The eCASM Algebra

A Structural Algebra for Program Semantics
and Instruction–Set Design

Extended Edition (Version 1)

Adrian Diamond
adriandiamond@yahoo.com

November 2025

Permanent Reference URL:
https://ecasm.aadsystems.com/static/ecasm-algebra.pdf

This document is part of the eCASM Research Series.

Table of Contents

Abstract

The eCASM (extended Concise Assembly System Model) research program explores minimal structural vocabularies for reasoning about program semantics, state evolution, and observation. This document presents a conceptual overview of the eCASM perspective, emphasizing algebraic roles and invariant structure rather than execution models or implementation details.

The paper is intended as a foundational and historical reference within the eCASM Research Series, situating the work in relation to categorical semantics, algebraic theories, and program analysis, without serving as a specification or compiler design document.

1 Conceptual Overview

The eCASM perspective is motivated by the observation that many computational systems, across classical and quantum domains, can be described in terms of a small number of structural roles: initialization, transformation, and observation.

Rather than committing to a specific execution model, instruction set, or evaluation strategy, eCASM treats these roles abstractly, allowing semantic interpretation to vary by context. This makes the framework suitable as a unifying conceptual lens rather than a concrete implementation blueprint.

The emphasis of the eCASM Research Series is structural clarity and semantic minimalism, not prescriptive compiler construction.

The eCASM perspective admits multiple downstream elaborations in specialized contexts, which are intentionally outside the scope of this public document.

2 Discussion: Minimality and Expressiveness

The eCASM algebra accomplishes an unusual combination:

- **Minimal syntax**: only four generator classes.
- **Maximal expressiveness**: arbitrary composites of structural roles.
- **Functorial semantics**: every compiler is just a semantic assignment.

The algebra deliberately avoids committing to:

- hardware assumptions,
- quantum or classical bias,
- effects or evaluation rules,
- specific instruction sets.

Instead, it provides a *structural skeleton* upon which arbitrary architectures may be realized.

Permanent Link: `https://ecasm.aadsystems.com/static/ecasm-algebra.pdf`
Version: 1 (Extended Edition)

References

- S. Mac Lane, *Categories for the Working Mathematician.* Springer, 1998.
- B. Jacobs, *Categorical Logic and Type Theory.* Elsevier, 1999.
- J. Baez and M. Stay, "Physics, Topology, Logic, and Computation: A Rosetta Stone," in *New Structures for Physics*, Springer, 2011.
- J. Power, "Enriched Lawvere Theories," *Theory and Applications of Categories*, 2000.