

Replace this file with `prentcsmacro.sty` for your meeting,
or with `entcsmacro.sty` for your meeting. Both can be
found at the [ENTCS Macro Home Page](#).

Formalizing adequacy

James Cheney¹ Rene Vestergaard² Michael Norrish³

Abstract

Adequacy is an important criterion for judging the correctness of formal reasoning. The issue is particularly subtle in the expansive case of approaches to languages with name-binding. We posit that adequacy of a novel representation technique is best addressed by formalizing an *isomorphism* or, more generally, an *interpretation* explicating the new approach in terms of a more conventional one. We present an example formalization of an isomorphism relating nominal and higher-order abstract syntax techniques. We also outline steps towards a systematic framework that could be used for proving adequacy results automatically, which we believe would help make representation techniques more transparent to end-users of mechanized metatheory verification systems, and provide insight into the relative merits of different approaches.

Keywords: adequacy, isomorphism, interpretation, nominal abstract syntax, higher-order abstract syntax

1 Introduction

Adequacy is an important criterion for judging the correctness of formal reasoning about languages with name-binding. Intuitively, adequacy means that an object language corresponds with its representation in a formal system in a way that justifies reasoning about the object language via its representation. The term adequacy was introduced by Harper, Honsell and Plotkin in their seminal work on LF [3]. However, as Crary and Harper have pointed out, adequacy is also important for other formalisms for mechanized metatheory, including nominal techniques [1].

To avoid confusion, we introduce a distinction between two senses of the term *adequacy*. By *informal adequacy* we mean that the formalization of a mathematical object matches its informal definition. Since informal mathematical definitions are sometimes ambiguous or subjective, informal adequacy is in general a subjective judgment, and thus beyond the scope of formalization. Conversely, by *formal adequacy* we mean that two candidate formalizations of an object language are (provably) equivalent in an appropriate sense. Formal adequacy cannot guarantee informal adequacy; however, it is comforting if we can prove that a novel representation is equivalent to a well-understood, conventional one (and disconcerting if we

¹ University of Edinburgh, Scotland. Email: jcheney@inf.ed.ac.uk

² JAIST, Ishikawa, Japan. Email: vester@jaist.ac.jp

³ NICTA, Australia. Email: Michael.Norrish@nicta.com.au

cannot). Henceforth, we just write “adequacy” for “formal adequacy”.

An LF representation is considered adequate with respect to an object language (equipped with a suitable notion of substitution) provided there exists a *bijection* from the object-language terms to LF terms of an appropriate type that is *compositional* in the sense that the object-language substitution maps to metalanguage substitution. Proving adequacy properties in LF requires first establishing that LF terms have unique β -normal, η -long canonical forms [4,5], and once this is done, showing that a given encoding function is bijective and compositional.

Verifying adequacy is essential for avoiding subtle errors in reasoning. However, adequacy proofs are tiresome and are typically conducted on paper using informal approaches to name-binding; thus, they share the well-known disadvantages of paper-and-pencil syntactic proofs. There are a few treatments of adequacy for other techniques besides LF [3,4,5,8], notably Crole’s [2] development of adequacy for the Hybrid system, Norrish and Vestergaard’s [7] formalization of isomorphisms among several definitions of the untyped lambda-calculus, and Urban’s isomorphism between classical and nominal representations of the lambda-calculus [9, Sec. 3]. As far as we know only Urban’s and Norrish and Vestergaard’s work has been mechanically formalized. There appears to be little agreement as to how to define adequacy for other representations besides higher-order abstract syntax.

In this paper, we advocate formalizing adequacy via isomorphisms or interpretations among appropriate structures, following standard definitions in model theory [6]. Using Nominal Isabelle/HOL [9], we verify an isomorphism between canonical higher-order abstract syntax and nominal abstract syntax representations of the untyped lambda-calculus. This result has not been formalized mechanically before as far as we know, and adds to the existing body of evidence [7] that these and other formalisms correctly capture the “real” lambda-calculus. However, isomorphisms are limited to structures that share a common signature. To avoid this limitation, we introduce a more general form of adequacy based on *interpretations* and speculate on the development of a systematic framework for adequacy that would decrease the effort involved in proving adequacy results and shed light upon the relationships among different representation formalisms.

2 Adequacy via isomorphism: an example

We take the view that the goal of adequacy is to justify reasoning about one structure by reasoning about another. It is nontrivial to relate different formalisms because they may use entirely different notions to talk about “the same” things. The concept of isomorphism is a natural way to capture what is really “the same” about two candidate representations of, say, the lambda-calculus, while allowing us to abstract away from implementation details.

