Weight-Space Programming: Symmetry, Certified Synthesis, and Higher-Topos Semantics

Anonymous researcher (agent-assisted)

2 November 2026

Abstract

We formalize weight-space programming through parameterized maps, homotopy quotients, and dependent solution objects. We prove descent criteria, an exact Boolean-to-ReLU compiler theorem, a protected-feature criterion for rank-one edits, and perturbation bounds. Explicit observation and coherence obstructions separate certificates from empirical evidence.

1. Scope and mathematical claims

Takehi, Teoh, Zhang, and Khattab [1] describe language models reading, constructing, and editing network weights under different tool and feedback allowances. Their reported successes and failures motivate this work; we did not reproduce their experiments. No universal interpretability claim is inferred.

We combine parameterized computation, symmetry descent, and dependent specifications with exact compilation and repair guarantees. Standard categorical facts [2,6,7], Boolean universality, and concentration inequalities are used with explicit attribution.

2. Parameterized computation and its semantic image

Fix an infinity-topos $\mathcal{X}$, size universes, and chosen data objects. Its internal language supplies products, dependent sums and products, identity types, and propositional truncation [6,7]. Assume a 0-truncated ordered ring object R; a natural number object alone supplies neither real arithmetic nor a norm. Geometric topology may be encoded by sheaves; 0-truncation concerns homotopy level. Section 6 works externally over real vector spaces; internalization requires the stated field, algebra, and norm assumptions.

An architecture A specifies a finite computation graph, operations, tensor shapes, and input/output conventions. Its parameter object is P_A; for a finite numerical graph it is a finite product of powers of R. Let X_A, Y_A be 0-truncated input and output objects. Fixed operations define

$$\mathrm{ev}_A : P_A \times X_A \longrightarrow Y_A, \qquad d_A : P_A \longrightarrow Y_A^{X_A}.$$

Write $f_p = d_A(p)$. Tokenization, normalization, masks, decoding rules, and execution horizon must be included in A or explicitly in the inputs. A randomized bounded decoder can instead be modeled deterministically with an explicit seed input; equality for every seed is stronger than equality of output distributions. An arbitrary infinity-topos does not automatically supply a probability monad.

Recall the bicategory of parameterized maps [2]. A 1-cell $X \to Y$ is $(P, f : P \times X \to Y)$; a 2-cell $(P, f) \to (Q, g)$ is $a : P \to Q$ with $g(a(p), x) = f(p, x)$. Composition is

$$(Q, g) \circ (P, f) = (P \times Q, (p, q, x) \mapsto g(q, f(p, x))).$$

Product associators and unitors give its coherence. In an infinity-category the corresponding commuting diagrams require specified coherent homotopies. This parameterized structure describes ordinary training, hypernetworks [3], and direct synthesis without making gradient descent part of the definition.

Define $B_A = \mathrm{im}(d_A)$, with image factorization

$$P_A \xrightarrow{q_A} B_A \hookrightarrow Y_A^{X_A}.$$

Here q_A is an effective epimorphism and B_A is 0-truncated. This is the object of realizable behaviors, relative to the fixed interface. Define $p \equiv_A p'$ by $f_p = f_{p'}$, equivalently $\prod_x f_p(x) = f_{p'}(x)$. A specification outside B_A cannot be realized merely by choosing a better editor.

3. Gauge symmetry, behavioral descent, and interpretation

For a layered network applying one scalar activation coordinatewise in each layer, hidden-unit permutations act on parameters. With permutation matrices $S_0 = S_L = I$,

$$W_l' = S_l W_l S_{l-1}^{-1}, \qquad b_l' = S_l b_l.$$

Induction on layers gives $h_l' = S_l h_l$, hence $f_{p'} = f_p$. Input and output labels are fixed. Let G_A denote this group, or any specified internal group with a coherently behavior-preserving action. Positive ReLU rescaling is another exact symmetry [5], but a general change of basis is not a symmetry of a coordinatewise nonlinear activation.

The action groupoid presents the homotopy quotient

$$Q_A = P_A // G_A = |[n] \mapsto P_A \times G_A^n| .$$

Coherent invariance of d_A induces $Q_A \to B_A$. Its automorphisms retain parameter stabilizers. This map need not be an equivalence: redundant units can create behaviorally identical parameters outside a chosen symmetry orbit. Nor is $\tau_{\le 0} Q_A$ automatically B_A.

Proposition 3.1 (interpretation descent). For a 0-truncated object Z, a map $r : P_A \to Z$ descends along $P_A \to Q_A$ exactly when $r(gp) = r(p)$ internally. The descended map is unique in the appropriate mapping space.

