COUNTEREXAMPLE-GUIDED DISCOVERY OF EXECUTABLE SEMANTIC STATE ABSTRACTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

A semantic representation in a neural network may predict outputs without containing enough information to determine how it evolves through subsequent computation, and therefore need not define an independently executable high-level model. We study how to select the necessary state from given semantic variables and candidate auxiliary variables, and how to synthesize a high-level program that reproduces the transitions and specified outputs of a fixed computational system. Selecting an auxiliary variable also requires determining its successor, so additional features can destroy closure. We give a verifiable sufficient condition under which a dependency graph yields the unique least admissible state. In general, minimum auxiliary-state selection is NP-complete for explicit finite tables even with an involutive transition and a constant response, and remains NP-hard on a rational box with quadratic supplied functions. State-pair and single-state counterexamples distinguish state insufficiency from program inexpressibility; for a fixed template affine in its p unknown coefficients, exact synthesis needs at most $p+1$ verification calls. Cost-ordered search is optimal within the prescribed finite program class. With a positive verification margin, numerical synthesis terminates and yields finite-execution error bounds under certified domain invariance. Experiments on polynomial instances and ten trained networks demonstrate exact discovery, infeasibility certification, and distinct greatest and minimum selections. Tests on 120 finite-table systems isolate the role of auxiliary updates in state selection, while alternative feature libraries on the same frozen networks separate state sufficiency from program expressibility and show multiple minimal selectors without a least one.

1 INTRODUCTION

Internal representations in neural networks are often studied to identify intermediate concepts or computational variables used by a model (Brinkmann et al., 2024). Yet the ability to read out semantic information from a representation does not by itself show that the representation contains the state required for subsequent computation. Causal abstraction places a stronger requirement on an explanation by asking high-level variables and their relations to correspond to computations or interventions in the original system, rather than merely to correlate with its outputs (Geiger et al., 2025). At the same time, if the extraction or alignment map is allowed to be arbitrarily expressive, it can perform part of the computation that the resulting representation is intended to explain, weakening the interpretation (Sutter et al., 2025). This motivates a more stringent question: once semantic variables have been specified, what additional state must be retained so that these variables can evolve with the original computation and form an independently executable high-level model?

We study this question for a fixed computational system in which each computational stage has a state domain and fixed state transitions. A Transformer can be represented in this form at specified computational boundaries by retaining the residual state, cache contents, and any other state required to determine subsequent computation. At each stage, we assume that the semantic variables are given, together with a finite library of candidate auxiliary variables computable from the current state and a finite class of permitted program forms. Our goal is to select the state that must be retained and to synthesize explicit update and output rules so that the resulting high-level program can run without calling the remaining computation of the original system while reproducing the specified transitions and outputs. We do not infer the meanings of the semantic variables, and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

candidate variables cannot invoke the next original transition, the remaining decoder computation, or a separately trained substitute.

Two distinct failure modes must be separated. First, the retained state may contain insufficient information: two low-level states can have the same high-level representation while requiring different successor states or outputs, in which case no deterministic high-level update can exist regardless of the permitted formula class. Second, the retained state may contain sufficient information while the prescribed program class cannot express the required update. We therefore use state-pair counterexamples to determine which auxiliary variables can be retained consistently and to remove variables that cannot occur in any valid high-level representation. Once the retained state is fixed, single-state counterexamples impose successive constraints on the unknown coefficients of the update rules. This separation turns the questions of what information must be retained and how that information evolves into two distinct verification problems.

Exact state abstraction already requires stability of the induced state partition; Dean & Givan (1997) also establish a unique coarsest stable refinement under Boolean-variable subset representations. Difference-set constraints and counterexample-driven refinement occur in unique column combination discovery (Birnick et al., 2020). Here, retaining an optional successor coordinate activates a conditional dependence requirement, so the targets to be determined change with the selection. Our candidate-elimination procedure is related to Houdini (Flanagan & Leino, 2001), while the synthesis stage follows the counterexample-guided program-synthesis paradigm (Jha et al., 2010). Positive-margin synthesis (Ravanbakhsh & Sankaranarayanan, 2015) and abstraction refinement (Clarke et al., 2000) provide related verification mechanisms. We use these established mechanisms to separate state sufficiency from expressibility in a prescribed syntax; our main structural and complexity results concern the additional update obligations created by state selection.

Our main contributions are fourfold. **First,** for general supplied feature functions, we give an intersection-closure condition under which a dependency graph yields the unique least admissible state, together with a finite verification procedure for recognizing the condition. **Second,** we establish a complexity boundary for transition closure. One distinguishing coordinate per explicit state-pair clause permits linear-time least selection or infeasibility detection, whereas two suffice for NP-completeness even with a single involutive transition, constant response, and only one auxiliary coordinate forced by mandatory conditions. Hardness extends to linear involutions and quadratic supplied functions on rational boxes. **Third,** we separate state sufficiency from program expressibility using state-pair and single-state counterexamples. Houdini-style elimination finds the greatest admissible selection; affine-coefficient synthesis gives an exact program or a finite inconsistency certificate using at most one more verification call than the number of unknown coefficients, and cost-ordered search is optimal within the declared finite syntax. **Fourth,** positive-margin numerical synthesis terminates with an explicit verification-call bound and, with certified invariant domains, finite-execution output-error guarantees. Experiments on polynomial instances, finite-table systems, and trained networks test exact discovery, auxiliary-update constraints, selector structure, and program expressibility. A terminal decoder-readout construction illustrates the specification without claiming closure of earlier decoder transitions.

2 THE DISCOVERY PROBLEM

2.1 FIXED COMPUTATION AND SUPPLIED SEMANTICS

Let $(\mathcal{V}, \mathcal{E})$ be a finite directed graph whose vertices are computational types and whose edges are permitted transitions. For $v \in \mathcal{V}$ the nonempty state domain is $X_v \subseteq \mathbb{R}^{n_v}$. An edge e has a source $\mathrm{src}(e)$, a destination $\mathrm{dst}(e)$, and a fixed map $F_e : X_{\mathrm{src}(e)} \longrightarrow X_{\mathrm{dst}(e)}$. Types record discrete information such as the computational component, token position, and cache size. Edges represent original or controlled operations. Cycles are allowed when the same type-specific domains and maps are reused; finite-horizon decoder execution can instead be unfolded into an acyclic graph. Every guarantee is restricted to the declared domains.

Each type has a fixed semantic map $s_v : X_v \to \mathbb{R}^{m_v}$, where m_v is the number of semantic coordinates. A set $\mathcal{O} \subseteq \mathcal{V}$ has scalar responses $r_v : X_v \to \mathbb{R}$; for a decoder, a response can be the log-probability difference between two tokens. The interpretation and validity of s_v on X_v are assumed, not inferred

from the synthesis constraints. Semantic labels defined only on prompt states do not by themselves specify these functions on neighboring activation states or under interventions.

For each type, a library supplies scalar functions $f_{v,j} : X_v \to \mathbb{R}$, $j = 1, \ldots, N_v$. Their syntax uses only current-state inputs and specified basic operations; it cannot call an original transition, the remaining decoder computation, or a learned surrogate model. A selector $J = (J_v)_{v \in \mathcal{V}}$ chooses $J_v \subseteq \{1, \ldots, N_v\}$ and defines the extractor $\alpha_v^J(x) = \big(s_v(x), (f_{v,j}(x))_{j \in J_v}\big)$, $d_v = m_v + |J_v|$. Selected indices keep their library order. Library functions need not be linearly independent, and d_v counts stored scalar coordinates, not the intrinsic dimension of their image.

2.2 State sufficiency and executable formulas

Definition 1 (Admissible selector)**.** A selector J is admissible if there are deterministic update maps g_e and readout maps o_v defined on the images of the corresponding extractors such that $\alpha^J_{\mathrm{dst}(e)} \circ F_e = g_e \circ \alpha^J_{\mathrm{src}(e)}$, $r_v = o_v \circ \alpha_v^J$ $(v \in \mathcal{O})$. No continuity or expression-size condition is imposed.

Admissibility requires deterministic updates (closure) and readouts, but not executable formulas. The supplied semantic variables are recovered directly by projection onto the semantic coordinates of the extractor. Definition 1 uses unrestricted maps to distinguish insufficient state information from insufficient expressiveness of the permitted program class. Selectors are ordered by inclusion at each type.

A finite set $\mathscr{S}$ specifies program templates. A template $S \in \mathscr{S}$ fixes a selector $J(S)$ and loop-free, branch-free expressions $g_e^S(z; \theta)$ and $o_v^S(z; \theta)$ with a joint coefficient vector $\theta \in \mathbb{R}^{p_S}$. Every scalar expression is affine in θ but may be nonlinear in z; for example, it may be a linear combination of selected monomials in z. Coefficients may be shared across edges. Products of unknown coefficients, and unknown coefficients inside nonlinear primitives, are excluded. All primitives must be defined on the reported evaluation domains.

Each template has a declared cost $c(S)$, for instance maximum auxiliary-state size followed by the operation count for extraction, temporary storage, updates, and response evaluation. Every unknown coefficient counts toward cost even when fitted to zero, so sparsity reduces cost only through distinct templates. We assume a fixed computable total order refining the cost order. A minimum is always relative to this finite syntax, never to all equivalent real-valued programs.

