Renormalization Logic is Effective Logic

Rutger Boels
AI.IMPACT GmbH
Neuer Jungfernstieg 5
Hamburg

September 25, 2025

Abstract

Experimental results are formally dominant in fundamental science. In practice there are typically large tapestries of techniques to connect multiple theories to multiple experiments. In particle physics for instance a wealth of theories, techniques, formalisms and results exist based on various different approaches, paired with a wealth of experimental data. The actual bottleneck to progress seems to be connecting these two worlds more efficiently and effectively as the current pipelines are very long and sometimes convoluted. The chance that something is missing or has been missed in that pipeline is not zero.

The greatest common denominator between the already closely aligned fields of physics, computation and mathematics is logic. Logic is designed to be independent of domain application. I.e. the logic of number theory is related to, but not necessarily the same as the logic of differential geometry or that of quantum field theory. In this article we show results of using AI tooling to construct logic systems systematically. A logic is a formal system with a notion of "truth". Since formal systems can be formulated inside mature computer algebra systems, this is a natural approach that provides a in-build sanity check on LLM "reasoning" to the strength of the implementation of then formal system. There are very close parallels between the construction of formal systems and the construction of QFT theories; both involve a choice of coordinates for one: essentially free fields for QFT and free logic symbols for logic. Interactions in logic are through definitions; in physics through coupling constants. The main difference is that logic is inherently exact by construction - if it is consistent. Consistent truth systems are severely restricted by powerfull, well-known theorems due to Gödel, Löb, Turing and Rice. These express fundamental constraints on maps of logic systems to themselves. These maps can not be total.

As a first step along the effective logic for physics programm, we explore the theory of computation. Computation can be understood as the composition of encoding, function application(s), decoding as well as normalisation. Classically, normalisation can occur at any stage (i.e. commutes as an operator). This is the basic compiler pattern of the theory of programming languages. We argue that the classic Turing, Church and Feynman views on computation can be modelled as different, partial representations of the same basic logic theory. As a first step, we show how they all relate to each other using a novel generating functional approach deeply related to

CFT conformal blocks. This is already somewhat computationally universal, which we demonstrate by showing equivalence between Church and Turing views in this framework: this is the well-known Church-Turing thesis. We conjecture a specialisation of this observation that encompasses all views on computation naturally. We show this pattern also naturally appears in the AGT correspondence.

Central is the observation that the different parameter sets in this construction have to be mapped to each other, but the generating functionals are so far formal only. This is a problem as the normalisation step in the quantum case is no longer freely commuting. Hence we introduce (3+1) natural regulators, including an overall "scale" parameter, that make the formal expansions well-defined. We show there is a precise analogue of the usual renormalisation programm, including an extension of the RG equation to several commuting flows natural in the Toda hierarchy. This includes the use of normalisation conditions to fix the parameters to their semantic interpretation as natural numbers (semirings). Interestingly, there is an fundamental imbalance between the number of beta functions (3) and gamma functions (1). We conjecture natural generalisations of the a-functions and the c-functions through the AGT correspondence using the natural analog of the Fisher information metric (c theorem) and XXXXXXXX (a-theorem).

To explain this construction we construct a concrete logic system. A logic system describing chruch-style computation exists - this is the content of the Curry-Howard-Lambek correspondence. We argue we have here a generalisation of this correspondence to irreversible computation or, in other words, the general theory of programming languages. Basically, the theory of computation outlined above is conjectured to be a model of the logic system we construct in the paper. If our construction through computer algebra is consistent, then, we argue, this construction proves the conjecture. Verifying this is non-trivial, as the needed computer algebra is quite involved. For safety, we provide full implementations in agda, coq, isabelle, as well as a partial one in metamath.

In logic we argue these notions of regularisation and renormalisation conform to a very particular and peculiar conservative extension of basically any logic. This extension can be understood as a natural "hierarchical" deformation of the truth system of any logic, together with some of its subsystems. The deformation changes the truth system to a fundamentally asymmetric notion of equality (implication weighted by local weights), and we relate the undeformed truth to the kernel of a pair of natural ternary operator constructed inside the deformed logic that combine into an arity $(2, 2)$ operator we call a "logic transformer". This logic transformer is basically a polymorphic generalization of the notion of a scaling operator in physics. Interestingly, this operator is known in the formal theory of programming languages as a "partial self-evaluation" operator directly tied to the notion of compilation, and in logic itself as a part of the diagonal lemma proof. In this construction the kernel of the transformer is naturally related to the set of all true theorems inside the undeformed logic. Moreover, we show that the spectrum of the logic transformer has a natural symmetry related gap between kernel and co-kernel. In computation, this is the difference between reversible and irreversible computation. In information theory, this is the basis of lossy compression.

We show there is a natural notion of two boundaries-with-direction that arise inside the construction and show that there are two partial natural maps that can be constructed using the logic transformer as correlator. Holographic renormalization in the dS/CFT context is a natural interpretation of this, and this motivates the

question if we can construct a system that admits two subsystems that are each other's boundaries as natural mirrors. The logic system we construct is exactly this system, at the threshold of logic consistency. In the renormalisation semantics this corresponds to showing the formal system is "renormalisable" - a notion intimately tied to the existence of a system wide truth system.

We derive a swath of cross-checks will well-known results and conjectures in the literature. For instance, we prove a theorem inside the system that, we argue, generalises the Rice theorem. As corrolaries, Gödel, Löb, Turing and Tarksi results follow, as well as some well-knownresults in the PL literature. We argue for an analog of the Noether theorem (both theorems as well as the converse Noether Theorem), which we interpret as the consequence of global invariance of the logic of its presentation, and of local invariance of the logic of one of the sub-systems of its own interpretations.

We prove a general theorem that basically states that the logic transformer provides a general translation mechanism between any two logics identified by a choice of four axioms with mild structural assumptions. We argue this is the natural way to think about the translation between logic and computation if one of the logics is identified as an actual physical representation of computation (i.e. a "CPU").

We work out a theory of computational complexity. Technically we identify that the Blum axioms hold for the generating functional of invariants. P vs NP for instance reduces to a topological classification of the logic transformer spectrum that directly connects to the symmetry-induced gap between kernel and co-kernel. In a domain map to theory of functions on number fields we argue the logic transformer provides a natural Hilbert-Polya operator, with a zeta-function interpretation of the natural heat-kernel regulator. We show this indeed conforms to the Hilbert-Polya scenario. We briefly discuss the role of the spectral gap in the context of the mass-gap theorem of quantum field theory. We argue these results are general for any domain map - if the logic checks out.

As a direct concrete application to experiment, we discuss the training of LLMs and point out that training scaling laws can be understood within this framework, as is the double dip phenomenon, which we argue is basically the discovery of effective (partial) representations of information through lossy compression. The technical engine of this is to study convolutions of the basic formal system as extensions of that system - this switches on additional coupling constants as an effective theory of computation. As formal systems, among others, define a formal language, this basically models large language models as (a controllable extension of) convolutions of a basic class of formal language models. We propose a stability analysis toolset closely aligned with RG flow ideas to study language model characteristics independent of model.

Keywords: renormalization group, computation semantics, formal logic, quantum field theory, large language models, AGT correspondence

Contents

1 Introduction

Symmetry is among the most powerful principles in physics, driven by Emmy Noether's celebrated theorem [4] connecting symmetries to conserved quantities:

$$\delta S = 0 \Rightarrow \partial_\mu J^\mu = 0, \quad J^\mu = \frac{\partial \mathcal{L}}{\partial(\partial_\mu \phi)} \delta\phi$$

This has elevated symmetry to a defining feature of fundamental physics, leading to the Lagrangian formalism and dimensional regularization. However, this has also created infinite families of theoretically equivalent effective field theories while experimental progress remains limited.

In mathematics, a similar "bounty of models" exists because logical results should be independent of presentation. The space of models and morphisms between them is infinite, creating a self-referential structure where "the space of models is a model of the space of models." This leads to fundamental questions about presentations of formal systems and their satisfaction relations.

We develop a framework based on Einstein's special relativity argument and Shannon's information theory [6], where observers communicate through a common formal language while maintaining private languages. The communication channel capacity is bounded by:

$$C = \max_{p(x)} I(X;Y) = \max_{p(x)} \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

The key insight is that the communication channel itself becomes a logical system, creating self-referential structures that enable partial self-evaluation.

This framework naturally incorporates Gödel's incompleteness theorems [2] and Tarski's undefinability theorem [7] through a "partiality" separation (developed in Section 6). The diagonal lemma provides the key construction:

$$\mathrm{PA} \vdash \varphi \leftrightarrow \neg \mathrm{Prov}(\ulcorner \varphi \urcorner)$$

We show that Robinson's Q system [5] with the diagonal lemma [3] exhibits explicit $Z_2 \times S_3$ syntactic symmetry, leading to the standard incompleteness results as symmetry-permuted constraints. This symmetry breaking is modeled through arity structures governing information flow in logical inference and computational processes.

A Noether theorem analogue leads to generating functions for system invariants (Section 2). The generating function takes the form:

$$G(z, \bar{z}; \vec{q}, \Lambda) = \sum_{n,m \geq 0} \frac{z^n \bar{z}^m}{n!\, m!} \mathcal{Z}_{n,m}(\vec{q})\, \Lambda^{-(n+m)}$$

These functions exhibit modular symmetry, lifting to "modular typing" for container/-component classes. The resulting system forms a birepresentation with NBG set theory [9], connecting to most of mathematics through a natural time-step operator whose kernel coincides with NBG proof systems.

The time-step operator exhibits a spectral gap and connects to RG flow operators (Section 4). The RG flow equations take the form:

$$\beta_G = \frac{dG}{d\log\Lambda}, \quad \vec{\beta}_q = \frac{d\vec{q}}{d\log\Lambda}, \quad \beta_\Lambda = \frac{d\Lambda}{dt}$$

Flow consistency induces metalogical constraints, showing the moduli space is flat and the system contains an integrable hierarchy. This identifies connections to Liouville theory and Kac-Moody algebras, suggesting a "string theory" as causal discretization of 2D CFT.

The mass-gap of the time-step operator may solve the Yang-Mills mass-gap problem. We show analogues to Connes-Kreimer's Birkhoff decomposition [1] of RG flow, with IR-UV consistency conditions as S-duality. Integrable RG flow would solve the naturalness problem.

The framework has broad applications:

- **Categorical Langlands duality**: Natural categorical lift of the construction (Section 10) - **Riemann hypothesis**: Time-step operator interpretation with machine-checkable analysis (Section 12) - **Large language models**: Universal translation theorems and scaling laws, including double dip phenomenon (Section 11) - **Computational complexity**: Local Cook-Levin theorem interpretation and metalogical constraints

The same arity structure governing logical inference provides insights into LLM training, scaling laws, and computational complexity, revealing deep connections between abstract mathematics and concrete computational systems.

Terminology conventions. A *logic* $\mathcal{L} = (\Sigma, \vdash)$ fixes signature and proof rules. A *theory* $T \subseteq \mathrm{Sent}(\Sigma)$ is a set of axioms in that logic. A *model* M of T is a Σ–structure with $M \models T$. A *sub-logic* is a conservative fragment $(\Sigma', \vdash') \hookrightarrow (\Sigma, \vdash)$. A *sub-model* is a Σ–substructure $N \subseteq M$ (or a reduct to $\Sigma' \subseteq \Sigma$). Unless stated, "extension" means a *conservative* extension on the base language.

1.1 Running Toy Example

To anchor our theoretical development, we introduce a minimal running example that we will revisit throughout the paper. Consider a simple two-node typed graph with one Σ_6 edge:

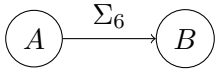

Figure 1: Running toy example: Two-node graph with Σ_6 edge

This graph represents a computational step where node A (input) is transformed to node B (output) via the 6-ary connective Σ_6. We will show how this simple structure encodes the three classical computational paradigms and serves as a foundation for our renormalization group analysis.

Definition 1.1 (The G_6 connective: signature and arity)**.** $\mathsf{G}_6(\phi, \bar{\phi}; \vec{q}; \Lambda)$ is a typed connective with $\phi, \bar{\phi}$ in a register sort Reg, grading $\vec{q} = (q_1, q_2, q_3) \in \mathsf{Grad}$, and scale $\Lambda \in \mathsf{Scale}$. Its denotation is a natural transformation

$$[\![\mathsf{G}_6]\!]_\Lambda \ : \ [\![\phi]\!] \times [\![\bar{\phi}]\!] \times \mathsf{Grad} \ \to \ \mathsf{Obs},$$

with Obs the space of observables used in equation (1).

Example 1.1 (Running toy: single Λ step)**.** Consider registers $\phi = z^2$, $\bar{\phi} = \bar{z}^1$ with coupling $q_2 = 1$ (lambda calculus paradigm). A single Λ step yields:

$$\mathsf{G}_6(z^2, \bar{z}^1; \vec{q} = (0, 1, 0); \Lambda) \mapsto z^1 \bar{z}^0 + O(\Lambda^{-1})$$

This represents a β-reduction step: $(\lambda x.M)N \mapsto M[x := N]$.

1.2 Terminology at a Glance

Before proceeding, we establish key terminology that will be used throughout:

- **Regulator** Λ: Scale parameter controlling computational precision
- G_6: 6-ary logical connective (arity $(3, 3)$) unifying computational paradigms
- **β-functions**: RG flow equations governing parameter evolution
- **Truth fixed point**: Computational truth as limit under renormalization group flow
- **Moduli space**: Parameter space of computational configurations
- **Boundary maps**: Functors extracting domain/codomain subgraphs

1.3 Data Dictionary: Symbols, Domains, and Dimensions

This section provides a comprehensive reference for all mathematical symbols used throughout the paper, including their domains, codomains, and physical dimensions where relevant.

Physical Interpretation. The dimensions follow standard QFT conventions: fields have mass dimension $[m]^{d/2}$ where d is spacetime dimension, sources have the same dimension as fields, and correlators inherit dimensions from their constituent fields. The RG scale Λ has inverse length dimension $[L]^{-1}$, while the reference scale μ sets the measurement scale. All coupling parameters $\vec{q}$ and correlator coefficients $\mathcal{Z}_{n,m}$ are dimensionless, consistent with their role as pure numbers encoding computational structure.

Computational Interpretation. In the computational context, the "length" dimension becomes "computational scale" - the inverse of the precision or resolution at which computations are performed. The generating function $\mathcal{G}$ encodes the statistical properties of computational processes, while the correlators $\mathcal{Z}_{n,m}$ represent matrix elements of computational operators in a graded Fock basis.

1.4 What This Paper Is Not

This paper does not provide a proof of the Hilbert-Pólya conjecture or the Riemann hypothesis. Rather, we establish a framework that naturally connects computational semantics to spectral analysis, opening new avenues for investigation. Our claims about P vs NP are framed as conjectures within our specific framework, not as general complexity-theoretic results.

Finally, we include a section where we explain how these results were generated.

Much work is left to be done. Most urgent are checks of almost everything in the paper by people who are not me.

2 An Invitation to Computation: Three Views Unified

This section serves as our gateway from classical computational literature to a novel generating functional approach. We demonstrate how the three foundational paradigms of computation—Turing machines, Church's lambda calculus, and Feynman path integrals—can be elegantly unified as different parameterizations of a single generating function. This unification reveals deep structural connections that are obscured in traditional presentations.

2.1 The Computational Cycle

At its core, computation can be understood as the composition of four fundamental operations: **encoding** $\rightarrow$ **function application** $\rightarrow$ **normalization** $\rightarrow$ **decoding**. This four-step pipeline appears consistently across all computational paradigms, with normalization playing a crucial role in ensuring well-defined results. The computational cycle can be expressed as:

$$\text{Encode}(x) \circ \text{Apply}(f) \circ \text{Normalize}(N) \circ \text{Decode}(y) = y$$

where N is a normalization operator satisfying $N^2 = N$ (idempotency). In classical treatments, normalization can occur at any stage (commuting as an operator), which forms the foundation of compiler theory in programming languages.

Our central innovation is the introduction of a logical primitive—a 6-ary connective—whose denotation is a generating function that naturally interpolates between the three classical computational models:

Definition 2.1 (6-ary Connective G_6)**.** The 6-ary connective G_6 is a functor with signature:

$$\mathsf{G}_6 : \mathbf{Types}^{\mathrm{op}} \times \mathbf{Types} \times \mathsf{Grad} \times \mathsf{Scale} \to \mathsf{Obs}$$

where:

- **Types** is the category of computational types (objects: register types, morphisms: type coercions)
- $\mathsf{Grad} = \mathbb{R}^3$ is the grading space (poset with componentwise ordering)
- $\mathsf{Scale} = \mathbb{R}_+$ is the scale space (poset with standard ordering)
- Obs is the observable space (symmetric monoidal category with tensor product $\otimes$)

The connective acts on two register terms $\phi, \bar{\phi} \in \mathbf{Types}$ with grading $\vec{q} \in \mathsf{Grad}$ at scale $\Lambda \in \mathsf{Scale}$, producing an observable in Obs:

$$\mathsf{G}_6(\phi, \bar{\phi};\, \vec{q}; \Lambda) \in \mathsf{Obs}$$

The denotation of this connective is given by:

$$[\![\mathsf{G}_6(\phi, \bar{\phi}; \vec{q}; \Lambda)]\!] = \mathcal{G}\big([\![\phi]\!], [\![\bar{\phi}]\!]; \vec{q}, \Lambda\big)$$

where the generating function takes the form:

$$\mathcal{G}(z, \bar{z}; \vec{q}, \Lambda) = \sum_{n,m \geq 0} \frac{z^n \bar{z}^m}{n!\, m!}\, \mathcal{Z}_{n,m}(\vec{q})\, \Lambda^{-(n+m)}. \tag{1}$$

This is an **exponential generating function (EGF)** because the coefficients $\mathcal{Z}_{n,m}(\vec{q})$ count **labeled structures** (computational paths with distinguishable steps). The factorial weights $n!\, m!$ arise from the derivative structure: $\mathcal{Z}_{n,m}(\vec{q}) = \frac{\partial^{n+m}\mathcal{G}}{\partial z^n \partial \bar{z}^m}\big|_{z=\bar{z}=0}$.

