ISAR: Invariant Scalable Fractal Addressable Representations

Fabian Schindler

September 4, 2026

Abstract

This monograph presents the complete mathematical foundations of the ISAR (Invariant Scalable Fractal Addressable Representation) framework, a formally verified computational substrate implemented in the Lean 4 proof assistant. We detail: (1) the core discrete ISAR combinator calculus, proving confluence and unique normal forms; (2) a category-theoretic terminality proof showing that observational quotients factor uniquely through the invariant layer; (3) relative set-theoretic interpretations in Hereditarily Finite sets; (4) dialect views and partial evaluation via Futamura projections; and (5) continuous completions of metric address spaces and the Universal Approximation Theorem.

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

Core Combinatory Calculus

In this chapter, we formalize the syntax, reduction semantics, and computational properties of the ISAR combinatory logic kernel. The core ISAR calculus is a substrate-independent combinator system that supports S-K combinator reductions, parallel reductions, complete developments, and normalization.

2.1 Syntax and Reduction Semantics

The definitions and operational behavior described in this section are mechanically verified in `src/ISAR/Kernel.lean`. The syntax consists of standard S-K combinators and the specialized ISAR terms.

Definition 1 (S-K Term Syntax)**.** *The inductive type of basic S-K combinator expressions is defined by:*

$$t, u \in SK ::= \mathbf{S} \mid \mathbf{K} \mid t\ u$$

Definition 2 (ISAR Term Syntax)**.** *The core term syntax of the ISAR kernel extends basic combinators with norms, constants, and custom operations:*

$$t, u \in ITerm ::= \mathbf{var}\ n \mid \mathbf{norm} \mid \mathbf{konst} \mid \mathbf{dup} \mid \mathbf{swap} \mid \mathbf{comp} \mid \mathbf{s_s} \mid \mathbf{app}\ t\ u$$

where $\mathbf{app}\ t\ u$ *denotes term application (also written* $t \cdot u$*).*

We define the reduction rules for both systems.

Definition 3 (SKStep Reduction)**.** *The reduction relation on SK terms represents the standard combinatory logic reduction rules:*

- $\mathbf{K}\ x\ y \to x$
- $\mathbf{S}\ x\ y\ z \to (x\ z)(y\ z)$

Definition 4 (IStep Reduction)**.** *The single-step reduction relation on ISAR terms defines the operational semantics of norms and constants. The reduction rules* ($\to_I$) *are:*

- $\mathbf{norm} \cdot x \to_I x$
- $\mathbf{konst} \cdot x \cdot y \to_I x$
- $\mathbf{comp} \cdot f \cdot g \cdot x \to_I f \cdot (g \cdot x)$
- $\mathbf{s_s} \cdot x \cdot y \cdot z \to_I (x \cdot z) \cdot (y \cdot z)$

together with congruence rules for application:

$$\frac{f \to_I f'}{f \cdot x \to_I f' \cdot x} \qquad \frac{x \to_I x'}{f \cdot x \to_I f \cdot x'}$$

The relation $\twoheadrightarrow_I$ (denoted `IRed`) is the reflexive-transitive closure of `IStep`.

2.2 Parallel Reduction and Confluence

The parallel reduction relation, Takahashi's Lemma, and the confluence theorems are formalized and proved in `src/ISAR/Kernel.lean`. To prove confluence, we define the parallel reduction relation `ParStep` ($\Rightarrow$), which allows simultaneous contractions of multiple redexes in a single step:

Definition 5 (Parallel Step)**.** *The parallel reduction relation is defined inductively by:*

- $x \Rightarrow x$
- $\mathbf{norm} \cdot x \Rightarrow x'$ *if* $x \Rightarrow x'$
- $\mathbf{konst} \cdot x \cdot y \Rightarrow x'$ *if* $x \Rightarrow x'$ *and* $y \Rightarrow y'$
- $\mathbf{comp} \cdot f \cdot g \cdot x \Rightarrow f' \cdot (g' \cdot x')$ *if* $f \Rightarrow f'$, $g \Rightarrow g'$, $x \Rightarrow x'$
- $\mathbf{s_s} \cdot x \cdot y \cdot z \Rightarrow (x' \cdot z') \cdot (y' \cdot z')$ *if* $x \Rightarrow x'$, $y \Rightarrow y'$, $z \Rightarrow z'$
- $f \cdot x \Rightarrow f' \cdot x'$ *if* $f \Rightarrow f'$ *and* $x \Rightarrow x'$

