Reverse Rosetta: Decoder Theory and Semantic Invariance in Closed and Open Systems

Fabian Schindler

September 4, 2026

Abstract

We explore the boundaries of semantic reconstructibility and representation in formal systems. We introduce the concept of Operational Closure to mathematically distinguish closed systems (which factor uniquely through a view-independent invariant layer) from open, anchor-dependent systems. We show that while closed dialects (such as lambda calculus and set-theoretic semantics) are fully reconstructible from their invariant quotients, open systems (such as natural languages or undeciphered historical scripts like Linear A) cannot be decoded without external contextual anchors.

Chapter 1

Introduction and Foundational Motivation

Modern programming language design, software engineering, and formal verification are plagued by a profound fragmentation. Disjoint computational models—such as the untyped lambda calculus, term rewriting systems (TRS), e-graphs, compiler intermediate representations (IR), and virtual machine bytecodes—are treated as separate worlds. They occupy distinct namespaces, employ incompatible metatheories, and rely on ad-hoc, pairwise compiler pipelines that leak implementation details and fail to preserve observational boundaries.

This paper argues that this fragmentation is unnecessary. We show that a broad class of closed formal systems can be represented through a single, quotient-mediated semantic kernel without loss of operational behavior. This kernel is not a new "universal language," but rather a view-independent substrate: different programming languages and formalisms become different *decoders* over the same invariant structure.

1.1 Primal Modeling Assumptions

To build a mathematically closed computational substrate, we establish a set of minimal design assumptions. These are not proposed as cosmological truths about reality, but rather as the foundational design requirements for a closed algebraic representation of computation.

Axiom 1 (Substrate Existence)**.** *The universe of discourse (carrier space) is non-empty:* $\Omega \neq \emptyset$.

Axiom 2 (Identity Context)**.** *The substrate contains a unique element representing the empty or inactive context, denoted by* $\emptyset$*, from which distinction begins.*

Axiom 3 (Self-Description Capability)**.** *The representation must support self-referential structures. This requires the algebraic capability to form pairs of carrier names, beginning with the initial pair:* $P = (\emptyset, \emptyset)$.

Axiom 4 (Operational Asymmetry)**.** *To drive computation, the system must distinguish the active operator from the operand it acts upon. In algebraic terms, this asymmetric relationship is modeled via Kuratowski pairing [5]:*

$$(x, y) \triangleq \{\{x\}, \{x, y\}\}$$

This asymmetry, which we denote by Δ*, serves as the operational drive for directed rewriting steps.*

1.2 The 4-Carrier System and Design Heuristics

The computational substrate is spanned by four primitive carriers: Identity (I), Rewrite (R), Adjacency (A), and State (S). We conjecture that exactly four independent roles are necessary and sufficient to form a closed, non-trivial computational substrate:

1. **Identity (C_I)**: Establishes a notion of stable normal forms and equivalence-preserving transformations. Without it, states dissolve into noise.
2. **Rewrite (C_R)**: Represents directed, non-invertible selection. Without it, transitions are symmetric, preventing one-way directed computation.
3. **State (C_S)**: Enables repeatable observables and state-space closure under self-application. Without it, the system collapses to a strictly linear, non-duplicating fragment that is sub-universal.
4. **Adjacency (C_A)**: Maps topological interaction and function application. Without it, parameters cannot be composed into ordered causal chains.

Under this design heuristic, no carrier can serve dual roles without collapsing the representation space:

- If $C_I = C_S$, the quotient structure collapses, identifying observational equivalence with syntactic identity.
- If $C_R = C_A$, causality becomes mutable, leading to non-deterministic, non-confluent rewrites.
- If $C_I = C_R$, the invariant becomes strategy-dependent, making evaluation order determine the semantic outcome.

Thus, a minimal carrier set size of exactly 4 is indicated.

