TYPED REPAIR: EMNIST FROM λ-DEFINABLE EVALUATION AND SPECIALIZATION GATE (DRAFT)

Milan Rosko
January 2026

ORCID: 0009-0003-1363-7158

Abstract

We present a finitary, semantics-locked learning and evaluation pipeline for multiclass EMNIST digit classification in which training, evaluation, and optimization steps admit explicit, checkable invariants. A deterministic preprocessing lock fixes a partial function from bitmaps to feature facts and a stable tie-broken prediction rule. Training is performed by an integer-valued margin-repair procedure that refutes violated margin obligations on an explicit finite witness table and propagates each update through a postings index to maintain a cached score matrix. When no witnesses remain, the system reaches a certified fixed point: the cached scores agree with definitional evaluation and all finite-table margin constraints are satisfied. The resulting perceptron-style classifier is re-expressed in a restricted logical intermediate representation with definitional interpreter semantics. A compiler emits specialized residual evaluators for recognized fragments, and speedups are reported only after a decidable finite equivalence gate verifies agreement of full score vectors between the compiled evaluator and the reference interpreter on a locked test family. The artifact is fully reproducible and includes configuration and weight hashes binding the certificate to the experimental context.

Subject: Logic, Proof Theory, Type Theory, Machine Learning

1 INTRODUCTION

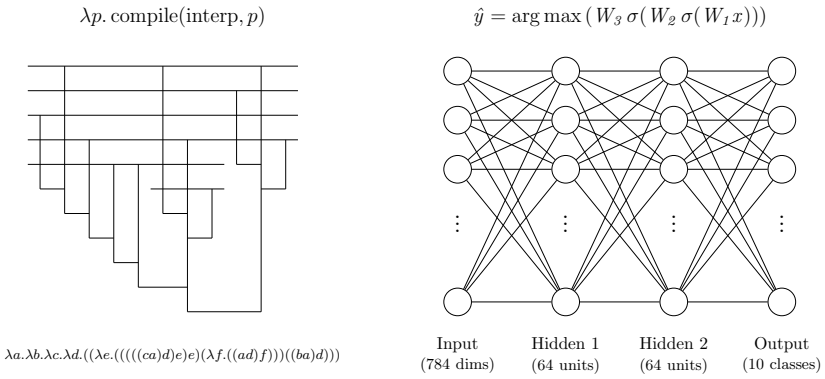

Figure 1. Illustration. *Left:* a combinatoric substrate, i.e. an SK-combinator Tromp Diagram, cf. Grygiel and Lescanne [2015]; in which finite programs denote partial computable functions under standard evaluation. *Right:* a topological substrate for neural evaluation, cf. Scott [1972], where each finite parameter tensor fixes an executable evaluator on a particular machine-and-runtime semantics.

Exposition. A persistent source of opacity in contemporary learning systems is not an exotic defect of particular architectures, but a familiar consequence of effectivity and program semantics. Once training yields a determinate

INPUT–OUTPUT behavior (relative to a fixed execution semantics), that behavior is (extensionally) representable as the denotation of a program; and many "interpretability" questions thereby become questions about semantic properties of program denotations.

2 PROJECT OVERVIEW

2.1 CONTRIBUTIONS

Exposition. This paper presents a finitary, semantics-locked learning and evaluation artifact for EMNIST digit classification, together with a decidable specialization (equivalence) gate for residual evaluators. The main contributions are:

(i) *Reproducible artifact.* The implementation produces stable configuration and weight hashes binding the certificate, evaluator behavior, and reported measurements to a concrete preprocessing lock, dataset identity, and parameter regime.

(ii) *Finitary learning and evaluation as a predicative artifact.* Working over a fixed semantics lock and an explicit finite witness table, we develop a learning-and-audit pipeline whose core predicates (well-formedness, cache correctness, margin satisfaction, and evaluator equivalence on a fixed gate) are total and decidable. Concretely:

   (a) *Semantics lock.* We fix a deterministic preprocessing and feature-extraction regime (canonicalization, binarization threshold, feature schema, and a stable tie-broken $\arg\max^*$) that induces determinate maps from EMNIST bitmaps to finite fact lists and from score vectors to predictions.

   (b) *Typed margin repair on a finite witness table.* We define an integer-valued repair procedure that enumerates violated one-vs-rest margin obligations on an explicit finite table and applies local updates determined by a stable best-competitor selector. Updates are propagated through postings indices to maintain a cached score matrix.

   (c) *Invariant preservation and certification.* We prove that each repair step preserves a well-formedness judgment consisting of postings correctness and cache correctness (agreement with definitional scores). This yields a decidable finite certificate predicate: cache correctness plus satisfaction of all margin obligations on the table.

   (d) *λ-definable evaluation.* For fixed bounds (finite table, fixed schema, and bounded vocabularies), the locked feature extraction, definitional scoring, repair functional, and certificate predicate are λ-definable under standard encodings of finite lists, bounded vectors, and integers.

   (e) *Verified specialization gate.* We re-express the learned classifier in a restricted logic-style intermediate representation with definitional interpreter semantics. A recognizer/compiler emits residual evaluators for recognized fragments, and speedups are reported only after a decidable finite equivalence gate verifies equality of

full score vectors between the compiled evaluator and the reference interpreter on a fixed, semantics-locked test family.

Definition 2.1 (Effectivity)**.** Throughout, the learned artifact is understood as the *implemented* evaluator on a physical machine. Parameters are stored as finite bitstrings (e.g. integers or floating-point words) and evaluation proceeds by the machine's effective transition dynamics. Hence the realized input–output behavior induces a partial computable function on the chosen concrete encodings.

Ansatz 2.2 (Realizability)**.** The reduction from learned artifacts to program semantics may be stated in the Brouwer–Heyting–Kolmogorov and Kleene tradition; cf. Troelstra and van Dalen [1988]. Any concrete learning system is realized by a physical machine; a machine implements effective procedures; effective procedures are representable in a formal calculus; and formal calculi admit mathematical interpretations. This yields the chain of partial encodings:

$$\begin{aligned}\text{Realization of Machine Learning} &\preceq_{\mathsf{enc}} \text{Symbolic Machine Code} \\ &\preceq_{\mathsf{enc}} \text{Neural Network Model} \preceq_{\mathsf{enc}} \text{Second-Order Arithmetic} \\ &\preceq_{\mathsf{enc}} \text{Computer Algebra} \preceq_{\mathsf{enc}} \text{Effective Logic.}\end{aligned} \tag{2.1}$$

However, second order arithmetic is not a computational necessity. The "machine" already realizes the effective content; the appeal to external mathematical semantics functions primarily as an explanatory bridge. In this sense—as an analogy rather than a literal sequent-calculus theorem—one may regard the external detour as a *cut* in the manner of Gentzen [1935]: it can be eliminated without loss of operational content, yielding a direct reduction to a universal syntactic model. Concretely, there exists a shorter route, e.g. in the sense of Solomonoff [1964a,b],

$$\begin{aligned}\text{Realization of Machine Learning} &\preceq_{\mathsf{enc}} \text{Symbolic Machine Code} \\ &\preceq_{\mathsf{enc}} \text{Combinatory Logic} \preceq_{\mathsf{enc}} \text{Effective Logic.}\end{aligned} \tag{2.2}$$

The claim is not that a trained network *is* a combinator term, but that there exists an *extensional extraction* into a universal formalism (e.g. λ-calculus or SK-combinators) such that, under standard encodings, the extracted term computes a partial function extensionally equal to the learned evaluator's realized input–output behavior.