Theorem 1 (Takahashi's Lemma)**.** *If $x \Rightarrow y$, then $y \Rightarrow cd(x)$, where cd is the complete development function.*

Theorem 2 (Confluence of IRed)**.** *The reduction relation IRed satisfies the Church-Rosser confluence property:*

$$\forall t, u_1, u_2, \quad t \to^* u_1 \wedge t \to^* u_2 \implies \exists v, \quad u_1 \to^* v \wedge u_2 \to^* v$$

Theorem 3 (Unique Normal Forms)**.** *If a term t reduces to two normal forms n_1 and n_2, then $n_1 = n_2$.*

2.3 Conservative Basis Translation

The derived combinator simulation and conservative translation are verified in `src/ISAR/Kernel.lean`. The distributive combinator s_s is not strictly necessary for the rewrite algebra. We constructively define the derived S combinator as:

$$\begin{aligned}\text{derived\_s} \triangleq\ & (\mathbf{comp} \cdot (\mathbf{comp} \cdot \mathbf{dup})) \\ & \cdot \big((\mathbf{swap} \cdot ((\mathbf{comp} \cdot \mathbf{comp}) \cdot ((\mathbf{comp} \cdot \mathbf{comp}) \cdot \mathbf{swap}))) \cdot \mathbf{norm}\big)\end{aligned}$$

Theorem 4 (Derived S reduction)**.** *The derived S combinator simulates the beta-reduction rule of S using only the basis operators (without s_s):*

$$\mathit{derived\_s} \cdot x \cdot y \cdot z \twoheadrightarrow_{I_{basis}} (x \cdot z) \cdot (y \cdot z)$$

Theorem 5 (Conservative Translation)**.** *The translation map* ***`translate_to_basis`*** *(which replaces s_s with* ***`derived_s`****) preserves reduction steps.*

2.4 The Invariant Layer

The operational quotient and application congruence are formalized and proved in `src/ISAR/InvariantLayer.lean`. Using the uniqueness of normal forms, we define the quotient space representing behavioral equivalence classes of terms.

Definition 6 (ISKSubtype)**.** *The subtype of ISAR terms belonging to the S-K fragment:*

$$\mathit{ISKSubtype} := \{t : \mathit{ITerm} \mid \mathit{ISKTerm}\ t\}$$

Definition 7 (Operational Equivalence)**.** *Two terms are operationally equivalent if they behave identically under reduction:*

$$t \sim_{op} u \iff \mathit{OperEq}\ t\ u$$

Definition 8 (Invariant Layer)**.** *The quotient type of* ***`ISKSubtype`*** *modulo* ***`OperEq`****:*

$$\mathit{InvariantLayer} := \mathit{Quotient}\ (\mathit{operEqSetoid})$$

Definition 9 (Invariant Layer Application)**.** *The application operator lifted to the quotient space:*

$$\mathit{app} : \mathit{InvariantLayer} \to \mathit{InvariantLayer} \to \mathit{InvariantLayer}$$

Definition 10 (Canonical Representative)**.** *A noncomputable function mapping each equivalence class to its canonical representative in* ***`ISKSubtype`****.*

2.5 Linear Duplication and Computable Normalization

Local duplication counts, size-decrease properties of complete development, and fuel correctness are verified in `src/ISAR/Kernel.lean` and `src/ISAR/InvariantLayer.lean`. We define the linear duplication fragment `LinearIKTerm` and its termination metrics, which share a direct isomorphism with linear interaction nets.

Definition 11 (Linear Duplication)**.** *The function* ***`dupCount`*** *tracks the duplication multiplicity of a term. For the linear fragment* ***`LinearIKTerm`****, we verify that duplication is strictly localized:*

$$dupCount\ t = 0$$

We prove that complete development (`cd`) strictly decreases term size for all non-fixed-point linear terms.

