Embedding Higher-Order Abstract Syntax in Type Theory

Steven Schäfer and Kathrin Stark

Saarland University, Saarbrücken, Germany
{schaefer,kstark}@ps.uni-saarland.de

Higher-order abstract syntax (HOAS) [10] offers a direct representation of higher-order languages, where capture-avoiding substitution is function application, substitution lemmas hold by definition and all derived constructions respect substitution. However, embedding HOAS directly into type theory (and thus profiting from its perks) is difficult, since HOAS relies on an intensional host language function space, while type theoretic function spaces are extensional [6]. We thus propose an indirect approach via embeddings.

An embedding of a HOAS signature into type theory is a term model of the signature. A model gives an interpretation of terms, well-behaved substitutions, and (substitution respecting) recursive definitions on terms. Previous approaches have focused on term representations and renaming - essentially modelling *weak* HOAS [4, 11, 3].

For full models, we extend Hofmann's presheaf semantics for a second-order signature [6], to a construction for arbitrary HOAS signatures (Figure 1). A model consists of a *category of contexts* $\mathbb{D}$ whose morphisms are substitutions and presheaves for every term sort. The action of a substitution on a term gives the notion of instantiation $s[\sigma]$.

The main obstacle to constructing a term model as an inductive type are negative occurrences (tm in lam). Hofmann's insight is that this problem disappears when Tm is representable, i.e., $\mathsf{Tm} = \mathrm{Hom}(\_, T)$ and $\mathbb{D}$ has finite products. In this case, the natural transformation $\mathsf{Lam} : \mathsf{Tm}^{\mathsf{Tm}} \to \mathsf{Tm}$ can equivalently be given by a natural transformation with components $\mathsf{Lam}_X : \mathsf{Tm}(T \times X) \to \mathsf{Tm}(X)$. For every signature, we have to construct a category $\mathbb{D}$.

As a first step, we create the corresponding weak HOAS signature (Figure 1 (b)), in which all non-strictly positive occurrences of a sort (tm) are replaced by a new type v of variables together with a *variable constructor* (var). We build a presheaf model for a weak HOAS signature by constructing a category $\mathbb{C}$ with enough distinct non-terminal objects to represent each sort of variables. In the case of the lambda calculus, any cartesian category with a non-terminal object T will work and we define $V = \mathrm{Hom}(\_, T)$. The canonical choice for $\mathbb{C}$ is the free cartesian category on one object. We model the term sorts by the corresponding inductive types.

We have previously considered presheaf models for weak HOAS signatures [8]. In the present context, the results of [8] show that this construction gives a model for a weak HOAS signature and moreover that such models always extend to models of the corresponding HOAS signature. For terms, this shows that $\mathsf{Tm} : \widehat{\mathbb{C}}$ is a monad relative to V. The construction then extends Tm

$\mathsf{tm} : *$	$\mathsf{Tm} : \widehat{\mathbb{D}}$
$\mathsf{app} : \mathsf{tm} \to \mathsf{tm} \to \mathsf{tm}$	$\mathsf{App} : \mathsf{Tm}^2 \to \mathsf{Tm}$
$\mathsf{lam} : (\mathsf{tm} \to \mathsf{tm}) \to \mathsf{tm}$	$\mathsf{Lam} : \mathsf{Tm}^{\mathsf{Tm}} \to \mathsf{Tm}$

(a) HOAS

$\mathsf{tm}, \mathsf{v} : *$	$\mathsf{Tm}, \mathsf{V} : \widehat{\mathbb{C}}$
$\mathsf{var} : \mathsf{v} \to \mathsf{tm}$	$\mathsf{Var} : \mathsf{V} \to \mathsf{Tm}$
$\mathsf{app} : \mathsf{tm} \to \mathsf{tm} \to \mathsf{tm}$	$\mathsf{App} : \mathsf{Tm}^2 \to \mathsf{Tm}$
$\mathsf{lam} : (\mathsf{v} \to \mathsf{tm}) \to \mathsf{tm}$	$\mathsf{Lam} : \mathsf{Tm}^{\mathsf{V}} \to \mathsf{Tm}$

(b) Weak HOAS

Figure 1: (Weak-) HOAS signatures and presheaf models for the lambda calculus.

to a (representable) presheaf over the Kleisli category of Tm (our chosen $\mathbb{D}$). Additionaly, we obtain a recursion principle with respect to other models of the signature.

The construction of [8] applies to simply typed second-order signatures. We extend this construction to essentially arbitrary HOAS signatures. In the general case, the reduction to a weak HOAS signature can be recursive, the construction has to be stratified in the case of several sorts of terms, and the final relative monad contains more than a single sort of terms. This construction yields several interesting cases, when we consider certain concrete HOAS signatures.

- **Well-typed Terms.** For well-typed terms, the resulting contexts contain a type for each variable. In this case, our construction yields the relative monad of well-typed terms described in [2].