This extraction claim is intentionally modest: it is an existence statement about representability in a universal calculus. Any stronger claim (e.g. that extraction is uniformly small, human-legible, or tractable) requires additional structural restrictions; the Notebook's restricted intermediate representation (IR) regime (Section 3.4) is where such operational extraction and audit become feasible.

Moreover, insofar as learning consists in escaping the maximally entropic baseline predictor by exploiting compressible regularities, it typically admits many extensionally equivalent realizations with sharply different computational profiles. This phenomenon is representation-theoretic rather than accidental: in our earlier work, as in Fibonacci-based codings and Φ-adic recursion, semantic operations become local and bounded by relocating structure into sparse additive supports [Rosko, 2025a,b], the same denotation may admit realizations whose verification, explanation, or execution costs differ by orders of magnitude. At the universal boundary this multiplicity is

not, in general, well-founded: for a fixed Blum complexity measure, there exist computable functions for which no asymptotically optimal extensional realization exists, in the sense of a Gödel–Blum Speedup defined bellow.

Lemma 2.3 (Gödel–Blum Speedup)**.** By Gödel [1986]; Blum [1967], for any Blum complexity measure $\mathcal{C}$ there exist total computable functions f with *no* $\mathcal{C}$-optimal implementation.

Concretely, for any $\mathcal{C}$ and any program p computing f, there exists a program q computing the same f such that

$$\mathcal{C}(q, x) \;<\; \mathcal{C}(p, x) \quad \text{for all but finitely many } x. \tag{2.3}$$

2.2 RELEGATION

Exposition. We relegate implementation details to the accompanying Notebook, using Project Jupyter [Granger and Pérez, 2021], Python 3 (3.13.7) [Van Rossum and Drake, 2011], and GitHub [Dabbish et al., 2012]. Experiments are conducted on EMNIST [Cohen et al., 2017].

GitHub repository: github.com/Milan-Rosko/typedlearn;
GitHub policy: docs.github.com/en/site-policy;
On EMNIST: biometrics.nist.gov/cs_links/EMNIST/Readme.txt.

2.3 SIMPLE RUNDOWN

Ansatz 2.4. In our realization a learned classifier is treated as an auditable family of evaluators, together with semantics-preserving speedups obtained by verified specialization.

We fix a deterministic canonicalization of EMNIST bitmaps, binarization threshold, a feature schema mapping each image to a finite set of ground facts $\mathsf{feat}(F)$ (active pixel indices), and a stable, tie-broken arg max rule (Lemma 3.2). This Semantics Lock induces determinate maps from raw inputs to feature facts and from score vectors to predictions, and is recorded by ate es for reproducibility.

First, a simple multiclass perceptron is trained. Each class score is the sum of weights associated with the active feature facts. The learned weights are saved together with dataset and schema fingerprints (SHA), pinning the realized function to a specific preprocessing and data regime.

The learned model is then re-expressed as an instance of a restricted *intermediate representation* (IR) family. A generic logic-style interpreter (unification and rule firing) provides definitional semantics for this IR. On top of it, a recognizer/compiler emits a specialized residual evaluator for recognized families (unary vote and a join-bearing vote2 variant), eliminating interpretive overhead while preserving both the computed score vector and the resulting prediction.

All performance reporting is conditioned on deterministic equivalence checks: before any timing is reported, the compiled evaluator must match the interpreter on (i) predictions and (ii) full score vectors over a fixed semantics-locked finite sample (Definition 3.8). The realization is then a median of per-example runtimes, it reports speedup factors, and checks compilation amortization. The methodological claim is that, once learning

yields determinate, program-representable behavior, verified compilation can change *how* the behavior is computed without changing *what* is computed.

The interpretive point is that the cost driver is the *worst-case* inference pattern. In the general IR, join-bearing rule firing is an analogue of a locally simple but globally expensive proof step: it is syntactically shallow yet can force adversarial search. By contrast, the unary-vote fragment behaves like bounded axiom checking. Specialization targets this worst case by collapsing generic join search into direct residual computation; the equivalence gate ensures that the optimization is purely intensional.

In summary:

(i) A semantics-locked EMNIST artifact that re-expresses a learned classifier as an auditable evaluator definable in a universal formalism (e.g. λ-calculus), together with deterministic verification of a compiled evaluator against a reference interpreter on an explicit finite gate.

(ii) A measured "speedup-by-compilation" account for two IR families (unary vote; join-bearing VOTE2), separating *what* is computed from *how* it is computed.

(iii) Synthesis: once learning yields program-representable behavior, universal interpretability requests about nontrivial extensional semantic properties inherit classical undecidability barriers, while the practical bottleneck in restricted interpreters is governed by worst-case inference (joins) even when typical instances are benign.

2.4 RESULT

Our NOTEBOOK run produces a single finitary training-and-audit artifact for EMNIST digits. The pipeline is discrete and explicit: images are streamed from IDX-gz, canonicalized, binarized by a fixed threshold, mapped to unary facts, and then subjected to a compiled discrete margin-repair loop that outputs an integer weight table together with an incrementally maintained score cache. The reported results therefore concern (i) the locked experimental context and (ii) the exact finite verification outcome by thresholding pixels at `bin_threshold`= 127 with an optional intensity-based cap `cap_unary`= 128. The training procedure below is executed on a finite witness table of size `finite_n`= 256 drawn from the streamed training set (under `quick` mode caps), and audited exactly on that table.

On the finitary fragment

`| EMNIST/digits | finite_n=256 | classes=10 | margin=1 | cap_unary=128 |`,

the compiled repair loop of Definition 3.9 produces an integer weight table W and a cached score matrix S intended to coincide with the definitional sums (Definition 3.6). The target obligation is the finite-table margin constraint

$$\text{score}(i, y_i) \;\geq\; \text{score}(i, c) + 1 \quad \text{for all } c \neq y_i, \tag{2.4}$$

with

$$\text{score}(i, c) = \sum_{f \in \text{facts}(i)} W[(c, f)], \tag{2.5}$$

and a stable first-maximum tie-break for prediction. The verifier reports that the learned weights satisfy the margin constraint on the entire finite witness table, and that the compiled cache is consistent with the definitional computation:

```
| Violations: 0/2304 (finite-table) | Cache: OK |.
```

Here $2304 = 256 \cdot (10 - 1)$ is the total number of one-vs-rest margin obligations audited.

For an out-of-table diagnostic, we additionally evaluate on a small balanced probe drawn deterministically from the streamed test set, with `probe_per_class=10` (100 items total). The best probe accuracy observed in this run occurs at epoch 6:

```
| probe_acc=0.710 | w_keys=4573 |.
```

This probe figure is not itself certified; it is reported only as a lightweight generalization sanity check under the same SEMANTICS LOCK.

The run emits a hash identifying the certified finite artifact:

```
| 5a5def3278dcac0ad827615e75374d8c66a81b8366285e4661ae0021ab39d8cc | v0.1.2 |
```

In summary, our NOTEBOOK yields a finitary, mechanically checkable result: exact satisfaction of the stated margin obligations on an explicit finite witness table, together with a verified correctness gate for the compiled score cache. The balanced probe accuracy is an auxiliary measurement and should be read separately from the certificate.