Theorem 6 (Complete Development Size Decrease)**.** *For any term t in the* ***`LinearIKTerm`*** *fragment:*

$$term\_size(cd\ t) \leq term\_size(t)$$

And if $t \neq cd\ t$, the inequality is strict.

Definition 12 (Computable Normalization Loop)**.** *A fuel-based normalization loop that computes the normal form.*

Theorem 7 (Fuel Correctness)**.** *The optimal fuel limit computed from the term size is sufficient to guarantee complete normalization at runtime.*

Chapter 3

Category-Theoretic Interface and Terminality

The ISAR kernel provides a category-theoretic interpretation of representation-free substrates and semantic views. We formalize this using the category of semantic kernels and morphisms, and prove that the canonical invariant quotient space is a terminal object in this category. This universal property of terminality establishes the mathematical foundation of view-independence.

3.1 The Category of Semantic Kernels

The definitions and structures in this section are mechanically verified in `src/ISAR/KernelCategory.lean` and `src/ISAR/DialectKernel.lean`. We define the category $\mathcal{K}$ of admissible semantic kernels over the substrate.

Definition 13 (Semantic Kernel)**.** *An admissible semantic kernel $K \in \mathcal{K}$ consists of:*

1. *A carrier type Carrier.*
2. *A view mapping view_of : ISKSubtype → Carrier.*
3. *An observational equivalence relation $\approx$ on Carrier.*
4. *A soundness property: operational equivalence in the substrate implies view equivalence:*

$$\forall t, u, \quad t \sim_{op} u \implies \mathit{view\_of}\ t \approx \mathit{view\_of}\ u$$

5. *A decoding/reconstruction mapping decode : Carrier → ISKSubtype.*
6. *Coherence axioms:*
   - *decode(view_of t) $\sim_{op}$ t*

- *view_of*(*decode* c) $\approx c$
- $c_1 \approx c_2 \implies$ *decode* $c_1 \sim_{op}$ *decode* c_2

Definition 14 (Canonical ISAR Kernel)**.** *The identity kernel mapping the substrate directly to itself is the canonical presentation.*

Definition 15 (Computable ISAR Kernel)**.** *The computable kernel parametrized by normalization fuel.*

Definition 16 (Optimal Computable Kernel)**.** *The computable kernel instantiated with optimal fuel computed from the term size for linearly-typed terms.*

Definition 17 (Kernel Morphism)**.** *A morphism $f : K_1 \to K_2$ in the category $\mathcal{K}$ is a carrier mapping $hom : K_1.Carrier \to K_2.Carrier$ that:*

- *Preserves the view representations:* $\forall t \in ISKSubtype, \quad K_2.view\_of\ t \approx_2 hom(K_1.view\_of\ t)$
- *Respects carrier equivalence:* $\forall c_1, c_2, \quad c_1 \approx_1 c_2 \implies hom(c_1) \approx_2 hom(c_2)$

Definition 18 (Canonical Homomorphism)**.** *For any kernel K, the canonical homomorphism into ISAR_Kernel is defined by the decoding mapping.*

3.2 Universal Factorization and Terminality

The uniqueness and terminality theorems are formalized and proved in `src/ISAR/KernelCategory.lean`. We prove that the canonical representation-free kernel is terminal, meaning that any admissible view can be uniquely decoded into it.

Theorem 8 (Morphism Uniqueness)**.** *Any structure-preserving morphism $f : K \to ISAR\_Kernel$ is observationally equivalent to the canonical decoding morphism:*

$$\forall c \in K.Carrier, \quad f(c) \sim_{op} decode\ c$$

Theorem 9 (ISAR Kernel Terminality)**.** *The quotient of the morphism space KernelHom K ISAR_Kernel modulo observational equivalence is a singleton:*

$$\exists! [f] \in KernelHom\ K\ ISAR\_Kernel/\sim_{op}$$

Thus, ISAR_Kernel is a terminal object in the category $\mathcal{K}$ of semantic kernels.