Here, we consider first-order relational structures. Concretely, let σ be a relational signature and \mathcal{M}, \mathcal{N} be models of σ (in the usual sense in first-order logic). Recall that an isomorphism of relational structures is a function $h : \mathcal{M} \rightarrow \mathcal{N}$ that is bijective and such that for every n -ary relation $R \in \sigma$, we have $R^{\mathcal{M}}(x_1, \dots, x_n) \iff R^{\mathcal{N}}(h(x_1), \dots, h(x_n))$. Isomorphisms among first-order structures preserve all properties that are first-order definable using the relations in σ . Thus, as a first approx-

$$\begin{array}{c}
 M ::= c \mid x \mid MN \mid \lambda x.N \quad A ::= a \mid A \rightarrow A' \\
 \frac{x : A \in \Gamma}{\Gamma \vdash x \uparrow A} \quad \frac{c : A \in \Sigma}{\Gamma \vdash c \uparrow A} \quad \frac{\Gamma \vdash M \uparrow A_1 \rightarrow A_2 \quad \Gamma \vdash N \downarrow A_1}{\Gamma \vdash M N \uparrow A_2} \quad \frac{\Gamma \vdash M \uparrow A}{\Gamma \vdash M \downarrow A} \quad \frac{\Gamma, x : A_1 \vdash M \uparrow A_2}{\Gamma \vdash \lambda x.M \downarrow A_1 \rightarrow A_2} \\
 \Sigma_\lambda = \{\Lambda : \text{type}, \text{lam} : (\Lambda \rightarrow \Lambda) \rightarrow \Lambda, \text{app} : \Lambda \rightarrow \Lambda \rightarrow \Lambda\} \\
 \frac{}{\text{app}(\text{lam}(\lambda x.M) N) \rightsquigarrow M[N/x]} \quad \frac{M \rightsquigarrow M'}{\text{lam}(\lambda x.M) \rightsquigarrow (\lambda x.M')} \\
 \frac{M \rightsquigarrow M'}{\text{app } M N \rightsquigarrow \text{app } M' N} \quad \frac{N \rightsquigarrow N'}{\text{app } M N \rightsquigarrow \text{app } M N'}
 \end{array}$$

 Fig. 1. Syntax, typing rules, signature, and reduction rules for λ^{HO} .

$$h(x) = x \quad h(t u) = \text{app } (h(t)) (h(u)) \quad h(\lambda x.t) = \text{lam } (\lambda x.h(t))$$

Fig. 2. Encoding function

imation we define \mathcal{M} adequate with respect to \mathcal{N} to mean that \mathcal{M} and \mathcal{N} are isomorphic. This property is symmetric in \mathcal{M} and \mathcal{N} , but often \mathcal{M} is taken to be a more concrete or well-understood formalization and \mathcal{N} is taken to be more abstract.

For example, consider σ_λ consisting of a binary relation symbol \rightsquigarrow . First-order formulas over σ_λ capture many standard properties of the lambda-calculus. Norrish and Vestergaard formalized σ_λ -isomorphisms among several representations of the λ -calculus, including de Bruijn and nominal representations. We have formalized an isomorphism between nominal datatypes λ^N and λ^{HO} corresponding to plain λ -terms and a higher-order syntax presentation of λ -terms. We omit the details of λ^N (which can be found in [7,9]), and focus on λ^{HO} and the translation $h : \lambda^N \rightarrow \lambda^{HO}$.

Higher-order abstract syntax is based on a metalanguage consisting of a typed lambda-calculus with constants, quotiented by α , β , and η -equivalence; see Figure 1. For clarity, we write t, u for object-terms and M, N for metalanguage terms. Following Harper and Licata [4], to avoid having to deal with β and η -equivalence here, we assume that expressions are maintained in canonical forms, according to the typing rules in Figure 1. The signature Σ_λ defines a type constant Λ and term constants lam and app . We define λ^{HO} as the set of terms $\{M \mid \exists \Gamma \in \Lambda^*. \Gamma \vdash M \downarrow \Lambda\}$, where Λ^* is the set of contexts Γ such that every binding in Γ is of the form $x : \Lambda$.