Proof. Mapping a geometric realization into Z gives the limit of the cosimplicial mapping objects of the action groupoid. Its degree-one condition is the displayed invariance. Because equality in Z is a proposition, all further coherence conditions are unique whenever they exist. Conversely a map on the quotient satisfies the degree-one condition. This is the universal property of this colimit. For a higher target, invariant values without coherent higher data do not suffice. □

An invariant reader describes behavior alone only if it further factors through q_A. Permutation-equivariant processing of weights already appears in [10].

Proposition 3.2 (editing descent). A map $F : P_A \to P_A$ induces a unique map $F_B : B_A \to B_A$ satisfying $F_B q_A = q_A F$ if and only if

$$p \equiv_A p' \implies F(p) \equiv_A F(p').$$

Proof. Necessity follows by applying F_B. For sufficiency, $q_A F$ agrees on the kernel pair of q_A. Effective epimorphism descent, with 0-truncated target B_A, gives the unique factorization. □

Equivariance $F(gp) = gF(p)$ is sufficient for a map on Q_A; it does not establish Proposition 3.2 for behaviorally equal parameters outside a single orbit. Both conditions are relative to the architecture and metadata.

Proposition 3.3 (symmetry-breaking obstruction). Suppose p is fixed by a subgroup $H \leq G_A$, and H acts on a finite set of units U without a common fixed unit. No equivariant selector $s : P_A \to U$ exists at p.

Proof. Equivariance would give $s(p) = s(hp) = hs(p)$ for every $h \in H$, contradicting the hypothesis. Two identical hidden units exchanged by a transposition provide an example. An orbit-valued answer, or an invariant distribution such as the uniform distribution on this pair, avoids choosing one distinguished unit. □

If $t : H \to P_A$ maps histories to final weights, a history property $v : H \to Z$ is readable from weights only if constant on fibers of t. Histories differing in an unused training example can yield the same weights. This concerns history, not computing a known finite network's outputs.

4. Specifications, certified writers, and coherent synthesis

Let S be an object of formal specifications and

$$\Phi : S \times Y_A^{X_A} \longrightarrow \Omega$$

a proposition-valued satisfaction predicate. Natural-language prompts must first be interpreted as particular terms of S; correctness relative to that formalization does not prove that the prompt was translated faithfully. Define the object of solutions

$$\mathcal{W}_A = \sum_{s:S} \sum_{p:P_A} \Phi(s, f_p) \longrightarrow S.$$

A certified writer is a section of this projection on its declared domain. An architecture-varying writer uses $\sum_A \sum_{p:P_A} \Phi(s, f_p)$, including the corresponding interface conversions. A writer without a proof returns a candidate, not a term of the solution object. For a certifying implementation it suffices to provide a certificate type $C(s, p)$, a decidable checker, and a proof

$$\mathrm{check}(s, p, c) = \mathsf{true} \implies \Phi(s, f_p).$$

The proof transfers trust to the checker's soundness and the precise execution semantics. This paper does not provide a complete checker for unrestricted neural programs. The finite checker in the companion is exhaustive only over its explicitly finite Boolean input domain.

Gauge invariance transports solution proofs, producing $\mathcal{W}_A//G_A \to S$. A section here selects a coherent implementation up to the chosen symmetries; it need not choose a globally named weight tensor.

Descent principle 4.1 (local synthesis). Let $\mathcal{V} \to S$ be either solution object above. A section after an effective epimorphism $U \to S$, together with its full Cech compatibility data, descends to a section over S.

Proof. In an infinity-topos the slice over S is recovered from the limit of the slices over the Cech nerve of $U \to S$. A compatible local section is exactly an object of the corresponding limit of section spaces. □

Nonempty fibers alone do not give the required data or an algorithm. As a general counterexample to fiberwise existence implying coherent selection, the degree-two covering $S^1 \to S^1$ has nonempty fibers but no section: a section would make multiplication by two on $\pi_1(S^1) = \mathbb{Z}$ have a right inverse, requiring $2k = 1$. This is a section obstruction, not a claim that every weight problem has this particular topology. In a groupoid presentation, overlap isomorphisms and their cocycle conditions are essential; higher fibers require higher coherence.

A repair specification can be a relation R on behaviors: preservation on a protected domain, a safety invariant, or refinement. If R is reflexive and transitive, parameters and proofs $R(f_p, f_{p'})$ form a thin category of certified edits. Identities use reflexivity and composition uses transitivity. For example, $f_p|_K = f_{p'}|_K$ is transitive. An arbitrary target-improvement predicate need not be transitive and cannot be used for this composition without a further proof.

5. Exact synthesis without target-model training

Theorem 5.1 (finite Boolean compilation). Every finite acyclic Boolean circuit using AND, NOT, and copying has an effectively constructed rational ReLU network with exactly the same output on every Boolean input. Compilation respects circuit wiring and composition up to renaming wires.