3 Discovering which coordinates can be retained

For $x, x' \in X_v$ with $s_v(x) = s_v(x')$, let $\Delta_v(x, x') = \{j : f_{v,j}(x) \neq f_{v,j}(x')\}$ be the library coordinates that distinguish them. Write $b_{v,j} \in \{0, 1\}$ for whether coordinate j is selected. If an edge $e : v \to w$ satisfies $s_w(F_e(x)) \neq s_w(F_e(x'))$, every admissible selector satisfies $\bigvee_{j \in \Delta_v(x,x')} b_{v,j}$. The same clause follows when $v \in \mathcal{O}$ and $r_v(x) \neq r_v(x')$. If the differing successor quantity is the optional coordinate $f_{w,k}$, the necessary clause is $\neg b_{w,k} \vee \bigvee_{j \in \Delta_v(x,x')} b_{v,j}$. An empty disjunction is false. Therefore, an empty distinguishing set excludes $f_{w,k}$ when the differing target is optional, and excludes every selector when the differing target is a semantic coordinate or response. These clauses are dual-Horn (at most one negated variable) and independent of the update formulas.

Theorem 2 (State-pair characterization)**.** *A selector is admissible if and only if it satisfies every clause $\bigvee_{j \in \Delta_v(x,x')} b_{v,j}$ and $\neg b_{w,k} \vee \bigvee_{j \in \Delta_v(x,x')} b_{v,j}$ obtained from the stated domains, edges, and responses. Admissible selectors are closed under componentwise union. If any exists, there is a unique greatest admissible selector.*

The factorization criterion requires each successor coordinate and response to be constant on each fiber, where a fiber is the set of states with the same extracted value. Union closure holds because the joint state of two valid abstractions determines the successors of both. Intersection closure need not hold, so the existence of a greatest admissible selector does not imply the existence of a least admissible selector.

A *fiber verifier* certifies these conditions or returns a witness pair consisting of two states with the same extracted value but different values of a required successor coordinate or response, together

with the quantity on which they differ. Algorithm 1 in Appendix B starts from the full library and deletes one optional coordinate per witness.

Theorem 3 (Greatest admissible selector)**.** *Assume the fiber verifier is sound, complete, and terminating. Algorithm 1 returns the greatest admissible selector, or correctly reports that none exists. With $N = \sum_v N_v$, it uses at most $N + 1$ verifier calls. Every deletion is valid for all admissible selectors in the original library, not only for the current candidate.*

The invariant is that every admissible selector is contained in the current one. A witness pair agreeing on the current source extractor agrees on every subset, so a differing optional successor coordinate is absent from every admissible selector; a mandatory difference rules out all of them. This is the Houdini argument (Flanagan & Leino, 2001) with functional-dependence clauses. Several optional-coordinate witnesses valid for the same current extractor justify deleting their coordinates together. The remaining obligations must then be rechecked, because deletion can make another target undetermined. Sequential deletion is a convenient implementation, not a condition for soundness. Appendix B gives the proofs and an example in which adding a coordinate destroys closure.

The procedure establishes state sufficiency but does not synthesize g_e. A smaller selector may admit a simpler executable program than the greatest selector. The $N + 1$ bound counts verifier calls rather than the cost of producing witnesses. Exact verification is effective for polynomial data on semialgebraic domains but is not assumed for general decoder expressions.

4 Exact synthesis of the update program

4.1 Coefficient constraints from single states

Fix a template S. Because the extractor is fixed by the template and the high-level expressions are affine in θ, define the residual for each scalar identity required by Def. 1 as

$$R_a(x;\theta) = u_a(x)^{\mathsf{T}}\theta - y_a(x) = 0,\ x \in D_a,\ a \in \mathcal{A}_S, \tag{1}$$

where D_a is a source domain, $u_a : D_a \to \mathbb{R}^{p_S}$ is a fixed coefficient-vector function, $y_a : D_a \to \mathbb{R}$ is a fixed target-value function, and the finite index set $\mathcal{A}_S$ indexes all required transition coordinates and responses. Coefficient sharing across identities is encoded in the rows u_a. Nothing here assumes that the entire candidate library is closed.

An *identity verifier* either certifies every identity in (1) or returns a pair (a, x) with nonzero residual. Given the returned pairs $\mathcal{Q}$, write their equations as $M_{\mathcal{Q}}\theta = y_{\mathcal{Q}}$. The inner loop solves this finite linear system, rejects the template if it is inconsistent, and otherwise proposes a solution and calls the verifier again. Algorithm 2 in Appendix C gives the full procedure.

Theorem 4 (Finite exact synthesis)**.** *Assume exact coefficient arithmetic and a sound, complete, terminating identity verifier. For a template with p_S coefficients, the inner loop terminates after at most $p_S + 1$ verification calls. It returns a program satisfying Def. 1, or a finite inconsistent subsystem of* (1)*. A rejection certificate uses at most $p_S + 1$ scalar equations and a vector λ with $\lambda^{\mathsf{T}} M_{\mathcal{Q}} = 0,\ \lambda^{\mathsf{T}} y_{\mathcal{Q}} \neq 0$.*

Each rejected proposal contributes an equation that is not valid throughout the current affine solution set, so after at most p_S strict dimension reductions the next verification call either accepts or proves inconsistency. This bound does not imply that random samples identify the program or that the accepted coefficients are unique; Appendix C gives the proof.

4.2 Cost-ordered discovery and its computable case

When an exact fiber verifier is available, Algorithm 1 can first be used to eliminate impossible selectors. The remaining templates are then considered in nondecreasing cost, with templates excluded by certified pair clauses skipped. The synthesis loop is applied to each remaining template until a verified program is found or all templates have been rejected. The first verified program is returned. If every template is rejected, the resulting pair witnesses and inconsistency certificates form the rejection record. Omitting the deletion phase does not affect correctness or completeness; it only removes a sound preprocessing step.

Theorem 5 (Minimum-cost discovery)**.** *Under the hypotheses of Theorem 4, and with sound preprocessing when used, cost-ordered search returns a minimum-cost exact program in $\mathscr{S}$ whenever one exists, and otherwise rejects the entire declared syntax. It uses at most $\sum_{S\in\mathscr{S}}(p_S+1)$ identity-verifier calls in addition to any fiber checks. If the domains are semialgebraic over the real algebraic numbers and all supplied functions and templates are polynomial with real algebraic coefficients, both verifiers and all synthesis steps are effective and terminate.*

For general Transformer expressions with transcendental operations, the exact result remains conditional on an identity verifier; solver failure or timeout therefore gives *unknown*, not infeasibility. Section 6 gives the positive-margin alternative, while Appendix C extends the effective case to fixed single-valued semialgebraic primitives.

5 Structure and complexity of minimum closed state

Minimum closed-state selection is a minimum-ones problem over the supplied feature library (Khanna et al., 2001). The following results connect its complexity to the requirements for transition closure.

5.1 A sufficient condition for a least selector

For each type v, put $L_v=\{1,\dots,N_v\}$ and let $\mathcal{T}_v$ contain every scalar successor semantic coordinate, optional successor coordinate, and response with source domain X_v. A target $q\in\mathcal{T}_v$ is *mandatory* if it is semantic or a response; otherwise it has an optional destination coordinate $t(q)=(w,k)$. Let $\mathcal{D}_q$ contain the sets $I\subseteq L_v$ for which q is constant on every fiber of $x\mapsto(s_v(x),(f_{v,j}(x))_{j\in I})$, and call q supported when this family is nonempty. The family is upward closed but can have incomparable minimal members. If it is nonempty and intersection-closed, its intersection $A_q=\bigcap_{I\in\mathcal{D}_q} I$ belongs to $\mathcal{D}_q$ and is its unique least member.

Theorem 6 (Least admissible selector)**.** *Suppose every nonempty $\mathcal{D}_q$ is intersection-closed. An empty $\mathcal{D}_q$ for a mandatory target implies infeasibility. Otherwise, form a directed graph on the auxiliary coordinates (v,j). For every supported optional target $q\in\mathcal{T}_v$, insert the edges $t(q)\to(v,j)$ for $j\in A_q$. Mark $t(q)$ forbidden for each unsupported optional target. Let $\mathcal{B}$ consist of the nodes (v,j) with $j\in A_q$ for a mandatory target, and let $\mathcal{R}$ be the vertices reachable from $\mathcal{B}$, including $\mathcal{B}$ itself. An admissible selector exists exactly when $\mathcal{R}$ contains no forbidden vertex. In that case, $J_v=\{j:(v,j)\in\mathcal{R}\}$ is the unique least admissible selector and hence the unique minimum-cardinality selector.*

The graph propagates the update obligations introduced by retaining a coordinate, including obligations around cycles. It also minimizes every coordinatewise nondecreasing selector cost. This concerns state sufficiency; the permitted program syntax can still exclude the resulting selector.

Proposition 7 (Recognizing the structural condition)**.** *With a sound, complete, terminating scalar fiber verifier, the hypothesis of Theorem 6 can be checked using at most $\sum_v(N_v+2)|\mathcal{T}_v|$ scalar fiber-verifier calls. For each target, first test L_v. If this succeeds, set $A_q=\{j\in L_v : L_v\setminus\{j\}\notin\mathcal{D}_q\}$ and test A_q. This last test succeeds exactly when $\mathcal{D}_q$ is intersection-closed. The obtained supports then give the graph in Theorem 6.*

The bound counts verifier calls, not runtime; explicit finite tables permit polynomial-time verification. A failed structural test does not establish infeasibility or rule out a least selector. The condition holds when each semantic fiber has a Cartesian-product feature image. For disjoint semantic and auxiliary Boolean coordinate blocks on a full product domain, the least-selector conclusion is already covered by the deterministic specialization of the fluentwise refinement result of Dean & Givan (1997, Theorem 8); our condition and its recognition procedure apply to general supplied functions. Appendix D proves the product case and gives a nonlinear cyclic example.