3 THEORY

3.1 EXTENSIONALITY

Definition 3.1 (Semantics Lock)**.** Fix a preprocessing map

$$\mathsf{canon} : \mathcal{I} \to \mathcal{I}, \tag{3.6}$$

a binarization threshold

$$\tau \in \{0, \ldots, 255\}, \tag{3.7}$$

a feature schema

$$\Phi : \mathcal{I} \to \mathcal{P}(\mathcal{F}), \tag{3.8}$$

producing "ground facts", and a deterministic tie-breaking rule $\arg\max^*$ (Lemma 3.2). The tuple

$$\mathcal{L} \equiv \{\, \mathsf{canon},\ \mathsf{bin}_\tau,\ \Phi,\ \arg\max^* \,\} \tag{3.9}$$

is the SEMANTICS LOCK. It induces determinate functions from raw inputs to feature facts and from score vectors to decisions, relative to the fixed evaluator semantics under which canon and bin_τ are executed. We work *throughout* relative to this fixed lock $\mathcal{L}$ and a fixed, explicitly listed finite table

$$(X_i, y_i)_{i<N}, \quad y_i \in \{0, \ldots, K-1\}, \tag{3.10}$$

which may be regarded as a (predicative) table over $\mathbb{N}$ under standard encodings. We write the locked feature list for X_i as

$$FI[i] \;\equiv\; \mathsf{Facts}_{\mathcal{L}}(X_i), \tag{3.11}$$

where $\mathsf{Facts}_{\mathcal{L}}$ includes the forced bias fact, hence

$$|FI[i]| \geq 1 \text{ for all } i < N. \tag{3.12}$$

Let F denote the finite feature universe induced by the schema on this artifact. For propagation, we fix a postings index

$$\mathsf{post}[f] \;\equiv\; \{\, i < N : f \in FI[i] \,\}. \tag{3.13}$$

All objects above are finitary and admit standard encodings (finite sequences, bounded matrices, and finitely supported tables).

Lemma 3.2 (Tie-break)**.** Let $K \in \mathbb{N}$ and let

$$s : \{0, \ldots, K-1\} \to \mathbb{R} \tag{3.14}$$

be a score function. Define

$$\begin{gathered} \arg\max^{*}_{C \in \{0,\ldots,K-1\}} s(C) \;\equiv \\ \min\Big\{\, C \;\Big|\; \forall C' \in \{0, \ldots, K-1\} :\; s(C) \geq s(C') \Big\}. \end{gathered} \tag{3.15}$$

Equivalently, $\arg\max^{*}$ is implemented by a left-to-right scan which updates only on strict improvement.

Observation. Write $\mathsf{Facts}_{\mathcal{L}}(X)$ for the finite, deterministically normalized list of active feature indices extracted from $X \in \mathcal{I}$ under $\mathcal{L}$.

The locked finite artifact supports two complementary ways of presenting learning and evaluation. Both can be rendered as (possibly partial) functionals on a finitary state space together with a target predicate: Typed Repair targets a decidable certificate gate on the finite artifact, whereas gradient methods target minimization of a real-valued empirical risk.

3.2 EVALUATION

Definition 3.3 (Compiled Evaluator)**.** A compiled evaluator is a residual program E_θ emitted by a recognizer/compiler targeting a restricted IR family, such that for all inputs X,

$$E_\theta(X) \;=\; \mathsf{Interp}(\mathcal{R}_\theta, X), \tag{3.16}$$

where equality is taken at the level of the full score vector (hence also the induced prediction under the stable tie-broken $\arg\max^{*}$).

Definition 3.4 (Finite Equivalence Gate)**.** Let $\mathcal{R}_\theta$ be an IR artifact with definitional meaning given by the reference interpreter Interp, and let E_θ be a compiled residual evaluator. Define the finite equivalence gate by

$$\mathsf{Eq}_{\mathcal{T}}(E_\theta, \mathcal{R}_\theta) \;\equiv\; \forall X \in \mathcal{T} :\; E_\theta(X) \;=\; \mathsf{Interp}(\mathcal{R}_\theta, X), \tag{3.17}$$

where equality is again taken at the level of the full score vector.

Ansatz 3.5. Consider a deliberately incomplete refuter (a finite sieve P), which runs a diagonal schedule $t = k + n$ to guarantee systematic coverage of a product search space under explicit caps, and reserves the expensive verifier, e.g., Miller [1976]; Rabin [1980], for the residue that the cheap refuter cannot dismiss. The procedure is therefore intentionally partial: it is engineered to produce auditable *certificates of failure* cheaply (divisibility witnesses) and to invest verification effort only where the sieve cannot decide.

On the *finite* side, the core constraint family (margin satisfaction on a fixed witness table under a locked semantics) is decidable and is in fact checked exactly by the certificate gate; here there is no undecidability, only explicit bookkeeping and cost control. On the *universal* side, however, interpretability demands that quantify over all inputs or all semantically equivalent realizations push toward nontrivial extensional predicates on programs, where uniform decision is blocked in the classical way.

In this reading, "undecidability on purpose" means: we exploit the gap between (i) local, cheaply refutable obligations and (ii) global, non-uniform semantic demands by replacing the latter with diagonally scheduled search plus promotion of discovered structure into fast filters (postings indices, caches, specialized evaluators). The analogy is thus not that learning reduces to primality, but that in both settings one makes progress by converting a universal decision posture into an auditable stream of refutations, survivors, and certificates, controlled by explicit resource bounds and a fixed notion of semantics, cf. Rosko [2025c].

Remark. The training loop is "diagonal" in the operational sense that it proceeds by refuting a finite universal condition: violations of the margin constraints are Σ-witnesses, and each step eliminates the earliest witness under a fixed enumeration while preserving a decidable well-formedness invariant. On the finite semantics-locked artifact, success is a decidable Π-fact and is certified by exhaustive checking. However, once the learned evaluator is regarded extensionally as an arbitrary effective procedure in a universal hypothesis space, many explanatory demands become nontrivial semantic predicates of program denotations; uniform decision of such Π-shaped claims would constitute a semantic oracle. Hence the methodology intentionally replaces unattainable universal completion by auditable streams of local refutations and finite-gate certificates.

Definition 3.6 (State Space and Definitional Score)**.** A repair state is a pair

$$\Sigma \equiv (W, S), \tag{3.18}$$

where

$$W : \{0, \dots, K-1\} \times F \to \mathbb{Z}, \tag{3.19}$$

$$S : \{0, \dots, N-1\} \times \{0, \dots, K-1\} \to \mathbb{Z}. \tag{3.20}$$

Here W is a sparse integer weight table (default 0), and S is a cached score matrix. The definitional score on the finite table is

$$\mathsf{score}_W(i, c) \equiv \sum_{f \in FI[i]} W(c, f). \tag{3.21}$$

Definition 3.7 (Well-formedness)**.** Define postings well-formedness by

$$\begin{aligned}\mathsf{PostOK}(FI, \mathsf{post}) \;\equiv\; \forall i < N\ \forall f \in F :\\ \big(f \in FI[i]\big) \;\leftrightarrow\; \big(i \in \mathsf{post}[f]\big).\end{aligned} \tag{3.22}$$

and cache correctness by

$$\begin{aligned}\mathsf{CacheOK}(FI, W, S) \;\equiv\; \forall i < N\ \forall c < K :\\ S(i,c) = \mathsf{score}_W(i,c).\end{aligned} \tag{3.23}$$

Write

$$\vdash \Sigma : \mathsf{WF} \;\equiv\; \mathsf{PostOK}(FI, \mathsf{post}) \wedge \mathsf{CacheOK}(FI, W, S). \tag{3.24}$$

Definition 3.8 (Finite Certificate Gate)**.** Fix $\gamma \in \mathbb{N}$. Define a finite margin predicate (cache-level) by

$$\mathsf{FULL.ok}_\gamma^{\mathrm{cache}}(S) \;\equiv\; \forall i < N\ \forall c \neq y_i :\; S(i, y_i) \geq S(i,c) + \gamma, \tag{3.25}$$

and define the gate

$$\mathsf{OK}_\gamma(FI, W, S) \;\equiv\; \mathsf{CacheOK}(FI, W, S) \wedge \mathsf{FULL.ok}_\gamma^{\mathrm{cache}}(S). \tag{3.26}$$

For fixed (FI, y), the predicate OK_γ is total and decidable by explicit checks.