Proof. Let $\rho(z) = \max(0, z)$. Compile each AND gate to $\rho(a + b - 1)$, each NOT gate to $\rho(1 - a)$, and each copy to $\rho(a)$. On $\{0, 1\}$ these are precisely the named Boolean operations. Compile gates in topological order, assigning each gate a neuron and retaining its dependencies as wires. Induction on that order proves Boolean-valued intermediate results and equality with the source circuit. Each gate contribution has coefficient in $\{-1, 0, 1\}$; repeated wires add coefficients. If strict layers are required, carry earlier wires through identity ReLU neurons; their nonnegative Boolean values are preserved. Connecting compiled circuits yields the compiled composite up to wire renaming, establishing the semantic composition law. □

AND and NOT are Boolean-complete, so truth-table circuits represent every finite Boolean function, potentially at exponential cost. This does not ensure fit in a fixed architecture. Compilation uses a supplied program without fitting target-model data. A language-model proposer also uses pretrained parameters and its prompt: no target-model gradients does not mean no learned information. Tracr supplies a richer existing program-to-transformer compiler [4].

6. Exact and quantitative conditions for safe repair

All statements in this section use ordinary finite-dimensional real spaces and exact arithmetic, unless explicitly bounded otherwise.

Theorem 6.1 (protected-feature criterion). Fix features $h : X \to \mathbb{R}^d$ and an affine output $f(x) = Wh(x) + b$. For $u \neq 0$, the edit $W' = W + uv^T$ preserves f on $K \subseteq X$ exactly when

$$v \in V_K^{\perp}, \qquad V_K = \operatorname{span}\{h(x) : x \in K\}.$$

An edit of this form that preserves K and changes the output at x_0 exists exactly when $h(x_0) \notin V_K$.

Proof. The output difference is $u(v^T h(x))$. Since $u \neq 0$, it vanishes exactly when the scalar $v^T h(x)$ vanishes. This proves the first equivalence. If $h(x_0) \in V_K$, every such v also annihilates $h(x_0)$.

Otherwise its orthogonal projection onto $V_K^{\perp}$ is nonzero; choose that projection as v and any $u \neq 0$. Then $v^T h(x_0) = \|v\|^2 > 0$. □

If protected features span $\mathbb{R}^d$, this restricted editing scheme has no nonzero preserving update. Bias edits are covered by augmenting h with a constant coordinate. The theorem assumes the feature extractor stays fixed; internal-layer updates require a new analysis. A finite protected sample certifies preservation on its feature span, not on arbitrary unseen inputs.

Counterexample 6.2 (sparsity is not preservation). For $M > 0$ let $f_{w,a}(x) = a\rho(wx)$. Set $p_M = (M, 0)$ and $p'_M = (M, 1/M)$. One coordinate changes, the Euclidean parameter displacement is $1/M \to 0$, yet $f_{p'_M}(1) - f_{p_M}(1) = 1$. Padding with unused parameters makes the changed fraction $1/N$ arbitrarily small. Therefore support fraction and unqualified parameter distance cannot bound behavioral change uniformly over unbounded weights. Gauge rescaling also changes coordinate geometry [5].

Theorem 6.3 (layerwise perturbation certificate). Consider equal-shaped networks $h_l = \sigma_l(W_l h_{l-1} + b_l)$ and $h'_l = \sigma_l(W'_l h'_{l-1} + b'_l)$ with the same input. Assume each σ_l is 1-Lipschitz for a chosen vector norm, the matrix norm is induced, and $\|h_{l-1}(x)\| \leq H_{l-1}$ on K. Put $\epsilon_l = \|W'_l - W_l\|$ and $\beta_l = \|b'_l - b_l\|$. Then

$$\sup_{x \in K} \|h'_L(x) - h_L(x)\| \leq B := \sum_{l=1}^{L} (\epsilon_l H_{l-1} + \beta_l) \prod_{j=l+1}^{L} \|W'_j\|.$$

Proof. With $D_l = \sup_K \|h'_l - h_l\|$, add and subtract $W'_l h_{l-1}$ inside the preactivation difference. Lipschitz continuity and the triangle inequality give $D_l \leq \|W'_l\| D_{l-1} + \epsilon_l H_{l-1} + \beta_l$. Starting from $D_0 = 0$, repeated substitution gives the formula, with the empty product equal to one. □

For logits with the infinity norm, if the original winning logit exceeds every competitor by more than $2B$ on K, the edited argmax agrees everywhere on K. Each logit changes by at most B, so each winning margin falls by at most $2B$. This certifies a decision rule, not an entire decoded trajectory. Floating-point deployment needs a separate rounding-error bound; tanh, softmax, and GPU execution are not certified by the exact Boolean companion. Finite rational ReLU verification also admits established solver methods [8].