5.2 When transition closure requires a combinatorial choice

Theorem 8 (A complexity boundary for transition closure)**.** *For explicitly listed finite rational domains and evaluation tables, deciding whether an admissible selector of size at most k exists is NP-complete even with one type, a constant response, and one involutive transition $F^2=\mathsf{I}$. Hardness*

holds when the mandatory state-pair conditions force just one designated auxiliary coordinate and every distinguishing set has size at most two. Replacing F by the identity makes the empty selector admissible. For polynomially represented functions, minimum admissible selection is also NP-hard for a linear involution on a full-dimensional rational box, with semantic and candidate functions of degree at most two. The distinguishing-set bound concerns the finite construction. On that finite-domain restriction and a compact union-of-segments extension, every admissible selector in the reduction admits polynomial updates of degree at most two. In contrast, when every distinguishing set in the explicit state-pair clauses has size at most one, a least admissible selector, or infeasibility, is computable in time linear in the clause representation.

The semantic successor forces one auxiliary coordinate. Updating that coordinate requires a vertex cover, while the reverse transition adds only implications back to it. Thus the combinatorial choice arises from auxiliary-state closure. Appendix E gives the constructions and proofs.

Theorem 18 in Appendix F also gives NP-hardness for identity transitions with linear supplied functions, and NP-completeness for the rational affine case. Its Exact Cover by 3-Sets reduction isolates the static sparse-representation obstruction (Natarajan, 1995; Jokar & Pfetsch, 2008); Appendix F retains its proof. In contrast, Theorem 8 makes the additional choices necessary to update the selected state. CLUE (Ovchinnikov et al., 2021) optimizes over linear extraction subspaces and allows nonlinear reduced dynamics; our results concern subsets of specified features and discrete transitions. Appendix I further distinguishes executable nonlinear state closure from finite-dimensional linear invariant lifting.

6 CERTIFIED NUMERICAL DISCOVERY

Exact equality is too strict for most decoder applications. A numerical procedure must control both the error of the proposed abstraction and the error incurred when evaluating the synthesis constraints. Replacing exact equality with a sampled least-squares objective provides neither a uniform guarantee nor a termination argument. We use a positive verification margin, as in delta-decision methods (Kong et al., 2018), and derive an explicit bound on the number of verification calls for templates that are affine in their parameters.

For a fixed template, let the coefficient vector θ range over the cube $K_S = [-B_S, B_S]^{p_S}$, where B_S is a positive rational number. The scalar constraints before relaxation, indexed by a finite set $\mathcal{H}_S$, have the form $h_\nu(t;\theta) = a_\nu(t)^\mathsf{T}\theta - \beta_\nu(t) \le 0,\ t \in \Omega_\nu,\ \nu \in \mathcal{H}_S$. The domains Ω_ν are compact rational boxes or compact sets equipped with computable finite nets and known covering radii. All functions admit certified numerical evaluation and have a supplied computable modulus of continuity in t, uniform over K_S. A known positive rational L_S bounds $\|a_\nu(t)\|_1$ for every constraint and point, so each constraint is L_S-Lipschitz in θ in the maximum norm.

Fix a rational margin $\delta > 0$. The sample solver solves the accumulated finite feasibility problem, returning either a coefficient vector satisfying all sampled inequalities with right-hand side $\delta/4$ or a certificate that no coefficient vector satisfies them with right-hand side zero. The verifier returns either a certificate that every constraint is at most δ on its full domain, or a point where some constraint exceeds $\delta/2$. The overlap between these thresholds ensures that no decision depends on exact equality at a threshold.

Theorem 9 (Terminating numerical synthesis)**.** *Under the preceding compactness and evaluation assumptions, both subroutines have finite implementations. The resulting loop terminates with a program satisfying $h_\nu \le \delta$ uniformly, or correctly rejects the existence of a program satisfying $h_\nu \le 0$ uniformly. For $p_S \ge 1$ the number of verification calls is at most $\left\lceil \left(1 + \frac{8B_S L_S}{\delta}\right)^{p_S} \right\rceil$. For $p_S = 0$ a single verification call suffices.*

A rejected vector violates some constraint by more than $\delta/2$, whereas every later proposal satisfies the corresponding sampled constraint with value at most $\delta/4$. The L_S-Lipschitz bound therefore separates distinct proposals by more than $\delta/(4L_S)$ in the maximum norm, and a packing argument on K_S yields $\left\lceil \left(1 + \frac{8B_S L_S}{\delta}\right)^{p_S} \right\rceil$. The same type of packing argument is used in robust counterexample-guided synthesis for control (Ravanbakhsh & Sankaranarayanan, 2015). A certified coefficient grid provides a finite implementation of the sample solver, while finite covers constructed from the

effective modulus provide a finite implementation of the verifier. Neither construction requires exact evaluation of transcendental numbers. Appendix G gives the precision thresholds and the termination proof.

Cost-ordered search compares feasibility before and after relaxation. If any template is feasible before relaxation, the search returns a program that is feasible after relaxation and whose cost is no greater than the minimum cost of a template feasible before relaxation. It need not minimize cost after relaxation, and numerical rejection does not establish nonexistence at the relaxed tolerance. Exact pair clauses may be reused only when they remain valid at the tolerance being searched.

Experiment 2 quantifies the size of the exhaustive verification cover implied by a reference modulus; the results are reported in Section 8.

6.1 Using state pairs at a nonzero tolerance

The exact deletions of Algorithm 1 do not transfer automatically to approximate synthesis. A feature that is not exactly predictable may still be predictable within an acceptable tolerance. State pairs remain useful when their target separation is compared with the error tolerance accepted by the numerical procedure.

Proposition 10 (Quantitative state-pair certificate)**.** *For a scalar target f_{tar}, extractor A, and proposed readout q, suppose $A(x) = A(x')$. Then every deterministic q satisfies $\max\{|f_{\mathrm{tar}}(x) - q(A(x))|, |f_{\mathrm{tar}}(x') - q(A(x'))|\} \geq \frac{1}{2}|f_{\mathrm{tar}}(x) - f_{\mathrm{tar}}(x')|$. If q is K_q-Lipschitz, exact equality of the extracted states can be relaxed. The corresponding lower bound is $\frac{1}{2}\max\{0, |f_{\mathrm{tar}}(x) - f_{\mathrm{tar}}(x')| - K_q \|A(x) - A(x')\|_\infty\}$.*

A semantic-successor separation greater than $2(\varepsilon_e + \delta)$ excludes any selector leaving the pair indistinguishable. For an optional successor coordinate the exclusion is conditional on retaining it; for a response the threshold is $2(\varepsilon_v^{\mathrm{out}} + \delta)$. These clauses are necessary pruning conditions, not certificates of expressibility in the prescribed program class. Appendix B gives the full statement.

7 From local errors to an executable abstraction

7.1 Domain invariance must be part of synthesis

For every selected extractor, let $Z_v \subseteq \mathbb{R}^{d_v}$ be a supplied compact box containing $\alpha_v^J(X_v)$, with containment certified independently of θ. Denote its coordinate endpoints by $\underline{z}_v$ and $\overline{z}_v$. A one-step error bound of at most $\varepsilon_e + \delta$ is obtained by including both signed coordinate residuals in $h_\nu(t;\theta) = a_\nu(t)^{\mathsf{T}}\theta - \beta_\nu(t) \leq 0,\ t \in \Omega_\nu,\ \nu \in \mathcal{H}_S$ with pre-relaxation threshold ε_e, and analogously for each output with threshold $\varepsilon_v^{\mathrm{out}}$. Exact abstract-domain invariance under the relaxed certificate is enforced by the additional constraints

$$\underline{z}_{\mathrm{dst}(e),j} + \delta - g_{e,j}^S(z;\theta) \leq 0,\ g_{e,j}^S(z;\theta) - \overline{z}_{\mathrm{dst}(e),j} + \delta \leq 0 \tag{2}$$

for every $z \in Z_{\mathrm{src}(e)}$. These constraints stay affine in the unknown coefficients, and relaxing their right-hand sides to δ gives $g_e^S(Z_{\mathrm{src}(e)}) \subseteq Z_{\mathrm{dst}(e)}$. The original constraints require an interior margin δ and may have no solution even when non-strict invariance is feasible.

Corollary 11 (Cost and execution certificate)**.** *Under Theorem 9, cost-ordered search returns a program feasible after relaxation or correctly rejects feasibility of every template before relaxation. If any template is feasible before relaxation, the returned cost is no greater than the minimum cost among templates feasible before relaxation. The returned program preserves the declared abstract boxes and satisfies $\left\|\alpha^J_{\mathrm{dst}(e)}(F_e(x)) - g_e^S(\alpha^J_{\mathrm{src}(e)}(x);\theta)\right\|_\infty \leq \varepsilon_e + \delta$ throughout each source domain, with output error at most $\varepsilon_v^{\mathrm{out}} + \delta$.*

The guarantee concerns actual iterations of the synthesized program. Once initialized, the high-level program uses its own state and formulas, and it is not re-extracted from the original activation at every boundary. The computational cost of extraction is accounted for separately.