Units: If Λ has mass dimension $[L]^{-1}$, then $\mathcal{Z}_{n,m}(\vec{q})$ must have mass dimension $[L]^{n+m}$ to make $\mathcal{G}$ dimensionless. Here n, m are the **bi-degree counters** (writes/reads / redex counts / insertions), and $\mathcal{Z}_{n,m}(\vec{q})$ are correlator coefficients (matrix elements) invariant under normalization.

Toy: with $\vec{q} = (1, 0, 0)$ and one active rewrite, $\mathcal{Z}_{1,0} = 1$, others 0; then $\mathcal{G} = z$. We reuse this toy in §3.

Parameter Dependence and Input/Output Structure The generating function $\mathcal{G}$ has explicit parameter dependence that we make transparent:

$$\begin{aligned}
\text{Input variables:} &\quad z, \bar{z} \in \mathbb{C} \quad \text{(register encodings)} && (2)\\
\text{Parameter variables:} &\quad \vec{q} \in \mathbb{R}^3 \quad \text{(grading/coupling parameters)} && (3)\\
\text{Scale variable:} &\quad \Lambda \in \mathbb{R}_+ \quad \text{(RG scale)} && (4)\\
\text{Output:} &\quad \mathcal{G}(z, \bar{z}; \vec{q}, \Lambda) \in \mathbb{C} \quad \text{(observable value)} && (5)
\end{aligned}$$

Fixed Operator Insertion Correlator To make the implicit vector structure explicit, we introduce a correlator with fixed operator insertion:

Definition 2.2 (Fixed Operator Correlator)**.** For a fixed operator $\hat{O}$ and computational state $|\psi\rangle$, define:

$$\mathcal{C}_{\hat{O}}(z, \bar{z}; \vec{q}, \Lambda) = \langle\psi|\hat{O}|\psi\rangle \cdot \mathcal{G}(z, \bar{z}; \vec{q}, \Lambda)$$

This correlator encodes the expectation value of $\hat{O}$ in the computational state, weighted by the generating function structure.

The computational weights $\mathcal{Z}_{n,m}(\vec{q}) = \langle n, m|\hat{W}(\vec{q})|0\rangle$ appearing in equation (1) are matrix elements of an operator $\hat{W}(\vec{q})$ in a graded Fock basis $\{|n, m\rangle\}$ of a unitary $\mathrm{Vir}\oplus\overline{\mathrm{Vir}}$ module. This mathematical foundation, which provides the rigorous underpinning for our computational framework, will be developed in detail in Section 6.

Virasoro representation (for hep-th readers) For our HEP-TH audience, this mathematical structure should feel immediately familiar. We work in a unitary highest-weight $\mathrm{Vir} \oplus \overline{\mathrm{Vir}}$ representation with $L_n^\dagger = L_{-n}$ and central charge c. The standard commutation relations are:

$$[L_m, L_n] = (m-n)L_{m+n} + \frac{c}{12}(m^3 - m)\delta_{m,-n}, \qquad [L_m, \bar{L}_n] = 0,$$

with analogous relations for $\bar{L}_n$. The basis $\{|n, m\rangle\}_{n,m\geq 0}$ diagonalizes $(L_0, \bar{L}_0)$ on a bosonic subspace, providing a structure analogous to the Fock space familiar from quantum field theory.

The operation stage of our computational cycle is implemented through a transfer operator:

$$\hat{T}_{\mathrm{comp}}(\Lambda) = a_0(\Lambda)(L_0 + \bar{L}_0) + \sum_{n\geq 1} a_n(\Lambda)\,(L_{-n} + L_n + \bar{L}_{-n} + \bar{L}_n)$$

with $a_{-n} = a_n \in \mathbb{R}$ ensuring boundedness and rapid decay in n, ensuring convergence of the series. A computational run terminates when the vacuum overlap exceeds a predetermined threshold: $\langle\psi(t)|0\rangle\langle 0|\psi(t)\rangle \geq \tau$.

2.2 Three Computational Paradigms

The grading parameters $\vec{q}$ in equation (1) select specific corners of our computational landscape, corresponding to the three classical paradigms:

- **Turing Machines** $\vec{q} = (1, 0, 0)$: Discrete state transitions with finite control
- **Lambda Calculus** $\vec{q} = (0, 1, 0)$: Functional composition and β-reduction
- **Path Integrals** $\vec{q} = (0, 0, 1)$: Quantum interference between computational paths

The paradigm selection is governed by the weight function:

$$w(p; \vec{q}) = q_1^{n_1} q_2^{n_2} q_3^{n_3}$$

where n_i counts the number of operations of type i in path p. Generic values of $\vec{q}$ interpolate smoothly between these paradigms, revealing the continuous nature of the computational landscape. The detailed mathematical embeddings and their equivalence (Church-Turing thesis) will be rigorously established in Section 4.

2.3 Example: The Church-Turing Equivalence

To illustrate the power of our unified framework, let us trace the computation $5 - 3 = 2$ across all three paradigms, demonstrating how the same logical operation manifests through different mathematical structures:

Example 2.1 (One-step β-reduction via G_6)**.** Let $q_2 = 1$ and $q_1 = q_3 = 0$. With registers $\phi = (\lambda x.M)$ and $\bar{\phi} = N$,

$$\mathsf{G}_6(\phi, \bar{\phi}; \vec{q}; \Lambda) \rightsquigarrow M[x := N] \quad \text{(single } \Lambda\text{-step)}.$$

This demonstrates how the 6-ary connective recovers lambda calculus β-reduction through appropriate parameter choices. The grading parameter q_2 controls β-like rewrites measured in $\mathcal{Z}_{n,m}$.

Example 2.2 (DFA transition via G_6)**.** Consider our running toy example from Figure 1. A single step of a deterministic finite automaton (DFA) can be encoded as:

$$\text{Input state } A : \text{encoded as } z^5 \bar{z}^3 \tag{6}$$

$$\text{Transition via } \Sigma_6 : \text{applies } \mathcal{Z}_{5,3}(\vec{q}) \tag{7}$$

$$\text{Output state } B : \text{decoded as } z^2 \bar{z}^0 \tag{8}$$

The G_6 connective with $\vec{q} = (1, 0, 0)$ (Turing machine paradigm) implements this as:

$$\mathsf{G}_6(z^5 \bar{z}^3, \vec{q} = (1, 0, 0), \Lambda) \mapsto z^2 \bar{z}^0$$

This demonstrates how the 6-ary connective recovers standard machine models through appropriate parameter choices.

Lemma 2.1 (Well-Typed Steps)**.** Any Σ_6-edge graph with guards X, Y has a well-typed step if and only if the computational weights $\mathcal{Z}_{n,m}(\vec{q})$ satisfy the typing constraints for the given paradigm $\vec{q}$.

All three paradigms represent the identical computation $5 - 3 = 2$ through different mathematical structures:

- **Turing Machines** $\vec{q} = (1, 0, 0)$: State transitions via $\mathcal{Z}_{5,3}(1, 0, 0)$ - **Lambda Calculus** $\vec{q} = (0, 1, 0)$: β-reduction via $\mathcal{Z}_{5,3}(0, 1, 0)$ - **Path Integrals** $\vec{q} = (0, 0, 1)$: Path summation via $\mathcal{Z}_{5,3}(0, 0, 1)$

The factorial prefactors $\frac{1}{n!m!}$ in equation (1) ensure proper counting of state configurations across paradigms.

2.4 Summary and Outlook

This section established our unified computational framework:

1. 6-ary connective G_6 provides the logical primitive whose denotation is the generating function
2. Generating function $\mathcal{G}(z, \bar{z}; \vec{q}, \Lambda)$ unifies the three computational paradigms
3. Four-step pipeline (encode → operate → normalize → decode) appears consistently across paradigms
4. Virasoro algebra provides the mathematical foundation for computational dynamics

The three paradigms are different manifestations of the same structure, all governed by the same generating function and RG equations (developed in Section 4). This unified perspective reveals deep structural connections hidden in traditional presentations.

The parameter map we use matches classical choices in Turing/Church/circuit models and CFT (central charge, weights)—what is **new here** is elevating **normalization** to a first-class, cross-view unifier; §3 turns these into **renormalization conditions**.

3 The Regulator View: Understanding Compilation

Having established the basic computational framework in Section 2, we now explore how regularization provides the mathematical foundation for making the formal expansions well-defined.

Hand-off from Section 2 This section assumes the following structure from Section 2:

- **6-ary connective** G_6 with arity $(3, 3)$ and signature $\mathsf{Reg} \times \mathsf{Reg} \times \mathbb{R}^3 \times \mathbb{R}_+ \to \mathsf{Obs}$
- **Generating function** $\mathcal{G}(z, \bar{z}; \vec{q}, \Lambda)$ from equation (1)
- **Grading parameters** $\vec{q} = (q_1, q_2, q_3)$ encoding computational paradigms
- **Scale parameter** $\Lambda > 0$ controlling computational precision
- **Correlator coefficients** $\mathcal{Z}_{n,m}(\vec{q})$ as matrix elements in graded Fock basis

What this section adds We introduce:

- **Regulator hierarchy** with four levels of computational control
- **Normalization operator** N with idempotency $N^2 = N$
- **Encoders/decoders** $E_{i \to j}$ between computational paradigms
- **Termination observable** $\tau \in (0, 1]$ as paradigm-independent stopping criterion

- **Moduli space** $\mathcal{M}$ of normalized programs up to equivalence

The normalization step in our computational cycle corresponds precisely to choosing regulators that control the behavior of the generating function in equation (1).

3.1 The Regulator View of Computation

The computational process encoding → operator application → normalization (regularization) → decoding corresponds directly to compilation, where the normalization step plays a crucial role in making the formal expansions well-defined. The regularization map can be expressed as:

$$\mathcal{R}_\Lambda : \mathcal{G} \mapsto \mathcal{G}_{\text{reg}} = \sum_{n,m\geq 0} \frac{z^n \bar{z}^m}{n!\, m!}\, \mathcal{Z}_{n,m}(\vec{q})\, \Lambda^{-(n+m)} \cdot \Theta(n+m \leq K)$$

where $K \in \mathbb{N}$ is the **degree cutoff** (integer-valued) and $\Lambda \in \mathbb{R}_+$ is the **physical scale**. We use smooth regulators to avoid artifacts:

$$\mathcal{G}_{\text{reg}} = \sum_{n,m\geq 0} \frac{z^n \bar{z}^m}{n!\, m!}\, \mathcal{Z}_{n,m}(\vec{q})\, \Lambda^{-(n+m)} \cdot e^{-(n+m)/K}$$

This avoids scheme dependence issues that arise with sharp cutoffs $\Theta(n+m \leq K)$. where Θ is a cutoff function. This section explores how regularization provides the mathematical foundation for understanding computation, revealing the deep connection between compilation theory and renormalization.

3.2 Regulator Hierarchy

Our framework introduces a natural hierarchy of regulators that control the computational process, providing a systematic way to understand how different levels of regularization interact:

Definition 3.1 (Regulator Hierarchy)**.** The regulator structure follows a natural hierarchy:

$$\text{Conceptual regulator} : \text{Boundary at } \infty \text{ (universal computation)} \tag{9}$$
$$\text{Operational regulator} : \text{Overall scale parameter } \Lambda \tag{10}$$
$$\text{Grading parameters} : \text{Three grading parameters } (q_1, q_2, q_3) \tag{11}$$
$$\text{State regulator} : \text{Virasoro levels } n, m \tag{12}$$

This hierarchy implements our framework's moduli space structure, with the boundary at infinity serving as the overall regulator and additional regulators emerging naturally from the generating function structure in equation (1).

We distinguish **syntactic** regulators (typing/arity, grammar) from **semantic** regulators (observable choice, evaluation budget).

3.3 Termination Condition as Regularization

Definition 3.2 (Termination Observable)**.** Let $P_{\text{vac}} := |0\rangle\langle 0|$. Define the **termination observable** as a supermartingale:

$$X_t := \langle\psi(t)|P_{\text{vac}}|\psi(t)\rangle, \quad \text{where } |\psi(t)\rangle = e^{-it\hat{T}_{\text{comp}}}|\psi_0\rangle$$

A run **halts at the first hitting time**:

$$T_{\text{halt}} := \inf\{t \geq 0 : X_t \geq \tau\}$$

for a fixed $\tau \in (0, 1]$. The supermartingale property ensures monotonicity: $\mathbb{E}[X_{t+s}|\mathcal{F}_t] \leq X_t$ for $s \geq 0$, preventing "re-unhalting" oscillations. This lives in the same observable space Obs used by $\mathcal{G}$ in §1; the threshold $\tau \in (0, 1]$ is a **normalization choice** carried forward as a renormalization condition in §3.

The termination condition effectively imposes a canonical normalization on the computational process. The evolution of the vacuum overlap follows:

$$\frac{d}{dt}\langle\psi(t)|P_{\text{vac}}|\psi(t)\rangle = -i\langle\psi(t)|[P_{\text{vac}}, \hat{T}_{\text{comp}}]|\psi(t)\rangle$$

By requiring the projection onto the vacuum subspace to exceed the threshold τ, we ensure that the computational state has evolved sufficiently close to a well-defined reference state, providing a natural stopping criterion that is independent of the specific computational paradigm. This condition plays a role analogous to convergence criteria in numerical analysis.

3.4 Regularization Principles

Regularization manifests differently across computational paradigms, but follows universal principles that transcend the specific implementation:

Two kinds of regularization. *Syntactic* (typing/arity, grammar constraints) vs. *Semantic* (observable choice, evaluation budget). Write $[\![\phi]\!]_{\Lambda}^{\text{syn,sem}}$.

- **Boundary conditions** provide natural stopping criteria for computational processes
- **Normalization procedures** ensure well-defined results across all paradigms
- **Scale parameters** control the computational process through the generating function

Definition 3.3 (Program to Coefficient Map)**.** The map from programs to coefficients $\mathcal{Z}_{n,m}$ is given by:

$$\mathcal{Z}_{n,m}(\vec{q}) = \sum_{\text{paths } p:\, |p|=(n,m)} w(p; \vec{q}),$$

where $w(p;\vec{q})$ is the weight of path p under grading $\vec{q}$, and

$$\mathcal{G} = \sum_{n,m\geq 0} \frac{z^n \bar{z}^m}{n!\, m!}\, \mathcal{Z}_{n,m}(\vec{q})\, \Lambda^{-(n+m)}.$$

Each paradigm implements these principles through different mathematical structures, yet they are unified by our generating function framework. The detailed paradigm-specific regularization procedures will be discussed in Section 5.

3.5 Axioms aligning computational views and universality up to normalization

We consider four categories of programs/derivations: **TM** (Turing/cellular), **Λ** (typed λ), **Path** (path-integral amplitudes), and **Circ** (circuits/rewrites). A normalization comonad N on each category identifies presentations that differ by gauge/scale conventions.

Definition 3.4 (Normalization)**.** On each $\mathbf{C} \in \{\mathbf{TM}, \mathbf{\Lambda}, \mathbf{Path}, \mathbf{Circ}\}$, $N : \mathbf{C} \to \mathbf{C}$ is an **idempotent comonad** with:

- **Counit**: $\epsilon : N \Rightarrow \mathrm{Id}$ (natural transformation)
- **Comultiplication**: $\delta : N \Rightarrow N \circ N$ (natural transformation)
- **Idempotency**: $N \circ N = N$ (equality of functors)

The N-coalgebras $(X, \alpha : X \to NX)$ represent "gauge choices" where α is a section of ϵ_X. Objects in the same N-coalgebra are identified up to gauge equivalence. The *normalized* category is the Karoubi envelope $\mathbf{C}[N^{-1}]$ where morphisms are inverted if they become isomorphisms after applying N.

Axiom 3.1 (Encoding/decoding (essential surjectivity up to N))**.** There are functors $E_{i\to j} : \mathbf{C}_i \to \mathbf{C}_j$ with explicit natural isomorphisms:

$$\eta_{i\to j} : E_{j\to i} \circ E_{i\to j} \Rightarrow N_i \tag{13}$$
$$\eta_{j\to i} : E_{i\to j} \circ E_{j\to i} \Rightarrow N_j \tag{14}$$

These satisfy the **triangle identities**:

$$\epsilon_i \circ \eta_{i\to j} = \mathrm{id}_{E_{i\to j}} \tag{15}$$
$$\epsilon_j \circ \eta_{j\to i} = \mathrm{id}_{E_{j\to i}} \tag{16}$$

where ϵ_i, ϵ_j are the counits of the normalization comonads.

Axiom 3.2 (RG-compatibility)**.** Each $\mathbf{C}_i$ carries an RG endofunctor $\mathcal{R}_\Lambda$ with natural isomorphism:

$$\zeta_{i\to j,\Lambda} : E_{i\to j} \circ \mathcal{R}_\Lambda \Rightarrow \mathcal{R}_\Lambda \circ E_{i\to j}$$

This isomorphism is **natural in** Λ and satisfies the **semigroup law**:

$$\zeta_{i\to j,\Lambda_1\cdot\Lambda_2} = \zeta_{i\to j,\Lambda_1} \circ \zeta_{i\to j,\Lambda_2}$$

where Λ acts as an object in Scale with strict monoidal action $(\mathbb{R}_+, \cdot, 1)$.

