ISAR: A Quotient-Mediated Semantic Kernel for Closed Dialects

Fabian Schindler

September 4, 2026

Abstract

This paper presents the ISAR framework, a closed combinatory substrate designed to unify disjoint computational models under a single algebraic setting. We formally prove that the quotient `InvariantLayer` is the view-independent invariant layer of the substrate. Furthermore, we establish that the representation process for any admissible closed dialect yields an Encoder $\rightarrow$ Kernel $\rightarrow$ Quotient $\rightarrow$ Decoder factorization, showing that different formalisms and programming languages function as decoders over the same invariant structure.

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

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.

5.1 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.

5.2 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.

5.3 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.