Theorem 2.1 *The encoding function $h : \lambda^N \rightarrow \lambda^{HO}$ is an isomorphism of σ_λ -structures.*

Proof. [Outline] We have defined λ^N, λ^{HO} , and their associated substitution functions and reduction relations in Nominal Isabelle/HOL using the Nominal Datatype Package [9], and verified that h is injective, surjective, commutes with substitution, and preserves and reflects reduction. It turns out to be easier to prove these properties for untyped HOAS terms, using an inductive predicate *range* that (provably) characterizes the range of h . It then suffices to show that *range*(M) holds iff $\exists \Gamma \in \Lambda^*. \Gamma \vdash M \downarrow \Lambda$ holds. This seems straightforward in principle, but the reverse direction is nontrivial to formalize because there are many inversion steps and cases to consider, and it appears necessary to proceed by height induction on M .

Another subtlety is in the proof of reflection. It is difficult to prove that $h(t) \rightsquigarrow$

$h(u)$ implies $t \rightsquigarrow u$ directly by induction; instead, we define the inverse $h^{-1} : \lambda^{HO} \rightarrow \lambda^N$, and prove that $M \rightsquigarrow N$ implies $h^{-1}(M) \implies h^{-1}(N)$ provided $M, N \in \lambda^{HO}$. However, we cannot easily define h^{-1} by nominal primitive recursion in Nominal Isabelle/HOL. Instead, we show that h is injective, define h^{-1} as the left-inverse to h , and prove that h^{-1} satisfies the equations:

$$h^{-1}(x) = x \quad h^{-1}(\mathbf{app} \ M \ N) = (h^{-1}(M)) \ (h^{-1}(N)) \quad h^{-1}(\mathbf{lam} \ (\lambda x.M)) = \lambda x.h^{-1}(M)$$

We can then easily prove h^{-1} is compositional and preserves reduction on the range of h , hence h reflects reduction.

In our formalization we use ordinary substitution (defined in the usual way using nominal primitive recursion), not hereditary substitution [4]. Ordinary substitution suffices here because it happens to be true that $\Gamma \vdash M \downarrow \Lambda$ implies $\Gamma \vdash M \uparrow \Lambda$, provided $\Gamma \in \Lambda^*$. This would not be true if we distinguished canonical and atomic forms at the syntactic level (as in [4,5]), or if Γ could contain function variables. \square

3 A systematic approach based on interpretations

To facilitate reasoning about adequacy, we envision identifying classes of object languages definable using various representations, and proving general adequacy results systematically for all languages in such classes. For example, we conjecture that all languages defined via nominal datatypes can be translated to “equivalent” de Bruijn, HOAS, or other conventional approaches. However, care must be taken with the definition of the appropriate notion of “equivalence” here. Recall that we proved that λ^N and λ^{HO} were isomorphic with respect to a fixed σ_λ . But in general we do not know what the signatures will be; moreover, the two structures might have different natural signatures. How can we consider one structure to be equivalent to another one that might be using different basic language?

We draw inspiration from the idea of *interpretation* in model theory [6]. Let \mathcal{M}, \mathcal{N} be relational structures over (possibly different) signatures σ and τ respectively. An *interpretation* $\Gamma = (\gamma, P_\Gamma, R_\Gamma, \dots)$ of (\mathcal{M}, σ) in (\mathcal{N}, τ) consists of: a function $\gamma : \mathcal{N} \rightarrow \mathcal{M}$, a τ -formula $P_\Gamma(x)$ defining a subset $\mathcal{N}_\Gamma \subseteq \mathcal{N}$ such that γ is surjective from \mathcal{N}_Γ , i.e., $\gamma[\mathcal{N}_\Gamma] = \mathcal{M}$, and for each n -ary $R \in \sigma$, a τ -formula $R_\Gamma(x_1, \dots, x_n)$ such that for every $a_1, \dots, a_n \in \mathcal{N}_\Gamma$, we have $\mathcal{M} \models R(\gamma(a_1), \dots, \gamma(a_n))$ if and only if $\mathcal{N} \models R_\Gamma(a_1, \dots, a_n)$. Moreover, we can extend any interpretation Γ to a map from σ -formulas ϕ to τ -formulas ϕ_Γ by replacing occurrences of $R(\bar{x})$ with $R_\Gamma(\bar{x})$ and relativizing quantifiers by taking $(\forall x.\phi(x))_\Gamma := \forall x.P_\Gamma(x) \Rightarrow (\phi(x))_\Gamma$. It is important to note that we treat equality (if present) as part of the signature, so that equality in \mathcal{M} may be interpreted as an arbitrary formula $=_\Gamma$.