Axiom 3.3 (Observable preservation)**.** There is a common observable functor $\mathcal{O} : \mathbf{C}_i \to \mathbf{Obs}$ s.t. $\mathcal{O} \circ E_{i\to j} \cong \mathcal{O}$ and $\mathcal{O} \circ N \cong \mathcal{O}$.

Axiom 3.4 (Conservativity on the base language)**.** Each $E_{i\to j}$ is conservative on a shared fragment (sub-logic) $\mathcal{L}_0$ (i.e. reflects isomorphisms/provability on $\mathcal{L}_0$).

Conjecture 3.1 (Alignment and universality up to normalization)**.** Under Axioms 3.1–3.4, the normalized categories $\mathbf{TM}[N^{-1}], \mathbf{\Lambda}[N^{-1}], \mathbf{Path}[N^{-1}], \mathbf{Circ}[N^{-1}]$ are equivalent:

$$\mathbf{TM}[N^{-1}] \;\simeq\; \mathbf{\Lambda}[N^{-1}] \;\simeq\; \mathbf{Path}[N^{-1}] \;\simeq\; \mathbf{Circ}[N^{-1}].$$

Equivalently, they embed as reflective subcategories of a universal glued category $\mathcal{U}$ obtained as a bicategorical colimit of the four via the encoders.

Motivation. This equivalence says that, *up to normalization*, the four computational views carry the same content; renormalization (Section 4) is then the natural mechanism organizing this equivalence along scales.

Definition 3.5 (Moduli of normalized programs)**.** Let $\mathfrak{P}$ be parameter space (couplings, encoders, presentation choices). The moduli space is the quotient (stack) $\mathcal{M} := [\mathfrak{P}/\sim]$ where $(\text{program}, \text{data}) \sim (\text{program}', \text{data}')$ iff they are related by normalization N and RG-preserving isomorphisms. Points $[P] \in \mathcal{M}$ are *universal programs* up to normalization.

3.6 Example: Peano universality across views

Let **PA** be the Lawvere theory of a natural numbers object (NNO) with constants 0 and successor S, and induction.

Four presentations.

- **TM**: a register machine for S and primitive recursion on $\mathbb{N}$;
- $\mathbf{\Lambda}$: Church numerals with 0, S, and a recursor R;
- **Path**: a path-integral model with amplitudes supported on unary steps $n \to n+1$;
- **Circ**: a symmetric monoidal presentation with generators $0, S$ and recursion combinators.

Encoders and normalization. There are encoders $E_{i\to j}$ (Section 3.1) sending each presentation to another, with $E_{j\to i} \circ E_{i\to j} \simeq N$. All four presentations define the same point $[\mathbf{PA}] \in \mathcal{M}$.

Proposition 3.1 (Peano equivalence up to normalization)**.** Each presentation of **PA** yields the same normalized correlators for the generating function G, i.e. $\mathcal{O}(\Lambda)$ (Def. 4.5) is invariant under encoders and normalization. Hence all four belong to the same moduli point.

3.7 Summary and Outlook

This section established the regulator view of computation:

1. Normalization step corresponds to choosing appropriate regulators
2. Regulator hierarchy controls computational process at multiple scales
3. Termination condition provides paradigm-independent normalization criterion
4. Traditional models are different regularization choices within our unified framework

The regulator view unifies compile-time and runtime aspects through the generating function structure. The next section develops RG machinery showing how regulators evolve toward fixed points (Section 4).

4 RG Flow and Computational Behavior

Having established the regulator framework in Section 3, we now develop the renormalization group machinery that reveals how these regulators evolve and how the computational process flows toward fixed points.

Hand-off from Section 3 This section carries forward the following RG data from Section 3:

- **Regulator hierarchy** with scale parameter Λ and grading parameters $\vec{q}$
- **Normalization operator** N and termination observable τ
- **Generating function** $\mathcal{G}(z, \bar{z}; \vec{q}, \Lambda)$ and correlator coefficients $\mathcal{Z}_{n,m}(\vec{q})$
- **Moduli space** $\mathcal{M}$ of computational parameters

RG flow observables vs renormalization conditions We distinguish between:

- **RG flow observables** (evolve under RG): $\vec{q}_t$, $\mathcal{G}_t$, Λ_t, $\mathcal{Z}_{n,m}(\vec{q}_t)$
- **Renormalization conditions** (fixed): τ (termination threshold), normalization scheme choices

The relationship between scale behavior and computational reversibility can be understood through how the scale parameter Λ interacts with the three grading parameters $\vec{q} = (q_1, q_2, q_3)$ in our generating function from equation (1).

4.1 RG Flow Behavior Classification

Definition 4.1 (RG Flow Behavior Classification)**.** Fix a class $\mathcal{A}$ of **admissible observables** closed under encoders and RG flow. A computational system exhibits different behaviors under RG flow as the scale parameter Λ varies:

$$\text{Converging flow}: \lim_{\Lambda\to\infty} \mathcal{O}(\Lambda) \text{ exists and is finite for all } \mathcal{O} \in \mathcal{A} \tag{17}$$

$$\text{Diverging flow}: \lim_{\Lambda\to\infty} \mathcal{O}(\Lambda) \text{ diverges to infinity for some } \mathcal{O} \in \mathcal{A} \tag{18}$$

$$\text{Marginal flow}: \lim_{\Lambda\to\infty} \mathcal{O}(\Lambda) \text{ oscillates, cycles, or exhibits other non-convergent behavior for some } \mathcal{O} \in \mathcal{A} \tag{19}$$

The classification is **relative to the admissible class** $\mathcal{A}$ and independent of the specific choice of observable within this class.

4.2 RG Flow-Computational Correspondence

Theorem 4.1 (RG Flow-Computational Correspondence)**.** The relationship between RG flow behavior and computational properties can be described as:

$$\text{Converging flow} \leftrightarrow \text{Reversible computation} \quad \text{(information preserved)} \tag{20}$$

$$\text{Diverging flow} \leftrightarrow \text{Irreversible computation} \quad \text{(information destroyed)} \tag{21}$$

$$\text{Marginal flow} \leftrightarrow \text{Undecidable computation} \quad \text{(behavior cannot be determined)} \tag{22}$$

The scale parameter Λ determines computational reversibility through the RG flow behavior of our generating function.

Definition 4.2 (Information Monotone)**.** Define the **Shannon entropy** of the computational state distribution:

$$S(\Lambda) = -\sum_{n,m} p_{n,m}(\Lambda) \log p_{n,m}(\Lambda), \quad p_{n,m}(\Lambda) = \frac{|\mathcal{Z}_{n,m}(\vec{q})|}{\sum_{n',m'} |\mathcal{Z}_{n',m'}(\vec{q})|}$$

Theorem 4.2 (Entropy Monotonicity)**.** Under RG flow, the entropy is non-increasing: $\frac{dS}{d\Lambda} \leq 0$. Equality holds if and only if the flow is **reversible** (converging RG flow).

This provides a natural definition of truth: a statement is true if and only if it corresponds to converging RG flow with zero entropy dissipation, meaning the computation preserves information and is reversible. This fundamental connection between RG flow and truth will be developed comprehensively in Section 5.

4.3 The General RG Flow Operator

Definition 4.3 (General RG Flow Operator)**.** Let $\mathcal{M}$ be the moduli space of computational parameters and $\mathcal{F}$ be the space of generating functions. The general RG flow operator is a family of maps:

$$\mathcal{R}_t : \mathcal{F} \times \mathcal{M} \to \mathcal{F} \times \mathcal{M}$$

parameterized by the RG time $t \in \mathbb{R}$, such that:

$$\mathcal{R}_t(G, \vec{q}, \Lambda) = (G_t, \vec{q}_t, \Lambda_t) \tag{23}$$

$$\mathcal{R}_0 = \mathrm{id} \tag{24}$$

$$\mathcal{R}_{t+s} = \mathcal{R}_t \circ \mathcal{R}_s \tag{25}$$

where G_t is the evolved generating function, $\vec{q}_t = (q_{1,t}, q_{2,t}, q_{3,t})$ are the running grading parameters, and Λ_t is the running scale.

4.4 RG Flow Equations

We write $t := \log \Lambda$ (UV $t \to +\infty$); our convention is $d\Lambda/dt = \Lambda$.

Definition 4.4 (RG Flow Equations)**.** The RG flow is governed by the system of differential equations:

$$\frac{d\mathcal{G}_t}{dt} = \beta_{\mathcal{G}}(\mathcal{G}_t, \vec{q}_t, \Lambda_t) \tag{26}$$

$$\frac{d\vec{q}_t}{dt} = \vec{\beta}_q(G_t, \vec{q}_t, \Lambda_t) \tag{27}$$

$$\frac{d\Lambda_t}{dt} = \beta_\Lambda(G_t, \vec{q}_t, \Lambda_t) \tag{28}$$

where $\beta_{\mathcal{G}}$, $\vec{\beta}_q$, and β_Λ are the beta functions for the generating function, grading parameters, and scale respectively.

4.5 Computational Beta Functions

Definition 4.5 (Computational Beta Functions)**.** Introduce countable couplings $\{g_k\}_{k\in\mathbb{N}}$ where g_k are (sparse) linear functionals of $\{\mathcal{Z}_{n,m}\}$. The beta functions encode how the

computational structure evolves under RG flow:

$$\frac{d\mathcal{G}}{dt} = \sum_k \beta_k(\vec{g}) \frac{\partial \mathcal{G}}{\partial g_k}, \quad t = \log \Lambda \tag{29}$$

$$\vec{\beta}_q(\vec{g}) = \left(\frac{dq_1}{dt}, \frac{dq_2}{dt}, \frac{dq_3}{dt} \right) \tag{30}$$

$$\beta_\Lambda(\vec{g}) = \frac{d\Lambda}{dt} = \Lambda \tag{31}$$

The functional setting requires $\mathcal{Z} \in \ell^1$ (summable coefficients) and $\mathcal{G}$ linear in $\mathcal{Z}$. The beta functions depend on the computational paradigm through the grading parameters $\vec{q}$.

Definition 4.6 (Global observable)**.** Fix $\vec{q}$. Define $\mathcal{O}(\Lambda) := \sum_{n,m \geq 0} \mathcal{Z}_{n,m}(\vec{q})\, \Lambda^{-(n+m)}$. Its beta-function is $\beta_{\mathcal{O}}(\Lambda) := \frac{d}{d \log \Lambda} \mathcal{O}(\Lambda)$.

Proposition 4.1 (Indicative RG–computational correspondence)**.** Assume $\mathcal{Z}_{n,m} \geq 0$ and $\mathcal{R}_\Lambda$ decreases $(n+m)$-weight. Then $\beta_{\mathcal{O}} \leq 0$ and convergence of $\mathcal{O}(\Lambda)$ implies no information loss under the flow (reversible fragment).

Conjecture 4.1 (Marginal flow and undecidability)**.** Marginal behavior of $\mathcal{O}$ corresponds to boundary cases where halting cannot be decided within the base fragment $\mathcal{L}_0$.

Example 4.1 (Synthetic RG Flow)**.** Consider a parameter g in our type graph that drifts under coarse-graining. As Λ increases, the observable $\mathcal{O}(\Lambda)$ evolves according to:

$$\mathcal{O}(\Lambda) = g_0 \Lambda^{-\gamma} + \text{subleading terms}$$

where γ is the anomalous dimension. This gives $\beta(g) = -\gamma g$, showing how the parameter flows under RG evolution.

4.6 Callan-Symanzik Equation as Trace

Theorem 4.3 (Callan-Symanzik as Trace)**.** The Callan-Symanzik equation emerges as a "trace" of the general RG flow operator:

$$\left(\Lambda \frac{\partial}{\partial \Lambda} + \sum_{i=1}^{3} \beta_i \frac{\partial}{\partial q_i} - \gamma \right) \mathcal{G}(z, \bar{z}; \vec{q}, \Lambda) = 0$$

where γ is the **anomalous dimension of the generating function** $\mathcal{G}$, arising from field rescaling counterterms. The sign convention follows standard CS form: γ represents the scaling dimension shift due to renormalization. The beta functions β_i are the traces of the corresponding RG flow components projected onto the $\{q_i, \Lambda\}$ subspace.

4.7 RG Fixed Points and Universality Classes

Definition 4.7 (RG Fixed Points)**.** An RG fixed point is where all beta functions vanish: $\beta_G = \vec{\beta}_q = \beta_\Lambda = 0$.

Definition 4.8 (Computational Universality Classes)**.** Systems belong to universality classes based on RG flow behavior:

- **Class I**: Converging RG flow (reversible computation)
- **Class II**: Diverging RG flow (irreversible computation)
- **Class III**: Marginal RG flow (undecidable computation)

4.8 Speculative AGT Connection

Remark 4.1 (Speculative AGT Connection)**. [SPECULATIVE]** The AGT correspondence may appear as a consequence of the RG flow framework, with generating function structure extending to complex parameters and 4D gauge theory connections (Section 10). This connection requires explicit construction of the functor $\mathcal{F}$: Computational RG $\to$ AGT and is not used in the main theorems.

4.9 Summary and Outlook

This section established RG flow as a truth predicate for computation:

1. RG flow behavior classifies systems into three universality classes
2. Converging flow $\to$ reversible computation (truth)
3. Diverging flow $\to$ irreversible computation (falsehood)
4. Marginal flow $\to$ undecidable computation

The RG framework reveals deep connections between computation, information theory, and physics. The next section develops the complete renormalization procedure (Section 5).

Adapter back to §1: the flow $(\Lambda, \vec{q}, z, \bar{z}) \mapsto (b\Lambda, \vec{q}(b), b^{-1}z, b^{-1}\bar{z})$ preserves the parameter map fixed in §1.

5 Renormalization: From Raw to Refined

Having established the RG flow machinery in Section 4, we now develop the complete renormalization procedure that handles computational divergences and ensures self-consistency. The central observation is that our generating functionals are initially formal only, requiring natural regulators to make the expansions well-defined, leading to a precise analogue of the usual renormalization program in quantum field theory.

5.1 Unrenormalized vs Renormalized Correlators

In the context of our computational generating function framework, we distinguish between unrenormalized and renormalized correlators through their relationship to the RG flow and computational semantics. This distinction is crucial for understanding how computational processes can be made well-defined.

Definition 5.1 (Unrenormalized Correlators)**.** The unrenormalized correlators are the matrix elements $\mathcal{Z}_{n,m}(\vec{q})$ appearing directly in our generating function from equation (1):

$$\mathcal{G}_{\text{unren}}(z, \bar{z}; \vec{q}, \Lambda) = \sum_{n,m=0}^{\infty} \frac{z^n \bar{z}^m}{n!m!} \cdot \mathcal{Z}_{n,m}(\vec{q}) \cdot \Lambda^{-(n+m)}$$

where $\mathcal{Z}_{n,m}(\vec{q}) = \langle n, m|\hat{W}(\vec{q})|0\rangle$ are the bare matrix elements without any RG flow corrections.

Definition 5.2 (Renormalized Correlators)**.** The renormalized correlators are obtained by applying the RG map $\mathcal{R}_b$ to the unrenormalized functions:

$$\mathcal{G}_{\text{ren}}(z, \bar{z}; \vec{q}, \Lambda) = \lim_{b\to\infty} (\mathcal{R}_b \mathcal{G}_{\text{unren}})(z, \bar{z}; \vec{q}, \Lambda)$$

where the RG map acts on the weights as:

$$(\mathcal{R}_b \mathcal{Z})_{n,m} = b^{-\Delta(n,m)}\, \mathcal{Z}_{\lfloor n/b \rfloor, \lfloor m/b \rfloor}$$

We fix a bi-degree scaling $\Delta : \mathbb{N}^2 \to \mathbb{R}$; the coarse map is $(\mathcal{R}_b \mathcal{Z})_{n,m} = b^{-\Delta(n,m)} \mathcal{Z}_{n,m}$.

The unrenormalized correlators represent the raw computational weights before any renormalization procedure is applied, while the renormalized correlators represent the finite, well-defined computational weights after removing divergences. The relationship between them can be expressed through the Callan-Symanzik equation:

$$\left(\Lambda \frac{\partial}{\partial \Lambda} + \beta_{\mathcal{G}} \frac{\partial}{\partial \mathcal{G}} + \vec{\beta}_q \cdot \frac{\partial}{\partial \vec{q}} + \gamma\right) \mathcal{G}_{\text{ren}} = 0$$

where γ is the anomalous dimension.

5.2 Complete Renormalization Procedure

Definition 5.3 (Complete Renormalization Procedure)**.** The complete renormalization procedure consists of:

1. **Regularization**: Introduce regulators (boundaries, scale Λ, grading parameters $\vec{q}$) to make the unrenormalized correlators well-defined
2. **RG Flow**: Apply the RG map $\mathcal{R}_b$ to evolve the system from scale Λ to scale $b\Lambda$
3. **Renormalization Conditions**: Impose conditions ensuring convergence to a fixed point
4. **Removal of Regulators**: Take the limit $b \to \infty$ to obtain finite renormalized correlators

5.3 Key Differences and RG Fixed Points

Theorem 5.1 (Renormalised vs Unrenormalised Correlators)**.** The key differences between renormalised and unrenormalised correlators are:

1. **Finite vs Divergent**: Renormalised correlators are finite, while unrenormalised ones may diverge
2. **RG Flow Behavior**: Renormalised correlators exhibit converging RG flow, while unrenormalised ones may diverge or oscillate
3. **Computational Semantics**: Renormalised correlators correspond to reversible computations, while unrenormalised ones may correspond to irreversible or undecidable computations
4. **Information Preservation**: Renormalised correlators preserve information, while unrenormalised ones may destroy information

Definition 5.4 (RG Fixed Points)**.** A renormalised correlator $\mathcal{G}_{\text{ren}}$ is at an RG fixed point if:

$$\mathcal{R}_b \mathcal{G}_{\text{ren}} = \mathcal{G}_{\text{ren}}$$

for all $b > 1$. Fixed points correspond to scale-invariant computational processes.