7.2 A SPECIFIED ORIGINAL OUTPUT

Consider an edge path $e_0, \dots, e_{T-1}$ with types $v_0, \dots, v_T$ and $v_T \in \mathcal{O}$, with the fitted coefficients suppressed from the notation. Let $Q_T = o^S_{v_T}$ and $Q_t = Q_{t+1} \circ g^S_{e_t}$ denote the suffix responses for the remaining abstract computation, and let Λ_t be a certified Lipschitz bound for Q_t on Z_{v_t} in the maximum norm. The two systems are initialized with $z_0 = \alpha^J_{v_0}(x_0)$ and evaluated along the same path.

Proposition 12 (Finite-execution response bound)**.** *Under Corollary 11, the terminal outputs satisfy* $|r_{v_T}(x_T) - o^S_{v_T}(z_T)| \le \varepsilon^{\mathrm{out}}_{v_T} + \delta + \sum_{t=0}^{T-1} \Lambda_{t+1}(\varepsilon_{e_t} + \delta)$*. If the perturbed initial state remains in* Z_{v_0}*, a discrepancy of at most* η_0 *contributes at most* $\Lambda_0\eta_0$ *additional error. For two admissible executions, the error in their response difference is bounded by the sum of the corresponding response bounds.*

Invariance of the abstract boxes ensures that every composition of exact and abstract steps appearing in the proof is well defined. If the response is a two-token log probability ratio, an estimated contrast larger in magnitude than its error bound certifies its sign in the original decoder. Explaining a sampled token or the truth of generated text requires a different specification.

A discovery instance at the original decoder readout The terminal-normalization construction in Appendix L retains the original vocabulary contrast and its positive normalization denominator. Under the stated semantic factorization and domain assumptions, one squared-norm feature is necessary and sufficient within the supplied library, and its readout is affine in the unknown coefficients. The construction provides an exact worked instance of the specification at the terminal decoder readout; closure under earlier attention or feed-forward transitions remains outside its scope.

8 EXPERIMENTS

Experimental setup. Experiments 1, 2, 4, and 5 use exact symbolic or combinatorial CPU computations. Experiment 3 uses deterministic FP64 training on an NVIDIA GeForce RTX 4090, followed by rationalization and exact CPU verification; Experiment 5 reuses the resulting networks. Acceptance and rejection rely on exact checks rather than floating-point residual thresholds. Appendix K gives the detailed protocols for all five experiments.

Experiment 1: exact discovery. We ran the deletion and counterexample-guided synthesis procedures on five polynomial instances. Four instances yielded exact identities; the affine-readout instance was rejected within its declared syntax. All 12 tested templates satisfied the p_S+1 identity-verification bound, and the reduced inconsistent subsystems contained at most three equations. Table 1 reports the final templates. Appendix K.2 gives the complete setup and query counts.

Table 1: Exact discovery. Fiber queries count scalar tests; calls/bound gives the final template's identity-verifier calls and $p_S + 1$ bound. Times are single-run CPU wall times.

Instance	Fiber queries	Templates	Outcome	Calls/bound	Time (s)
Square update	2	2	Accepted	$6/7$	0.122
Affine readout	2	1	Rejected	$5/5$	0.048
Feature deletion	5	1	Accepted	$5/5$	0.035
Greatest vs. minimum	4	4	Accepted	$7/10$	0.180
Auxiliary update	3	4	Accepted	$8/10$	0.147

Experiment 2: verification cover size. For a reference box of dimension $d_{\mathrm{res}} + 1$ and reference modulus $\omega_{\mathrm{ref}}(h) = h$, exact rational and integer calculations give approximately 1.052×10^{16} and 1.049×10^{21} grid points at $d_{\mathrm{res}} = 4$ for $\delta = 10^{-2}$ and 10^{-3}, respectively. Appendix K.3 reports all tested dimensions. This calculation concerns a specified exhaustive grid, not measured verification time or a certified modulus for a pretrained decoder.

Experiment 3: trained polynomial networks. We trained ten six-dimensional polynomial networks and rationalized the frozen weights before exact state-pair and polynomial-identity checks.

Seven instances admitted exact programs, while three were rejected by state-pair witnesses. Among the feasible instances, the minimum auxiliary-state size was 1.00 ± 1.00, ranging from zero to two coordinates. The greatest admissible selector could be strictly larger: for seed 701 it retained all four candidates, whereas the minimum program used only x_2. All seven accepted programs required degree two. FP64 least-squares fits of the same templates had mean test RMSE 5.07×10^{-17} but no identity certificates, so this comparison concerns certification rather than numerical fit. Appendix K.4 gives the complete protocol and results.

Experiment 4: auxiliary-update constraints. We instantiated the finite-table construction in Appendix E using graphs with $8, 12, 16, 20, 32, 48$ vertices, edge probabilities 0.2 and 0.5, and ten seeds per configuration. Each system had one involutive transition and a constant response. Solving the state-pair clauses generated from the tables gave $\min |J| = 1 + \tau(H)$ in all 120 instances, where the minimum is over admissible selectors and $\tau(H)$ is the independently computed minimum vertex-cover size. Exhaustive search agreed on the optimal cardinality in all 40 instances with at most 12 vertices. Removing optional-successor constraints reduced every optimum to the single forced auxiliary coordinate, whose own successor then failed the fiber condition. Replacing the transition by identity made the empty selector admissible in every instance, while the original systems' greatest selectors retained the full library. These controls isolate the additional state requirements imposed by auxiliary updates; they validate the finite construction rather than establish its complexity lower bound empirically.

Experiment 5: alternative feature libraries. We reused the ten rationalized networks from Experiment 3 without retraining, comparing the base library $\{x_i : 1 \le i \le 4\}$, the cubed library $\{x_i^3 : 1 \le i \le 4\}$, and their union. Selector enumeration recovered the same seven feasible instances and minimum auxiliary sizes in all three libraries, with mean 1.00 ± 1.00 over feasible instances. The base library also reproduced the accepted selectors and polynomial degrees from Experiment 3. Degree-two polynomial templates admitted exact programs in seven instances for the base library but only three for the cubed library; all three required no auxiliary state. For the other four feasible instances, a fixed real cube-root operation recovered the retained base coordinates, after which the verified base updates and cubing of selected auxiliary successors yielded exact programs. Polynomial identity checks verified these constructions, whose polynomial parts have degree at most six in the recovered coordinates. These constructions provide exact verification without a separate synthesis run or minimum-degree analysis.

The recognizer in Proposition 7 agreed with determining-family enumeration for all 50 base-library successor targets, using at most six queries per target. With both feature representations available, seeds $211, 307, 701, 1009$ had respectively $4, 4, 2, 4$ incomparable minimal admissible selectors and no least selector. The three semantic-only feasible instances retained the empty least selector despite failure of the all-target intersection-closure condition. Thus the condition is sufficient rather than necessary: interchangeable features can preserve minimum storage while eliminating a least selector, and an invertible feature transformation can preserve state sufficiency while changing expressibility in a fixed syntax.

9 Scope and conclusion

Our results are relative to the declared state domains, semantics, candidate library, and program syntax. They do not guarantee compact abstractions for every checkpoint or domains covering all natural-language contexts. Exact verification can be costly, and minimum-cost search exponential. The synthesis guarantees require fixed extraction maps and affine dependence on unknown coefficients. Applying the method to Transformers requires supported semantics, certified extraction ranges, and verification of transition and readout expressions on the declared domains. Softmax and other transcendental operations are covered by the numerical result under its assumptions, while finite-precision execution additionally requires rounding-error bounds and valid primitive domains.

Within these limits, the structural condition identifies when state selection reduces to dependency propagation, while the involutive construction shows how auxiliary updates create combinatorial choices. The synthesis procedures distinguish insufficient state from insufficient program syntax and provide finite certificates. With certified domain invariance, accepted numerical programs execute independently with finite-horizon error bounds for specified original outputs.

AI USE STATEMENT

Generative AI tools were used to aid and polish the writing of this paper, to support the retrieval and discovery of related work, and to assist in checking and revising portions of the mathematical arguments and proofs, and to support the verification of portions of the experimental code. All AI-assisted mathematical work and experimental code were independently verified by the authors against the stated assumptions, derivations, and reported results, and all references identified with AI assistance were checked against the original publications. The authors take full responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Proofs and verification details are provided in Appendices B–H, and Appendix K reports the experimental protocols and detailed results for all five experiments. The supplementary material contains the source code and recorded outputs used for all five experiments.

REFERENCES

Johann Birnick, Thomas Bläsius, Tobias Friedrich, Felix Naumann, Thorsten Papenbrock, and Martin Schirneck. Hitting set enumeration with partial information for unique column combination discovery. *Proc. VLDB Endow.*, 13(12):2270–2283, July 2020. ISSN 2150-8097. 10.14778/3407790.3407824. URL https://doi.org/10.14778/3407790.3407824.

Jannik Brinkmann, Abhay Sheshadri, Victor Levoso, Paul Swoboda, and Christian Bartelt. A mechanistic analysis of a transformer trained on a symbolic multi-step reasoning task. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4082–4102. Association for Computational Linguistics, 2024. 10.18653/v1/2024.findings-acl.242.

Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In E. Allen Emerson and Aravinda Prasad Sistla (eds.), *Computer Aided Verification*, pp. 154–169, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45047-4.

Thomas Dean and Robert Givan. Model minimization in markov decision processes. In *AAAI/IAAI*, pp. 106–111, 1997.

Cormac Flanagan and K. Rustan M. Leino. Houdini, an annotation assistant for ESC/Java. In *FME 2001: Formal Methods for Increasing Software Productivity*, volume 2021 of *Lecture Notes in Computer Science*, pp. 500–517. Springer, 2001. 10.1007/3-540-45251-6_29.

Atticus Geiger, Duligur Ibeling, Amir Zur, Maheep Chaudhary, Sonakshi Chauhan, Jing Huang, Aryaman Arora, Zhengxuan Wu, Noah Goodman, Christopher Potts, and Thomas Icard. Causal abstraction: A theoretical foundation for mechanistic interpretability. *Journal of Machine Learning Research*, 26(83):1–64, 2025. URL https://jmlr.org/papers/v26/23-0058.html.

Susmit Jha, Sumit Gulwani, Sanjit A. Seshia, and Ashish Tiwari. Oracle-guided component-based program synthesis. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, pp. 215–224. ACM, 2010. 10.1145/1806799.1806833.

Sadegh Jokar and Marc E. Pfetsch. Exact and approximate sparse solutions of underdetermined linear equations. *SIAM Journal on Scientific Computing*, 31(1):23–44, 2008. 10.1137/070686676.

Richard M. Karp. *Reducibility among Combinatorial Problems*, pp. 85–103. Springer US, Boston, MA, 1972. ISBN 978-1-4684-2001-2. 10.1007/978-1-4684-2001-2_9. URL https://doi.org/10.1007/978-1-4684-2001-2_9.

Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing*, 30(6):1863–1920, 2001. 10.1137/S0097539799349948. URL https://doi.org/10.1137/S0097539799349948.

Soonho Kong, Armando Solar-Lezama, and Sicun Gao. Delta-decision procedures for exists-forall problems over the reals. In *Computer Aided Verification*, volume 10982 of *Lecture Notes in Computer Science*, pp. 219–235. Springer, 2018. 10.1007/978-3-319-96142-2_15.

B. K. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2): 227–234, 1995. 10.1137/S0097539792240406.

Alexey Ovchinnikov, Isabel Pérez Verona, Gleb Pogudin, and Mirco Tribastone. CLUE: Exact maximal reduction of kinetic models by constrained lumping of differential equations. *Bioinformatics*, 37(12):1732–1738, 2021. 10.1093/bioinformatics/btab010.

Hadi Ravanbakhsh and Sriram Sankaranarayanan. Counter-example guided synthesis of control Lyapunov functions for switched systems. In *54th IEEE Conference on Decision and Control*, pp. 4232–4239. IEEE, 2015. 10.1109/CDC.2015.7402879.

Denis Sutter, Julian Minder, Thomas Hofmann, and Tiago Pimentel. The non-linear representation dilemma: Is causal abstraction enough for mechanistic interpretability? In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/dbb98528c9870377f3f0d133aae6050b-Abstract-Conference.html.

A Operational specification and notation

The low-level system is fixed before discovery. Each edge has a fixed source and destination type, and its input domain includes every state on which the proposed guarantee is claimed. A source state need not be prompt-reachable, but this choice must be explicit. Conversely, a guarantee restricted to a finite collection of prompt states is not a statement about an open activation neighborhood. Neither the exact selector results nor their finite synthesis bounds require differentiability. The numerical realization uses the effective compact-domain assumptions stated separately.

An original Transformer may be represented by its full residual and cache state between computational components. Alternatively, let T_e denote the original transition on edge e. Suppose a supplied extraction Φ_v from original states satisfies $\Phi_{\mathrm{dst}(e)} \circ T_e = F_e \circ \Phi_{\mathrm{src}(e)}$ and preserves the selected responses. An accepted program then composes with Φ_v to give an abstraction of that original model. The composition follows by substitution; it does not transfer any state-minimality theorem from a larger intervention family. Readout injectivity is not needed for the forward synthesis identities.

Only finitely many types and edges are used in a discovery instance. For bounded generation, token positions, remaining horizon, and cache sizes can be retained in the type, with a finite unfolding of the controlled execution. The present results apply to the declared finite family of transitions and do not replace a continuous family of controls by a finite one.

The extraction library is finite as syntax, not merely as a list of learned functions accessible through an unrestricted neural network. Each primitive specifies its arguments and arithmetic cost. Allowed primitives can include fixed residual projections, prescribed cache statistics, squared norms, and normalization expressions with certified denominators. Evaluating an edge to obtain an offline target is permitted; using the edge itself as a primitive of the *online* extraction program is not. The same distinction applies to remaining-decoder calls.

A.1 NOTATION

Symbol	Meaning
$\mathcal{V}, \mathcal{E}, \mathcal{O}$	Types, fixed transitions, and response types.
X_v, n_v, F_e	Low-level domain, ambient dimension, and transition.
s_v, m_v, r_v	Supplied semantics, their coordinate count, and response.
$f_{v,j}, N_v, N$	Optional expressions, per-type count, and total count.
J_v, α_v^J, d_v	Selected indices, extractor, and stored dimension.
$b_{v,j}, \Delta_v$	Boolean selection variable and distinguishing-index set.
$\mathcal{T}_v, \mathcal{D}_q, A_q$	Scalar targets, determining sets, and their intersection.
$\mathcal{B}, \mathcal{R}$	Required auxiliary nodes and their dependency closure.
$S, \mathscr{S}, c(S)$	Program template, finite template family, and cost.
θ, p_S	Joint unknown coefficient vector and its dimension.
g_e^S, o_v^S	Explicit high-level update and response formulas.
R_a, u_a, y_a, D_a	Exact residual, coefficient row, target, and test domain.
$\mathcal{Q}, M_{\mathcal{Q}}, y_{\mathcal{Q}}$	Learned tests and their finite linear system.
λ	Left-nullspace certificate for an inconsistent system.
$h_\nu, a_\nu, \beta_\nu, \Omega_\nu$	Numerical inequality, its affine data, and domain.
K_S, B_S, L_S	Coefficient cube, its radius, and coefficient sensitivity bound.
$\delta, \varepsilon_e, \varepsilon_v^{\mathrm{out}}$	Verification margin and error tolerances before relaxation.
$Z_v, \underline{z}_v, \overline{z}_v$	Abstract box and its coordinate endpoints.
Q_t, Λ_t, T	Abstract suffix response, its sensitivity, and path length.
η_0	Initialization discrepancy.

All semantic coordinate maps remain fixed when a selector or coefficient vector changes. Auxiliary coordinates do not acquire semantic names merely because they are necessary for closure.

B STATE-PAIR CERTIFICATES AND GREATEST-SELECTOR DISCOVERY

Algorithm 1 Greatest admissible selection from a fixed library

Require: The semantic maps, candidate functions, transitions, responses, and an exact fiber verifier.

$J_v \leftarrow \{1, \dots, N_v\}$ for each $v \in \mathcal{V}$.
loop
  Verify all successor and response fiber conditions for α^J.
  if all conditions hold **then**
    return J and the verified fiber conditions.
  end if
  Receive a source pair (x, x') and a differing required target.
  if the target is a semantic coordinate or a response **then**
    return infeasible, with the witness and previous deletion record.
  end if
  Let $f_{w,k}$ be the optional successor coordinate named by the witness.
  Record the witness and set $J_w \leftarrow J_w \setminus \{k\}$.
end loop

B.1 A FACTORIZATION LEMMA

Lemma 13. *Let $A : X \to Z$ and $B : X \to Y$ be arbitrary maps. There is a map $C : A(X) \to Y$ with $B = C \circ A$ if and only if $A(x) = A(x')$ implies $B(x) = B(x')$. When it exists, C is unique on $A(X)$.*

Proof. Necessity follows by applying C to equal values of A. Conversely, for any $z \in A(X)$, define $C(z)$ as the common value of B on the nonempty set $\{x \in X : A(x) = z\}$. The implication makes this value independent of the representative. Any map satisfying $B = C \circ A$ must therefore take this value at z, which proves uniqueness. □

The lemma is set-theoretic and imposes no smoothness or executable-expression guarantee.

B.2 Proof of Theorem 2

Proof. First suppose that J is admissible. Take $x, x' \in X_v$ such that $s_v(x) = s_v(x')$. If all selected optional source coordinates also agree, then $\alpha_v^J(x) = \alpha_v^J(x')$. Def. 1 forces equality of every required successor coordinate and of the response when present. Consequently, a difference in a successor semantic coordinate or in the response requires that some member of $\Delta_v(x, x')$ be selected. This is $\bigvee_{j \in \Delta_v(x,x')} b_{v,j}$. A difference in optional successor coordinate k has the same consequence whenever k is selected at the destination, $\neg b_{w,k} \vee \bigvee_{j \in \Delta_v(x,x')} b_{v,j}$.

Conversely, suppose every clause holds. For two states with equal extracted source state, their current semantics agree and no selected index belongs to $\Delta_v(x, x')$. A difference in a successor semantic coordinate would violate $\bigvee_{j \in \Delta_v(x,x')} b_{v,j}$. A difference in a selected optional successor coordinate would violate $\neg b_{w,k} \vee \bigvee_{j \in \Delta_v(x,x')} b_{v,j}$. At a response type, a response difference would also violate $\bigvee_{j \in \Delta_v(x,x')} b_{v,j}$. Lemma 13 therefore supplies each map in Def. 1 on the attained image.