Remark 1 (Sizing Hierarchies: Nibbles and Bytes)**.** *This 4-carrier structure provides a clean algebraic justification for the token hierarchies of digital computing:*

- ***1 Bit**: Distinguishes existence from absence (atom vs. void).*
- ***2 Bits**: Identifies one of the four active roles (C_I, C_R, C_A, C_S).*
- ***4 Bits (One Nibble)**: Represents a directed pair of names (an input pointer to a carrier role and an output pointer from a carrier role), expressing adjacency.*
- ***8 Bits (One Byte)**: Represents a pair of pairs—a source context and a target context—expressing a directed rewrite step.*

1.3 Occam and the Description-Length Heuristic

For a candidate substrate model M, we motivate the minimality of a 4-carrier substrate using a description-length marginal likelihood heuristic:

$$p(D \mid M) = \int p(D \mid \theta, M) p(\theta \mid M) d\theta$$

where θ represents the parameter space (matrices and rules) and D represents a reference computation. Given the 4-carrier limit, any substrate with $n > 4$ carriers introduces redundant gauge flexibility that increases description length. The terminal ISAR kernel minimizes this description length, acting as a Minimum Description Length (MDL) substrate.

Chapter 2

Decoder Theory and the Reverse Rosetta Boundary

In this chapter, we explore the semantic boundaries of representation in formal systems. We analyze the distinction between closed computational systems—which factor through a view-independent invariant layer—and open, anchor-dependent systems.

2.1 Operational Closure vs. Environmental Anchor-Dependence

We classify computational and semantic systems into two fundamental categories based on their transition dynamics and semantic recovery:

Definition 1 (Transition System)**.** *An autonomous transition system consists of a type of states and a step relation:*

$$\mathit{step} : \mathit{State} \to \mathit{State} \to \mathit{Prop}$$

Definition 2 (Operational Closure)**.** *A subset of states C is operationally closed (or forward invariant) under the transition system if all transitions starting within C remain in C:*

$$\mathit{IsClosed}(TS, C) \triangleq \forall(s_1, s_2 : \mathit{State}), C(s_1) \to \mathit{step}(s_1, s_2) \to C(s_2)$$

We prove that states starting in an operationally closed subsystem remain within it under any sequence of transitions:

Theorem 1 (Forward Invariance of Closed Subsystems)**.** *For any transition system TS and closed subsystem C, any state reachable from a state in C remains in C:*

$$\forall(s_1, s_2 : \mathit{State}), C(s_1) \to \mathit{Reachable}(TS, s_1, s_2) \to C(s_2)$$

Operational closure forms the basis for reconstructibility. In a closed system (such as lambda calculus, SKI combinators, and set-theoretic semantics), the transition dynamics can be fully projected onto the InvariantLayer quotient, allowing the decoder to reconstruct semantics at any point without loss of meaning.

In contrast, open systems depend on external parameters:

Definition 3 (Anchor-Dependent System)**.** *A transition system where transitions depend on an external environment or context:*

$$step : State \to Anchor \to State \to Prop$$

In an anchor-dependent system, trace semantics under sequences of external anchors are non-deterministic from the perspective of the state alone:

Theorem 2 (Referential Openness Requires Anchors)**.** *If a state can lead to different observations under different anchor sequences, then the trace semantics are not recoverable from the state alone:*

$$\exists (as : List\ Anchor), \neg\, (\forall (s' : State), Trace(s, as, s') \to decode(s') = decode(s'_1))$$

2.2 The Boundary of Decodability (Reverse Rosetta)

The **Reverse Rosetta** principle represents the boundary of what can be decoded when only the syntax is preserved but the semantic context is lost.

For closed systems (like lambda calculus or hereditarily finite sets), we can reconstruct their semantics from the operational quotient layer because their reductions are self-contained. For open systems (such as natural languages, undeciphered scripts like Linear A, or APIs with external state), the characters and syntax do not suffice for decoding because the meaning is tied to external cultural or environmental anchors.

2.3 Formal Distinction between Invariant Layer and Decoders

The distinction between the **Invariant Layer** and the **Decoder** is formalized through three distinct mechanisms:

2.3.1 Category-Theoretic Terminality

As proved in Theorem 9, the quotient InvariantLayer is the terminal object in the category of semantic kernels. Any other admissible computational view is guaranteed to factor uniquely through it via a unique morphism:

$$h : K \to \text{ISAR\_Kernel}$$

2.3.2 Geometrical Gauge Invariance

As formalized in the matrix representation of carriers, the decoder corresponds to a coordinate system (or basis choice), while the Invariant Layer represents the coordinate-free operator. Two matrix representations are isomorphic under basis conjugation:

$$P \cdot K_1 \cdot P^{-1} = K_2$$

proving that representation details are gauge-dependent shadows.

2.3.3 Empirical Decoupling

The type InvariantLayer remains static, yet we implement entirely separate decoders for set membership, lambda terms, and stack-machine bytecodes. These map disjoint syntactic types into the same quotient, proving that the invariant layer holds the view-independent semantics while representation remains strictly delegated to the decoder.