- **Stratified and Mutual Inductive Types.** Applying the construction to signatures with several sorts with binders, such as the types and terms of System F, yields vector parallel substitutions [7, 8]. The category of contexts is the finite product of the respective contexts for all occuring subsorts. In the case of System F we find that the product of the presheaf of types and terms together form a relative monad on the category of contexts.

- **Third-order signatures.** While most practical work has focused on second-order signatures, some applications require higher-order signatures. Consider the $\lambda\mu$-calculus [9, 1], which extends the lambda calculus with an additional constructor.

  $$\mu : ((\mathsf{tm} \to \mathsf{cont}) \to \mathsf{cont}) \to \mathsf{tm}$$

  Our construction yields a weak version with additional variable sort w and variable constructor $\mathsf{var} : \mathsf{w} \to \mathsf{tm} \to \mathsf{cont}$. The final result of our translation corresponds to Parigot's original first-order definition of $\lambda\mu$-terms [9], with a novel notion of instantiation which generalizes the structural substitution of $\lambda\mu$.

- **Type Systems.** Since presheaves can model dependent types we can extend our construction to HOAS predicates, i.e., type systems. The result of the translation is a form of relational typing as in [8], with general "hypothetical" judgments in the context.

  Our translation comes with a suitable notion of substitution on the type system, e.g.:

  $$\frac{\Gamma \vdash s : A \qquad \forall (t : B) \in \Gamma.\ \ \Delta \vdash t[\sigma] : B}{\Delta \vdash s[\sigma] : A}$$

  This corresponds to the notion of *context morphism lemmas* – which in [5] are defined as "parallel substitutions with additonal typing and well-formedness information". Our approach pins down exactly *what* additional information is necessary and identifies context morphism lemmas as the correct notion of substitution on typing judgments.

- **Induction Principles.** Applying the translation to the trivial predicate on terms which states that a term is built from constructors yields a useful induction principle for terms. For the lambda calculus, the induction principle states that given a substitutive predicate $P : \mathsf{tm} \to \mathsf{Prop}$ for which $P\,s \to P\,t \to P(\mathsf{app}\,s\,t)$ and $(\forall t\sigma.\ P\,t \to P\,s[t \cdot \sigma]) \to P(\mathsf{lam}\,s)$ we have $P\,s[\sigma]$ for all terms s and substitutions σ which satisfy $P(\sigma\,i)$ for all i.

Future Work. Ultimately, we are not interested in an arbitrary presheaf model, but in a "free" model. What this should mean is currently an open question. On the practical side we are currently implementing our construction as part of a tool to embed HOAS signatures into Coq.

References

[1] Andreas Abel. A third-order representation of the $\lambda\mu$-calculus. *Electronic Notes in Theoretical Computer Science*, 58(1):97–114, 2001.

[2] Thorsten Altenkirch, James Chapman, and Tarmo Uustalu. Monads need not be endofunctors. In *International Conference on Foundations of Software Science and Computational Structures*, pages 297–311. Springer, 2010.

[3] Adam Chlipala. Parametric higher-order abstract syntax for mechanized semantics. In *ACM Sigplan Notices*, volume 43, pages 143–156. ACM, 2008.

[4] Joëlle Despeyroux, Amy Felty, and André Hirschowitz. Higher-order abstract syntax in coq. In *International Conference on Typed Lambda Calculi and Applications*, pages 124–138. Springer, 1995.

[5] Healfdene Goguen and James McKinna. Candidates for substitution. *LFCS report series-Laboratory for Foundations of Computer Science ECS LFCS*, 1997.

[6] Martin Hofmann. Semantical analysis of higher-order abstract syntax. In *Logic in Computer Science, 1999. Proceedings. 14th Symposium on*, pages 204–213. IEEE, 1999.

[7] Jonas Kaiser, Steven Schäfer, and Kathrin Stark. Autosubst 2: Towards reasoning with multi-sorted de bruijn terms and vector substitutions. In *Proceedings of the Workshop on Logical Frameworks and Meta-Languages: Theory and Practice*, LFMTP '17, pages 10–14, New York, NY, USA, 2017. ACM.

[8] Jonas Kaiser, Steven Schäfer, and Kathrin Stark. Binder aware recursion over well-scoped de bruijn syntax. *Certified Programs and Proofs - 7th International Conference, CPP 2018, Los Angeles, USA, January 8-9, 2018*, Jan 2018.

[9] Michel Parigot. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 190–201. Springer, 1992.

[10] Frank Pfenning and Conal Elliott. Higher-order abstract syntax. In *Proceedings of the ACM SIGPLAN'88 Conference on Programming Language Design and Implementation (PLDI), Atlanta, Georgia, USA, June 22-24, 1988*, pages 199–208. ACM, 1988.

[11] Geoffrey Washburn and Stephanie Weirich. Boxes go bananas: Encoding higher-order abstract syntax with parametric polymorphism. *Journal of Functional Programming*, 18(1):87–140, 2008.