This category-theoretic result is not a naming convention: it is an algebraic theorem stating that the Invariant Layer is the unique (up to isomorphism) cofinal quotient that retains only the information visible to observational equivalence. Any other view is mathematically guaranteed to factor uniquely through it, separating the dialect-specific representation from the coordinate-free invariant semantics.

3.3 The Dialect Subsystem

The Dialect structures in this section are verified in `src/ISAR/DialectKernel.lean`. A Dialect is an alternative abstract view over the substrate.

Definition 19 (Dialect)**.** *A Dialect specifies:*

1. *A type of dialect objects.*
2. *A type of observations and observational equivalence.*
3. *An evaluation mapping, an encoder to substrate terms, and a decoder from the substrate.*
4. *A coherence law: evaluation commutes with encoding and decoding.*

Chapter 4

Symbolic Views and Dialects

The ISAR architecture supports multiple symbolic views or "dialects" over the representation-free substrate. Each view compiles to the substrate and decodes back, satisfying a preservation law. In this chapter, we formalize several dialects: the Lambda calculus fragment, ZFC hereditarily finite sets, term rewriting systems, bytecode compilation, and Barker's Iota combinator.

4.1 The Lambda Calculus Fragment

The bracket abstraction compiler and simulation properties in this section are mechanically verified in `src/ISAR/LambdaFragment.lean`. We define the standard λ-calculus with de Bruijn indices.

Definition 20 (Lambda Terms)**.** *Lambda terms with variables (represented by natural number de Bruijn indices), applications, and abstractions:*

$$t, u \in \mathit{LTerm} ::= \mathbf{var}\ n \mid \mathbf{app}\ t\ u \mid \mathbf{lam}\ t$$

Definition 21 (Lambda Step)**.** *Single-step β-reduction on `LTerm` using de Bruijn shifts and substitutions.*

Definition 22 (Lambda Compiler)**.** *Compiles lambda terms into the ISAR substrate via bracket abstraction:*

$$\mathit{compile} : \mathit{LTerm} \to \mathit{ITerm}$$

Theorem 10 (Compiler Simulation)**.** *The compiler faithfully simulates lambda reductions in the ISAR kernel:*

$$\forall t, u \in \mathit{LTerm}, \quad t \to_\beta u \implies \mathit{compile}\ t \to^* \mathit{compile}\ u$$

4.2 Hereditarily Finite Sets and ZFC

The set-theoretic encoding and relative ZFC interpretation verified in this section are mechanically verified in `src/ISAR/HFSet.lean`, `src/ISAR/HFSetEncoding.lean`, and `src/ISAR/HFSetSemantics.lean`. This relative interpretation is strictly limited to the Hereditarily Finite (HF) set fragment (ZFC without the axiom of infinity). We formalize HF Sets, their encoding, and show they satisfy set-theoretic axioms.

Definition 23 (HF Sets)**.** *The type of hereditarily finite sets, built inductively from the empty set:*

$$x, y \in HF ::= \emptyset \mid x \cup \{y\}$$

Definition 24 (Extensional Equality)**.** *Set-theoretic extensional equivalence ($x =_{ext} y$) defined via Ackermann's bijective encoding 'toNat':*

$$x =_{ext} y \iff ExtEq\ x\ y$$

Definition 25 (HF Set Encoding)**.** *Encodes HF sets into substrate invariant layers:*

$$HF\_encode : HF \to InvariantLayer$$

Theorem 11 (HF Encoding Correctness)**.** *The encoding is sound and invertible:*

$$\forall c \in HF, \quad decode\_layer(HF\_encode\ c) = c$$

We package HF Sets as a semantic kernel.

Definition 26 (HF Set Kernel)**.** *The set-theoretic admissible semantic kernel HF_Kernel.*

Theorem 12 (ZFC Interpretation Factorization)**.** *The HF set kernel factors uniquely through `ISAR_Kernel`:*

$$\forall f : KernelHom\ HF\_Kernel\ ISAR\_Kernel, \quad f(c) \sim_{op} encode\_raw\ c$$

This faithful factorization proves that the ZFC HF set fragment faithfully interprets into the ISAR substrate, satisfying empty set, pairing, and union axioms.