Definition 3.9 (Local Repair Step)**.** Fix $\gamma \in \mathbb{N}$ and an overshoot $\Delta \in \mathbb{N}$. Given $\Sigma = (W, S)$ and an index $i < N$, define the stable best competitor

$$c^*(\Sigma, i) \;\equiv\; \arg\max^*_{c \neq y_i} S(i,c), \tag{3.27}$$

the slack

$$\mathsf{slack}(\Sigma, i) \;\equiv\; (S(i, c^*) + \gamma) - S(i, y_i), \tag{3.28}$$

and $\ell \equiv |FI[i]|$ (so $\ell \geq 1$). If $\mathsf{slack} \leq 0$, define

$$\mathsf{Repair}(\Sigma, i, \gamma, \Delta) = \Sigma. \tag{3.29}$$

If $\mathsf{slack} > 0$, set

$$\mathsf{steps} \;\equiv\; \left\lceil \frac{\mathsf{slack}}{\ell} \right\rceil + \Delta, \tag{3.30}$$

and produce $\Sigma' = (W', S')$ by, for each $f \in FI[i]$ and each $j \in \mathsf{post}[f]$,

$$\begin{aligned} W'(y_i, f) = W(y_i, f) + \mathsf{steps}, \quad W'(c^*, f) = W(c^*, f) - \mathsf{steps},\\ S'(j, y_i) = S(j, y_i) + \mathsf{steps}, \quad S'(j, c^*) = S(j, c^*) - \mathsf{steps},\end{aligned} \tag{3.31}$$

leaving all other entries unchanged. Define

$$\mathsf{Repair}(\Sigma, i, \gamma, \Delta) \equiv \Sigma'. \tag{3.32}$$

Definition 3.10 (Repair Functional)**.** Fix $\gamma \in \mathbb{N}$ and an overshoot $\Delta \in \mathbb{N}$. Define the *first-refutation index* (if it exists) by

$$i^*(\Sigma) \;\equiv\; \min\Big\{ i < N : \exists c \neq y_i \ \big(S(i, y_i) < S(i,c) + \gamma\big) \Big\}. \tag{3.33}$$

If no such i exists, define $\mathsf{Repair}_{\gamma,\Delta}(\Sigma) = \Sigma$. Otherwise define

$$\mathsf{Repair}_{\gamma,\Delta}(\Sigma) \;\equiv\; \mathsf{Repair}(\Sigma, i^*(\Sigma), \gamma, \Delta). \tag{3.34}$$

Lemma 3.11 (Typedness)**.** Typing is the art of not asking questions when answers are known to be insufficient. If

$$\vdash \Sigma : \mathsf{WF} \quad \text{and} \quad \Sigma' = \mathsf{Repair}(\Sigma, i, \gamma, \Delta), \tag{3.35}$$

then

$$\vdash \Sigma' : \mathsf{WF}. \tag{3.36}$$

In particular, cache correctness is preserved:

$$\forall j, c: \;\; S'[j, c] = \mathsf{score}_{W'}(j, c). \tag{3.37}$$

Proof Sketch. By *invariant preservation,* since PostOK is unchanged and for $\mathsf{CacheOK}$, each primitive weight update

$$W[(c, f)] \mathrel{+}= \delta \tag{3.38}$$

is paired with exactly the induced cache updates

$$S[j, c] \mathrel{+}= \delta \tag{3.39}$$

for all j such that $f \in FI[j]$, equivalently $j \in \mathsf{post}[f]$ by PostOK. The linearity of

$$\sum_{f \in FI[j]} W[(c, f)] \tag{3.40}$$

yields the claim. ∎

Theorem 3.12 (Learning by Repair)**.** Assume $\vdash \Sigma : \mathsf{WF}$ and $\mathsf{slack}(\Sigma, i) > 0$ at i in Definition 3.9. Let

$$\Sigma' = \mathsf{Repair}(\Sigma, i, \gamma, \Delta) \tag{3.41}$$

with updated weights W'. Then $\mathsf{Margin}_{\gamma}(W', i)$ holds. Moreover, for every competitor $c \neq y_i$,

$$\mathsf{score}_{W'}(i, y_i) \;\geq\; \mathsf{score}_{W'}(i, c) + \gamma + \Delta \cdot |FI[i]|. \tag{3.42}$$

Proof Sketch. By *inequality-chasing.* Write $\ell = |FI[i]|$ (or 1 if empty). Since c^* is a stable maximizer among competitors and $\mathsf{CacheOK}$ holds, for every $c \neq y_i$ we have

$$S(i, c) \leq S(i, c^*), \tag{3.43}$$

hence the slack with respect to any competitor is at most $\mathsf{slack}(\Sigma, i)$. The update increases the true-class score at i by $\mathsf{steps} \cdot \ell$ and decreases the c^* score by $\mathsf{steps} \cdot \ell$; all other competitor scores are unchanged. Because

$$\mathsf{steps} \cdot \ell \geq \mathsf{slack}(\Sigma, i) + \Delta \cdot \ell, \tag{3.44}$$

the margin constraints are satisfied with the stated overshoot. ∎

Exposition. Philosophically, we exploit the asymmetry inherent in Reflection Principles, cf. Feferman [1960], between RE and co-RE sets by stating

a global semantic claim up front, then immediately discharging only its finitary, checkable fragment via explicit search for counterexamples. Below, we look at some of our entities in detail.

Lemma 3.13. If $\mathsf{Margin}_\gamma(W, i)$ fails in a well-formed state, then $(i, c^*(W, i))$ is a refutation witness. Applying Repair eliminates that witness while preserving well-formedness. Thus the training loop is an explicit refutation-guided search for W satisfying

$$\forall i \ \mathsf{Margin}_\gamma(W, i) \tag{3.45}$$

on the finite table.

Remark. This development is finitary by design: it depends only on locked preprocessing, finite witness tables, integer arithmetic, and decidable invariants (WF and the finite-table margin predicate). Smoothness and derivatives may be invoked externally to explain why analogous dynamics scale or generalize, but they are not required to state, execute, or audit the update procedure.

3.4 SPEEDUP THESIS

Definition 3.14 (Finite Certificate Predicate)**.** Given (FI, y, W, S, γ), define

$$\mathsf{FULL.ok} \ \equiv \ \forall i < N \ \forall c \neq y_i : \ S(i, y_i) \geq S(i, c) + \gamma, \tag{3.46}$$

$$\mathsf{CACHE.ok} \ \equiv \ \mathsf{CacheOK}(FI, W, S), \tag{3.47}$$

$$\mathsf{ok} \ \equiv \ \mathsf{FULL.ok} \wedge \mathsf{CACHE.ok}. \tag{3.48}$$

A *certificate* binds ok together with stable hashes of configuration, dataset identity, and weights (stable JSON + SHA256), so that the audited claim is replayable and checkable.