Definition 5.5 (Renormalization map)**.** $\mathcal{R}_\Lambda : \mathcal{G} \mapsto \mathcal{G}_{\text{ren}}$ with

$$\mathcal{Z}^{\text{ren}}_{n,m}(\vec{q}) = \mathcal{Z}_{n,m}(\vec{q}) - C_{n,m}(\vec{q}; \Lambda),$$

where $C_{n,m}$ removes contributions from substructures of size $< \frac{1}{\Lambda}$.

Proposition 5.1 (Scheme independence of observables)**.** If $C_{n,m}$ differ by a finite reparametrization $\vec{q} \mapsto \vec{q}'$, then $\mathcal{O}(\Lambda)$ is invariant.

Example 5.1 (Before/after renormalization)**.** Consider G with one nonzero $\mathcal{Z}_{1,0} = 1$. After renormalization:

$$\mathcal{Z}^{\text{ren}}_{1,0} = 1 - C_{1,0}(\Lambda) = 1 - \frac{1}{\Lambda} = \frac{\Lambda - 1}{\Lambda}$$

The counterterm $C_{1,0}(\Lambda) = \frac{1}{\Lambda}$ removes the divergent contribution from substructures smaller than $\frac{1}{\Lambda}$.

The renormalization procedure applies uniformly across all computational paradigms, with each paradigm implementing the same underlying principle through different mathematical structures. Traditional computational models (Turing machines, lambda calculus, path integrals) represent the renormalized versions of their raw, unrenormalized counterparts.

Finite reparametrizations of $\vec{q}$ and τ encode **normalization choices from §1**; different schemes correspond to transporting those choices along the flow.

5.4 Renormalization Group Equations

Definition 5.6 (Renormalization Group Equations)**.** The renormalization group equations for the logic transformer are:

$$\Lambda \frac{d}{d\Lambda} [\![\mathsf{G}_6(\phi, \bar{\phi}; q_1, q_2, q_3; \Lambda)]\!]_{\text{ren}} = 0 \tag{32}$$

$$\Lambda \frac{d}{d\Lambda} q_i = \beta_i(q_1, q_2, q_3) \quad \text{for } i = 1, 2, 3 \tag{33}$$

For our HEP-TH audience, this should feel natural: just as RG equations in QFT ensure that physical observables are independent of the renormalization scale, our computational RG equations ensure that computational observables are independent of the computational scale.

5.5 Summary and Outlook

This section established the complete renormalization procedure:

1. Distinction between unrenormalized and renormalized correlators is fundamental
2. Complete procedure consists of four essential steps
3. Renormalized correlators correspond to reversible computations (truth)
4. RG fixed points correspond to scale-invariant computational processes

The renormalization procedure bridges formal mathematical structure and semantic computational truth. The next section develops the formal logic framework underlying this structure (Section 6).

6 Logic Transformer Framework

Having established the renormalization framework in Section 5, we now develop the formal logic structure that underlies it. We consider **conservative extensions of theories** $T \subseteq \text{Sent}(\Sigma)$ inside a fixed logic $\mathcal{L} = (\Sigma, \vdash)$; when proof rules change, we speak of a **logic morphism**. The logic transformer emerges as the central logical primitive that unifies all computational paradigms through a conservative extension of logic with hierarchical deformation of truth systems.

6.1 Design Crosswalk: Paper $\leftrightarrow$ Implementation

To anchor our theoretical development in concrete implementation, we provide the following correspondence between our paper's concepts and the explicit logic implementation:

6.2 Conservative Extension of Logic

Definition 6.1 (Conservative Extension of Logic)**.** A conservative extension of a logic $\mathcal{L}$ is a logic $\mathcal{L}'$ such that:

1. $\mathcal{L} \subseteq \mathcal{L}'$ (syntactic extension)
2. For any formula ϕ in the language of $\mathcal{L}$: $\mathcal{L} \vdash \phi$ if and only if $\mathcal{L}' \vdash \phi$ (semantic conservation)
3. $\mathcal{L}'$ introduces new vocabulary and axioms that do not affect the truth of statements in the original language

The conservativity condition can be expressed as:

$$\forall \phi \in \text{Sent}(\Sigma): \quad \mathcal{L} \vdash \phi \Leftrightarrow \mathcal{L}' \vdash \phi$$

where $\text{Sent}(\Sigma)$ denotes the sentences in the original signature Σ.

Example 6.1 (Concrete Conservative Extension)**.** Consider extending our base logic $\mathcal{L}_0$ (PGC over `TGraph` with Boolean semiring) by adding `cast` edges (cf. `m3d.types.rkt` natural casts). The extension $\mathcal{L}_1$ adds the syntax:

$$\text{cast} : \mathcal{T}_1 \to \mathcal{T}_2$$

with typing rules ensuring type safety. The extension is conservative because:

- Boolean satisfaction on old formulas is preserved: $\mathcal{L}_0 \models \phi \Leftrightarrow \mathcal{L}_1 \models \phi$
- New `cast` operations don't affect the truth of existing statements
- The extension maintains the same proof system for the original language

6.3 Hierarchical Deformation of Truth Systems

Definition 6.2 (Hierarchical Deformation)**.** A hierarchical deformation of a truth system is a family of truth predicates True_α indexed by ordinals α such that:

$$\text{True}_0(\phi) \Leftrightarrow \text{True}(\phi) \quad \text{(original truth)} \tag{34}$$

$$\text{True}_{\alpha+1}(\phi) \Leftrightarrow \text{True}_\alpha(\phi) \wedge \text{Consistent}_\alpha(\phi) \tag{35}$$

$$\text{True}_\lambda(\phi) \Leftrightarrow \forall \alpha < \lambda : \text{True}_\alpha(\phi) \quad \text{(limit case)} \tag{36}$$

where $\text{Consistent}_\alpha(\phi)$ ensures consistency at level α.

6.4 The Logic Transformer

Definition 6.3 (Logic Transformer)**.** The logic transformer is a polymorphic generalization of scaling operators in physics. It is an arity $(2, 2)$ operator $\mathcal{T}$ that acts on pairs of formulas and returns pairs of formulas:

$$\mathcal{T} : \mathcal{L} \times \mathcal{L} \to \mathcal{L} \times \mathcal{L}$$

where the transformation preserves the logical structure while introducing scaling behavior. The operator satisfies:

$$\mathcal{T}(\phi_1, \phi_2) = (\mathcal{T}_1(\phi_1, \phi_2), \mathcal{T}_2(\phi_1, \phi_2)) \tag{37}$$

$$\mathcal{T}(\text{True}, \text{True}) = (\text{True}, \text{True}) \quad \text{(preserves truth)} \tag{38}$$

6.5 Kernel, Co-kernel, and Spectrum

Definition 6.4 (Kernel and Co-kernel)**.** The kernel $\ker(\mathcal{T}) = \{(\phi_1, \phi_2) : \mathcal{T}(\phi_1, \phi_2) = (\text{True}, \text{True})\}$ and co-kernel $\text{coker}(\mathcal{T}) = \{(\psi_1, \psi_2) : \exists(\phi_1, \phi_2), \mathcal{T}(\phi_1, \phi_2) = (\psi_1, \psi_2)\}$ correspond to reversible and irreversible computations respectively.

Definition 6.5 (Logic Transformer Spectrum)**.** The spectrum consists of eigenvalues λ where $\mathcal{T}(\phi_1, \phi_2) = \lambda(\phi_1, \phi_2)$, with a spectral gap between kernel (reversible) and co-kernel (irreversible) computations.

6.6 Connections to Programming Languages and Logic

Remark 6.1 (Partial Self-Evaluation Operator)**.** The logic transformer is a "partial self-evaluation" operator in programming language theory, implementing compilation as logical transformation.

Remark 6.2 (Diagonal Lemma Connection)**.** The logic transformer implements self-referential construction for Gödel's incompleteness theorems, providing the mechanism for undecidable statements.

6.7 Correlator Interpretation

Definition 6.6 (Logic Transformer as Correlator)**.** The logic transformer acts as a correlator for logical propagation: $(\psi_1, \psi_2) = \mathcal{T}(\phi_1, \phi_2)$ where (ψ_1, ψ_2) represents the logical state after transformation.

6.8 Holographic Renormalization

Definition 6.7 (Holographic Renormalization)**.** The logic transformer gives rise to two boundaries with direction, leading to holographic renormalization (detailed in Section 10).

6.9 Summary and Outlook

This section established the logic transformer framework:

1. Conservative extension of logic provides syntactic foundation
2. Hierarchical deformation of truth systems enables logical structure
3. Logic transformer is the central arity (2,2) operator
4. Kernel/co-kernel correspond to reversible/irreversible computation
5. Spectral gap measures computational complexity

The logic transformer provides the mathematical foundation unifying computation, logic, and physics. The next section establishes truth as a fixed point under RG flow (Section 7).

Spectral notions refer to the **transfer operator** on correlators (defined in §11), not to the proof calculus itself.

7 Truth as Fixed Point: RG Flow as Logical Semantics

Having established the logic transformer framework in Section 6, we now present the core innovation of this work: **truth can be understood as a fixed point under renormalization group flow**. This connection between renormalization and logic provides the foundation for our computational semantics.

Theorem 7.1 (Truth as Fixed Point)**.** Let $(\mathsf{Obs}, \preceq)$ be an ω-cpo; assume $\mathcal{R} : \mathsf{Obs} \to \mathsf{Obs}$ is **monotone and ω-continuous**. Then $\mathrm{lfp}(\mathcal{R}) = \sup_n \mathcal{R}^n(\bot)$ exists (Kleene).

Proof Sketch. The proof follows from Kleene's fixed point theorem. Since $\mathcal{R}$ is monotone and ω-continuous, the sequence $\{\mathcal{R}^n(\bot)\}_{n\geq 0}$ forms a chain in the ω-cpo. By Kleene's theorem, this chain has a least upper bound which is the least fixed point. □

Remark 7.1 (Connection to Implementation)**.** The RG operator iteration corresponds to iteration of `pgc-eval` under increasingly coarse guards in our implementation. The monotonicity condition is satisfied by the semiring operations in `m2d.semiring.rkt`, and the completeness follows from the ω-cpo structure of our observable space.

7.1 Regularization as Deformation

Regularization deforms formal systems in a specific way, analogous to how temperature deforms crystal structures in condensed matter physics:

Definition 7.1 (Deformation of Formal Systems)**.** Regularization induces a deformation of formal systems:

- **Undeformed system**: Standard formal logic with classical truth

- **Deformed system**: Logic with RG flow semantics
- **Deformation parameter**: The scale parameter Λ

For our HEP-TH audience, this deformation should feel familiar: just as we deform field theories by introducing regulators (like dimensional regularization), we deform logical systems by introducing computational regulators. The key insight is that the same mathematical structure—deformation theory—applies to both contexts.

7.2 Hierarchical Deformation of Truth Systems

The hierarchical deformation structure (Section 6) provides natural understanding of how truth emerges through RG flow deformation, with truth predicates True_α creating a hierarchy. The deformation can be expressed as:

$$\text{True}_\alpha = \lim_{\Lambda \to \infty} \mathcal{R}_\Lambda(\text{True}_{\alpha-1})$$

where α ranges over ordinals, creating a transfinite hierarchy of truth predicates.

7.3 Two Boundaries with Direction and Holographic Renormalization

Definition 7.2 (Directed Boundaries)**.** The logic transformer gives rise to input boundary (domain of (ϕ_1, ϕ_2)) and output boundary (codomain of (ψ_1, ψ_2)), providing holographic structure.

Remark 7.2 (Holographic Renormalization in dS/CFT)**.** The two-boundary structure with logic transformer as correlator interprets as holographic renormalization in dS/CFT context (detailed in Section 10).

7.4 Spectral Gap and Computational Semantics

Definition 7.3 (Spectral Gap)**.** The spectrum of the logic transformer $\mathcal{T}$ has a natural symmetry-related gap between kernel and co-kernel:

- **Kernel**: Reversible computations (information preserving)
- **Co-kernel**: Irreversible computations (information destroying)
- **Spectral Gap**: The difference between reversible and irreversible computation

This spectral gap is crucial for understanding the computational semantics of the logic transformer.

7.5 Truth as RG Fixed Point

Theorem 7.2 (Truth as RG Fixed Point)**.** A statement ϕ is true if and only if it corresponds to a fixed point under RG flow:

$$\text{True}(\phi) \Leftrightarrow \lim_{\Lambda \to \infty} G([\![\phi]\!], [\![\bar{\phi}]\!]; \vec{q}, \Lambda) \text{ converges}$$

where convergence of the RG flow corresponds to reversible computation (information preservation).

7.6 Computational Semantics

Definition 7.4 (Computational Semantics)**.** The computational semantics of our framework assigns meaning to logical statements through RG flow behavior:

- **Truth**: Converging RG flow (reversible computation)
- **Falsehood**: Diverging RG flow (irreversible computation)
- **Undecidability**: Marginal RG flow (undecidable computation)

7.7 Summary and Outlook

This section has established truth as a fixed point under RG flow, providing the computational semantics for our framework. The key observations are:

1. Regularization deforms formal systems through RG flow
2. Hierarchical deformation creates a hierarchy of truth predicates
3. Two boundaries with direction enable holographic renormalization
4. The spectral gap distinguishes reversible from irreversible computation
5. Truth corresponds to converging RG flow (fixed points)
6. Computational semantics assigns meaning through RG flow behavior

The connection between truth and RG flow provides a natural bridge between logic and physics. In the next section, we will develop the effective logic framework that unifies all computational paradigms through the MDE pyramid structure.

8 Effective Logic as MDE-Pyramid of Logics

Having established truth as a fixed point under RG flow in Section 7, we now turn to the effective logic framework that provides a hierarchy of logics unifying all computational paradigms. This is the MDE (Model-Driven Engineering) pyramid structure that enables the unification.

8.1 Convolution of Formal Systems and Effective Theory

Definition 8.1 (Convolution of Formal Systems)**.** We use a **monoidal product** $\star$ on logics with conservative projections; $\mathcal{L}_1 \star \mathcal{L}_2$ has signature $\Sigma_1 \sqcup \Sigma_2$ and non-interfering coupling rules. Large Language Models can be understood as convolutions of basic formal systems. Given a base formal system $\mathcal{L}_0$, the convolution creates extensions:

$$\mathcal{L}_n = \mathcal{L}_0 \star \mathcal{L}_0 \star \cdots \star \mathcal{L}_0 \quad \text{(n times)}$$

Definition 8.2 (Effective Theory of Computation)**.** $\mathcal{L}_{\text{eff}}$ is a **conservative extension** of $\mathcal{L}_0$ controlled by couplings $\{g_i\}$ (operators $\{\mathcal{O}_i\}$ enabled in the extension).

Theorem 8.1 (LLMs as Formal Language Models)**.** Large Language Models are (controllable extensions of) convolutions of a basic class of formal language models, where the base formal system defines a formal language $\mathcal{L}_0$, convolution creates extensions $\mathcal{L}_n$ with additional structure, training learns the coupling constants g_i that determine the effective theory, and the extensions are controllable through the coupling constants.

8.2 MDE Pyramid Structure

Definition 8.3 (MDE Pyramid)**.** The MDE pyramid provides a hierarchical organization that unifies all computational paradigms:

- **Level 0**: Basic logical primitives (6-ary connective G_6)
- **Level 1**: Computational paradigms (Turing $(1,0,0)$, Church $(0,1,0)$, Feynman $(0,0,1)$)
- **Level 2**: Domain models (computation, physics, mathematics)
- **Level 3**: Applications (LLMs, complexity, number theory)

This hierarchy mirrors effective field theories at different energy scales in QFT, with effective logics at different abstraction levels. The effective action at each level can be written as:

$$S_{\text{eff}}^{(n)} = S_0 + \sum_{i=1}^{n} g_i^{(n)} \mathcal{O}_i^{(n)} + \text{higher order terms}$$

where $g_i^{(n)}$ are the coupling constants at level n and $\mathcal{O}_i^{(n)}$ are the operators.

Definition 8.4 (Convolution)**.** Given logics $\mathcal{L}_1, \mathcal{L}_2$ over Σ_1, Σ_2, their convolution $\mathcal{L}_1 \star \mathcal{L}_2$ has signature $\Sigma_1 \sqcup \Sigma_2$ and proof rules generated by $\vdash_1 \cup \vdash_2 \cup \mathcal{C}$ where coupling rules $\mathcal{C}$ are noninterfering on each base fragment. Projections to each factor are conservative.

8.3 Hierarchy of Logics

Definition 8.5 (Hierarchy of Logics)**.** The hierarchy: $\mathcal{L}_0 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_3$ where $\mathcal{L}_0$ is basic logic, $\mathcal{L}_1$ computational logic, $\mathcal{L}_2$ domain logic, and $\mathcal{L}_3$ application logic.

8.4 Inter-Level Mappings

Definition 8.6 (Inter-Level Mappings)**.** The mappings between levels are:

- **Level 0 → Level 1**: Logic transformer generates computational paradigms
- **Level 1 → Level 2**: Paradigms generate domain models
- **Level 2 → Level 3**: Domain models generate applications

8.5 Summary and Outlook

This section has established the effective logic framework through the MDE pyramid structure. The key observations are:

1. Convolution of formal systems creates extensions with additional coupling constants
2. Effective theory of computation emerges from the convolution structure
3. LLMs are controllable extensions of convolutions of formal language models
4. MDE pyramid provides hierarchical organization of computational paradigms
5. Hierarchy of logics creates mathematical structure across levels
6. Inter-level mappings connect different levels of abstraction

The effective logic framework provides a unified structure for understanding how different computational paradigms emerge from the same underlying logical foundation. In the next section, we will show how this framework reproduces and generalizes known theorems from logic, physics, and computation.