Syntactically distinct terms representing the same set (e.g., $\{a, b\}$ and $\{b, a\}$) project to the exact same element in the 'InvariantLayer' quotient, demonstrating extensional equivalence.

4.3 Term Rewriting, Bytecode, and Iota Views

The SKI TRS view, stack bytecode compiler, and Barker's iota dialect are verified in `src/ISAR/TRSView.lean`, `src/ISAR/BytecodeView.lean`, and `src/ISAR/IotaView.lean` respectively. We define dialects for other standard computational models.

Definition 27 (SKI TRS Dialect)**.** *The term rewriting dialect modeling pure SKI combinator terms* `TTerm`*.*

Definition 28 (Bytecode Dialect)**.** *The dialect representing stack-based virtual machine instruction bytecode* `Instruction` *compiled and decompiled.*

Definition 29 (Barker's Iota Dialect)**.** *The dialect modeling the single-combinator* `IotaTerm` *utilizing Barker's universal combinator* $\iota = \lambda x.x\ S\ K$*.*

In Barker's iota combinator view, iterating iota-application produces the closed orbit:

$$I \to A \to K \to S \to X \to I$$

where:

- I = identity ($Ix = x$)
- A = flip constant ($Axy = y$)
- K = constant ($Kxy = x$)
- S = substitution ($Sfgx = fx(gx)$)
- X = self-application kernel

These four distinct active combinators map directly onto the four matrices (I, R, A, S) of the ISAR substrate, proving the ontological completeness of the 4-carrier representation.

Chapter 5

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.

5.1 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 30 (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 31 (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 13 (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 32 (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 14 (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:*

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

5.2 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.

5.3 Formal Distinction between Invariant Layer and Decoders

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

5.3.1 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:

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

5.3.2 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.

5.3.3 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 6

Continuous Approximation and Representation Space

In this chapter, we bridge the discrete combinatory logic of the ISAR substrate with continuous analysis and matrix spaces. We formalize dimensional physical quantities, matrix representations, expressive completeness, the tensor semantics, and named analytic axioms for universal approximation. Unique addresses in a completion of `KernelAddress` remain axiomatic: there is no metric on `KernelAddress`.

6.1 Physical Quantities and Matrix Representations

We define dimensional quantities and 4x4 matrix algebra.

Definition 33 (Quantities and Uncertainty)**.** *A quantity represents a physical parameter with:*

1. *A value (rational scalar).*
2. *A physical dimension (exponent exponents).*
3. *An epistemic uncertainty expression* `Uncertainty` *and* `Correlation`*.*

Covariance propagation propagates exact metric-epistemic covariance equations under addition and multiplication.

Definition 34 (Matrix4 Carriers)**.** *The concrete integer 4x4 matrix represen-*

tation used for basis combinator signatures. The primitive matrices are:

$$I_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad R_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$
$$A_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad S_1 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Theorem 15 (Idempotency and Nilpotency)**.** *The identity matrix I_1 is idempotent:*

$$I_1^2 = I_1$$

The core rewrite operator $K_1 = I_1 \cdot R_1 \cdot A_1 \cdot S_1$ is nilpotent:

$$K_1^2 = \mathbf{0}$$

Theorem 16 (Gauge Similarity)**.** *The two alternative matrix representations K_1 and K_2 are conjugate (similar) via the lower-triangular gauge transformation matrix P:*

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

proving that representation details are gauge shadows, while the terminal quotient state is invariant.

Definition 35 (Matrix View Map)**.** *The structural homomorphism mapping each ISAR term to its concrete 4x4 integer matrix signature:*

$$\mathit{term\_signature\_val} : \mathit{ITerm} \to \mathit{Matrix4}$$

Theorem 17 (Expressive Completeness)**.** *Every matrix generated by the basis combinator algebra is the image of some term:*

$$\forall M \in \mathit{ISKAlgebra}, \quad \exists t \in \mathit{ISKSubtype}, \quad \mathit{term\_signature\_val}\ t = M$$

6.2 The Categorical Matrix Bridge

We package the matrix representations as an admissible semantic kernel.