Lemma 3.15 (Gate Decidability)**.** For fixed K, finite N, and fixed lock $\mathcal{L}$, the predicate ok is total and decidable. If ok holds, then the cached evaluator S agrees with definitional semantics on the finite artifact and all stated margin obligations are satisfied.

Lemma 3.16 (Score Definability)**.** Fix K, a lock $\mathcal{L}$, and finite bounds N and cap. Under standard encodings of finite lists, bounded vectors, and integers, the functions computing $\mathsf{Facts}_\mathcal{L}$ on concrete inputs, the definitional scorer score_W, the stable best-competitor selector, the repair step Repair, and in particular, the finite certificate predicate ok are λ-definable and decidable on the finite artifact.

Thesis 3.17 (Speedup from Σ/Π Polarity)**.** Two orthogonal speedups are exposed by the pipeline, each admitting an arithmetic reading, in the sense of Kleene [1943]:

(i) *Evaluator speedup.* Compilation replaces a generic interpreter by a residual evaluator while aiming to preserve extensional behavior (Def. 3.3). Non-equivalence is witnessed by a concrete counterexample input X on which score vectors differ (a Σ-style object). By contrast, extensional equivalence is a global Π-style claim (universal over inputs) and, at the universal boundary, is not uniformly decidable. We therefore replace

the unattainable global oracle by an auditable gate: equivalence is required only on an explicit, finite, semantics-locked test family.

(ii) *Training speedup.* Refutation-guided repair replaces uninformed search for W by a specialized update that propagates local refutations through a typed dependency structure (Lemma 3.13). A violated margin constraint is a Σ-fact (there exists a witness (i, c)), whereas zero violations is a Π-condition (for all i and all $c \neq y_i$, the margin holds). On the finite witness table this Π-condition is decidable and is exactly what the certificate predicate ok attests.

Lemma 3.18 (Primitive Recursion)**.** We realize the repair trajectory by primitive recursion:

$$\boxed{\Sigma_0 \equiv (0,0), \quad \Sigma_{t+1} \equiv \mathsf{Repair}_{\gamma,\Delta}(\Sigma_t).} \tag{3.49}$$

Methodologically: if the iteration reaches a fixed point $\Sigma_{t+1} = \Sigma_t$, cf. Kleene [1952], then the decidable certificate predicate $\mathsf{OK}_\gamma(FI, W_t, S_t)$ can be checked exactly.

3.5 CROSS-ENTROPY

To present gradient methods in the same "functional + objective" idiom, we fix feature vectors

$$\varphi_i \in \mathbb{R}^F \quad (i < N), \tag{3.50}$$

derived from the same locked schema (e.g. bias plus unary indicators), and let

$$W \in \mathbb{R}^{K \times F}. \tag{3.51}$$

Define logits and probabilities by

$$z_i(W) \equiv W\varphi_i \in \mathbb{R}^K, \quad p_i(W) \equiv \mathsf{softmax}(z_i(W)). \tag{3.52}$$

The per-example loss and empirical risk are

$$\ell_i(W) \equiv -\log\big(p_i(W)_{y_i}\big), \quad L(W) \equiv \frac{1}{N}\sum_{i<N} \ell_i(W). \tag{3.53}$$

Fix a learning rate $\eta > 0$ and a deterministic schedule i_t (e.g. $i_t = t \bmod N$). One-step Stochastic Gradient Descent (SGD) is the primitive recursion

$$W_{t+1} \equiv W_t - \eta\,\big(p_{i_t}(W_t) - e_{y_{i_t}}\big)\,\varphi_{i_t}^\top, \tag{3.54}$$

where e_y is the one-hot basis vector. In "full-batch" form, this is gradient descent on L:

$$W_{t+1} \equiv W_t - \eta\,\nabla L(W_t). \tag{3.55}$$

To align prediction with the locked decision semantics elsewhere, define

$$\hat{y}_i(W) \equiv \arg\max^*_{c<K}\ z_i(W)_c, \tag{3.56}$$

using the same tie-broken $\arg\max^*$ (Lemma 3.2).

Proposition 3.19 (Parallel Condensation)**.** On the fixed semantics-locked finite artifact, both methods admit a compact presentation as iteration

schemes together with their respective targets:

$$\begin{aligned}&\textsc{Typed Repair}\\&\Sigma_{t+1} = \mathsf{Repair}_{\gamma,\Delta}(\Sigma_t) \;\rightsquigarrow\; \big(\Sigma_{t+1} = \Sigma_t\big) \Rightarrow \mathsf{OK}_\gamma(FI, W_t, S_t),\\&\textsc{Stochastic Gradient Descent}\\&W_{t+1} = W_t - \eta\,(p_{i_t}(W_t) - e_{y_{i_t}})\varphi_{i_t}^\top \;\rightsquigarrow\; \text{minimization of } L(W).\end{aligned} \tag{3.57}$$

In particular, Typed Repair is engineered so that success on the finite artifact is a decidable, mechanically checkable certificate predicate, whereas gradient descent is naturally specified by its recurrence together with a real-valued objective and typically requires additional analytic assumptions to obtain convergence guarantees.

4 REALIZATION

4.1 SEMANTICS

Realization 4.1 (Argumentum Maximi)**.** The stable arg max (Lemma 3.2) ensures predictions are total and deterministic; in ties it returns the least class index among maxima.

Algorithm

Input: score vector scores$[0..K-1] \in \mathbb{R}^K$ with fixed K.
Output: $\arg\max^*$ (least index among maxima).

```
procedure StableArgmaxStar(scores)
    best ← 0
    bestv ← scores[0]
    for C ← 1 to K − 1 do
        if scores[C] > bestv then
            best ← C
            bestv ← scores[C]
        end if
    end for
    return best
end procedure
```

Realization 4.2 (Unary Vote)**.** We realize semantics by generic proof steps (unification on atoms and rule firing). The implementation is intentionally overhead-heavy but auditable.

Algorithm

Input: weights $\theta : \{0,\dots,K-1\} \times \{0,\dots,783\} \to \mathbb{Z}$ (default 0), image X.
Output: $(\hat{y}, \text{scores})$.

```
procedure UnarySpec(θ, X)
    L ← Facts_C(X)
    for C ← 0 to K − 1 do
        scores[C] ← 0
    end for
    for all F ∈ L do
```

```
        for C ← 0 to K − 1 do
            scores[C] ← scores[C] + θ(C, F)
        end for
    end for
    ŷ ← StableArgmaxStar(scores)
    return (ŷ, scores)
end procedure
```

Definition 4.3 (Vocabulary)**.** Fix a finite offset vocabulary R and a deterministic boundary convention inducing a total operation

$$F \oplus r \in \{0, \dots, 783\} \text{ for } F \in \{0, \dots, 783\},\ r \in R. \tag{4.58}$$

Let

$$\text{rel\_id} : R \to \{0, \dots, |R| - 1\} \tag{4.59}$$

be the index map. Define

$$\mathsf{Active}(G, L) \tag{4.60}$$

as membership of G in the list-induced set of L.