Now let J and J' be admissible and form their componentwise union J''. Equality of $\alpha_v^{J''}$ implies equality of both α_v^J and $\alpha_v^{J'}$. Their separate closure identities imply equality of every successor coordinate selected by J''. Responses are preserved by either representation. Thus J'' is admissible. There are at most 2^N selectors. If at least one is admissible, taking the finite union of all of them gives an admissible selector containing every other one, and it is uniquely greatest. □

B.3 Proof of Theorem 3

Proof. Write $J^{(0)}$ for the full library and $J^{(t)}$ for the current selector after t deletions. We maintain the invariant that every admissible selector I satisfies $I_v \subseteq J_v^{(t)}$ at every type. The invariant holds initially.

Suppose the verifier produces an edge $e : v \to w$, two source states with the same extracted value under $\alpha_v^{J^{(t)}}$, and an optional successor coordinate $f_{w,k}$ with $k \in J_w^{(t)}$ whose values differ at the two successor states. Since $I_v \subseteq J_v^{(t)}$, equality under the current extractor also implies equality under α_v^I for every admissible selector I. If such an I contained k at w, the admissibility condition Def. 1 would force $f_{w,k}$ to have the same value at the two successor states, contradicting the witness. Therefore every admissible I omits k, and deleting it preserves the invariant.

If the witness instead gives a difference in a successor semantic coordinate or response, the same argument contradicts admissibility of every selector satisfying the invariant. Reporting nonexistence is therefore sound. If the verifier finds no violation, Lemma 13 establishes admissibility of the current selector. The invariant then proves that it is greatest. Each nonterminal call deletes one previously present optional index, so there are at most N such calls and one final accepting or rejecting call. □

Witnesses obtained for a common current selector each exclude their named coordinate from every admissible selector, so their deletions can also be applied as a batch. Deleting one or several coordinates can cause another retained coordinate to violate the admissibility conditions. The verifier must therefore recheck all remaining successor and response conditions, including those on incoming edges of affected types. Checking only the edge that produced the most recent witness would be insufficient. The algorithmic bound assumes each verifier call checks all current requirements, or that an equivalent implementation maintains and eventually processes all pending checks.

B.4 Why adding a feature can destroy closure

For example, with $F(x, y, w) = (x, w, w)$, $s(x, y, w) = r(x, y, w) = x$, and candidate $f_1 = y$, the empty selector is admissible whereas retaining y is not: $(0, 0, 0)$ and $(0, 0, 1)$ have identical extracted states but different successor y values.

Intersection closure also fails. On $[-1, 1]^2$, take the identity transition, semantic map $s(x, y) = x$, response $r(x, y) = y$, and optional features y and y^3. Either singleton selector is admissible with

unrestricted deterministic readouts, but their empty intersection is not. Their union is admissible, as Theorem 2 requires.

B.5 Quantitative use of a pair witness

Full statement and proof of Proposition 10. Let a scalar target $f_{\mathrm{tar}} : X \to \mathbb{R}$ be approximated by $q \circ A$. If $A(x) = A(x')$, then

$$\max\{|f_{\mathrm{tar}}(x) - q(A(x))|, |f_{\mathrm{tar}}(x') - q(A(x'))|\} \geq \tfrac{1}{2}|f_{\mathrm{tar}}(x) - f_{\mathrm{tar}}(x')|. \tag{3}$$

If q is K_q-Lipschitz and the two extracted states need not agree, the right-hand side can instead be replaced by

$$\tfrac{1}{2}\max\{0, |f_{\mathrm{tar}}(x) - f_{\mathrm{tar}}(x')| - K_q \|A(x) - A(x')\|_\infty\}. \tag{4}$$

Proof. Insert and subtract $q(A(x))$ and $q(A(x'))$ in the target difference and use the triangle inequality. With equal extracted states the middle term is zero. Otherwise the Lipschitz assumption bounds it by $K_q \|A(x) - A(x')\|_\infty$. Each remaining term is at most their maximum, which proves both inequalities. □

C Exact coefficient synthesis and certificates

Algorithm 2 Exact program discovery in a finite syntax

Require: Cost-ordered templates and a terminating exact identity verifier.

for each template S not excluded by certified selector constraints **do**
  Initialize the finite constraint set $\mathcal{Q} \leftarrow \varnothing$.
  loop
    Solve $M_{\mathcal{Q}}\theta = y_{\mathcal{Q}}$ exactly.
    if the system is inconsistent **then**
      Store a certificate as in Theorem 4 and terminate the current inner loop.
    end if
    Let θ be a solution and verify every identity in (1).
    if all identities hold **then**
      return the instantiated template and its identity certificate.
    end if
    Receive (a, x) with $R_a(x; \theta) \neq 0$.
    Append $u_a(x)^{\mathsf{T}}\theta = y_a(x)$ to $\mathcal{Q}$.
  end loop
end for
return infeasible within the declared syntax, with all rejection records.

C.1 Constructing the affine residual system

A scalar coordinate of a template has a fixed expression of the form

$$g^S_{e,j}(z; \theta) = \gamma^S_{e,j,0}(z) + \sum_{i=1}^{p_S} \theta_i \gamma^S_{e,j,i}(z), \tag{5}$$

where the γ functions are known. A zero function indicates that the corresponding coefficient is unused in that scalar expression, while coefficient sharing is represented by the same θ_i appearing in multiple expressions. Terminal response templates have the same form. Substitution of the fixed extractor into (5) produces the coefficient row $u_a(x)$; the successor target minus the fixed term gives $y_a(x)$. The expression may be nonlinear in the state and may contain multiple nonlinear operations while the residual remains affine in θ.

Only one-step identities must be affine in θ; correctness of their compositions follows from the separately verified identities.

C.2 Proof of Theorem 4

Proof. Let $V_{\mathcal{Q}} = \{\theta : M_{\mathcal{Q}}\theta = y_{\mathcal{Q}}\}$. Whenever this affine set is nonempty, the proposed vector belongs to it. Suppose the identity verifier returns the equation $u_a(x)^{\mathsf{T}}\theta = y_a(x)$ violated by that proposal. The equation is not true on all of $V_{\mathcal{Q}}$. Its intersection with $V_{\mathcal{Q}}$ is therefore either empty or an affine set of strictly smaller dimension. There can be at most p_S strict dimension decreases while the solution set remains nonempty. At dimension zero, the next call either verifies the unique remaining proposal or returns an equation making the system inconsistent. Thus at most $p_S + 1$ verification calls occur.

If a coefficient vector satisfying all identities in (1) exists, it satisfies every returned equation, so the finite linear system can never be inconsistent. Termination therefore ends in successful verification. Conversely, a verified vector satisfies all required functional identities by the verifier specification. An inconsistent finite subsystem cannot be satisfied by any globally valid coefficient vector, so rejection is sound.

For the certificate size, an inconsistent system in p_S unknowns has an augmented matrix whose rank exceeds the coefficient rank. Select a basis of at most p_S coefficient rows. At least one further augmented row fails the corresponding affine consistency relation. These at most $p_S + 1$ rows are inconsistent. Equivalently, the right-hand side lies outside the column space of the coefficient matrix, so there is a left-null vector λ whose inner product with the right-hand side is nonzero. This is exactly $\lambda^{\mathsf{T}} M_{\mathcal{Q}} = 0,\ \lambda^{\mathsf{T}} y_{\mathcal{Q}} \neq 0$. Exact elimination can produce such a vector. For $p_S = 0$, the sole coefficient vector is the empty tuple. One verification call either accepts it or returns a scalar equation $0 = y_a(x) \neq 0$, which is already a rejection certificate. □

A vector witness may add several independent equations at once and can only reduce the number of verification calls. Acceptance certifies the returned coefficient vector, not every vector remaining in the accumulated affine solution set.

C.3 Proof of Theorem 5

Proof. By Theorems 2 and 3, every preprocessing deletion and every stored pair clause is valid for all admissible selectors. Any exact executable template is admissible, so preprocessing cannot remove a feasible exact program. The inner loop decides each unpruned template under its stated verifier assumptions. Since $\mathscr{S}$ is finite, the outer loop terminates. If it accepts a template, every strictly cheaper template has already been ruled out by a valid rejection certificate or excluded by a sound selector constraint. The returned cost is therefore minimal. If no template is accepted, every member has a certificate ruling it out. Summing the per-template verification bounds gives the displayed total bound.

For the effective specialization, every violation of a fiber identity is an existential semialgebraic condition in two source states. For example, for an edge $e : v \to w$ and a chosen successor coordinate j, it has the form

$$x, x' \in X_v,\ \alpha_v^J(x) = \alpha_v^J(x'),\ [\alpha_w^J(F_e(x))]_j \neq [\alpha_w^J(F_e(x'))]_j. \tag{6}$$

The response case is analogous. There are finitely many such coordinate tests. Real quantifier elimination decides emptiness and provides real algebraic points when a counterexample exists.

