An Effectful Way to Eliminate Addiction to Dependence

P.-M. Pédrot[1] and N. Tabareau[2]

[1] University of Ljubljana
[2] Inria Rennes - Bretagne Atlantique

The gap between dependent type theories such as CIC and mainstream programming languages comes to a large extend from the absence of effects in type theories, because of its complex interaction with dependency. For instance, it has already been noticed that inductive types and dependent elimination do not scale well to CPS translations and classical logic [1, 3]. Furthermore, the traditional way to integrate effects from functional programming using monads does not scale to dependency because the monad leaks in the type during substitution, preventing any straightforward adaptation.

To solve this conundrum, we propose Baclofen Type Theory (BTT), a stripped-down version of CIC, together with a family of syntactic models of BTT that allows for a large range of effects in dependent type theory, amongst which exceptions, non-termination, non-determinism or writing operation. By syntactic models, we mean a model directly expressed in a type theory through a program transformation, as advocated in a previous paper [2].

The key feature of BTT lies in the fact it has a restricted version of dependent elimination to overcome the difficulty to marry effects and dependency. Essentially, the restriction appears as a side-condition on the return predicates of dependent pattern-matching, which must be *linear*, in the sense of Munch-Maccagnoni [7]. For instance, the elimination rule for booleans is of the following shape.

$$\frac{\Gamma \vdash M : \mathbb{B} \qquad \begin{array}{l}\Gamma \vdash N_1 : P\{b := \texttt{true}\} \\ \Gamma \vdash N_2 : P\{b := \texttt{false}\}\end{array} \qquad \Gamma, b : \mathbb{B} \vdash P : \square \qquad P \text{ linear in } b}{\Gamma \vdash \texttt{if } M \texttt{ as } b \texttt{ return } P \texttt{ then } N_1 \texttt{ else } N_2 : P\{b := M\}}$$

Linearity is a property of functions in an ambient call-by-name language. Intuitively, it captures the fact that a function is semantically call-by-value, or alternatively, in a more categorical parlance, that it is an algebra homomorphism. As showed by Levy in a recent paper [5], it is possible to provide syntactic underapproximations for linearity, so that the above side-condition can be understood as a *guard condition* similar to the one used for fixpoint productivity in practical CIC implementations. This guard condition is totally oblivious of the ambient effect and does not mention it at all. Furthermore, this restriction is a generalisation of the one we required for the call-by-name forcing translation [4], which was based on storage operators. It turns out that, at least in the non-recursive case, storage operators syntactically turn any predicate into a linear one.

The syntactic models are given by the *weaning translation* of BTT into CIC, using a variant of the traditional monadic translation. The need for this variant can be explained by analyzing the call-by-push-value (CBPV) decomposition of call-by-value and call-by-name reduction strategies. Indeed, the key observation is that the traditional monadic interpretation dating back to Moggi [6] is call-by-value whereas type theories such as CIC are fundamentally call-by-name because they feature an unrestricted conversion rule. Therefore, any effectful model of CIC ought to factor through a call-by-name decomposition in CBPV.

In fact, the weaning translation is somehow dual to the forcing translation through this decomposition, in the sense that they respectively trivialize the $\mathcal{U}$ and $\mathcal{F}$ functors which decompose the ambient monad $\mathtt{T}$ as an adjunction. Most notably, the weaning translation can

be thought of as a variant of the Eilenberg-Moore construction on steroids, where types are translated as plain algebras (i.e. without coherence requirement), which can be easily expressed by the dependent sum

$$\mathbb{T}_i = \Sigma A : \square_i.\, \mathtt{T}\ A \to A.$$

But in CIC, universes satisfy a kind of self-enrichment expressed as $\square_i : \square_{i+1}$. Thus, to get a correct interpretation of universes, the monad needs to satisfy the additional requirement that the type of algebras needs to be itself an algebra of the monad. A monad satisfying this property is said to be *self-algebraic*. We then show how very common monads satisfy this property and thus give rise to effects that can be integrated to BTT. In particular, all free monads are also self-algebraic.

The exception monad is in particular self-algebraic, which allows us to adapt Friedman's A-translation to CIC. We recover the following theorem, which shows that Markov's rule is admissible in CIC, a fact which was, to the best of our knowledge, not known.

Theorem 1. *If* $\vdash_{\mathrm{CIC}} t : \neg\neg A$ *and* A *is a first-order type, then there exists* $\vdash_{\mathrm{CIC}} t^\bullet : A$.

As a matter of fact, in addition to weaning, BTT is also the source theory of our previous forcing translation, even though the two translations sit on two extreme points of CBPV decompositions. This leads us to postulate the following thesis.

BTT *models effectful type theories.*

As it is the case for other syntactic models [4, 2], it is possible to implement the weaning translation as a Coq plugin, thanks to the fact that it is a program translation preserving amongst other things conversion. The plugin is available at `https://github.com/CoqHott/coq-effects`, and allows to give the impression to the user that she lives in an impure theory while everything she writes is compiled on the fly to actual Coq terms.

References

[1] G. Barthe and T. Uustalu. Cps translating inductive and coinductive types. In *Proceedings of Partial Evaluation and Semantics-based Program Manipulation*, pages 131–142. ACM, 2002.

[2] S. Boulier, P.-M. Pédrot, and N. Tabareau. The next 700 syntactical models of type theory. In *Proceedings of Certified Programs and Proofs*, pages 182–194. ACM, 2017.

[3] H. Herbelin. On the degeneracy of sigma-types in presence of computational classical logic. In P. Urzyczyn, editor, *Seventh International Conference, TLCA '05, Nara, Japan. April 2005, Proceedings*, volume 3461 of *Lecture Notes in Computer Science*, pages 209–220. Springer, 2005.

[4] G. Jaber, G. Lewertowski, P.-M. Pédrot, M. Sozeau, and N. Tabareau. The definitional side of the forcing. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 367–376, 2016.

[5] P. B. Levy. Contextual isomorphisms. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, pages 400–414, New York, NY, USA, 2017. ACM.

[6] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, July 1991.

[7] G. Munch-Maccagnoni. Models of a Non-Associative Composition. In A. Muscholl, editor, *17th International Conference on Foundations of Software Science and Computation Structures*, volume 8412, pages 396–410, Grenoble, France, Apr. 2014. Springer.