Realization 4.4 (Specification)**.** VOTE2 introduces a join pattern over pairs of active features. A fixed relation vocabulary R (grid offsets) mediates the join; the compiler specializes the join into an offset scan, reducing the baseline quadratic join cost.

Algorithm

Input: weights $\theta^{(2)} : \{0, \dots, K-1\} \times \{0, \dots, |R|-1\} \to \mathbb{Z}$ (default 0), image X, fixed R.
Output: $(\hat{y}, \mathsf{scores}^{(2)})$.

```
procedure VoteTwoSpec(θ^(2), X, R)
    L ← Facts_L(X)
    for C ← 0 to K − 1 do
        scores^(2)[C] ← 0
    end for
    for all F ∈ L do
        for all r ∈ R do
            G ← F ⊕ r
            if Active(G, L) then
                rid ← rel_id(r)
                for C ← 0 to K − 1 do
                    scores^(2)[C] ← scores^(2)[C] + θ^(2)(C, rid)
                end for
            end if
        end for
    end for
    ŷ ← StableArgmaxStar(scores^(2)) return (ŷ, scores^(2))
end procedure
```

Lemma 4.5 (λ-definability)**.** For fixed K and fixed finite R, the procedures StableArgmaxStar, UnarySpec, and VoteTwoSpec are λ-definable under standard encodings of bounded vectors and finite lists. If $\mathsf{Facts}_{\mathcal{L}}$ is effective (as it is for concrete preprocessing and feature extraction), then the

full locked mapping

$$X \mapsto (\hat{y}, \mathsf{scores}) \tag{4.61}$$

is representable as a closed term in a universal formalism (e.g. λ-calculus or SK).

4.2 MAIN FIXED POINT ALGORITHM

Realization 4.6 (Finite Witness Table)**.** The diagonalization phase operates on a finite, auditable training table built from the streamed dataset. It extracts witness facts per example and compiles a postings index for fast propagation of local weight repairs into cached scores.

Specification

Inputs: training stream $\mathcal{D}_{train}$, parameters (N, τ, cap), and facts function $\mathsf{Facts}_{\mathcal{L}}$ induced by the SEMANTICS LOCK $\mathcal{L}$.
Output: finite table $(X_i, y_i)_{i=1}^{N}$, witness lists $FI[i] = \mathsf{Facts}_{\mathcal{L}}(X_i)$, postings $\mathsf{post}[f] = \{\, i : f \in FI[i] \,\}$, and cached score matrix $S[i, c]$ initialized to 0.

procedure BUILDFINITETABLE($\mathcal{D}_{train}, N, \mathsf{cap}, \mathcal{L}$)
  $(X_1, y_1), \ldots, (X_N, y_N) \leftarrow$ first N items from $\mathcal{D}_{train}$
  for $i \leftarrow 1$ to N **do**
    $FI[i] \leftarrow \mathsf{Facts}_{\mathcal{L}}(X_i)$ ▷ locked canonicalize, binarize, schema
    if $\mathsf{cap} \neq \infty$ **then**
      $FI[i] \leftarrow$ CAPFACTS($FI[i], \mathsf{cap}$) ▷ deterministic cap; stable tie by index
    end if
    $FI[i] \leftarrow$ UNIQPRESERVEORDER($FI[i]$)
    for all $f \in FI[i]$ **do**
      append i to $\mathsf{post}[f]$
    end for
  end for
  for $i \leftarrow 1$ to N **do**
    for $c \leftarrow 0$ to $K - 1$ **do**
      $S[i, c] \leftarrow 0$
    end for
  end for
  return $(X_{1..N}, y_{1..N}, FI, \mathsf{post}, S)$
end procedure

Realization 4.7 (Exact Margin Constraint)**.** Diagonalization targets an explicitly finite constraint family (hence auditable): each training example must have a fixed margin over every competing class under the same deterministic scoring semantics.

Definition

Weights are integer-valued and sparse: $W[(c, f)] \in \mathbb{Z}$ with default 0. Witness facts $FI[i]$ are finite lists of ground facts (e.g. unary pixel facts plus a bias).
Score:

$$\mathsf{score}_W(i, c) \equiv \sum_{f \in FI[i]} W[(c, f)].$$

Margin constraints (fixed margin $\gamma \in \mathbb{N}$):

$$\forall i \in \{1..N\}\ \forall c \neq y_i : \quad \mathsf{score}_W(i, y_i) \ \geq \ \mathsf{score}_W(i, c) + \gamma.$$

Stable competitor selection:

$$c^*(i) \ \equiv \ \arg\max^*_{c \neq y_i} \ \mathsf{score}_W(i, c),$$

where $\arg\max^*$ is the stable (first-maximum) rule of Lemma 3.2.

Realization 4.8 (Typed Repair)**.** Our λ-definable procedure provides a discrete analogue of BACKPROPAGATION, cf. Rumelhart et al. [1986] without gradients. It treats violated margin constraints as explicit error witnesses, computes an integer repair magnitude, and propagates the resulting update along compiled postings indices so that all affected cached scores are updated in bulk.

Diagonalization

Inputs: finite witnesses $(y_{1..N}, FI, \mathsf{post}, S)$, parameters (γ, Δ, E, U) where γ is margin, Δ is overshoot, E max epochs, U max updates.
State: sparse integer weights $W[(c, f)]$ (initialized to 0), cached scores $S[i, c]$ (initialized to 0).
Output: repaired weights W, epoch curve (diagnostics), and verification artifacts.

procedure DIAGONALIZEREPAIR($y, FI, \mathsf{post}, S, \gamma, \Delta, E, U$)
  initialize sparse $W[(c, f)] \leftarrow 0$
  updates $\leftarrow 0$; curve $\leftarrow [\,]$
  for *epoch* $\leftarrow 1$ to E **do**
    viol_before $\leftarrow$ COUNTVIOLATIONSFROMCACHE(S, y, γ)
    ep_updates $\leftarrow 0$
    for $i \leftarrow 1$ to N **do**
      $y_i \leftarrow y[i]$
      $c^* \leftarrow$ BESTCOMPETITORSTABLE($S[i, \cdot], y_i$)
      slack $\leftarrow (S[i, c^*] + \gamma) - S[i, y_i]$
      if slack ≤ 0 **then**
        continue
      end if
      $\ell \leftarrow |FI[i]|$; $\ell \leftarrow \max(1, \ell)$
      $steps \leftarrow \lceil \mathsf{slack}/\ell \rceil + \Delta$
                                  ▷ minimal integer repair + overshoot
      for all $f \in FI[i]$ **do** ▷ compiled propagation through postings
        $W[(y_i, f)] \leftarrow W[(y_i, f)] + steps$
        $W[(c^*, f)] \leftarrow W[(c^*, f)] - steps$
        for all $j \in \mathsf{post}[f]$ **do**
          $S[j, y_i] \leftarrow S[j, y_i] + steps$
          $S[j, c^*] \leftarrow S[j, c^*] - steps$
        end for
      end for
      ep_updates $\leftarrow$ ep_updates $+ 1$; updates $\leftarrow$ updates $+ 1$
      if updates $\geq U$ **then**
        break
      end if
    end for
    viol_after $\leftarrow$ COUNTVIOLATIONSFROMCACHE(S, y, γ)

```
        append to curve (epoch, ep_updates, viol_before, viol_after)
        if updates ≥ U then
            break
        end if
        if viol_after = 0 then
            break
        end if
    end for
    return (W, S, curve)
end procedure
```

Realization 4.9 (Certificate)**.** All claims made by the diagonalization phase are conditioned on exact verification on the finite table. A certificate binds configuration, finite dataset identity, and learned weights to a stable artifact hash.