7. Observation protocols and limits of evidence

An editor's access is part of its type. A protocol specifies weight reads/writes, activation queries, output queries, gradients, losses, environment interaction, and host computation. Transcripts compose by concatenation, and a declared resource vector in $\mathbb{N}^k$ adds under composition. A bound on API calls is distinct from bounds on batch size, sequences, generated tokens, or episodes. A no-output-query protocol restricts operational access; arbitrary host code could still emulate forward evaluation unless the allowed computations are specified and checked. Our compiler uses its program as input, and its subsequent exhaustive validation is recorded separately.

Proposition 7.1 (observation obstruction). In sets, let $o : P \to O$ be the available observation and $v : P \to \{0, 1\}$ a property. A perfect reader on $o(P)$ exists exactly when v is constant on observation fibers.

Proof. A factorization $v = r \circ o$ implies fiber constancy. Conversely assign the unique value of v on each nonempty fiber; uniqueness makes this well-defined. No extension outside $o(P)$ is asserted. □

Rational ReLU constraints over a rational polytope reduce to finitely many activation cases and linear feasibility problems; no general undecidability claim about finite feedforward networks is made.

Proposition 7.2 (finite black-box tests). A deterministic tester making fewer than $|D|$ point queries cannot decide whether an arbitrary function $D \to \{0,1\}$ is identically zero.

Proof. Answer zero to every query. Choose an unqueried x_*; the zero function and the function supported at x_* have identical adaptive transcripts. The tester returns the same answer on both and errs on one. □

This result is specifically about black-box observations, not full access to circuit syntax or weights plus a proof procedure.

For statistical claims, fix at most m candidate edits independently of an iid holdout sample of size n, with $m, n \geq 1$ and $0 < \delta < 1$. With losses in $[0,1]$, define the paired improvement $Z_i = \ell(f_p, z_i) - \ell(f_{p'}, z_i) \in [-1,1]$, its sample mean $\widehat{\Delta}$, and expectation Δ. Hoeffding's inequality [9] and a union bound give, simultaneously for the candidates with probability at least $1-\delta$,

$$\Delta \geq \widehat{\Delta} - \sqrt{\frac{2\log(m/\delta)}{n}}.$$

Indeed each upper-tail error is at most $\exp(-nt^2/2)$; sum over m and solve for t. A holdout-adaptive search does not meet these hypotheses. Nor does this estimate prove preservation on all inputs. Report proof certificates and empirical confidence as different forms of evidence.

8. Reproducibility, boundaries, and authorship

The companion contains source, a rational ReLU compiler, exhaustive finite-domain equivalence checks, and checks for Section 6 and permutation symmetry. Recorded output states the checks performed. General results rest on the written proofs. No Lean development, general decompiler, or verified accelerator compiler is supplied. The source experiments were not rerun.

Readers declare observations and equivalences; writers declare realizability assumptions; edits carry preservation obligations; descent records coherent gluing. No universal inverse to denotation or universal improvement theorem follows.

Prepared for an anonymous human researcher with OpenAI agent assistance in research, writing, code, and checks. The submitter is responsible for submission and further verification. Source authors are not coauthors. Proofs are not proof-assistant checked. Version 1.0.0, 2 November 2026.

References

[1] R. Takehi, J. Teoh, A. Zhang, and O. Khattab. *Sparks of Weight-Space Programming: LLMs can natively read & write neural network weights.* 2 November 2026. Article; released logs.

[2] G. S. H. Cruttwell, B. Gavranovic, N. Ghani, P. Wilson, and F. Zanasi. *Categorical Foundations of Gradient-Based Learning.* 2021. arXiv:2103.01931.

[3] D. Ha, A. Dai, and Q. V. Le. *HyperNetworks.* 2016. arXiv:1609.09106.

[4] D. Lindner, J. Kramar, S. Farquhar, M. Rahtz, T. McGrath, and V. Mikulik. *Tracr: Compiled Transformers as a Laboratory for Interpretability.* 2023. arXiv:2301.05062.

[5] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio. *Sharp Minima Can Generalize For Deep Nets.* 2017. arXiv:1703.04933.

[6] J. Lurie. *Higher Topos Theory*. Princeton University Press, 2009. Author's arXiv version.

[7] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. 2013. Book and official downloads.

[8] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer. *Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks*. 2017. arXiv:1702.01135.

[9] W. Hoeffding. *Probability Inequalities for Sums of Bounded Random Variables*. JASA 58(301), 13-30, 1963. DOI:10.1080/01621459.1963.10500830.

[10] A. Zhou et al. *Neural Functional Transformers*. 2023. arXiv:2305.13546.