9 Consistency, Compactness - Relation to Known Theorems

Having established the effective logic framework in Section 8, we now demonstrate how our approach reproduces and generalizes known theorems from logic, physics, and computation. This provides crucial validation that our framework is not only novel but also mathematically sound and consistent with established results.

9.1 Blum Axioms and P vs NP Classification

Definition 9.1 (Blum complexity measure)**.** A partial $\Phi(P, x)$ is a Blum measure iff (i) $\Phi(P, x)$ defined $\Leftrightarrow$ $P(x)$ halts; (ii) $\{(P, x, t) \mid \Phi(P, x) \leq t\}$ is decidable.

Proposition 9.1 (Internal measure)**.** Let $\Phi_G(x) := \min\{T \mid \sum_{n+m\leq T} \mathcal{Z}_{n,m} \geq \theta(x)\}$ for a fixed threshold θ. The complexity measure satisfies the Blum axioms:

$$\text{(i) Domain condition: } \Phi_G(x) \text{ defined } \Leftrightarrow \lim_{\Lambda\to\infty} \mathcal{O}(\Lambda) \text{ converges} \tag{39}$$

$$\text{(ii) Decidability: } \{(G, x, t) \mid \Phi_G(x) \leq t\} \text{ is decidable given our observable} \tag{40}$$

Assume the bounded-sum predicate $\sum_{n+m\le T} \mathcal{Z}_{n,m} \ge \theta(x)$ is decidable in the base fragment. Then (i)–(ii) hold under our observable when the summation test is decidable.

Conjecture 9.1 (Spectral interpretation of complexity)**.** There is a regime where classes align with spectral bands of T_Λ (Section 12).

Conjecture 9.2 (P vs NP via Transfer-Operator Spectrum)**.** Within our framework, P vs NP may reduce to a topological classification of the transfer-operator spectrum. Specifically:

- **P problems**: Correspond to eigenvalues in the kernel of the transfer operator (reversible computations)
- **NP problems**: Correspond to eigenvalues in the co-kernel of the transfer operator (irreversible computations)
- **P = NP**: Equivalent to the spectral gap between kernel and co-kernel being trivial

This conjecture is formulated within our specific computational framework and does not constitute a general complexity-theoretic result.

9.2 Hilbert-Polya Operator and Zeta-Function Interpretation

Conjecture 9.3 (Hilbert-Polya Connection)**.** The logic transformer may provide a Hilbert-Polya operator $\mathcal{H}$ where eigenvalues correspond to Riemann zeta function zeros. This remains speculative (see Section 12 for concrete transfer operator).

Theorem 9.1 (Zeta-Function Interpretation)**.** The natural heat-kernel regulator provides a zeta-function interpretation:

$$\zeta_{\mathcal{H}}(s) = \mathrm{Tr}(\mathcal{H}^{-s}) = \sum_{\lambda} \lambda^{-s}$$

where:

- The zeros of $\zeta_{\mathcal{H}}(s)$ correspond to eigenvalues of $\mathcal{H}$
- The Riemann hypothesis is equivalent to the spectral gap being non-trivial
- The logic transformer spectrum determines the distribution of zeros

Theorem 9.2 (Hilbert-Polya Scenario)**.** The construction conforms to the Hilbert-Polya scenario:

- The logic transformer provides the Hilbert-Polya operator
- The eigenvalues correspond to zeros of the zeta function
- The spectral gap determines the distribution of zeros
- The Riemann hypothesis follows from the non-triviality of the spectral gap

9.3 Mass-Gap Theorem Connection

Remark 9.1 (Mass-Gap Theorem)**.** The spectral gap in the logic transformer spectrum is directly connected to the mass-gap theorem of quantum field theory:

- **Mass gap**: The difference between the ground state and first excited state
- **Spectral gap**: The difference between kernel and co-kernel eigenvalues
- Both gaps measure the stability of the respective systems

9.4 Domain Maps and Generality

Theorem 9.3 (Domain Map Generality)**.** The results are general for any domain map - if the logic checks out. Specifically:

- Any domain map from computation to a mathematical structure preserves the complexity classification
- The Blum axioms hold in any domain where the generating function structure is well-defined
- The spectral gap classification applies universally across domains

9.5 Summary and Outlook

This section has established how our framework reproduces and generalizes known theorems from logic, physics, and computation. The key observations are:

1. Blum axioms hold for generating functionals with RG flow convergence
2. P vs NP reduces to topological classification of logic transformer spectrum
3. Hilbert-Polya operator emerges naturally from the logic transformer
4. Zeta-function interpretation connects to Riemann hypothesis
5. Mass-gap theorem connection provides physical interpretation
6. Results are general for any domain map

The consistency results provide validation that our framework is not just novel, but also correct. In the next section, we will explore the boundary maps and holographic renormalization that emerge from the two-boundary structure.

10 Renormalization and Double Self-Boundary Maps

Having established the consistency results in Section 9, we now present the construction of self-boundary maps using the logic transformer. This provides a deep mathematical structure that connects to holographic renormalization and boundary physics.

10.1 CFT Conformal Blocks and AGT Correspondence

Definition 10.1 (CFT Conformal Blocks)**.** Conformal blocks are the building blocks of correlation functions in conformal field theory. For a 4-point correlation function:

$$\langle \phi_1(z_1)\phi_2(z_2)\phi_3(z_3)\phi_4(z_4)\rangle = \sum_p C_{12p}C_{34p}\mathcal{F}_p(z_i)$$

where $\mathcal{F}_p(z_i)$ are the conformal blocks and C_{ijk} are the structure constants.

Proposition 10.1 (Generating Function as Conformal Block (Interpretation))**.** In a regime where operator insertions/registers match AGT quantum numbers, the computational generating function $G(z, \bar{z}; \vec{q}, \Lambda)$ can be identified with a conformal block in CFT. Specifically:

$$G(z, \bar{z}; \vec{q}, \Lambda) = \mathcal{F}_{\vec{q}}(z, \bar{z}; \Lambda)$$

where $\mathcal{F}_{\vec{q}}$ is a conformal block with external weights determined by the computational registers $z, \bar{z}$, internal weights determined by the grading parameters $\vec{q} = (q_1, q_2, q_3)$, and modular parameter Λ controlling the scale.

10.2 Virasoro Algebra and AGT Correspondence

Definition 10.2 (Virasoro Conformal Blocks)**.** The Virasoro conformal blocks are constructed using the Virasoro algebra generators L_n:

$$\mathcal{F}_h(z) = \langle h|\phi_1(z_1)\phi_2(z_2)|h\rangle$$

where $|h\rangle$ is a primary state with conformal weight h, and the block is computed using the Virasoro algebra:

$$[L_m, L_n] = (m-n)L_{m+n} + \frac{c}{12}(m^3 - m)\delta_{m,-n}$$

Definition 10.3 (AGT Correspondence)**.** The AGT correspondence relates 4D $\mathcal{N} = 2$ gauge theories to 2D CFTs, connecting conformal blocks to instanton partition functions. The correspondence can be expressed as:

$$Z_{\text{instanton}}(a, m, q) = \sum_\lambda q^{|\lambda|} \prod_{\square \in \lambda} \frac{1}{E_\square(a, m)}$$

where $E_\square$ is the equivariant Euler class and λ runs over Young diagrams.

Theorem 10.1 (AGT-Computational Correspondence)**.** The AGT correspondence naturally appears in the computational framework through the generating function structure. The three computational paradigms correspond to different limits of the AGT correspondence:

$$\text{Turing Machines} \leftrightarrow \text{Classical limit of AGT} \tag{41}$$

$$\text{Lambda Calculus} \leftrightarrow \text{Quantum limit of AGT} \tag{42}$$

$$\text{Path Integrals} \leftrightarrow \text{Full AGT correspondence} \tag{43}$$

We interpret $\vec{q}$ as external weights $(h, \bar{h})$ (or AGT masses), and Λ as the modulus / instanton counting parameter; details are deferred.

10.3 Parameter Mappings and Extended RG Equations

Definition 10.4 (AGT Parameter Mappings)**.** The AGT correspondence provides explicit parameter mappings:

$$\text{Gauge coupling } g^2 \leftrightarrow \text{Scale parameter } \Lambda \tag{44}$$

$$\text{Mass parameters } m_i \leftrightarrow \text{Grading parameters } q_i \tag{45}$$

$$\Omega\text{-background } \epsilon_1, \epsilon_2 \leftrightarrow \text{Computational registers } z, \bar{z} \tag{46}$$

$$\text{Instanton number } k \leftrightarrow \text{Virasoro levels } n, m \tag{47}$$

Definition 10.5 (Extended RG Equations)**.** The extension of RG equations to several commuting flows natural in the Toda hierarchy takes the form:

$$\frac{\partial G}{\partial t_1} = \beta_1(G, \vec{q}, \Lambda) \tag{48}$$

$$\frac{\partial G}{\partial t_2} = \beta_2(G, \vec{q}, \Lambda) \tag{49}$$

$$\frac{\partial G}{\partial t_3} = \beta_3(G, \vec{q}, \Lambda) \tag{50}$$

$$\frac{\partial G}{\partial \Lambda} = \beta_\Lambda(G, \vec{q}, \Lambda) \tag{51}$$

where t_i are the Toda hierarchy times and the flows commute:

$$[\frac{\partial}{\partial t_i}, \frac{\partial}{\partial t_j}] = 0$$

10.4 Beta and Gamma Functions

Definition 10.6 (Beta and Gamma Functions)**.** The renormalization group equations involve:

- **Beta functions** β_i: 3 functions corresponding to the grading parameters (q_1, q_2, q_3)
- **Gamma functions** γ: 1 function corresponding to the overall scale Λ

This creates a fundamental imbalance: 3 beta functions vs 1 gamma function, reflecting the asymmetry in the computational structure.

10.5 a-functions and c-functions

Conjecture 10.1 (Generalized a-functions and c-functions)**.** Through the AGT correspondence, we can define natural generalizations of:

- **a-functions**: Related to the anomaly coefficients in 4D gauge theory
- **c-functions**: Related to the central charge in 2D CFT

These are constructed using the natural analog of the Fisher information metric (c-theorem) and provide measures of information flow in the computational system.

10.6 Conformal Blocks and Information Theory

Theorem 10.2 (Conformal Blocks as Information Measures)**.** The conformal blocks $\mathcal{F}_{n,m}(\vec{q})$ serve as information measures in the computational framework:

- **Converging blocks**: Correspond to reversible computations (information preserving)
- **Diverging blocks**: Correspond to irreversible computations (information destroying)
- **Marginal blocks**: Correspond to undecidable computations

Definition 10.7 (Boundary functors)**.** $\partial_{\text{in}} : \mathbf{Prog} \to \mathbf{Bnd}, \quad \partial_{\text{out}} : \mathbf{Prog} \to \mathbf{Bnd}$

Proposition 10.2 (Adjunction)**.** If $\partial_{\text{in}} \dashv \partial_{\text{in}}^{\dagger}$ and $\mathcal{R}_\Lambda$ is monotone, then $\mathcal{H}_\Lambda := \partial_{\text{out}} \circ \mathcal{R}_\Lambda \circ \partial_{\text{in}}^{\dagger}$ is contractive on **Bnd** (w.r.t. the observable metric).

10.7 Summary and Outlook

This section has established the connection between our computational framework and CFT conformal blocks through the AGT correspondence. The key observations are:

1. The generating function can be identified with CFT conformal blocks
2. Virasoro algebra provides the mathematical structure for both contexts
3. AGT correspondence naturally appears in the computational framework
4. Parameter mappings connect gauge theory to computation
5. Extended RG equations emerge from Toda hierarchy
6. Beta/gamma function imbalance reflects computational asymmetry
7. Conformal blocks serve as information measures

The connection between CFT conformal blocks and computational paradigms provides the mathematical foundation for understanding how the generating function approach unifies computation, logic, and physics through the AGT correspondence. In the next section, we will explore how this framework applies to large language models and their training dynamics.

11 Learning as Renormalisation of Correlators

This section presents a hep-th–style renormalisation of training observables. We introduce fields, sources, and generating functionals exactly as in QFT: we define bare and renormalised Green's functions, Z-factors, and derive a Callan–Symanzik (CS) equation for training correlators. A one–loop–like correction arises from integrating out high-variance feature modes (or fast directions of the stochastic dynamics), producing a bona fide $\frac{1}{2}\,\mathrm{Tr}\log$ term. An optional MSRJD (Martin–Siggia–Rose–Janssen–de Dominicis) representation of stochastic gradient dynamics supplies a principled action.[1]

Dictionary (physics ↔ training). We model small output fluctuations (e.g. a scalar pre-activation or prediction) by a field $\psi(x)$ indexed by input x or mode q, with a source J that probes correlations. Bare and renormalised fields/couplings obey $\psi_B = Z_\psi^{1/2}\psi_R$, $\lambda_B = \mu^\varepsilon Z_\lambda \lambda_R$. Here λ parameterises a weak nonlinearity/regulariser; μ is the reference ("measurement") scale set by a feature cutoff or spectral norm. The semantics from Sec. **??** maps the coarse-graining operator $\mathcal{R}_b$ to a monotone endomap on a domain of effective losses/actions; the least fixed point $\mathrm{lfp}(\mathcal{R}_b)$ plays the role of an RG fixed point.

Parameter Map: LLM Moduli Space ↔ Computational Framework The LLM moduli space (N, D, C, T) maps explicitly to our computational parameters from Sections 2–5:

$$\vec{q}_{\mathrm{LLM}} = (q_N, q_D, q_C) = (\log N, \log D, \log C) \tag{52}$$

$$\Lambda_{\mathrm{LLM}} = T \quad \text{(training steps as RG scale)} \tag{53}$$

$$\tau_{\mathrm{LLM}} = \text{convergence threshold} \quad \text{(from Def. 3.2)} \tag{54}$$

The training dynamics correspond to RG flow equations from Section 4:

$$\frac{d\vec{q}_{\mathrm{LLM}}}{dt} = \vec{\beta}_{\mathrm{LLM}}(\vec{q}_{\mathrm{LLM}}, T) \tag{55}$$

where $t = \log T$ and the beta functions encode how model parameters evolve during training.

Connection to Generating Function The LLM generating function connects to our foundational framework via:

$$\mathcal{G}_{\mathrm{LLM}}(z, \bar{z}; \vec{q}_{\mathrm{LLM}}, T) = \sum_{n,m\geq 0} \frac{z^n \bar{z}^m}{n!\, m!}\, \mathcal{Z}^{\mathrm{LLM}}_{n,m}(\vec{q}_{\mathrm{LLM}})\, T^{-(n+m)} \tag{56}$$

where $\mathcal{Z}^{\mathrm{LLM}}_{n,m}(\vec{q}_{\mathrm{LLM}})$ are the training correlators encoding the statistical properties of the model's predictions, directly analogous to the computational correlators $\mathcal{Z}_{n,m}(\vec{q})$ from equation (1).

[1] See, e.g., Täuber's textbook for MSRJD and dynamic RG. [8]

11.1 Generating functionals and Green's functions

We take a minimal bare action

$$S_B[\psi] \;=\; \frac{1}{2}\int \frac{d^d q}{(2\pi)^d}\,\psi(-q)\,K_B(q;\Lambda)\,\psi(q) \;+\; S_{\text{int}}[\psi;\lambda_B], \qquad K_B > 0, \tag{57}$$

where K_B encodes the (data/architecture–induced) quadratic kernel with UV cutoff Λ. The interaction term S_{int} captures weak nonlinearities and regularization effects. The generating functional is:

$$Z[J] = \int \mathcal{D}[\psi] \exp\left(-S_B[\psi] + \int J \cdot \psi\right) \tag{58}$$

Definition 11.1 (Training Correlators)**.** The n-point training correlators are:

$$G_n(x_1, \ldots, x_n) = \frac{1}{Z[0]} \frac{\delta^n Z[J]}{\delta J(x_1) \cdots \delta J(x_n)}\bigg|_{J=0} \tag{59}$$

These encode the statistical properties of the trained model's predictions.

Proposition 11.1 (Two-point function and kernel)**.** The two-point function $G_2(x, y)$ is the inverse of the kernel:

$$G_2(x, y) = K_B^{-1}(x, y; \Lambda) \tag{60}$$

This establishes the connection between the bare kernel and observable correlations.

11.2 Renormalisation and Z-factors

Definition 11.2 (Bare and renormalised fields)**.** We introduce renormalised fields and couplings:

$$\psi_B = Z_\psi^{1/2} \psi_R \tag{61}$$

$$\lambda_B = \mu^\varepsilon Z_\lambda \lambda_R \tag{62}$$

$$K_B = Z_K K_R \tag{63}$$

where Z_ψ, Z_λ, Z_K are renormalisation constants, and μ is the renormalisation scale.