Definition 36 (Matrix Kernel)**.** *The semantic kernel MatrixKernel whose carrier is `Matrix4` and whose view map is `term_signature_val`.*

Theorem 18 (Nilpotent Collapse)**.** *Applying `konst` to itself maps to the zero matrix, showing the nilpotent collapse of the basis algebra:*

$$\mathit{kernelMatrixView}(\mathbf{K}\ \mathbf{K}) = \mathbf{0}$$

Theorem 19 (Matrix Terminality)**.** *Every structure-preserving morphism from* `MatrixKernel` *to* `ISAR_Kernel` *is observationally equivalent to the canonical decoding morphism.*

Theorem 20 (Unreachability of R and A)**.** *The structural homomorphism* `kernelMatrixView` *never produces the rotation matrix* R_1 *or adjacency matrix* A_1 *from a pure ISK term.*

6.3 Tensor Denotation Semantics

We bridge the discrete combinatory logic of the ISAR substrate with continuous analysis and matrix spaces by formalizing the denotational mapping of symbolic ISAR terms into an abstract rank-4 tensor space, where reductions map to extensional equality.

Definition 37 (Tensor Space Primitives)**.** *The abstract tensor space* $\mathcal{T}$ *is characterized by five primitive operators:*

- *Identity:* `t_norm`
- *Constant:* `t_konst`
- *Duplicator:* `t_dup`
- *Swapper:* `t_swap`
- *Composition:* `t_comp`

Definition 38 (Derived S in Tensor Space)**.** *The distributive combinator* **S** *in the tensor space is constructively derived using swapper, composition, and duplicator operators:*

$$t\_s_s \triangleq t\_app\Big(t\_app\big(t\_comp, t\_app(t\_comp, t\_dup)\big),\\ t\_app\big(t\_app(t\_swap, P), t\_norm\big)\Big)$$

where:

$$P \triangleq t\_app\big(t\_app(t\_comp, t\_comp), t\_app(t\_app(t\_comp, t\_comp), t\_swap)\big)$$

Definition 39 (Denotational Mapping)**.** *The structural denotational homomorphism mapping symbolic terms to the tensor space:*

$$denot : ITerm \to TensorSpace$$

Theorem 21 (Denotational Soundness)**.** *One-step reduction in the symbolic calculus maps to extensional equivalence of denotations:*

$$\forall t, u, \quad t \to_I u \implies denot\ t \approx_{ext} denot\ u$$

Thus, the denotational mapping factors uniquely through the Invariant Layer quotient:

$$InvariantLayer.toExtTensor : InvariantLayer \to ExtTensor$$

6.3.1 Sparse Matrix Operator Composition

In the concrete implementation of the axiomatic kernel (`scratch/isar_ski_kernel.py`), the emergent operator combinators are represented as products of the basic sparse matrices I, R, A, S:

$$\begin{aligned} \text{NORM} &\triangleq S \cdot I \\ \text{APP} &\triangleq A \cdot R \\ \text{DUP} &\triangleq S \cdot S \\ \text{COMP} &\triangleq R \cdot A \cdot S \\ \text{SWAP} &\triangleq R \cdot S \\ \text{CONST} &\triangleq I \cdot S \end{aligned}$$

6.3.2 Plex-Agent Tensor Primitives and Special Forms

Alternatively, the runtime environment and compiler stack defined in `tensor_primitives.py` specify four basic tensor primitives and four special forms derived from them:

$$\begin{aligned} \text{ATOM } (0) &\triangleq I \\ \text{PAIR } (1) &\triangleq S \\ \text{JOIN } (2) &\triangleq R \cdot A \\ \text{NORM } (3) &\triangleq I \cdot R \cdot A \cdot S \end{aligned}$$

The special forms are defined as compositions of these primitives:

$$\begin{aligned} \text{LAMBDA} &\triangleq \text{JOIN} \cdot \text{PAIR} \cdot \text{NORM} \\ \text{QUOTE} &\triangleq \text{ATOM} \cdot \text{NORM} \end{aligned}$$

along with conditional branching (`IF`) and variable bindings (`DEF`).