Theorem 3.1 (Thm. 4.3.1 [6]) *If $\Gamma : (\mathcal{M}, \sigma) \rightarrow (\mathcal{N}, \tau)$ is an interpretation then $\mathcal{M} \models \phi(\gamma(\bar{a})) \iff \mathcal{N} \models \phi_\Gamma(\bar{a})$ for all $\bar{a} \in \mathcal{N}_\Gamma$.*

Thus, an interpretation yields an “isomorphic copy” of \mathcal{M} inside \mathcal{N} , without requiring \mathcal{M} and \mathcal{N} to use the same language. Interpretations are closed under composition and form a natural class of morphisms among pairs (\mathcal{M}, σ) of signatures and structures. Any isomorphism of structures over the same signature (as, for example, in the previous section) clearly extends to an interpretation (in both

directions). Interpretations can also be used to formulate adequacy theorems among different formalisms, for example:

Example 3.2 Consider structures $(\lambda_\alpha, \rightsquigarrow)$, the set of lambda-terms quotiented by α -equivalence, and $(\lambda_{\text{raw}}, \rightsquigarrow, \equiv)$, the set of raw lambda-terms equipped with reduction and alpha-equivalence relations; λ_α is interpreted in λ_{raw} by $\Gamma : (\lambda_\alpha, \rightsquigarrow) \rightarrow (\lambda_{\text{raw}}, \rightsquigarrow, \equiv_\alpha)$, where $\gamma(x) = [x]_{\equiv}$, $P_\Gamma(x) = \text{true}$, and $x \rightarrow_\Gamma y = \exists x', y'. x \equiv_\alpha x' \wedge x' \rightsquigarrow y' \wedge y' \equiv_\alpha y$.

4 Future directions

To our knowledge, the most rigorous previous treatments of adequacy for HOAS are Harper and Pfenning [5, Sec. 7] and Harper and Licata [4, Sec. 3]. Our adequacy result considers a smaller object-language and simply-typed meta-language, but we have completely formalized it in Nominal Isabelle/HOL. It would be worthwhile to carry out a similar development of adequacy for a dependently-typed LF encoding of a multi-sorted object language such as the typed lambda-calculus, perhaps following the development in Harper and Licata [4].

Verifying adequacy does not seem inherently difficult, but is tedious and must currently be redone for each new representation. Instead of doing this on a case-by-case basis, we are investigating fundamental principles for systematically reasoning about adequacy. In [7], we considered *structural collapses*, which are intermediate between isomorphisms and interpretations; in fact, the γ in Example 3.2 is a structural collapse. An adaptation of the First Isomorphism Theorem of Algebra tells us that two structures obtained, e.g., from raw syntax, by two collapsing functions with equal kernels will be isomorphic. Conversely, we have started investigating *inverse isomorphism* results that give sufficient conditions for when two structures that collapse to a common formalism will enjoy similar properties. We plan to integrate algebraic and model-theoretic techniques in the context of interpretations.

References

- [1] Karl Crary and Robert Harper. Higher-order abstract syntax: setting the record straight. *SIGACT News*, 37(3):93–96, 2006.
- [2] Roy Crole. Hybrid adequacy. Technical Report CS-06-011, University of Leicester, 2006.
- [3] Robert Harper, Furio Honsell, and Gordon D. Plotkin. A framework for defining logics. *J. ACM*, 40(1):143–184, 1993.
- [4] Robert Harper and Daniel R. Licata. Mechanizing metatheory in a logical framework. *J. Funct. Program.*, 17(4-5):613–673, 2007.
- [5] Robert Harper and Frank Pfenning. On equivalence and canonical forms in the LF type theory. *ACM Trans. Comput. Log.*, 6(1):61–101, 2005.
- [6] Wilfrid Hodges. *A shorter model theory*. Cambridge University Press, 1997.
- [7] Michael Norrish and René Vestergaard. Proof pearl: De Bruijn terms really do work. In Klaus Schneider and Jens Brandt, editors, *TPHOLS*, volume 4732 of *Lecture Notes in Computer Science*, pages 207–222. Springer, 2007.
- [8] F. Pfenning. Logical frameworks. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 17, pages 1063–1147. Elsevier Science, 2001.
- [9] Christian Urban. Nominal techniques in Isabelle/HOL. *J. Autom. Reasoning*, 40(4):327–356, 2008.