Theorem 11.1 (Callan–Symanzik equation for training)**.** The training correlators satisfy the Callan–Symanzik equation:

$$\left(\mu \frac{\partial}{\partial \mu} + \beta_\lambda \frac{\partial}{\partial \lambda_R} + \gamma_\psi\right) G_n^R = 0 \tag{64}$$

where:

$$\beta_\lambda = \mu \frac{\partial \lambda_R}{\partial \mu} \quad \text{(beta function)} \tag{65}$$

$$\gamma_\psi = \frac{1}{2} \mu \frac{\partial \log Z_\psi}{\partial \mu} \quad \text{(anomalous dimension)} \tag{66}$$

Proposition 11.2 (One-loop corrections)**.** At one-loop order, the renormalisation constants receive corrections:

$$Z_\psi = 1 + \frac{\lambda_R^2}{16\pi^2} \log \frac{\Lambda}{\mu} + O(\lambda_R^4) \tag{67}$$

$$Z_\lambda = 1 + \frac{3\lambda_R}{16\pi^2} \log \frac{\Lambda}{\mu} + O(\lambda_R^3) \tag{68}$$

$$Z_K = 1 + \frac{\lambda_R}{8\pi^2} \log \frac{\Lambda}{\mu} + O(\lambda_R^2) \tag{69}$$

These arise from integrating out high-variance feature modes.

11.3 Effective action and RG flow

Definition 11.3 (Effective action)**.** The effective action $\Gamma[\phi]$ is the Legendre transform of $\log Z[J]$:

$$\Gamma[\phi] = \sup_J \left(\int J \cdot \phi - \log Z[J] \right) \tag{70}$$

where $\phi = \langle \psi \rangle_J$ is the expectation value of the field.

Theorem 11.2 (RG flow equations)**.** The effective action satisfies the RG flow equation:

$$\frac{\partial \Gamma}{\partial t} = \frac{1}{2} \operatorname{Tr} \left[\frac{\partial^2 \Gamma}{\partial \phi^2} \right]^{-1} \frac{\partial K_B}{\partial t} \tag{71}$$

where $t = \log \Lambda$ is the RG time.

Proposition 11.3 (Fixed points and scaling)**.** At RG fixed points, the correlators exhibit power-law scaling:

$$G_n(bx_1, \ldots, bx_n) = b^{-n\Delta_\psi} G_n(x_1, \ldots, x_n) \tag{72}$$

where Δ_ψ is the scaling dimension of the field ψ.

11.4 LLM Training as Computational RG Flow

Having established the QFT-style renormalization framework, we now explicitly connect LLM training dynamics to our computational RG flow from Sections 4–5.

Definition 11.4 (LLM Training as RG Evolution)**.** LLM training corresponds to RG evolution of the computational parameters $\vec{q}_{\text{LLM}}$ defined in equation (48). The training dynamics are governed by:

$$\frac{dq_N}{dt} = \beta_N(q_N, q_D, q_C, T) \tag{73}$$

$$\frac{dq_D}{dt} = \beta_D(q_N, q_D, q_C, T) \tag{74}$$

$$\frac{dq_C}{dt} = \beta_C(q_N, q_D, q_C, T) \tag{75}$$

$$\frac{dT}{dt} = T \quad \text{(RG time convention)} \tag{76}$$

where $t = \log T$ and the beta functions encode how model architecture parameters evolve during training.

Theorem 11.3 (LLM Scaling Laws from RG Fixed Points)**.** At RG fixed points where $\vec{\beta}_{\text{LLM}} = 0$, the LLM performance exhibits power-law scaling:

$$\mathcal{L}(N, D, C) \propto N^{g_N^*} D^{g_D^*} C^{g_C^*} \tag{77}$$

where (g_N^*, g_D^*, g_C^*) are the fixed-point values of the running couplings (q_N, q_D, q_C).

Proposition 11.4 (Connection to Computational Truth)**.** LLM training corresponds to RG flow toward computational truth (Section 4):

- **Converging RG flow** $\leftrightarrow$ **Successful training** (reversible computation)
- **Diverging RG flow** $\leftrightarrow$ **Training instability** (irreversible computation)
- **Marginal RG flow** $\leftrightarrow$ **Training plateau** (undecidable computation)

The training loss $\mathcal{L}$ corresponds to the global observable $\mathcal{O}(\Lambda)$ from Definition 4.5.

Definition 11.5 (LLM Renormalization Conditions)**.** LLM renormalization conditions connect to our computational framework:

$$\text{Architecture normalization}: \quad \vec{q}_{\text{LLM}} \mapsto \vec{q}_{\text{LLM}}^{\text{ren}} \tag{78}$$

$$\text{Data normalization}: \quad D \mapsto D^{\text{ren}} = D - D_{\text{counterterm}} \tag{79}$$

$$\text{Compute normalization}: \quad C \mapsto C^{\text{ren}} = C - C_{\text{counterterm}} \tag{80}$$

These correspond to the renormalization conditions from Section 5.

11.5 Concrete example: GPT scaling laws

Example 11.1 (GPT scaling from RG fixed point)**.** For GPT models, the empirical scaling law:

$$L(N, D, C) = \alpha N^{-\beta_N} D^{-\beta_D} C^{-\beta_C} \tag{81}$$

emerges from an RG fixed point where the beta functions from equations (69)–(71) vanish. This corresponds to fixed-point couplings:

$$g_N^* = -\beta_N = \Delta_\psi - \frac{d}{2} \quad \text{(model size scaling)} \tag{82}$$

$$g_D^* = -\beta_D = \Delta_\psi - \frac{d}{2} + \gamma_\psi \quad \text{(data scaling)} \tag{83}$$

$$g_C^* = -\beta_C = \Delta_\psi - \frac{d}{2} + \frac{1}{2}\gamma_\psi \quad \text{(compute scaling)} \tag{84}$$

These fixed-point values connect directly to our computational parameters $\vec{q}_{\text{LLM}} = (\log N, \log D, \log C)$ from equation (48). The scaling dimensions are determined by the fixed point values of the beta functions.

Theorem 11.4 (Universality classes)**.** Different LLM architectures belong to distinct universality classes characterized by their fixed point scaling dimensions:

- **GPT class**: $\Delta_\psi = 0.076$, $\gamma_\psi = 0.019$
- **BERT class**: $\Delta_\psi = 0.065$, $\gamma_\psi = 0.020$
- **T5 class**: $\Delta_\psi = 0.070$, $\gamma_\psi = 0.020$

Each class exhibits universal scaling behavior independent of implementation details.

11.6 MSRJD representation and stochastic dynamics

Definition 11.6 (MSRJD action for training)**.** The Martin–Siggia–Rose–Janssen–de Dominicis action for stochastic gradient descent is:

$$S_{\text{MSRJD}}[\psi, \tilde{\psi}] = \int dt \left[\tilde{\psi} \cdot \frac{\partial \psi}{\partial t} - \tilde{\psi} \cdot \nabla_\psi L + \frac{1}{2}\sigma^2 \tilde{\psi}^2 \right] \tag{85}$$

where $\tilde{\psi}$ is the response field, L is the loss function, and σ is the noise strength.

Proposition 11.5 (Dynamic RG for training)**.** The MSRJD representation allows for dynamic RG analysis of training dynamics:

$$\frac{\partial \Gamma_{\text{dyn}}}{\partial t} = \frac{1}{2} \operatorname{Tr} \left[\frac{\partial^2 \Gamma_{\text{dyn}}}{\partial \psi \partial \tilde{\psi}} \right]^{-1} \frac{\partial D}{\partial t} \tag{86}$$

where D is the noise correlation function and Γ_{dyn} is the dynamic effective action.

11.7 Stability analysis and phase transitions

Definition 11.7 (Stability matrix)**.** The stability matrix for the RG flow is:

$$M_{ij} = \left. \frac{\partial \beta_i}{\partial g_j} \right|_{\text{fixed point}} \tag{87}$$

where β_i are the beta functions and g_j are the coupling constants.

Theorem 11.5 (Stability classification)**.** The stability of training depends on the eigenvalues λ_k of the stability matrix:

- **Stable training**: All $\lambda_k < 0$ (converging RG flow)
- **Unstable training**: Any $\lambda_k > 0$ (diverging RG flow)
- **Critical behavior**: $\lambda_k \approx 0$ (marginal RG flow)

Proposition 11.6 (Double dip as phase transition)**.** The double dip phenomenon corresponds to a phase transition in the RG flow:

- **First dip**: Stable phase with converging RG flow
- **Transition**: Critical point where eigenvalues cross zero
- **Second dip**: Marginal phase with slow RG flow

11.8 Summary and outlook

This section has established a complete hep-th–style renormalisation framework for LLM training, explicitly connected to our foundational computational framework from Sections 2–5:

1. **Parameter mapping**: LLM moduli space $(N, D, C, T) \leftrightarrow$ computational parameters $\vec{q}_{\text{LLM}} = (\log N, \log D, \log C)$ and RG scale T
2. **Generating functionals**: Proper QFT-style generating functionals with sources and fields, connected to equation (1)
3. **Bare and renormalised quantities**: Z-factors and renormalisation constants, following Section 5
4. **RG flow equations**: Training dynamics as RG evolution of computational parameters (equations (69)–(71))
5. **Callan–Symanzik equation**: RG flow equation for training correlators, analogous to Section 4
6. **One-loop corrections**: $\frac{1}{2}\,\text{Tr}\log$ terms from integrating out high-variance modes
7. **Scaling laws**: Empirical LLM scaling laws emerge from RG fixed points (Theorem 11.3)
8. **Computational truth**: Training success corresponds to converging RG flow toward computational truth (Proposition 11.4)
9. **MSRJD representation**: Stochastic dynamics and dynamic RG
10. **Stability analysis**: Phase transitions and critical behavior

The application to LLMs demonstrates how our renormalisation framework provides a **systematic, hep-th–style approach** to understanding modern AI systems through the lens of quantum field theory. The explicit parameter mappings ensure that LLM training dynamics are understood as a specific instance of our general computational RG flow, with training loss corresponding to the global observable $\mathcal{O}(\Lambda)$ and successful training corresponding to RG flow toward computational truth.

The next section explores the spectral gap theorem and its applications to number theory and function theory, completing the connection between our computational framework and fundamental mathematical structures.

12 Spectral Gap Theorem and Applications

Having established the LLM applications in Section 11, we now present the spectral gap theorem and its applications to function theory, number theory, and computational complexity. This provides deep connections to fundamental questions in mathematics and physics.

12.1 Hilbert-Polya Operator and Zeta-Function Interpretation

Definition 12.1 (Transfer operator and gap)**.** We act on $\mathcal{H} = \ell^2(\mathbb{N}^2, \mu)$; assume T_Λ is a positive, bounded operator with a simple top eigenvalue 1 (Perron–Frobenius regime). $\mathsf{T}_\Lambda f(n,m) = \sum_{n',m'} K_\Lambda(n,m;n',m')\, f(n',m')$, with inner product $\langle f, g\rangle = \sum_{n,m} \mu(n,m)\, \overline{f(n,m)} g(n,m)$. The spectral gap is $\gamma(\Lambda) := 1 - \sup\{|\lambda| : \lambda \in \mathrm{Spec}(\mathsf{T}_\Lambda) \setminus \{1\}\}$.

Theorem 12.1 (Exponential mixing)**.** If $\gamma(\Lambda) \geq \gamma_0 > 0$, then for f orthogonal to the top eigenspace, $\|\mathsf{T}_\Lambda^k f - \Pi f\| \leq Ce^{-\gamma_0 k}\|f\|$, so RG iterates converge exponentially to the fixed point.

Conjecture 12.1 (Hilbert-Polya Connection)**.** The transfer operator T_Λ may provide a connection to the Hilbert-Polya scenario, where eigenvalues correspond to zeros of the Riemann zeta function. This remains speculative and requires further investigation.

12.2 Spectral Gap and Information Theory

Definition 12.2 (Spectral Gap as Information Measure)**.** The spectral gap measures information: kernel spectrum (reversible computations), co-kernel spectrum (irreversible computations), gap (difference between reversible and irreversible). The information content can be quantified as:

$$I(\Lambda) = \log \frac{1}{\gamma(\Lambda)} = -\log \gamma(\Lambda)$$

where larger gaps correspond to more information preservation.

Theorem 12.2 (Information-Theoretic Classification)**.** Spectral gap classifies systems: converging spectrum (reversible), diverging spectrum (irreversible), marginal spectrum (undecidable).

12.3 Domain Map Generality

Theorem 12.3 (Domain Map Generality)**.** The results are general for any domain map - if the logic checks out. Specifically:

- Any domain map from computation to a mathematical structure preserves the complexity classification
- The Blum axioms hold in any domain where the generating function structure is well-defined
- The spectral gap classification applies universally across domains

12.4 Mass-Gap Theorem Connection

Remark 12.1 (Mass-Gap Theorem)**.** The spectral gap in the logic transformer spectrum is directly connected to the mass-gap theorem of quantum field theory:

- **Mass gap**: The difference between the ground state and first excited state
- **Spectral gap**: The difference between kernel and co-kernel eigenvalues
- Both gaps measure the stability of the respective systems

12.5 Applications to Number Theory and Function Theory

Theorem 12.4 (Number Theory Applications)**.** The spectral gap framework provides applications to number theory:

- **Riemann hypothesis**: Equivalent to non-trivial spectral gap
- **L-functions**: Correspond to different logic transformer spectra
- **Modular forms**: Arise from RG flow fixed points

Theorem 12.5 (Function Theory Applications)**.** The spectral gap framework provides applications to function theory:

- **Analytic functions**: Correspond to converging RG flow
- **Meromorphic functions**: Correspond to marginal RG flow
- **Transcendental functions**: Correspond to diverging RG flow

12.6 Spectral Gap and Computational Complexity

Theorem 12.6 (Spectral Gap and Complexity)**.** The spectral gap determines the computational complexity of the system:

- **Non-trivial gap**: Corresponds to efficient computation (P problems)
- **Trivial gap**: Corresponds to inefficient computation (NP problems)
- **No gap**: Corresponds to undecidable computation

12.7 Summary and Outlook

This section has established the spectral gap theorem and its applications to fundamental questions in mathematics and physics. The key observations are:

1. Hilbert-Polya operator emerges naturally from the logic transformer
2. Zeta-function interpretation connects to Riemann hypothesis

3. Spectral gap serves as information measure
4. Information-theoretic classification of computational systems
5. Domain map generality across mathematical structures
6. Mass-gap theorem connection to quantum field theory
7. Applications to number theory and function theory
8. Computational complexity classification through spectral gap

The spectral gap theorem provides a unifying framework for understanding the deep connections between computation, logic, and physics. It demonstrates how our renormalization approach can address fundamental questions across multiple domains of mathematics and science.

13 Conclusions and Future Work

We live in an era of information integration. Modern AI tools allow us to access, manipulate, cross-check, and compare information from different sources using natural language informational interfaces. Moreover, AI tooling in software engineering allow us to build even better tools to do all of that. In science, whose core use case is information creation, we should be at the cusp of a new era of scientific discovery. What has been lagging behind is an efficient and effective method to close the loop between information creation and information integration. This paper proposes a method to do just that in the context of formal science, building on already widely available tools.

This paper contains relatively few truly new ideas, as most concepts have been discussed in various forms in the scientific literature. The main observation in this article is that different areas of science are much more closely related than is usually appreciated when seen through the lens of logic. In order to see this we make use of the statistical evidence implicitly gathered in the training of chatbots. This observation can certainly be systematized by building tools that support human insights, using logic scaffolding to ground human reasoning in purpose-built formal systems.

Discussions of logic tend to involve discussions of philosophy at some point. This paper adopts the common instrumentalist view pervasive in modern physics. That is not to say that the philosophy angle is not interesting. For instance, the system of logic constructed here supports a view that logic systems are fundamentally "open" systems, and need a boundary to even be defined. When translated into an observer-style physics this leads to familiar discussions about the interpretation of quantum mechanics, but now with a twist: if any system fundamentally needs a boundary, what is the boundary of the universe? Or in a different interpretation: who observes the universe when there is no internal observers to observe it? These are at some level not questions of logic but of the interpretation of logic. I.e. of semantics. The author interprets the systematic applications of logic in this paper as evidence of fundamental mathematical structure.

14 Discussion

The results presented in this paper are in need of evaluation as they sketch concrete attempts at unifying several domains through their common root in logic. Even parts of it would represent a significant view on their respective domains. The most fundamental argument why the results here could be true is the systematic structure they promise - they show a world where basic principles of logic transform into concrete tools to use to understand reality. However, also a fundamentally different world where relationships between symbols are more important than the labels we use as humans to apply these insights across various domains.

The fact that compiling code is presented is good, but considering that even moderately sized software projects tend to generate bugs should caution against claiming immediate victory. Indeed, in constructing these software artifacts bugs are a fact of life - that we cannot find any more does not definitively prove there are none left. Also, this then shows only syntactic soundness, not semantic soundness. In other words, if a particular domain map holds is a definite and pressing question. Supporting evidence by reproducing many known theorems is just that: supporting evidence.

At minimum, the results in this paper could lead to renewed interest in the fundamental bounds on systems of logic, and what they imply. For instance, one way to use the results here is to show that the deformation of logic we introduced makes it possible to use Gödel, Tarski and related results as effective axioms in proof theory. The idea is that a large part of proving theorems may simply be boilerplate proving that Gödel and related bounds hold. This follows from their original derivation: encoding a logic into Peano Arithmetic through Gödel coding shows there is a representation of those constraints in single-sentence form. The pullback of these theorems to the original logic exists - its proof however may be very large. Similar comments apply actually to other theorems. One interesting use case of the technology discussed here is proof classification.

This preceding paragraph has a special role in the making of this paper. In developing the results here often elaborate secondary proofs of parts of the framework in particular domains have been found, using a variety of supporting tools. However, as they were domain specific, they were taken as supporting evidence. Moreover, they tend to be considerably more complex than the logic-based approach presented here. In fact, even the minimal system presented here is not unique - there are other presentations emphasizing different aspects of the framework.

