Mechanized Proofs of Type Safety for a Family of Lambda Calculi with References

Michalis A. Papakyriakou Nikolaos S. Papaspyrou
School of Electrical and Computer Engineering, National Technical University of Athens, Greece.
mpapakyr@softlab.ntua.gr nickie@softlab.ntua.gr

Introduction Type systems typically guarantee a number of interrelated safety properties, e.g. memory safety (programs can only access appropriate memory locations) and control safety (programs can only transfer control to appropriate program points). Given a formal definition of a programming language, researchers are interested in proving *type safety*, in the sense that the static type system precludes classes of dynamic errors when executing programs.

This work presents our experience with the mechanical proof of type safety for the following family of λ-calculi, using the Isabelle/HOL proof assistant [8].

- $\lambda^{\rightarrow}$ is the simply typed λ-calculus (used as reference);
- λ^{ref} is an extension of $\lambda^{\rightarrow}$ with ML-style mutable references supporting reference allocation, dereferencing and type-preserving (weak) assignment [10, ch.13];
- $\lambda^{\forall,\,\mathrm{ref}}$ is an extension of F_2 with (polymorphic) references, as above, under the value restriction; and
- $\lambda^{\mathrm{ref,\,free}}$ is an extension of λ^{ref} with a linear type system supporting safe deallocation (`free`) of references.

Related Work The study of languages with polymorphic references and their metatheory is not new. Tofte proved type safety for polymorphic references using co-induction and Harper showed how a proof can be arranged so that there is no need for co-induction Recent research has focused on fully-fledged languages with mutable references, such as ML [6, 2] and Java [12, 5]. Substructural type systems in relation to reference and region management have also been investigated by many researchers recently [14, 4]. Mechanized proofs exist in the Twelf proof assistant.

Our Approach and Results The proofs for the first three languages do not deviate much from the standard style for call-by-value typed λ-calculi [10], with the exception of *substitution functions* for types that were used in the proof for $\lambda^{\forall,\,\mathrm{ref}}$. In the proof for $\lambda^{\mathrm{ref,\,free}}$, however, which combines references with a substructural type system [11, ch.1], we encountered several issues pertaining to proof engineering that are worth mentioning (and cannot be presented in this confined space). For the representation of bound variables we used the "locally nameless" technique [7, 1]. Table 1 gives an estimate of the size of the proofs.[1]

Conclusion To the best of our knowledge, this is first fully mechanized type safety proof in Isabelle/HOL for a language with mutable references and impredicative polymorphism, $\lambda^{\forall,\,\mathrm{ref}}$, and for a language with linear references and safe deallocation, $\lambda^{\mathrm{ref,\,free}}$.

Future Work We intend to investigate the interplay between linear references (used in deallocation and strong assignment) and unrestricted references (used in dereferencing and weak assignment). The language $\lambda^{\mathrm{ref,\,free}}$ is not useful without a safe way to convert between these two. However, the `let!` construct in Wadler's work [13] and other similar constructs [9, 3] usually impose severe restrictions to maintain type safety or excessively complicate the language. We are currently looking for natural, type safe extensions of $\lambda^{\mathrm{ref,\,free}}$ that do not suffer from these problems.

[1] The complete Isabelle/HOL source code is available from ftp://ftp.softlab.ntua.gr/pub/users/nickie/papers/wmm2007.tar.gz.

Table 1. Lines of code in Isabelle/HOL, per language and file.

File	$\lambda^{\rightarrow}$	λ^{ref}	$\lambda^{\forall,\,\mathrm{ref}}$	$\lambda^{\mathrm{ref,\,free}}$
`Environ.thy`	46	46	46	46
`Syntax.thy`	94	116	699	139
`Typing.thy`	74	83	738	366
`Semantics.thy`	47	143	138	231
`Metatheory.thy`	153	553	1151	2865
Total	414	941	2772	3647

References

[1] B. Aydemir, A. Charguéraud, B. C. Pierce, and S. Weirich. Engineering aspects of formal metatheory, 2007. Manuscript, available from www.cis.upenn.edu/~bcpierce/papers/binders.pdf.

[2] C. Dubois. Proving ML type soundness within Coq. In *Proc. TPHOLs '00*, pp. 126–144, 2000.

[3] M. Fahndrich and R. DeLine. Adoption and focus: Practical linear types for imperative programming. In *Proc. PLDI '02*, pp. 13–24, 2002.

[4] M. Fluet, G. Morrisett, and A. Ahmed. Linear regions are all you need. In *Proc. ESOP'06*, pp. 7–21, 2006.

[5] G. T. Leavens, D. A. Naumann, and S. Rosenberg. Preliminary definition of core JML. Technical Report CS 2006-07, Stevens Institute of Technology, 2006.

[6] D. K. Lee, K. Crary, and R. Harper. Towards a mechanized metatheory of Standard ML. In *Proc. POPL '07*, pp. 173–184, 2007.

[7] C. McBride and J. McKinna. Functional pearl: I am not a number; I am a free variable. In *Proc. ACM SIGPLAN Haskell Workshop*, pp. 1–9, 2004.

[8] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, LNCS vol. 2283. Springer, 2002. Updated documentation available from www.cl.cam.ac.uk/research/hvg/Isabelle/documentation.html.

[9] M. Odersky. Observers for linear types. In *Proc. ESOP '92*, pp. 390–407, 1992.

[10] B. C. Pierce. *Types and Programming Languages*. MIT Press, 2002.

[11] B. C. Pierce, editor. *Advanced Topics in Types and Programming Languages*. MIT Press, 2005.

[12] D. von Oheimb. *Analyzing Java in Isabelle/HOL: Formalization, Type Safety and Hoare Logic*. PhD thesis, Technische Universität München, 2001.

[13] P. Wadler. Linear types can change the world! In *Proc. IFIP TC 2 Working Conference on Programming Concepts and Methods*, pp. 347–359, 1990.

[14] D. Walker and K. Watkins. On regions and linear types. In *Proc. ICFP '01*, pp. 181–192, 2001.