressing that the set of strongly normalizing terms is the well-founded part of the reduction relation. A more explicit formulation of strong normalization is allowed, see for example [Joachimski and Matthes 2003], but then an equivalence proof should be provided. Note that reasoning with the above rule *SN-WF* cannot proceed by structural induction, since it is not the case that M' is a sub-term of M.

The logical predicates have the following structure:

- $\Gamma \vdash M \in \mathcal{R}_T$, and
- $\Gamma' \vdash \sigma \in \mathcal{R}_\Gamma$.

We use a Kripke-style logical relations definition where we *witness* the context extension using a *weakening* substitution ρ. This can be seen as a *shift* in de Bruijn terminology, while other encodings may use different (or no particular) implementation techniques for handling context extensions.

Definition 2.1 (Reducibility Candidates).

- $\Gamma \vdash M \in \mathcal{R}_B$ iff $\Gamma \vdash M : B$ and $\Gamma \vdash M \in \mathsf{SN}$:
- $\Gamma \vdash M \in \mathcal{R}_{T \to S}$ iff $\Gamma \vdash M : T \to S$ and for all N, Δ such that $\Gamma \leq_\rho \Delta$, if $\Delta \vdash N \in \mathcal{R}_T$ then $\Delta \vdash ([\rho]M)\ N \in \mathcal{R}_S$.

As usual, we lift reducibility to substitutions:

Definition 2.2 (Reducible Substitutions).

- $\Gamma' \vdash \epsilon \in \mathcal{R}$.
- $\Gamma' \vdash \sigma, N/x \in \mathcal{R}_{\Gamma, x:T}$ iff $\Gamma' \vdash \sigma \in \mathcal{R}_\Gamma$ and $\Gamma' \vdash N \in \mathcal{R}_T$.

We now give an outline of the proof as a sequence of lemmas — the reader will find all the details in the forthcoming full version of this paper.

Lemma 2.3 (Semantic Function Application).
If $\Gamma \vdash M \in \mathcal{R}_{T \to S}$ *and* $\Gamma \vdash N \in \mathcal{R}_T$ *then* $\Gamma \vdash M\ N \in \mathcal{R}_S$.

Proof. Immediate, by definition. □

Lemma 2.4 (SN Closure under Weakening).
If $\Gamma_1 \leq_\rho \Gamma_2$ *and* $\Gamma_1 \vdash M \in \mathsf{SN}$ *then* $\Gamma_2 \vdash [\rho]M \in \mathsf{SN}$.

Proof. By induction on the derivation of $\Gamma_1 \vdash M \in \mathsf{SN}$. □

Lemma 2.5 (Closure of Reducibility under Weakening).
If $\Gamma_1 \leq_\rho \Gamma_2$ *and* $\Gamma_1 \vdash M \in \mathcal{R}_T$ *then* $\Gamma_2 \vdash [\rho]M \in \mathcal{R}_T$.

Proof. By cases on the definition of reducibility using the above Lemma 2.4 and weakening for typing. □

Lemma 2.6 (Weakening of Reducible Substitutions).
If $\Gamma_1 \leq_\rho \Gamma_2$ *and* $\Gamma_1 \vdash \sigma \in \mathcal{R}_\Phi$ *then* $\Gamma_2 \vdash [\rho]\sigma \in \mathcal{R}_\Phi$.

Proof. By induction on the derivation of $\Gamma_1 \vdash \sigma \in \mathcal{R}_\Phi$ using Closure of Reducibility under Weakening. □

Lemma 2.7 (Closure under Beta Expansion).
If $\Gamma \vdash N \in \mathsf{SN}$ *and* $\Gamma \vdash [N/x]M \in \mathcal{R}_S$ *then* $\Gamma \vdash (\lambda x{:}T.M)\ N \in \mathcal{R}_S$.

Proof. By induction on S after a suitable generalization. □

Theorem 2.8 (Fundamental Theorem).
If $\Gamma \vdash M : T$ *and* $\Gamma' \vdash \sigma \in \mathcal{R}_\Gamma$ *then* $\Gamma' \vdash [\sigma]M \in \mathcal{R}_T$.