Specification

Inputs: (FI, y, W, S), margin γ, and stable encodings/hashes.
Output: OK iff (i) all margin constraints hold under the definitional score and (ii) the cache equals definitional sums; plus a certificate record.

5 DISCUSSION

5.1 RELATED WORK

Exposition. Typed Repair is a perceptron-style, mistake-driven margin update, cf. Rosenblatt [1958]; Crammer and Singer [2001]; Shalev-Shwartz and Singer [2005], presented here in a finitary predicative form that targets a decidable certificate predicate (finite-table margin satisfaction plus cache correctness) rather than minimization of a surrogate loss Rumelhart et al. [1986]. The evaluation phase places the learned classifier in a restricted logic-style IR with definitional interpreter semantics, connecting to logic programming and Datalog evaluation where joins dominate cost, cf. Ullman [1988]. Our compiler specializes recognized join-bearing fragments into residual evaluators, as partial evaluation, cf. Jones et al. [1993], and reports speedups only after a finite equivalence gate validates score-vector agreement with the reference interpreter, akin to translation validation, cf. [Necula, 1997; Poetzsch-Heffter and Gawkowski, 2005].

5.2 INTERPRETATION AS A LIMIT

Observation. The results stem from a controlled comparison between two regimes: In the *universal* regime, a learned system is treated as program-representable in a Turing-complete formalism, and explanation demands thereby become questions about semantic properties of arbitrary programs. In the *restricted* regime, the learned object is forced into syntactically recognizable IR families with a fixed reference semantics, so that explanation and verification reduce to tractable, witness-producing procedures.

Many "perfect" explanation demands can be formalized as requiring a procedure that decides or constructs witnesses for nontrivial semantic properties of the learned program (e.g. invariances, counterfactual guarantees, minimal faithful summaries, universal error characterization). Assume a universal program model and let P be any nontrivial semantic predicate (true for some programs and false for others).

Thesis 5.1 (Limit of Universal Explanation)**.** Any method that, for *arbitrary* learning in a universal hypothesis space, promises perfect decision of a nontrivial extensional explanatory property (or perfect prediction of all future behaviors in the strong, uniform sense) cannot be realized as a total computable function without additional oracle assumptions. Consequently, "interpretability" is not solely an engineering problem; at the universal boundary it is constrained by predicate logic.

Ansatz 5.2 (Interpretability by Typedness)**.** Rather than attempting a *universal semantic oracle*, the project commits to a restricted hypothesis space: IR families are syntactically recognizable; semantics are fixed by a reference interpreter; and compilation targets a tractable fragment in which equivalence checks, certificate generation, and faithful summaries are mechanically feasible. The intended notion of explanation is therefore *witness-based*: an explanation is a certificate that can be checked against the reference semantics, typically relative to an explicit finite gate.

Exposition. Within the restricted regime, cost concentrates in closure rather than in recognition. Local feature checks and axiom matching are uniform and shallow, but global propagation requires iterating a saturation condition. In operational terms, closure under rule application is the worst case: it forces repeated search and dominates the verification/explanation budget when rules interact densely.

The NOTEBOOK avoids the universal barrier not by refuting it, but by working in a deliberately restricted regime: IR families are recognizable, semantics are pinned down, and compilation targets a tractable subset where equivalence checks and explanations are feasible. This is interpretability-by-restriction plus verification.

Lemma 5.3. For the unary-vote family, specialization eliminates generic unification overhead and yields a constant-factor speedup. For the join-bearing family (VOTE2), specialization replaces a generic join baseline of the form

$$O(|F|^2 \cdot |R|) \tag{5.62}$$

with an offset-scan evaluator of the form

$$O(|F| \cdot |R|), \tag{5.63}$$

where $|F|$ is the number of active features and $|R|$ is the size of the fixed relation vocabulary.

Logic 5.4 (Speedup)**.** The point is not merely performance engineering. Specialization converts an implicit semantic question,

> Does there exist a join witness that activates this consequence?

into an explicit, checkable enumeration over a fixed vocabulary. This keeps explanation aligned with verification: the same witnesses that justify conse-

quences are those that the evaluator checks. The asymptotic improvement is thus a concrete instance of the broader methodological claim: by restricting the language and fixing semantics, explanatory artifacts become both computable and efficiently auditable.

Corollary 5.5 (Blackboxness)**.** In universal hypothesis spaces, the demand for perfect semantic explanation runs into undecidability (modulo trivial fixed-point degeneracies). In the NOTEBOOK's restricted IR families, "blackboxness" is reduced not by interpretive magic but by design: semantics are pinned down, and explanations are certificates for a tractable fragment. The relevant notion of transparency is therefore conditional: it is as strong as the restriction is tight, and it degrades as one approaches to $\exists\forall$-shaped demands.

Definition 5.6 (Uniform Operator)**.** Fix a SEMANTICS LOCK $\mathcal{L}$ and a BLUM COMPLEXITY measure $\mathcal{C}$ for the chosen machine model. Assume a total effective transformer

$$\mathsf{S} : \mathbb{N} \to \mathbb{N} \tag{5.64}$$

on program indices such that, for every index e,

(i) *Extensional preservation:* $\varphi_{\mathsf{S}(e)} = \varphi_e$ as partial functions on the locked encodings;

(ii) *Terminal certificate:* either $\mathsf{S}(e)$ returns an index provably $\mathcal{C}$-optimal among all indices computing φ_e, or else S returns a mechanically checkable certificate that *no* extensionally equivalent index is $\mathcal{C}$-optimal.

Theorem 5.7 (Diminishing Returns from Necessity)**.** Fix a universal hypothesis space (capable of encoding arbitrary effective evaluators under standard encodings) and fix a BLUM COMPLEXITY measure $\mathcal{C}$. There is no total computable *completion operator* S which, given an index e, halts and returns either

(i) an index $e^\star$ such that $\varphi_{e^\star} = \varphi_e$ and $e^\star$ is $\mathcal{C}$-optimal among all indices computing φ_e, or

(ii) a certificate that φ_e admits no $\mathcal{C}$-optimal implementation.

Consequently, any effective optimization pipeline can only produce improvements as a stream of local, witness-producing gains (candidate rewrites, counterexamples, refutations, or finite-gate certificates); it cannot uniformly and conclusively "finish" optimization in the strong, global sense.

Proof Sketch. By *reductio ad absurdum.* Assume for contradiction that such a total S exists. Define the extensional semantic predicate on partial computable functions

$$\mathsf{OPT}(f) \iff \text{"there exists a } \mathcal{C}\text{-optimal index computing } f\text{."} \tag{5.65}$$

This predicate depends only on the denotation f, not on any particular syntactic presentation.

By Lemma 2.3, for any BLUM COMPLEXITY measure $\mathcal{C}$ there exist total computable functions f such that $\neg\mathsf{OPT}(f)$. On the other hand, $\mathsf{OPT}(f)$ holds for some f (e.g. sufficiently trivial total functions admit minimal evaluators under standard machine models), so OPT is nontrivial.