Chapter 3

Discussion, Related Work, and Limitations

We present a review of related literature, establish the formal boundaries of validity for our theorems, and outline directions for future research.

3.1 Related Work

The ISAR framework intersects several foundational areas of computer science and logic:

- **Combinatory Logic and Lambda Calculus**: The bracket abstraction algorithm compiled to combinator form is a classical approach pioneered by Curry and Feys [2] and Barendregt [1]. While traditional implementations suffer from code size explosion (e.g., $O(n^3)$ for standard abstraction), the linear fragment of ISAR maintains bounded sizing, sharing a direct connection to optimal reduction.
- **Optimal Evaluation and Interaction Nets**: Yves Lafont's interaction nets [6] provide a graphical formulation of computation where reductions are local and symmetric. Our formalization of the linear fragment (`LinearIKTerm`) and its bounds mirrors the behavior of optimal virtual machines like the Higher-Order Virtual Machine 2 (HVM2) [10], which compiles terms to interaction nets for parallel execution.
- **Categorical Semantics of Computation**: Lawvere's functorial semantics [7] established algebraic theories as categories. Our formulation of semantic kernels ($\mathcal{K}$) and the proof of terminality of the Invariant Layer quotient formalizes view-independence as a universal factorization property.
- **Proof Assistant Kernels**: The verification and automated generation of documentation utilizing Lean 4 quotients and metadata mappings (e.g.,

doc-gen4 [8]) represent the state-of-the-art in proof design. Our work extends this by proving formal simulation theorems directly against the proof assistant's environment.

3.2 Limitations and Scope Boundaries

The theorems presented in this work are accompanied by strict boundary conditions:

1. **Closed Dialects Only**: The terminality and factorization theorems apply exclusively to computational dialects that are *operationally closed*—meaning they possess a total encoding function into the substrate and their evaluation dynamics can be modeled as deterministic, confluent rewrite steps. This excludes open, dynamic, or referentially open systems (such as natural languages, interactive APIs, or non-deterministic transition systems) that require contextual anchors.

2. **ZFC HF Set Scope**: The set-theoretic relative interpretation is verified solely for the Hereditarily Finite set fragment. Extension to infinite sets (ZFC with the axiom of infinity) requires different ordinals and is outside the scope of the current Lean 4 proofs.

3. **Interpretive Physical Assumptions**: The application of the Universal Approximation Theorem to physical systems rests on the interpretive hypothesis that a physical system's state space can be modeled as an admissible continuous kernel. This hypothesis is a modeling choice and is not mechanically verified by the Lean 4 proof assistant.

3.3 Future Work

This paper forms the core substrate of a multi-part series of monographs:

- **Paper 2 (Continuous Approximation)**: Extension of the discrete kernel to real-valued physical quantities and continuous address spaces. Unique representation in a completion of `KernelAddress` is axiomatic (`ISAR_UAT` stays named; there is no metric on `KernelAddress`). Classical citations: Cybenko [3] and Leshno et al. [9].

- **Paper 3 (Hardware Compilation)**: Verification of the compilation path from the algebraic matrix representations to optimal virtual machines (HVM2) and relational database schemas. We also formalize partial evaluation and dialect specialization via the first, second, and third Futamura projections [4].

Bibliography

[1] Hendrik Pieter Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103. Elsevier, 1984.

[2] Haskell Brooks Curry and Robert Feys. *Combinatory Logic*, volume 1. North-Holland Publishing Company, 1958.

[3] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[4] Yoshihiko Futamura. Partial evaluation of computation process–an approach to a compiler-compiler. *Systems, Computers, Controls*, 2(5):45–50, 1971.

[5] Kazimierz Kuratowski. Sur la notion de l'ordre dans la théorie des ensembles. *Fundamenta Mathematicae*, 2(1):161–171, 1921.

[6] Yves Lafont. Interaction nets. *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 95–108, 1990.

[7] F William Lawvere. *Functorial semantics of algebraic theories*. PhD thesis, Columbia University, 1963.

[8] Lean Prover Community. doc-gen4: Documentation generator for lean 4. `https://github.com/leanprover/doc-gen4`, 2023.

[9] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.

[10] Victor Taelin. Hvm2: Higher-order virtual machine 2. `https://github.com/HigherOrderCO/HVM2`, 2024.