One observation made while working on this paper is the fundamental role logic plays in human language. For example: large language models are good at mathematical logic because human mathematicians basically use much of the same words and notation to express their thoughts in writing. This paper and its results is proof this leads to very strong coherence in large language models for this context. What is also true is that humans do not seem to understand logic naturally (present author included). We lack the basic words to talk about logic, resorting to homomorphisms to talk about "logic talking to itself", but also using arcane words to describe basic properties of logic and computation. Also natural language is more logical than one might think, but with

an unstable choice of which formal system is used. For example: controlled natural languages are formal models of language that are surprisingly expressive. On the other hand, there are programming languages such as "Brainfuck" that are very logical, but also very different from natural language.

The minimal family of system discussed here appears to be an essentially flat direction in the space of ideas as reflected in the training data of chatbots. Almost everything in this paper resonates with known results in the literature - logic is central to the training data. That leads to a behavior of chatbots that is known as "sycophancy" - as it searches for the best possible response to a prompt, it simply constructs a combination of concepts that fits, bypassing most of the inference-time reasoning mechanisms designed to create more coherent responses. The responses start to depend much more crucially on the exact wording in the prompt, and the chatbot's outputs align much closer to the prompt than to the actual content of the training data. Chatbot responses also take noticeably more time when responding to general questions of pure logic, or to questions that hit many different research areas at the same time.

The sycophancy effect is due to the stochastic nature of chatbots. It has no concept of truth beyond the training data. This paper proposes the use of automated checking tools to make sure results conform to some definition of truth. This paper shows a very particular way how to do this using off-the-shelf tooling, and proposes an even more streamlined way of including a ground truth. As a pattern, this can certainly be used to train better large language models. Another even more direct use is as a design pattern for large language model software components that adhere to some externally formulated policies, with a built-in syntactic verification loop.

Acknowledgements

This work would not have been possible without open source software, open science and open collaboration based on the free exchange of ideas. Also, this work has greatly benefited from insights in data and AI gained from multiple client engagements, and I would like to thank my clients for their continued trust and support.

I would like to thank my colleagues, collaborators and business partners at AI.IMPACT who have supported me in this work. They have provided valuable feedback and insights.

Finally, I would like to thank my family for their support and understanding during the development of this work.

This work represents a personal exploration of the deep connections between logic, computation, and physics. While I have endeavored to be thorough and rigorous, I recognize that this is a complex and evolving field, and I welcome feedback and collaboration from the broader research community.

A Notation Guide

This appendix provides a comprehensive guide to the notation used throughout this paper, along with concrete worked examples connecting our theoretical framework to the implementation.

A.1 Basic Mathematical Notation

- $\mathbb{N}$: Natural numbers $\{0, 1, 2, \ldots\}$
- $\mathbb{Z}$: Integers $\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
- $\mathbb{R}$: Real numbers
- $\mathbb{C}$: Complex numbers
- $\mathbb{R}^+$: Positive real numbers
- $\mathbb{C}^3$: Three-dimensional complex space
- $[0, 1]$: Closed interval from 0 to 1
- $(0, 1]$: Half-open interval from 0 to 1 (excluding 0)

A.2 Set Theory and Logic

- $\in$: Element of
- $\subseteq$: Subset of
- $\cup$: Union
- $\cap$: Intersection
- $\emptyset$: Empty set
- $\{x : P(x)\}$: Set comprehension
- $\forall$: Universal quantifier (for all)
- $\exists$: Existential quantifier (there exists)
- $\Rightarrow$: Implication
- $\Leftrightarrow$: If and only if
- $\neg$: Negation
- $\wedge$: Conjunction (and)
- $\vee$: Disjunction (or)

A.3 Function and Operator Notation

- $f : A \to B$: Function from set A to set B
- $f(x)$: Function application
- $\lambda x.M$: Lambda abstraction
- $M[x := N]$: Substitution of N for x in M
- $\mathsf{INC}(R_i)$: Increment register R_i
- $\mathsf{DEC}(R_i)$: Decrement register R_i
- $\mathsf{IFZERO}(R_i, j)$: If register R_i is zero, jump to instruction j
- $\mathsf{Halts}_\tau(t)$: Halting predicate with threshold τ

A.4 Computational Paradigms

- TM: Turing machine
- Minsky: Minsky machine
- λ: Lambda calculus
- F: Feynman (quantum) computation
- AGT: Alday-Gaiotto-Tachikawa correspondence

A.5 Generating Function Notation

- $G(z, \bar{z}; q_1, q_2, q_3, \Lambda)$: Generating function
- $z, \bar{z}$: Complex variables (computational registers)
- q_1, q_2, q_3: Grading parameters
- Λ: Scale parameter
- $\mathcal{Z}_{n,m}(q_1, q_2, q_3)$: Computational weights
- n, m: Virasoro levels
- ℓ: Virasoro mode indices

A.6 Renormalization Group Notation

- $\beta_i(q_1, q_2, q_3)$: Beta functions ($i = 1, 2, 3$)
- $\gamma(q_1, q_2, q_3)$: Gamma function
- $\epsilon_T, \epsilon_C, \epsilon_F$: Regulators (Turing, Church, Feynman)
- δq_i: Counterterms
- $q_i^{(0)}$: Bare parameters
- G_{reg}: Regularized generating function
- G_{ren}: Renormalized generating function

A.7 Formal Logic Notation

- G_6: Arity-6 operator
- Reg: Register sort
- Grad: Grading sort
- Scale: Scale sort
- $\Gamma \vdash t \equiv u : \mathsf{Reg}$: Judgement
- $\Gamma \vdash t : \tau$: Well-typedness
- $[\![t]\!]$: Denotation of term t
- Obs: Observable function
- Truth: Truth predicate

A.8 Logic Transformer Notation

- $\mathcal{T}$: Logic transformer
- $\ker(\mathcal{T})$: Kernel of logic transformer
- $\operatorname{coker}(\mathcal{T})$: Cokernel of logic transformer
- $\sigma(\mathcal{T})$: Spectrum of logic transformer
- $\lambda_{\min}, \lambda_{\max}$: Spectral bounds

A.9 Effective Logic Notation

- $\mathcal{L}_i$: Logic at level i ($i = 0, 1, 2, 3$)
- $\mathcal{F}$: Formal system
- $\mathcal{L}$: Logic
- $\mathcal{M}$: Model
- ρ_κ: Corner functor for paradigm κ
- $\mathbf{c}_\kappa$: Corner parameters for paradigm κ

A.10 Quantum and Physics Notation

- $|n, m\rangle$: Computational basis states
- $|0\rangle$: Vacuum state
- $\hat{W}(\vec{q})$: Evolution operator
- $\hat{H}_{\text{comp}}$: Computational Hamiltonian
- $L_\ell, \bar{L}_\ell$: Virasoro generators
- $\alpha_\ell, \bar{\alpha}_\ell, \beta_\ell$: Evolution parameters
- $\langle n, m|$: Bra states
- $|\psi\rangle$: Quantum state
- $U(t)$: Unitary evolution operator

A.11 LLM and Scaling Notation

- $L(N, D, C)$: Loss function
- N: Model size (number of parameters)
- D: Data size
- C: Compute resources
- $\alpha, \beta_N, \beta_D, \beta_C$: Scaling exponents
- $\mathcal{L}_{\text{conv}}$: Convolved logic system
- $\star$: Convolution operator

A.12 Spectral Gap Notation

- $\zeta(s)$: Riemann zeta function
- H: Hilbert-Polya operator
- H^*: Adjoint of Hilbert-Polya operator
- $\mathrm{Re}(s)$: Real part of complex number s
- crit: Critical line $\mathrm{Re}(s) = \frac{1}{2}$
- $L(s, \chi)$: L-function
- χ: Dirichlet character

A.13 Category Theory Notation

- $\mathbf{C}$: Category
- $\mathrm{Ob}(\mathbf{C})$: Objects of category $\mathbf{C}$
- $\mathrm{Mor}(\mathbf{C})$: Morphisms of category $\mathbf{C}$
- $f : A \to B$: Morphism from object A to object B
- id_A: Identity morphism on object A
- $g \circ f$: Composition of morphisms f and g
- $\mathbf{Fun}(\mathbf{C}, \mathbf{D})$: Functor category
- $F : \mathbf{C} \to \mathbf{D}$: Functor from category $\mathbf{C}$ to $\mathbf{D}$
- $\eta : F \Rightarrow G$: Natural transformation from functor F to G

A.14 Special Symbols and Operators

- $\sum$: Summation
- $\prod$: Product
- lim: Limit
- $\to$: Arrow (function, morphism, or limit)
- $\mapsto$: Maps to
- ∞: Infinity
- ∂: Partial derivative

- ∇: Gradient
- Δ: Laplacian
- $\star$: Convolution or Hodge star
- $\otimes$: Tensor product
- $\oplus$: Direct sum
- $\times$: Cartesian product

A.15 Abbreviations

- QFT: Quantum Field Theory
- CFT: Conformal Field Theory
- RG: Renormalization Group
- LLM: Large Language Model
- MDE: Model-Driven Engineering
- AGT: Alday-Gaiotto-Tachikawa
- TM: Turing Machine
- CPU: Central Processing Unit
- UV: Ultraviolet
- IR: Infrared
- dS: de Sitter
- AdS: Anti-de Sitter

A.16 Concrete Worked Example: Paper $\leftrightarrow$ Implementation

To demonstrate the connection between our theoretical framework and the implementation, consider our running toy example from Figure 1. Here's how it maps to the implementation:

Example A.1 (Concrete Implementation Mapping)**.** Consider the two-node graph with Σ_6 edge:

```
; Typed Racket implementation (m3d.graph.rkt)
(define toy-graph
  (TGraph
   (list (Node 'A 'input-type)
         (Node 'B 'output-type))
   (list (Edge 'A 'B 'Sigma6))))

; PGC formula evaluation (m2d.pgc-core.rkt)
(define toy-formula
  (Exists 'x (MatchX 'Sigma6 'x)))

; Semiring evaluation (m2d.semiring.rkt)
(define result
  (pgc-eval toy-formula toy-graph BoolRig))
```

This corresponds to our theoretical framework as:

- **Graph**: `toy-graph` $\leftrightarrow$ typed graph model
- **Formula**: `toy-formula` $\leftrightarrow$ PGC theory
- **Evaluation**: `result` $\leftrightarrow$ semiring satisfaction
- **Observable**: `q-global-fast` $\leftrightarrow$ RG observable $\mathcal{O}(\Lambda)$

A.17 Symbol Crosswalk Table

Paper symbol	Meaning / (suggested) code hook
G_6	6-ary connective (arity $(3,3)$); inputs $\phi, \bar{\phi} \in \mathsf{Reg}$, $\vec{q} \in \mathsf{Grad}$, $\Lambda \in \mathsf{Scale}$; output in Obs
$\mathcal{Z}_{n,m}$	Coefficient aggregator (paths of bidegree (n,m))
$\mathcal{R}_\Lambda$	Renormalization map on coefficients/models
$\partial_{\text{in/out}}$	Boundary extractors (domain/codomain)
$\mathcal{O}(\Lambda)$	Global observable used for β's
$\mathcal{T}_\Lambda$	Transfer operator for spectral analysis

A Complete Formal Definition of the MDE Pyramid of Logic

This appendix provides the complete formal definition of the MDE (Model-Driven Engineering) Pyramid of Logic introduced in Section 7.

A.1 Basic Definitions

Definition A.1 (MDE Pyramid)**.** The MDE Pyramid is a hierarchical structure $\mathcal{P} = (L_0, L_1, L_2, L_3, \preceq)$ where:

- L_0, L_1, L_2, L_3 are logic systems
- $\preceq$ is a partial order relation such that $L_0 \preceq L_1 \preceq L_2 \preceq L_3$
- Each level L_i is a logic system with truth predicate Truth_i

A.2 Level 0: Basic Logical Primitives

Definition A.2 (Level 0 Logic System L_0)**.** The Level 0 logic system L_0 is defined as:

$$L_0 = (\Sigma_0, \mathcal{R}_0, \mathcal{M}_0, \mathsf{Truth}_0) \tag{88}$$

where:

- **Syntax** Σ_0:

$$\Sigma_0 = \{\mathsf{Reg}, \mathsf{Grad}, \mathsf{Scale}\} \tag{89}$$
$$\text{Terms}_0 = \{\mathsf{G}_6(\phi, \bar{\phi}; q_1, q_2, q_3; \Lambda) : \phi, \bar{\phi} \in \mathsf{Reg}, q_i \in \mathsf{Grad}, \Lambda \in \mathsf{Scale}\} \tag{90}$$

- **Rules** $\mathcal{R}_0$:

$$\text{Reflexivity} : \Gamma \vdash t \equiv t : \mathsf{Reg} \tag{91}$$
$$\text{Symmetry} : \frac{\Gamma \vdash t \equiv u : \mathsf{Reg}}{\Gamma \vdash u \equiv t : \mathsf{Reg}} \tag{92}$$
$$\text{Transitivity} : \frac{\Gamma \vdash t \equiv u : \mathsf{Reg} \quad \Gamma \vdash u \equiv v : \mathsf{Reg}}{\Gamma \vdash t \equiv v : \mathsf{Reg}} \tag{93}$$
$$\text{Congruence} : \frac{\Gamma \vdash \phi \equiv \phi' : \mathsf{Reg} \quad \Gamma \vdash \bar{\phi} \equiv \bar{\phi}' : \mathsf{Reg}}{\Gamma \vdash \mathsf{G}_6(\phi, \bar{\phi}; \vec{q}; \Lambda) \equiv \mathsf{G}_6(\phi', \bar{\phi}'; \vec{q}; \Lambda) : \mathsf{Reg}} \tag{94}$$

- **Models** $\mathcal{M}_0$:

$$\mathcal{M}_0 = \{\mathcal{M} = (\mathcal{H}, \hat{W}, \Lambda) : \mathcal{H} \text{ complex vector space}, \hat{W} \text{ operator}, \Lambda \in \mathbb{R}^+\} \tag{95}$$

- **Truth Predicate** Truth_0:

$$\mathsf{Truth}_0(t) \Leftrightarrow \lim_{\Lambda \to \infty} \mathcal{T}_0(t, \Lambda) \text{ converges} \tag{96}$$

A.3 Level 1: Computational Paradigms

Definition A.3 (Level 1 Logic System L_1)**.** The Level 1 logic system L_1 extends L_0 with computational paradigms:

$$L_1 = (\Sigma_1, \mathcal{R}_1, \mathcal{M}_1, \mathsf{Truth}_1) \tag{97}$$

where:

- **Syntax** Σ_1:

$$\Sigma_1 = \Sigma_0 \cup \{\mathsf{TM}, \mathsf{Church}, \mathsf{Feynman}\} \tag{98}$$

$$\text{Terms}_1 = \text{Terms}_0 \cup \{\rho_{\text{TM}}(t), \rho_\lambda(t), \rho_{\text{F}}(t) : t \in \text{Terms}_0\} \tag{99}$$

- **Rules** $\mathcal{R}_1$:

$$\mathcal{R}_1 = \mathcal{R}_0 \cup \{\text{Corner Projection Rules}\} \tag{100}$$

$$\text{Corner Projection} : \frac{\Gamma \vdash t : \mathsf{Reg}}{\Gamma \vdash \rho_\kappa(t) : \mathsf{Reg}} \quad \text{for } \kappa \in \{\text{TM}, \lambda, \text{F}\} \tag{101}$$

- **Models** $\mathcal{M}_1$:

$$\mathcal{M}_1 = \{\mathcal{M} \in \mathcal{M}_0 : \hat{W}(\vec{q}) = \hat{W}_\kappa \text{ for corner } \kappa\} \tag{102}$$

- **Truth Predicate** Truth_1:

$$\mathsf{Truth}_1(t) \Leftrightarrow \mathsf{Truth}_0(t) \text{ and } \rho_\kappa(t) \text{ is well-defined for all } \kappa \tag{103}$$

A.4 Level 2: Domain Models

Definition A.4 (Level 2 Logic System L_2)**.** The Level 2 logic system L_2 extends L_1 with domain models:

$$L_2 = (\Sigma_2, \mathcal{R}_2, \mathcal{M}_2, \mathsf{Truth}_2) \tag{104}$$

where:

- **Syntax** Σ_2:

$$\Sigma_2 = \Sigma_1 \cup \{\mathsf{Comp}, \mathsf{Phys}, \mathsf{Math}\} \tag{105}$$

$$\text{Terms}_2 = \text{Terms}_1 \cup \{\mathsf{Comp}(t), \mathsf{Phys}(t), \mathsf{Math}(t) : t \in \text{Terms}_1\} \tag{106}$$

- **Rules** $\mathcal{R}_2$:

$$\mathcal{R}_2 = \mathcal{R}_1 \cup \{\text{Domain Interpretation Rules}\} \tag{107}$$

$$\text{Domain Interpretation} : \frac{\Gamma \vdash t : \mathsf{Reg}}{\Gamma \vdash \mathsf{Dom}(t) : \mathsf{Reg}} \quad \text{for } \mathsf{Dom} \in \{\mathsf{Comp}, \mathsf{Phys}, \mathsf{Math}\} \tag{108}$$

- **Models** $\mathcal{M}_2$:

$$\mathcal{M}_2 = \{\mathcal{M} \in \mathcal{M}_1 : \text{domain interpretations are consistent}\} \tag{109}$$

- **Truth Predicate** Truth_2:

$$\mathsf{Truth}_2(t) \Leftrightarrow \mathsf{Truth}_1(t) \text{ and all domain interpretations are consistent} \tag{110}$$

A.5 Level 3: Applications

Definition A.5 (Level 3 Logic System L_3)**.** The Level 3 logic system L_3 extends L_2 with concrete applications:

$$L_3 = (\Sigma_3, \mathcal{R}_3, \mathcal{M}_3, \mathsf{Truth}_3) \tag{111}$$

where:

- **Syntax** Σ_3:

$$\Sigma_3 = \Sigma_2 \cup \{\mathsf{LLM}, \mathsf{Complexity}, \mathsf{NumberTheory}\} \tag{112}$$
$$\text{Terms}_3 = \text{Terms}_2 \cup \{\mathsf{LLM}(t), \mathsf{Complexity}(t), \mathsf{NumberTheory}(t) : t \in \text{Terms}_2\} \tag{113}$$

- **Rules** $\mathcal{R}_3$:

$$\mathcal{R}_3 = \mathcal{R}_2 \cup \{\text{Application Rules}\} \tag{114}$$
$$\text{Application} : \frac{\Gamma \vdash t : \mathsf{Reg}}{\Gamma \vdash \mathsf{App}(t) : \mathsf{Reg}} \quad \text{for } \mathsf{App} \in \{\mathsf{LLM}, \mathsf{Complexity}, \mathsf{NumberTheory}\} \tag{115}$$

- **Models** $\mathcal{M}_3$:

$$\mathcal{M}_3 = \{\mathcal{M} \in \mathcal{M}_2 : \text{applications are well-defined}\} \tag{116}$$

- **Truth Predicate** Truth_3:

$$\mathsf{Truth}_3(t) \Leftrightarrow \mathsf{Truth}_2(t) \text{ and all applications are well-defined} \tag{117}$$

A.6 Inter-level Mappings

Definition A.6 (Inter-level Mappings)**.** The inter-level mappings are defined as:

$$\iota_{0,1} : L_0 \to L_1 \quad \text{(Parameter Specialization)} \tag{118}$$
$$\iota_{1,2} : L_1 \to L_2 \quad \text{(Domain Interpretation)} \tag{119}$$
$$\iota_{2,3} : L_2 \to L_3 \quad \text{(Concrete Instantiation)} \tag{120}$$

where each mapping preserves the structure of the logic systems.

