

Adapting Abstract Machine Equations to Dependent Typing

Alexandra Aiello

December 22, 2025

Contents

1	Overview	1
2	Equational Semantics in the CAM	2
2.1	TODO: Consolidate	2
2.2	TODO: Consolidate—The CAM does not have type-checking by default	2

1 Overview

Lafont’s linear abstract machine (*LAM*) [Laf88] and the categorical abstract machine devised by Cousineau et al. (*CAM*) [CCM85] provide efficient execution schemes for λ -like calculi. Notably, the LAM provides an execution environment for *classical intuitionistic logic*, thereby corresponding to symmetric monoidal closed categories with finite products and coproducts [Laf88, p. 161]. It is unclear what advantages classical linear logic would provide for zero-knowledge proofs, but Lafont’s adaptation of the CAM may provide insight to our adaptation of the CAM.

Ideally, we would extend the CAM’s correspondence with cartesian closed categories (*CCC*’s) [CCM85, p. 178] to locally cartesian closed categories (*LCCC*’s), thereby enabling Martin-Löf Type Theory (*MLTT*). Here, I outline a potential approach for adapting the CAM to achieve this goal by adding explicit dependent type arguments with corresponding meta-combinator registers.

$$\begin{aligned} 0!(x, y) &= y, \quad (n + 1)!(x, y) = n!x, \\ ('x)y &= x, \\ S(x, y)z &= xz(yz), \\ \Lambda(x)yz &= x(y, z) \end{aligned} \tag{1}$$

Figure 1: Syntactic rules motivating the CAM, bearing similarity to the *SK*-calculus [CCM85, p. 176].

2 The CAM Achieves its Correspondence via Equational Semantics

2.1 TODO: Consolidate

- Cousineau et al. motivate the design of the CAM through a well-known semantics for the untyped λ -calculus. [CCM85, p. 175].
- The design of the CAM is derived from a transformation of the semantics for λ -calculus to an equational, syntactic interpretation with some similarity to the *SK*-calculus [CCM85, p. 176].
- They extend the CAM with explicit pairs to more closely align execution with a performant memory-model (i.e., binary-tree environments), thus yielding a correspondence to CCC's [CCM85, p. 178].
- We can keep most of the CAM's design the same, thus inheriting its performance benefits and its support for variable environments.
- As we have seen before, we can extend the *SK*-calculus with arguments for explicit types, enabling polymorphism, or even dependent typing.

2.2 TODO: Consolidate—The CAM does not have type-checking by default

- The CAM does not support explicit typing, but still corresponds to CCC's *by construction*. For example, Cousineau et al. give a compilation from ML to the CAM, but rely on ML's type-checking to achieve this [CCM85, p. 196]. However, in dependent typing, types are somewhat computationally relevant, so this will not be sufficient for our use case.
- The first 3 equations (1) correspond to the *K* rule, while the second two correspond to *S*.

- We don't extend SK with additional combinators, and we will still need pairs. But, our β -reduction rules for SK differ, in that they have explicit type arguments that are also “forgotten”, except at the meta-level.
- We will likely add some kind of meta register with its own pattern-matching rules to enforce our type discipline.
- The linear abstract machine doesn't have types at its base level, either. We are somewhat forced to do our typing *by construction*.
- Doing dependent typing is kind of difficult, since it requires beta reduction. This contradicts the “by construction” viewpoint. There is no beta reduction required in standard linear logic.
- LCCC's are just a category where all the slice categories are CCC's.

References

- [CCM85] G. Cousineau, P-L. Curien, and M. Mauny. “The categorical abstract machine”. In: *Functional Programming Languages and Computer Architecture*. Ed. by Jean-Pierre Jouannaud. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 50–64. ISBN: 978-3-540-39677-2.
- [Laf88] Y. Lafont. “The linear abstract machine”. In: *Theoretical Computer Science* 59.1 (1988), pp. 157–180. ISSN: 0304-3975. DOI: [https://doi.org/10.1016/0304-3975\(88\)90100-4](https://doi.org/10.1016/0304-3975(88)90100-4). URL: <https://www.sciencedirect.com/science/article/pii/0304397588901004>.