If S existed, we could decide $\mathsf{OPT}(\varphi_e)$ uniformly in e by running $\mathsf{S}(e)$ and observing whether it returns an optimal index or a certificate of nonexistence. Thus OPT would be decidable as a nontrivial extensional property, contradicting classical limitative results, in particular, Rice's theorem; cf. Rice [1953], and the surrounding foundations Gödel [1931]; Church [1936]; Blum [1967]; Chaitin [1974], therefore no such S exists. ■

Corollary 5.8 (Arithmetic Polarity of Optimization)**.** In the universal regime, *improvements* are Σ-shaped:

There exists an extensionally equivalent implementation with strictly better measured behavior.

is witnessed by a concrete alternative program together with a checkable gate (typically finite). By contrast, *completion* and *optimality* are Π-shaped:

For all extensionally equivalent implementations, none is better (eventually).

is a global universal claim. The absence of a total semantic oracle forces optimization to proceed by bounded search plus certificates on restricted gates; hence marginal gains are pushed into increasingly instance-dependent search as low-cost witnesses are exhausted.

Observation. On a fixed finite witness table under a semantics lock, completion *is* decidable: "no remaining violations" is a finite universal fact and can be certified exactly (as in the margin-and-cache certificate). Diminishing Returns here is therefore not folklore but precise: it reflects a structural distinction between finite, auditable gates (where saturation can be proved by exhaustive checking) and universal regimes (where uniform saturation would amount to a completion oracle for nontrivial extensional properties). At the level of generality where learned artifacts range over a hypothesis class, uniform interpretability and uniform optimization demands *must* become questions about program semantics.

ACKNOWLEDGMENTS

REFERENCES

M. Blum. A Machine-Independent Theory of the Complexity of Recursive Functions. *J. ACM*, 14(2):322–336, 1967. URL https://doi.org/10.1145/321386.321395.

G. J. Chaitin. Information-Theoretic Limitations of Formal Systems. *J. ACM*, 21(3):403–424, 1974. URL https://doi.org/10.1145/321832.321839.

A. Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, 1(1):40–41, 1936. URL https://doi.org/10.2307/2269326.

G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926, 2017. URL https://doi.org/10.1109/IJCNN.2017.7966217.

K. Crammer and Y. Singer. Ultraconservative Online Algorithms for Multiclass Problems. In *Computational Learning Theory*, volume 2111 of *Lecture Notes in Computer Science*, pages 99–115. Springer, Berlin, Heidelberg, 2001. URL https://doi.org/10.1007/3-540-44581-1_7.

L. Dabbish, C. Stuart, J Tsay, and J. Herbsleb. Social coding in GitHub: transparency and collaboration in an open software repository . In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, page 1277–1286, New York, NY, USA, 2012. Association for Computing Machinery. URL https://doi.org/10.1145/2145204.2145396.

S. Feferman. Arithmetization of metamathematics in a general setting. *Fundamenta Mathematicae*, 49(1):35–92, 1960. ISSN 0016-273600.

G. Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39: 176–210, 1935. URL https://doi.org/10.1007/BF01201353.

B. E. Granger and F. Pérez. Jupyter: Thinking and Storytelling With Code and Data. *Computing in Science & Engineering*, 23(2):7–14, 2021. URL https://doi.org/10.1109/MCSE.2021.3059263.

K. Grygiel and P. Lescanne. Counting and generating terms in the binary lambda calculus. *Journal of Functional Programming*, 25:e24, 2015. URL https://doi.org/10.1017/S0956796815000271.

K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38(1):173–198, 1931. URL https://doi.org/10.1007/BF01700692.

K. Gödel. On the length of proofs. In *Collected Works, Vol. I: Publications 1929–1936*, pages 396–399. Oxford University Press, Oxford, 1986. (Published posthumously).

N. D. Jones, C. K. Gomard, and P. Sestoft. *Partial Evaluation and Automatic Program Generation*. Prentice Hall, 1993. ISBN 978-0130202499.

S. C. Kleene. On the interpretation of intuitionistic number theory. *Journal of Symbolic Logic*, 8(2):109–124, 1943. URL https://doi.org/10.2307/2269016.

S. C. Kleene. *Introduction to Metamathematics*. North-Holland, 1952. ISBN 9780444896230.

G. L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13(3):300–317, 1976. URL https://doi.org/10.1016/S0022-0000(76)80043-8.

G. C. Necula. Proof-carrying code. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '97, page 106–119, New York, NY, USA, 1997. Association for Computing Machinery. ISBN 0897918533. URL https://doi.org/10.1145/263699.263712.

A. Poetzsch-Heffter and M. Gawkowski. Towards Proof Generating Compilers. *Electronic Notes in Theoretical Computer Science*, 132(1):37–51, 2005. URL https://doi.org/10.1016/j.entcs.2005.03.023.

M. O. Rabin. Probabilistic Algorithm for Testing Primality. *Journal of Number Theory*, 12 (1):128–138, 1980. URL https://doi.org/10.1016/0022-314X(80)90084-0.

H. G. Rice. Classes of Recursively Enumerable Sets and Their Decision Problems. *Transactions of the American Mathematical Society*, 74(2):358–366, 1953. URL https://doi.org/10.2307/1990888.

F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, 65(6):386–408, 1958. URL https://doi.org/10.1037/h0042519.

M. Rosko. A Fibonacci-Based Gödel Numbering: Δ_0 Semantics Without Exponentiation, 2025a. URL https://doi.org/10.48550/arXiv.2509.10382. arXiv preprint.

M. Rosko. The Fractal Logic of Φ-adic Recursion, 2025b. URL https://doi.org/10.48550/arXiv.2510.08934. arXiv preprint.

M. Rosko. On the Realizability of Prime Conjectures in Heyting Arithmetic, 2025c. URL https://doi.org/10.48550/arXiv.2511.07774. arXiv preprint.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. URL https://doi.org/10.1038/323533a0.

D. Scott. Continuous lattices. In F. W. Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, pages 97–136, Berlin, Heidelberg, 1972. Springer Berlin Heidelberg. ISBN 978-3-540-37609-5.

S. Shalev-Shwartz and Y. Singer. A New Perspective on an Old Perceptron Algorithm. In P. Auer and R. Meir, editors, *Learning Theory*, volume 3559 of *Lecture Notes in Computer Science*, pages 264–278. Springer, Berlin, Heidelberg, 2005. URL https://doi.org/10.1007/11503415_18.

R. J. Solomonoff. A Formal Theory of Inductive Inference. Part I. *Information and Control*, 7 (1):1–22, 1964a. URL https://doi.org/10.1016/S0019-9958(64)90223-2.

R. J. Solomonoff. A Formal Theory of Inductive Inference. Part II. *Information and Control*, 7(2):224–254, 1964b. URL https://doi.org/10.1016/S0019-9958(64)90131-7.

A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics, Vol. 2*, volume 123 of *Studies in Logic and the Foundations of Mathematics*. North-Holland (an imprint of Elsevier Science), Amsterdam, New York, 1988. ISBN 0444703586.

J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press, 1988. ISBN 978-0716781585.

G. Van Rossum and F. L. Drake. *The Python Language Reference Manual*. Network Theory, 2011. ISBN 978-1906966140.

VERSION

This manuscript is the initial release of the preprint. The accompanying project artifact (the NOTEBOOK) corresponds to version 0.1.1.

FINAL REMARKS

The author welcomes scholarly correspondence and constructive dialogue. No conflicts of interest are declared. This research received no external funding.

Milan Rosko is from University of Hagen, Germany

Email: Q1012878 @ studium.fernuni-hagen.de, or hi @ milanrosko.com