6.4 Continuous Universal Approximation

We formalize the continuous parameter space of the ISAR update.

Definition 40 (Non-Polynomial Activation)**.** *An activation function $\sigma \in C(\mathbb{R}, \mathbb{R})$ is non-polynomial:*

$$\forall P \in \mathit{Polynomial}\, \mathbb{R}, \quad \sigma \neq P$$

Definition 41 (Raw Address Space)**.** *The raw parameter configuration space $\mathit{RawAddress}\ d\ k$ representing a neural network configuration:*

$$\begin{aligned} \mathit{RawAddress}\ d\ k \triangleq \Sigma(N, T : \mathbb{N}), \quad & (\mathit{Fin}\ T \to \mathit{Fin}\ 4 \to \mathbb{R}) \times C(\mathbb{R}^d, \mathit{GridState}\ N) \\ & \times C(\mathit{GridState}\ N, \mathbb{R}^k) \end{aligned}$$

Definition 42 (Realization Map)**.** *The continuous function realized by a raw address configuration:*

$$\mathit{realizeRaw}\ d\ k\ \sigma : \mathit{RawAddress}\ d\ k \to C(\mathbb{R}^d, \mathbb{R}^k)$$

Definition 43 (Address Equivalence)**.** *Two raw parameter configurations are equivalent if they realize the same continuous function.*

Definition 44 (Kernel Address Space)**.** *The quotient parameter space of raw addresses modulo observational equivalence:*

$$\mathit{KernelAddress}\ d\ k\ \sigma := \mathit{Quotient}\ (\mathit{addressSetoid}\ d\ k\ \sigma)$$

We state the Universal Approximation Theorem as a **named axiom** (Leshno-style; not a theorem in this repository).

Axiom 5 (ISAR Universal Approximation)**.** *Let $K \subset \mathbb{R}^d$ be compact, $f \in C(\mathbb{R}^d, \mathbb{R}^k)$ a target continuous function, σ a non-polynomial activation, and $\varepsilon > 0$. Then there exists a raw address $\theta \in \mathit{RawAddress}\ d\ k$ such that:*

$$\forall x \in K, \quad \|\mathit{realizeRaw}\ d\ k\ \sigma\ \theta\ x - f(x)\| < \varepsilon$$

6.5 Quotient Completion and Unique Representation

We complete the parameter space only as a named interface.

Axiom 6 (Quotient Completion Space)**.** *The named placeholder type for a completion of `KernelAddress`. `KernelAddress` has no metric or uniform structure in Lean; this is **not** Mathlib `Metric.Completion`.*

Axiom 7 (Continuous Realization Limit)**.** *The named realization map on `KernelAddressLimit` (not a constructed Mathlib extension).*

Axiom 8 (Completion Embedding)**.** *Named embedding i : KernelAddress $\to$ KernelAddressLimit (not a metric-completion inclusion).*

Theorem 22 (Embedding Injectivity)**.** *The named embedding is injective (from `continuousRealization` injectivity and the commuting axiom).*

Theorem 23 (Embedding Density)**.** *Compact-open density relative to the named axiom `ISAR_UAT`, not a metric-space density theorem.*

Axiom 9 (Topological Extension Bijection)**.** *Named extension axiom. Would follow from a metric on `KernelAddress` plus density; that metric does not exist in the formalization.*

Theorem 24 (ISAR Representation Theorem)**.** *Relative to the named axioms* `ISAR_UAT` *and* `topological_extension_bijection`*: every continuous* $f \in C(\mathbb{R}^d, \mathbb{R}^k)$ *has a unique address* $\theta_{limit} \in KernelAddressLimit$ *such that:*

$$continuousRealizationLimit\ d\ k\ \sigma\ \theta_{limit} = f$$

This does not prove UAT or construct a metric completion.

Theorem 25 (Physical System Address)**.** *Wrapper:* `ISAR_representation` *applied to an arbitrary continuous map. Adds no physical content and no extra axiom.*

Chapter 7

Applications and Advanced Properties

In this final chapter, we formalize several advanced computational and structural properties of the ISAR framework: Futamura projections for partial evaluation, observational isomorphisms between dialects, and the classification of referentially open systems.