Proof. By induction on $\Gamma \vdash M : T$. In the case for functions, we use Closure under Beta Expansion (Lemma 2.7) and Weakening of reducible substitution (Lemma 2.6). □

3 BEYOND THE CHALLENGE

There is an ongoing tension between weak and strong logical frameworks [de Bruijn 1991], with which we can encode our benchmarks. Weak frameworks are designed to accommodate advanced infrastructural features for binders (HOAS/nominal syntax etc.) and for judgments (hypothetical and parametric), but may struggle on other issues, such as facilities for computation or higher-order quantification/impredicativity. There are at least two coordinates in which we can directly extend our benchmark, to further highlight this dilemma:

- Logical relations for dependent types [Abel and Vezzosi 2014; Rabe and Sojakova 2013], up to the Calculus of Constructions. Here we need to go beyond first-order quantification, which is typically what is on offer in weak frameworks.
- Proof by logical relation via *step-indexing* [Appel and McAllester 2001]. Here we have two issues:
  (1) the logical relation may even be harder to be accepted by the meta-language as an inductive definition than with simple types; in fact, the work around the negative occurrence of the defined relation cannot be based on structural induction on types, but it has to use some form of course-of-value induction.
  (2) It involves a limited amount of arithmetic reasoning: "definitions and proofs have a tendency to become cluttered with extra indices and even arithmetic, which are really playing the role of construction line." ([Benton and Hur 2010]).

  This latter point may be problematic for frameworks such as Abella and Beluga, which do not (yet) have extensive libraries, nor computational mechanisms (rewriting, reflection) for those tasks.

$\boxed{\Gamma \vdash M : T}$ Term M has type T in context Γ

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T}\ u \qquad \frac{\Gamma, x : T \vdash M : S}{\Gamma \vdash (\lambda x{:}T.M) : (T \to S)}\ \text{T-Abs}^x \qquad \frac{\Gamma \vdash M : (T \to S) \quad \Gamma \vdash N : T}{\Gamma \vdash (M\ N) : S}\ \text{T-App}$$

$\boxed{\Gamma \vdash M \longrightarrow M'}$ Term M steps to term M' in context Γ

$$\frac{\Gamma, x{:}T \vdash M \longrightarrow M'}{\Gamma \vdash \lambda x{:}T.M \longrightarrow \lambda x{:}T.M'}\ \text{E-Abs}^x \qquad \frac{}{\Gamma \vdash (\lambda x{:}T.M)\ N \longrightarrow [N/x]M}\ \text{E-App-Abs} \qquad \frac{\Gamma \vdash M \longrightarrow M'}{\Gamma \vdash M\ N \longrightarrow M'\ N}\ \text{E-App2} \qquad \frac{\Gamma \vdash N \longrightarrow N'}{\Gamma \vdash M\ N \longrightarrow M\ N'}\ \text{E-App1}$$

Figure 1: Typing and reduction rules for the STLC

4 CALL FOR ACTION

We ask the community to submit solutions and we plan to invite everyone who does so to contribute towards a joint paper discussing trade-offs between them. The authors commit themselves to produce solutions in Agda, Abella and Beluga. To resurrect the slogan from the PC, a small step (excuse the pun) for us, a big step for bringing mechanized meta-theory to the masses!

REFERENCES

A. Abel. Normalization for the simply-typed lambda-calculus in twelf. *Electronic Notes in Theoretical Computer Science*, 199:3 – 16, 2008. ISSN 1571-0661. doi: http://dx.doi.org/10.1016/j.entcs.2007.11.009. Proceedings of the Fourth International Workshop on Logical Frameworks and Meta-Languages (LFM 2004).

A. Abel and A. Vezzosi. A formalized proof of strong normalization for guarded recursive types. In *APLAS*, volume 8858 of *Lecture Notes in Computer Science*, pages 140–158. Springer, 2014.

A. Ahmed. Verified compilers for a multi-language world. In *SNAPL*, volume 32 of *LIPIcs*, pages 15–31. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

T. Altenkirch. A formalization of the strong normalization proof for system F in LEGO. In *TLCA*, volume 664 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 1993.

T. Altenkirch and A. Kaposi. Type theory in type theory using quotient inductive types. In *POPL*, pages 18–29. ACM, 2016.

A. W. Appel and D. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst.*, 23(5):657–683, Sept. 2001. ISSN 0164-0925. doi: 10.1145/504709.504712.

B. E. Aydemir, A. Bohannon, M. Fairbairn, J. N. Foster, B. C. Pierce, P. Sewell, D. Vytiniotis, G. Washburn, S. Weirich, and S. Zdancewic. Mechanized metatheory for the masses: The POPLMARK challenge. In *Eighteenth International Conference on Theorem Proving in Higher Order Logics*, volume 3603 of *LNCS*, pages 50–65. Springer, 2005.

D. Baelde, K. Chaudhuri, A. Gacek, D. Miller, G. Nadathur, A. Tiu, and Y. Wang. Abella: A system for reasoning about relational specifications. *J. Formalized Reasoning*, 7 (2):1–89, 2014.

