Project Literature Review

Alexandra Aiello

December 19, 2025

Contents

1 CAM

Hypothesis: we don't need the LAM. It has some advantages, but its instruction set is quite large. The CAM supports intuitionistic logic by construction. We might do the same by adapting it to LCCC's.

Slice categories (C/A) are easy to identify in the LAM. They are the canonical combinators for the type A. We might do the same for the CAM.

To upgrade to LCCC's, we need every slice category to also be cartesian closed. So, what are the slice categories in CAM? We can just adapt the notion of canonical combinators to CAM:

- $\mathrm{id} : A \to A$
- 50

2 LAM

For this section of the literature review, I focus on Lafont's Linear Abstract Machine [Laf88].

2.1 Hypothesis: LAM Supports Dependent Typing

A system corresponding with locally cartesian-closed categories would support DTT. The LAM has identifiable **slice categories** (C/A), corresponding to *canonical combinators* for some primitive type A.

However, the LAM does not actually correspond to CCC's, but symmetric monoidal closed categories (with finite products and coproducts). These categories are subsets of CCC's.

Indexed monoidal categories are we want to support.

- The *canonical combinators* for some type A correspond roughly to the **slice category** C/A [Laf88, p. 165].
- Canonical combinators are function terms. For example, id : $A \to A$ is the canonical combinator for an primitive type A [Laf88, p. 165].
- Atomic types correspond to builtin basic terms, but pairs (i.e, tensor products) of them are more compatible with a binary tree execution, and constitute *primitive types* [Laf88, p. 165].

2.2 Hypothesis: we can do debruijn decomposition, like in the CAM, to realize dependent typing

2.3 Hypothesis: We can appropriate the Munchhausen method to make our dependent typing for combinators more robust

Notes on the Munchausen method:

- Can "postpone" value assignments when a type depends on itself. For example, church-encoding a Σ type: $\Sigma AB = \lambda b, \text{if } b \text{ then } A \text{ else } \texttt{decl}$
- `decl` gets set "later", after the church-encoding is defined.
- They also give an application to the SK combinators to define dependent types, using the munchausen method.
- Tm represents a well-typed term
- I don't think we can do what's done in the paper in Lean
- I think this is only possible in Agda, not Lean, although we can use Quotients maybe to do this.
- Lean doesn't actually support forward declarations, or rewrite rules the way Agda does, but we can probably do this with quotients