7.1 Partial Evaluation and Futamura Projections

We formalize partial evaluation and compile-time optimization via Futamura projections.

Definition 45 (Specializer)**.** *The meta-level specializer (partial evaluator) substituting static variables:*

$$\mathit{specialize} : \mathit{ITerm} \to (\mathbb{N} \to \mathit{Option}\ \mathit{ITerm}) \to \mathit{ITerm}$$

Definition 46 (PE Setup)**.** *Object-level mix setup: an evaluator, a meta specializer, its reflection as an object-language term* `specTerm`*, the mix equation, and* `selfApp` *(evaluating* `specTerm` *implements* `spec`*). Mix+selfApp alone admit a trivial residualizer;* `Nontrivial` *is a separate obligation. Jones–Gomard–Sestoft 1993 polyvariant offline mix remains open.*

We prove the three Futamura projections in this mix formulation.

Theorem 26 (First Futamura Projection)**.** *Mix equation at the substitution layer: specializing then substituting dynamic data equals substituting the full environment, under a coherence hypothesis on static/dynamic environments.*

Theorem 27 (Second Futamura Projection)**.** *Mix instantiation: evaluating the specialized specializer on source data yields the specialized interpreter. Holds for any* `PESetup`*; does not by itself give a compiler-generator of Jones–Gomard–Sestoft quality.*

Theorem 28 (Third Futamura Projection)**.** *Self-application mix instantiation: evaluating* `specTerm` *specialized to itself yields a compiler generator* at the mix-equation level. *Online* `JGS_PE` *covers the* `IStep` *signature; 1993 polyvariant cogen remains open.*

7.1.1 System Generation vs. Compilation

It is important to distinguish the ontological role of the ISAR kernel from the epistemological role of the Futamura projections. - **ISAR Kernel (**$U(K) = I \cdot R \cdot A \cdot S$**)**: Acts as a **System Generator**. It creates the computational substrate itself, establishing the tensor manifold in which computation exists and from which operators (norm, app, composition) emerge constructively. - **Futamura Kernel (**$I \times S \times A$**)**: Acts as a **Compiler Generator**. It operates within an established space to transform program representations (interpreter $\rightarrow$ compiler).

7.2 Dialect Isomorphisms and Unification

We formalize structural unifications and observational equivalence between different dialects.

Definition 47 (Observational Isomorphism)**.** *An observational isomorphism between two dialects D_1 and D_2 is a pair of maps between their objects preserving the evaluation and coding semantics.*

Definition 48 (Admissible Dialect)**.** *A dialect packaged with its admissibility proofs, allowing it to induce a semantic kernel.*

Theorem 29 (Isomorphism Unification)**.** *An observational isomorphism between two admissible dialects induces an isomorphism between their corresponding semantic kernels in the category of kernels.*

7.3 Referential Openness and Anchor Dependency

We analyze systems that are operationally closed versus referentially open (anchor-dependent).

Definition 49 (Transition System)**.** *A standard operationally closed autonomous transition system.*

Theorem 30 (Forward Invariance)**.** *If a state-space subset C is closed under the step relation (forward invariant), any state reachable from C remains in C.*

This applies to closed computational models (combinator engines, lambda calculus, stack VMs, set theory interpreters) where the state alone is sufficient to reconstruct the semantics at any step.

Definition 50 (Anchor-Dependent System)**.** *A referentially open transition system whose transitions depend on an external environment context (anchor).*

Theorem 31 (Anchor Dependency)**.** *If a state can lead to different observations under different anchor sequences, the semantics cannot be resolved from the state alone without environmental anchors.*

This establishes the boundary of decodability (the "Reverse Rosetta" problem). For open systems like natural language or undeciphered scripts (e.g., Linear A), analyzing syntax is insufficient to decode meaning because the external semantic anchors are lost. Thus, referentially open systems cannot be reconstructed from syntactic relationships alone.

Chapter 8

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.

8.1 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.

8.2 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.

8.3 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.