Coefficient synthesis can be initialized with any algebraic vector, such as zero. Given algebraic counterexamples, all entries of the accumulated finite linear system are algebraic. Gaussian elimination over the real algebraic numbers decides consistency and returns an algebraic solution whenever the system is consistent over $\mathbb{R}$. For such a proposed vector, the set $\{x \in D_a : R_a(x;\theta) \neq 0\}$ is semialgebraic over the real algebraic numbers. Its emptiness and an algebraic witness are again decidable. By induction, every counterexample, coefficient-system entry, and proposed coefficient vector remains real algebraic. The termination bound therefore gives an effective decision procedure even when a real solution to the universal system exists but is not given in advance. □

C.4 Semialgebraic primitives and checkable rejection

Polynomial syntax is sufficient for the effective theorem but not necessary. A fixed primitive may instead be given by a semialgebraic graph with a unique real output on its declared input domain,

described with real algebraic constants. A positive square root, for example, can be represented by a quadratic equation together with a nonnegativity condition. Such a primitive is a fixed operation, not an unknown function.

Corollary 14 (Exact synthesis with semialgebraic primitives)**.** *The effective conclusion of Theorem 5 remains valid when the supplied functions and templates have single-valued semialgebraic graphs over the real algebraic numbers, provided the dependence of the templates on the unknown coefficients remains affine. Domain and branch conditions are included in each verification query.*

The proof introduces variables for the intermediate primitive values and applies quantifier elimination to the graph constraints. At algebraic source points, these intermediate values are algebraic, so the resulting finite coefficient systems remain effective. The extension covers the positive normalization denominator of Section 7.2. It does not cover a general exponential primitive.

C.5 Proof of Corollary 14

Proof. For each primitive in the fixed expression graph, let a real variable represent its output, with its inputs and output constrained by the corresponding semialgebraic graph. Composition is represented by conjunction and existential quantification over these intermediate variables. Because each graph is single-valued on the declared domain, the resulting constraints describe the corresponding function with the specified branch. They therefore do not admit an alternative value for a square root or any other primitive. Thus, when the proposed coefficient vector is algebraic, every fiber violation and every single-state identity violation is a semialgebraic condition over the real algebraic numbers.

Real quantifier elimination decides these conditions and produces algebraic source witnesses when they are satisfiable. Evaluating a single-valued semialgebraic function at algebraic arguments gives algebraic outputs. Its graph fiber is a singleton semialgebraic set over the real algebraic numbers. Such a nonempty semialgebraic set contains an algebraic point, which in this case must be the unique output. Consequently all coefficient rows and target values at returned points are algebraic. Exact elimination therefore yields algebraic coefficient vectors for subsequent verification. The rank-decrease proof of Theorem 4 is unchanged because affine dependence on the coefficients is retained. The finite outer search and its cost proof then apply without modification. □

A primitive with several graph branches is allowed only after the intended branch has been specified so that the function is single-valued. A graph describing both signs of a square root would not satisfy the premise. Real algebraic constants can be represented effectively, for example by a defining integer polynomial together with an isolating interval. The corollary makes no claim for arbitrary uncomputable real constants.

C.6 An explicit formula-level rejection certificate

The following example gives an explicit formula-level rejection certificate. On $[-1, 1]$ let the semantic state be $s(x) = x^3$ and the response $r(x) = x$. There is no state-pair obstruction because s is injective, but no affine readout $o(z; \theta) = \theta_1 + \theta_2 z$ can represent the response exactly. Evaluations at $x = 0, 1, 1/2$ impose $\theta_1 = 0,\ \theta_1 + \theta_2 = 1,\ \theta_1 + \theta_2/8 = 1/2$, and multiplying by $-7, -1, 8$ and summing gives $0 = 3$. The certificate rejects the affine readout rather than the semantic state. A cube-root primitive would express the response exactly. Every equation in the certificate is obtained by exact evaluation of the target relation, so the contradiction does not depend on an optimizer. Appendix I gives the full polynomial-language obstruction.

C.7 A finite verifier for polynomial box domains

The general semialgebraic result is deliberately broader than necessary in the full-dimensional polynomial case. The following elementary specialization gives an explicit identity verifier without an abstract equality oracle.

Proposition 15. *Suppose a scalar residual is polynomial in n source coordinates, with degree at most D in each coordinate, on a box with nonempty interior. Choose $D + 1$ distinct rational nodes in the interior of each coordinate interval. The residual vanishes on the box if and only if it vanishes on the Cartesian product of these node sets.*

Proof. A univariate polynomial of degree at most D that vanishes at $D+1$ distinct points is zero. Apply this fact to the first variable with all other variables fixed at their grid nodes. Every coefficient polynomial in the remaining variables vanishes at the remaining tensor grid. Induction over the number of variables shows that all coefficients vanish. The converse is immediate. □

Thus one can compute a degree bound from the known expression, evaluate the finite grid exactly, and return the first nonzero value as a counterexample. This can require $(D+1)^n$ evaluations; it is an implementation of the verifier, not a polynomial-time guarantee. Symbolic coefficient comparison often avoids the tensor grid. Both approaches establish a functional identity within a degree-bounded polynomial class, unlike unrestricted finite testing of analytic functions.

C.8 Certificates and incomplete solvers

A successful exact output records the selected program and its functional-identity evidence, while rejection records an applicable state-pair certificate or a finite inconsistent coefficient subsystem. A timeout or uncertified floating-point inconsistency cannot establish infeasibility or minimum cost.

D Least admissible selectors and their structural condition

We regard auxiliary coordinates as the finite set $\mathcal{I} = \{(v, j) : v \in \mathcal{V},\ j \in L_v\}$ and identify a selector J with $\widehat{J} = \{(v, j) : j \in J_v\} \subseteq \mathcal{I}$. The finite target list $\mathcal{T}_v$ contains one occurrence of every scalar coordinate of $s_w \circ F_e$ and $f_{w,k} \circ F_e$ for each outgoing edge $e : v \to w$, together with r_v if $v \in \mathcal{O}$. Occurrences may be kept separately even when their functions coincide. For an optional occurrence $q = f_{w,k} \circ F_e$, the destination coordinate is $t(q) = (w, k)$. A target with $\mathcal{D}_q \neq \varnothing$ is called supported. These definitions do not impose continuity or an expression class on a factorization.

Lemma 16 (Determining sets)**.** *The family $\mathcal{D}_q$ is upward closed. If it is nonempty, then $A_q := \bigcap_{I \in \mathcal{D}_q} I = \{j \in L_v : L_v \setminus \{j\} \notin \mathcal{D}_q\}$. The following are equivalent: $\mathcal{D}_q$ is closed under pairwise intersection; $A_q \in \mathcal{D}_q$; and $\mathcal{D}_q$ has a unique inclusion-minimal member. When these hold, $\mathcal{D}_q = \{I \subseteq L_v : A_q \subseteq I\}$.*

Proof. If $I \in \mathcal{D}_q$ and $I \subseteq I'$, agreement on the semantics and the features in I' implies agreement on those in I, and therefore implies agreement on q. This proves upward closure. A coordinate j does not belong to the intersection of $\mathcal{D}_q$ exactly when some determining set excludes it. By upward closure, this is equivalent to $L_v \setminus \{j\} \in \mathcal{D}_q$, proving the displayed identity.

Because the ground set L_v is finite, a nonempty family $\mathcal{D}_q$ is finite. Pairwise intersection closure therefore implies that the intersection of all its members belongs to the family. Conversely, if $A_q \in \mathcal{D}_q$, upward closure and the definition of intersection give $I \in \mathcal{D}_q \iff A_q \subseteq I$, which implies pairwise intersection closure and makes A_q the unique inclusion-minimal member. If there is a unique inclusion-minimal member A, every $I \in \mathcal{D}_q$ contains an inclusion-minimal member of $\mathcal{D}_q$, obtained by deleting elements while possible. Uniqueness gives $A \subseteq I$. As A itself belongs to the family, its intersection is exactly A, proving the remaining implication. The argument includes $L_v = \varnothing$: a nonempty determining family is then $\{\varnothing\}$ and its least support is empty. □

Proof of Theorem 6. By the fiber factorization criterion in Theorem 2, a selector is admissible exactly when every mandatory target $q \in \mathcal{T}_v$ has $J_v \in \mathcal{D}_q$, and every optional target with $t(q) \in \widehat{J}$ has $J_v \in \mathcal{D}_q$. A mandatory target with an empty determining family consequently rules out every selector. Assume none exists and let $\mathcal{F} \subseteq \mathcal{I}$ be the set of forbidden vertices in the theorem.

Lemma 16 changes the remaining conditions into the following three finite requirements: $\mathcal{B} \subseteq \widehat{J}$; every graph edge from a vertex of $\widehat{J}$ has its destination in $\widehat{J}$; and $\widehat{J} \cap \mathcal{F} = \varnothing$. Indeed, the first requirement imposes each mandatory support. An edge $t(q) \to (v, j)$ imposes the necessary source feature $j \in A_q$ whenever the destination feature is selected. When the optional target is unsupported, its destination feature cannot be selected, which is exactly the third requirement. These conditions are also sufficient, because each supported required target then has its whole least support in the selected source state.