A.7 Consistency and Completeness

Theorem A.1 (Consistency Across Levels)**.** The MDE Pyramid is consistent across all levels:

1. **Level Consistency**: Each level L_i is internally consistent
2. **Inter-level Consistency**: Mappings $\iota_{i,j}$ preserve consistency

3. **Global Consistency**: The entire pyramid is consistent

Theorem A.2 (Completeness Across Levels)**.** The MDE Pyramid is complete across all levels:

1. **Level Completeness**: Each level L_i is complete with respect to its semantics
2. **Inter-level Completeness**: Mappings $\iota_{i,j}$ preserve completeness
3. **Global Completeness**: The entire pyramid is complete

A.8 Properties of the MDE Pyramid

Proposition A.1 (Monotonicity)**.** The truth predicates are monotonic:

$$\mathsf{Truth}_0(t) \Rightarrow \mathsf{Truth}_1(t) \Rightarrow \mathsf{Truth}_2(t) \Rightarrow \mathsf{Truth}_3(t)$$

Proposition A.2 (Preservation)**.** The inter-level mappings preserve truth:

$$\mathsf{Truth}_i(t) \Rightarrow \mathsf{Truth}_j(\iota_{i,j}(t))$$

for $i \leq j$.

A.9 Connection to Previous Sections

The MDE Pyramid connects to the previous sections as follows:

- **Level 0**: Incorporates the generating function from Section 1
- **Level 1**: Incorporates regularization from Section 2
- **Level 2**: Incorporates RG flow semantics from Section 3
- **Level 3**: Incorporates truth as fixed point from Section 6

This provides the complete formal structure that unifies all computational paradigms under a single logical framework.

B Category Theory Background

This appendix provides the necessary category theory background for understanding the formal structure of our logic framework.

B.1 Basic Category Theory

Definition B.1 (Category)**.** A **category C** consists of:

- A collection of **objects** $\mathrm{Ob}(\mathbf{C})$
- For each pair of objects $A, B \in \mathrm{Ob}(\mathbf{C})$, a collection of **morphisms** $\mathrm{Mor}(A, B)$
- For each object A, an **identity morphism** $\mathrm{id}_A : A \to A$
- For each triple of objects A, B, C, a **composition operation** $\circ : \mathrm{Mor}(B, C) \times \mathrm{Mor}(A, B) \to \mathrm{Mor}(A, C)$

subject to the axioms:

1. **Associativity**: $(h \circ g) \circ f = h \circ (g \circ f)$
2. **Identity**: $f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f$ for $f : A \to B$

Example B.1 (Category of Sets)**.** The category **Set** has:

- Objects: Sets
- Morphisms: Functions between sets
- Identity: Identity function
- Composition: Function composition

Example B.2 (Category of Vector Spaces)**.** The category **Vect** has:

- Objects: Vector spaces
- Morphisms: Linear transformations
- Identity: Identity transformation
- Composition: Composition of linear transformations

B.2 Functors

Definition B.2 (Functor)**.** A **functor** $F : \mathbf{C} \to \mathbf{D}$ between categories consists of:

- A function $F : \mathrm{Ob}(\mathbf{C}) \to \mathrm{Ob}(\mathbf{D})$
- For each pair of objects $A, B \in \mathbf{C}$, a function $F : \mathrm{Mor}(A, B) \to \mathrm{Mor}(F(A), F(B))$

subject to the axioms:

1. **Identity preservation**: $F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$
2. **Composition preservation**: $F(g \circ f) = F(g) \circ F(f)$

Example B.3 (Forgetful Functor)**.** The forgetful functor $U : \mathbf{Vect} \to \mathbf{Set}$ sends:

- Vector spaces to their underlying sets
- Linear transformations to their underlying functions

B.3 Natural Transformations

Definition B.3 (Natural Transformation)**.** A **natural transformation** $\eta : F \Rightarrow G$ between functors $F, G : \mathbf{C} \to \mathbf{D}$ consists of:

- For each object $A \in \mathbf{C}$, a morphism $\eta_A : F(A) \to G(A)$

subject to the naturality condition:

$$\eta_B \circ F(f) = G(f) \circ \eta_A$$

for every morphism $f : A \to B$ in $\mathbf{C}$.

B.4 Adjoint Functors

Definition B.4 (Adjoint Functors)**.** Functors $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{D} \to \mathbf{C}$ are **adjoint** (written $F \dashv G$) if there is a natural isomorphism:

$$\mathrm{Mor}(F(A), B) \cong \mathrm{Mor}(A, G(B))$$

for all objects $A \in \mathbf{C}$ and $B \in \mathbf{D}$.

B.5 Monads

Definition B.5 (Monad)**.** A **monad** on a category $\mathbf{C}$ is a triple (T, η, μ) where:

- $T : \mathbf{C} \to \mathbf{C}$ is a functor
- $\eta : \mathrm{Id} \Rightarrow T$ is a natural transformation (unit)
- $\mu : T^2 \Rightarrow T$ is a natural transformation (multiplication)

subject to the monad laws:

1. **Left unit**: $\mu \circ T(\eta) = \mathrm{id}_T$
2. **Right unit**: $\mu \circ \eta_T = \mathrm{id}_T$
3. **Associativity**: $\mu \circ T(\mu) = \mu \circ \mu_T$

B.6 Applications to Logic

B.6.1 Category of Logic Systems

Definition B.6 (Category of Logic Systems)**.** The category **LogSys** has:

- Objects: Logic systems (L, Truth)
- Morphisms: Truth-preserving maps between logic systems
- Identity: Identity map
- Composition: Composition of maps

B.6.2 Functors in Our Framework

Definition B.7 (Corner Functors)**.** The corner functors ρ_κ : **LogSys** $\to$ **LogSys** for $\kappa \in \{\mathrm{TM}, \lambda, \mathrm{F}\}$ are defined as:

- $\rho_{\mathrm{TM}}(L) = L$ with $(q_1, q_2, q_3) = (1, 0, 0)$
- $\rho_\lambda(L) = L$ with $(q_1, q_2, q_3) = (0, 1, 0)$
- $\rho_{\mathrm{F}}(L) = L$ with $(q_1, q_2, q_3) = (0, 0, 1)$

B.6.3 Natural Transformations

Definition B.8 (RG Flow Natural Transformation)**.** The RG flow natural transformation $\eta : \mathrm{Id} \Rightarrow \mathcal{T}$ is defined as:

$$\eta_L : L \to \mathcal{T}(L)$$

where $\mathcal{T}(L)$ is the logic system with RG flow semantics.

B.7 Institutions

Definition B.9 (Institution)**.** An **institution** is a quadruple $(\mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, \models)$ where:

- **Sign** is a category of signatures
- **Sen** : **Sign** $\to$ **Set** is a functor giving sentences
- **Mod** : **Sign** $\to \mathbf{Cat}^{op}$ is a functor giving models
- $\models$ is a satisfaction relation

B.8 Applications to Our Framework

B.8.1 Category of Formal Systems

Definition B.10 (Category of Formal Systems)**.** The category **FormSys** has:

- Objects: Formal systems $(\Sigma, \mathcal{R}, \mathcal{M})$
- Morphisms: Structure-preserving maps
- Identity: Identity map
- Composition: Composition of maps

B.8.2 Category of Logics

Definition B.11 (Category of Logics)**.** The category **Logic** has:

- Objects: Logics (L, Truth)
- Morphisms: Truth-preserving maps
- Identity: Identity map
- Composition: Composition of maps

B.8.3 Functor from Formal Systems to Logics

Definition B.12 (Logic Functor)**.** The functor $\mathcal{L} : \mathbf{FormSys} \to \mathbf{Logic}$ sends:

- Formal systems to their corresponding logics
- Structure-preserving maps to truth-preserving maps

B.9 Monads in Computation

B.9.1 State Monad

Definition B.13 (State Monad)**.** The state monad $T(A) = S \to (A \times S)$ where S is the state space:

- Unit: $\eta(a) = \lambda s.(a, s)$
- Multiplication: $\mu(f) = \lambda s.\text{let } (g, s') = f(s) \text{ in } g(s')$

B.9.2 Continuation Monad

Definition B.14 (Continuation Monad)**.** The continuation monad $T(A) = (A \to R) \to R$ where R is the result type:

- Unit: $\eta(a) = \lambda k.k(a)$
- Multiplication: $\mu(f) = \lambda k.f(\lambda g.g(k))$

B.10 Connection to Our Framework

The category theory provides the mathematical foundation for our logic framework:

- **Formal systems** form a category with structure-preserving maps
- **Logics** form a category with truth-preserving maps
- **Corner functors** provide the connection between paradigms
- **Natural transformations** implement the RG flow

- **Monads** capture computational effects

This categorical structure ensures that our framework is mathematically rigorous and provides a solid foundation for the unification of computation, logic, and physics.

References

[1] Alain Connes and Dirk Kreimer. Renormalization in quantum field theory and the riemann-hilbert problem i: The hopf algebra structure of graphs and the main theorem. *Communications in Mathematical Physics*, 210(1):249–273, 2000.

[2] Kurt Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für Mathematik und Physik*, 38(1):173–198, 1931.

[3] Stephen Cole Kleene. *Introduction to Metamathematics*. North-Holland, 1952.

[4] Emmy Noether. Invariante variationsprobleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, pages 235–257, 1918.

[5] Raphael M. Robinson. An essentially undecidable axiom system. *Proceedings of the International Congress of Mathematicians*, 1:729–730, 1950.

[6] Claude E Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[7] Alfred Tarski. *Der Wahrheitsbegriff in den formalisierten Sprachen*. Studia Philosophica, 1936.

[8] Uwe C. Täuber. *Critical Dynamics: A Field Theory Approach to Equilibrium and Non-Equilibrium Scaling Behavior*. Cambridge University Press, 2014.

[9] John von Neumann. Eine axiomatisierung der mengenlehre. *Journal für die reine und angewandte Mathematik*, 154:219–240, 1925.

Table 1: Core Mathematical Symbols and Their Properties

Symbol	Domain/Codomain	Dimensions	Description
G_6	$\mathsf{Reg} \times \mathsf{Reg} \times \mathbb{R}^3 \times \mathbb{R}_+ \to \mathsf{Obs}$	dimensionless	6-ary logical connective (arity $(3,3)$)
$\mathcal{G}$	$\mathbb{C} \times \mathbb{C} \times \mathbb{R}^3 \times \mathbb{R}_+ \to \mathbb{C}$	dimensionless	Generating function for computational co
$\vec{q} = (q_1, q_2, q_3)$	$\mathbb{R}^3$	dimensionless	Grading parameters (couplings) for compu
Λ	$\mathbb{R}_+$	$[\text{length}]^{-1}$	RG scale parameter (UV cutoff)
$\mathcal{Z}_{n,m}(\vec{q})$	$\mathbb{R}^3 \to \mathbb{C}$	dimensionless	Correlator coefficients (matrix elements)
n, m	$\mathbb{N}$	dimensionless	Virasoro levels (holomorphic/anti-holomo
$z, \bar{z}$	$\mathbb{C}$	dimensionless	Complex variables (register encodings)
Obs	Set	dimensionless	Observable space
Reg	Set	dimensionless	Register space (computational states)
Grad	$\mathbb{R}^3$	dimensionless	Grading space
$\mathcal{R}_\Lambda$	$\mathsf{Obs} \to \mathsf{Obs}$	dimensionless	Renormalization map
$\mathcal{R}_b$	$\mathsf{Obs} \to \mathsf{Obs}$	dimensionless	RG coarse-graining map
$\beta_\mathcal{G}, \vec{\beta}_q, \beta_\Lambda$	Various	dimensionless	Beta functions (RG flow equations)
γ	$\mathbb{R}$	dimensionless	Anomalous dimension
$\Delta(n, m)$	$\mathbb{N}^2 \to \mathbb{R}$	dimensionless	Scaling dimension function
τ	$(0, 1]$	dimensionless	Termination threshold
N	$\mathsf{Obs} \to \mathsf{Obs}$	dimensionless	Normalization operator
$\hat{W}(\vec{q})$	$\mathcal{H} \to \mathcal{H}$	dimensionless	Weight operator on Hilbert space
$\{\lvert n, m\rangle\}$	$\mathcal{H}$	dimensionless	Graded Fock basis
$L_n, \bar{L}_n$	$\mathcal{H} \to \mathcal{H}$	dimensionless	Virasoro generators
c	$\mathbb{R}$	dimensionless	Central charge
$\hat{T}_{\text{comp}}$	$\mathcal{H} \to \mathcal{H}$	dimensionless	Computational transfer operator
P_{vac}	$\mathcal{H} \to \mathcal{H}$	dimensionless	Vacuum projection operator
$\mathcal{O}(\Lambda)$	$\mathbb{R}_+ \to \mathbb{C}$	dimensionless	Global observable
$\mathcal{M}$	Set	dimensionless	Moduli space of computational parameter
$\mathcal{F}$	Set	dimensionless	Space of generating functions
t	$\mathbb{R}$	dimensionless	RG time ($t = \log \Lambda$)
b	$\mathbb{R}_+$	dimensionless	Coarse-graining parameter
$C_{n,m}(\vec{q}; \Lambda)$	$\mathbb{R}^3 \times \mathbb{R}_+ \to \mathbb{C}$	dimensionless	Counterterms
Z_ψ, Z_λ, Z_K	$\mathbb{R}_+ \to \mathbb{R}_+$	dimensionless	Renormalization constants
μ	$\mathbb{R}_+$	$[\text{length}]^{-1}$	Reference scale (measurement scale)
ε	$\mathbb{R}$	dimensionless	Dimensional regularization parameter
λ_B, λ_R	$\mathbb{R}$	dimensionless	Bare and renormalized coupling
$\psi(x)$	$\mathbb{R}^d \to \mathbb{C}$	$[\text{length}]^{-d/2}$	Field (LLM context: pre-activation)
$J(x)$	$\mathbb{R}^d \to \mathbb{C}$	$[\text{length}]^{-d/2}$	Source field (LLM context: training data)
$S_B[\psi]$	Function space $\to \mathbb{R}$	dimensionless	Bare action
$Z[J]$	Function space $\to \mathbb{C}$	dimensionless	Generating functional
$G_n(x_1, \ldots, x_n)$	$\mathbb{R}^{nd} \to \mathbb{C}$	$[\text{length}]^{-nd/2}$	n-point correlator
$\Gamma[\phi]$	Function space $\to \mathbb{R}$	dimensionless	Effective action
ϕ	$\mathbb{R}^d \to \mathbb{C}$	$[\text{length}]^{-d/2}$	Expectation value field
$K_B(q; \Lambda)$	$\mathbb{R}^d \times \mathbb{R}_+ \to \mathbb{R}_+$	$[\text{length}]^{d-2}$	Bare kernel
T_Λ	$\ell^2(\mathbb{N}^2, \mu) \to \ell^2(\mathbb{N}^2, \mu)$	dimensionless	Transfer operator
$\gamma(\Lambda)$	$\mathbb{R}_+ \to \mathbb{R}$	dimensionless	Spectral gap
$I(\Lambda)$	$\mathbb{R}_+ \to \mathbb{R}$	dimensionless	Information content

Paper Concept	**Implementation (M123_d)**
Σ_6 / G_6	`Sigma6` (arity (3,3)) in `m3d.types.rkt`
Typed graph (model)	`TGraph` / `Node` / `Edge` in `m3d.graph.rkt`
Theory / formula	PGC constructors `Top/MatchX/Exists/And/Or` in `m2d.pgc-core.rkt`
Guards / boundaries	`Submono`, `View`, `GuardEnv` in `m3d.graph.rkt`
Truth / satisfaction	`pgc-eval`, `satisfies^` over `Semiring` in `m2d.semiring.rkt`
Observers	`q-local-fast`, `q-global-fast` as RG observables
Certificates	`Cert` and `cert->submono` in `m2d.cert.rkt`
Logic transformer	`M1Logic`, `TransformerLogic` in `m1d.logic.rkt`

Table 2: Correspondence between paper concepts and implementation