N. Benton and C. Hur. Step-indexing: The good, the bad and the ugly. In *Modelling, Controlling and Reasoning About State*, volume 10351 of *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2010.

W. J. Bowman and A. Ahmed. Noninterference for free. In *ICFP*, pages 101–113. ACM, 2015.

A. Cave and B. Pientka. A case study on logical relations using contextual types. In *LFMTP*, volume 185 of *EPTCS*, pages 33–45, 2015.

T. Coquand. An algorithm for testing conversion in type theory. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 255–279. Cambridge University Press, 1991.

K. Crary. Logical relations and a case study in equivalence checking. In B. C. Pierce, editor, *Advanced Topics in Types and Programming Languages*. The MIT Press, 2005.

N. de Bruijn. A plea for weaker frameworks. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 40–67. Cambridge University Press, 1991.

K. Donnelly and H. Xi. A formalization of strong normalization for simply-typed lambda-calculus and system F. *Electr. Notes Theor. Comput. Sci.*, 174(5):109–125, 2007.

A. Felty, A. Momigliano, and B. Pientka. The next 700 challenge problems for reasoning with higher-order abstract syntax representations: Part 2—a survey. *Journal of Automated Reasoning*, 55(4):307–372, 2015a. ISSN 0168-7433. doi: 10.1007/s10817-015-9327-3.

A. Felty, A. Momigliano, and B. Pientka. Benchmarks for reasoning with syntax trees containing binders and contexts of assumptions. *MATHEMATICAL STRUCTURES IN COMPUTER SCIENCE*, 2017. doi: 10.1017/S0960129517000093.

A. P. Felty, A. Momigliano, and B. Pientka. An open challenge problem repository for systems supporting binders. In *LFMTP*, volume 185 of *EPTCS*, pages 18–32, 2015b.

J.-Y. Girard, Y. Lafont, and P. Tayor. *Proofs and types*. Cambridge University Press, 1990.

R. Harper and F. Pfenning. On equivalence and canonical forms in the LF type theory. *ACM Transactions on Computational Logic*, 6(1):61–101, 2005.

F. Joachimski and R. Matthes. Short proofs of normalization for the simply- typed λ-calculus, permutative conversions and Gödel's T. *Archive for Mathematical Logic*, 42(1):59–87, Jan 2003. ISSN 1432-0665. URL http://dx.doi.org/10.1007/s00153-002-0156-9.

J. Narboux and C. Urban. Formalising in nominal isabelle crary's completeness proof for equivalence checking. *Electr. Notes Theor. Comput. Sci.*, 196:3–18, 2008.

G. Neis, C. Hur, J. Kaiser, C. McLaughlin, D. Dreyer, and V. Vafeiadis. Pilsner: a compositionally verified compiler for a higher-order imperative language. In *ICFP*, pages 166–178. ACM, 2015.

B. Pientka. Proof pearl: The power of higher-order encodings in the logical framework LF. In *Twentieth International Conference on Theorem Proving in Higher-Order Logics*, LNCS, pages 246–261. Springer, 2007.

B. Pientka and A. Cave. Inductive beluga: Programming proofs. In *CADE*, volume 9195 of *Lecture Notes in Computer Science*, pages 272–281. Springer, 2015.

F. Rabe and K. Sojakova. Logical relations for a logical framework. *ACM Trans. Comput. Logic*, 14(4):32:1–32:34, Nov. 2013. doi: 10.1145/2536740.2536741.

M. H. Sorensen. Strong normalization from weak normalization in typed λ-calculi. *Information and Computation*, 133(1):35 – 71, 1997. ISSN 0890-5401. doi: http://dx.doi.org/10.1006/inco.1996.2622. URL http://www.sciencedirect.com/science/article/pii/S089054019692622X.

R. Statman. Logical relations and the typed lambda-calculus. *Information and Control*, 65(2/3):85–97, 1985.

G. Sutcliffe. The TPTP problem library and associated infrastructure. *Journal of Automated Reasoning*, 43(4):337–362, 2009.

C. Urban, J. Cheney, and S. Berghofer. Mechanizing the metatheory of lf. *ACM Trans. Comput. Logic*, 12(2):15:1–15:42, Jan. 2011. ISSN 1529-3785. doi: 10.1145/1877714.1877721.

Y. Wang, K. Chaudhuri, A. Gacek, and G. Nadathur. Reasoning about higher-order relational specifications. In *Fifteenth International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming*, pages 157–168. ACM Press, 2013.