Every admissible $\widehat{J}$ contains $\mathcal{B}$ and is closed under the directed edges. Induction on path length shows $\mathcal{R} \subseteq \widehat{J}$. Thus $\mathcal{R} \cap \mathcal{F} \neq \varnothing$ implies infeasibility. If $\mathcal{R} \cap \mathcal{F} = \varnothing$, the set $\mathcal{R}$ contains $\mathcal{B}$, is closed under every edge by the definition of reachability, and avoids $\mathcal{F}$. It satisfies all three requirements and is admissible. Because it is contained in every admissible selector, it is the unique least selector. Any distinct admissible selector strictly contains it and hence has greater cardinality. If a cost is coordinatewise nondecreasing, inclusion instead gives that its cost at $\mathcal{R}$ is no greater than at any admissible selector; cost minimizers need not be unique when some coordinates have zero cost. Cycles require no additional assumption: reachability uses finite paths, and edge closure already includes every successor of a reachable vertex. □

Proof of Proposition 7. Upward closure implies $\mathcal{D}_q = \varnothing$ exactly when $L_v \notin \mathcal{D}_q$, so the first scalar fiber query recognizes unsupported targets. For a supported target, the next N_v queries compute the intersection A_q by Lemma 16, whether or not the structural condition holds. The final query checks that this intersection determines q. By the same lemma, it succeeds exactly when the determining family is intersection-closed. A failed final query is a witness that the structural condition fails; it does not by itself prove that the discovery problem is infeasible. A successful final query certifies the least support. If $N_v = 0$, the same full-set query can also serve as the final query, and the stated upper bound remains valid.

There are at most $N_v + 2$ queries for each target in $\mathcal{T}_v$. Once the supports are known, building the graph and computing reachability take $O\Big(|\mathcal{I}| + \sum_v |\mathcal{T}_v| + \sum_{q:\mathcal{D}_q \neq \varnothing} |A_q|\Big)$ elementary operations. For explicitly tabulated finite state domains with exact rational entries, one determinacy query can enumerate all state pairs and compare the semantic values, selected feature values, and target values. The number and bit complexity of these comparisons are polynomial in the table length. Evaluating the supplied transitions and target functions is included in the table representation; no polynomial-time claim is made for succinctly represented functions or continuous-domain verification. □

Proposition 17 (Product feature images)**.** *For $a \in s_v(X_v)$, define $Y_{v,a} = \{(f_{v,j}(x))_{j \in L_v} : x \in X_v,\ s_v(x) = a\}$. Suppose for every v and every $a \in s_v(X_v)$ this nonempty feature image satisfies $Y_{v,a} = \prod_{j \in L_v} \pi_j(Y_{v,a})$. Then every nonempty determining family $\mathcal{D}_q$ is intersection-closed, for every target function $q : X_v \to \mathbb{R}$.*

Proof. Take $I, K \in \mathcal{D}_q$ and states $x, x' \in X_v$ that have the same semantic value a and agree on all features in $I \cap K$. Specify a feature vector by taking its coordinates in I from x, its coordinates in $K \setminus I$ from x', and its remaining coordinates from x. Agreement on $I \cap K$ makes these assignments consistent. Every assigned value belongs to the appropriate coordinate projection of $Y_{v,a}$, so the product condition gives a state $z \in X_v$ with this feature vector and semantic value a. The states x, z agree on all features in I, while z, x' agree on all features in K. Therefore $q(x) = q(z) = q(x')$, proving $I \cap K \in \mathcal{D}_q$. This argument does not require the full feature extractor to be injective. For an empty library the image is the singleton containing the empty tuple, as is the product indexed by the empty set, and the claim still holds. □

The product condition is sufficient but not necessary and may fail when candidate functions substitute for one another; Proposition 7 remains exact in that case.

A nonlinear cyclic example. Take one type with $X = [0,1]^3$, semantic coordinate $s(x) = x_0$, response $r = s$, candidate features $f_1(x) = x_1^2$, $f_2(x) = x_2^2$, and the self-loop $F(x_0, x_1, x_2) = \big((x_0 + x_1^2)/2,\ x_1 x_2,\ x_1\big)$. Every coordinate of $F(x)$ lies in $[0,1]$, so this is a transition from X to itself. For each semantic value, the joint candidate image is $[0,1]^2$, satisfying Proposition 17. The least supports are $A_{s \circ F} = \{1\},\ A_r = \varnothing,\ A_{f_1 \circ F} = \{1,2\},\ A_{f_2 \circ F} = \{1\}$. For the optional targets these follow from $f_1 \circ F = f_1 f_2$ and $f_2 \circ F = f_1$. Each stated nonempty support is necessary: to test dependence on one feature, hold the other feature at one and vary the tested feature between zero and one; for $s \circ F$ hold x_0 fixed and vary x_1. The mandatory semantic successor first requires feature 1, whose update then requires feature 2; updating feature 2 again requires feature 1. The unique least selector is consequently $\{1,2\}$, with exact update $g(z_0, z_1, z_2) = \big((z_0 + z_1)/2,\ z_1 z_2,\ z_1\big)$. The response alone needs no auxiliary coordinate, whereas reproducing all subsequent semantic states needs both. The example therefore uses the interaction between selection and recursive update requirements.

If a third feature $f_3(x) = x_2^4$ is supplied, the joint feature image no longer has the product form. Both $\{1, 2\}$ and $\{1, 3\}$ are admissible: for the latter, use $g(z_0, z_1, z_3) = \big((z_0 + z_1)/2,\ z_1\sqrt{z_3},\ z_1^2\big)$ on its extractor image. Their intersection $\{1\}$ is not admissible, since states $(0, 1, 0)$ and $(0, 1, 1)$ agree on s, f_1 but have different values of $f_1 \circ F$. Each of the two selectors is inclusion-minimal: without feature 1, the semantic successor is not determined, and without its other coordinate, the update of feature 1 is not determined. This is a statement about unrestricted deterministic admissibility; a prescribed program syntax may exclude the square-root update.

E A COMPLEXITY BOUNDARY FOR TRANSITION CLOSURE

Proof of Theorem 8. Reduce from Vertex Cover (Karp, 1972). Let $H = (V_H, E_H)$ be a finite simple undirected graph. Introduce a label set $\mathcal{G} = \{A, B\} \cup \{P_e, Q_e : e \in E_H\}$ and put $d = |\mathcal{G}|$. For $g \in \mathcal{G}$, let $\mathbf{e}_g \in \mathbb{R}^d$ be the corresponding standard basis vector. The low-level state is $(a, t) \in \mathbb{R}^{d+1}$, with domain

$$X = \{(\mathbf{e}_g, t) : g \in \mathcal{G},\ 0 \le t \le 1\}. \tag{7}$$

This is a nonempty compact rational semialgebraic set. A polynomial-size description is given by $a_g(a_g - 1) = 0$ for every $g \in \mathcal{G}$, $\sum_{g\in\mathcal{G}} a_g = 1$, and $0 \le t \le 1$. Let Π be the permutation matrix swapping A with B and P_e with Q_e for every edge e. The sole transition is

$$F(a, t) = (\Pi a, t). \tag{8}$$

It is linear, maps X onto itself, and satisfies $F^2 = \mathsf{I}$. The semantic map and response are

$$s(a, t) = (a, a_B t),\ r(a, t) = 0. \tag{9}$$

Index the candidate library by $\{0\} \cup V_H$, which can be relabeled as $\{1, \dots, |V_H| + 1\}$ without changing the instance, and define

$$f_0(a, t) = t\left(a_A + \sum_{e\in E_H} a_{Q_e}\right),\ f_i(a, t) = t \sum_{e\in E_H : i\in e} a_{P_e}\ (i \in V_H). \tag{10}$$

All supplied functions have rational coefficients and degree at most two. The graph incidence lists give a polynomial-size sparse description. For every vertex incident to an edge, f_i is not the restriction of an affine function of (a, t) to X: its slope in t is one on an incident P_e segment and zero on the A segment, whereas an affine function of (a, t) has the same slope in t on every segment. The same argument applies to f_0, using the A and B segments. Isolated vertices can be removed from the source graph without changing its minimum vertex-cover size.

We classify every pair with equal semantics. The semantic coordinates a force the pair to have the same label. On the B segment, the last semantic coordinate is t, so equal semantics imply equal low-level states and produce no clause. On any other segment, two states with distinct values of t have equal semantics. For these pairs, every nonconstant candidate or successor quantity is equal to t, so its values differ precisely when the corresponding coefficient is one. The complete list of nontrivial clauses is as follows.

On the A segment, only f_0 varies with t, while the successor lies on B and has last semantic coordinate t. This yields the mandatory clause b_0. Every optional successor coordinate vanishes on B, so the A segment yields no further clause. On P_e, where $e = \{u, v\}$, exactly f_u and f_v vary with t. The successor lies on Q_e, where exactly f_0 varies with t; the successor semantics are constant in t. This yields $\neg b_0 \vee b_u \vee b_v$. On Q_e, exactly f_0 varies with t, while the successor lies on P_e and exactly f_u and f_v vary with t. This yields $\neg b_u \vee b_0$ and $\neg b_v \vee b_0$. Responses give no clauses because they are constant. Pairs with identical values of t give no clause. Consequently the complete nontrivial state-pair condition is

$$b_0 \ \wedge \bigwedge_{\{u,v\}\in E_H} \left[(\neg b_0 \vee b_u \vee b_v) \wedge (\neg b_u \vee b_0) \wedge (\neg b_v \vee b_0)\right]. \tag{11}$$

Theorem 2 therefore gives

$$J \text{ is admissible } \iff 0 \in J \text{ and } J \cap V_H \text{ is a vertex cover of } H. \tag{12}$$

In particular, all mandatory conditions are satisfied by the singleton selector $\{0\}$. The need to cover edges arises entirely from the successor requirements for this selected coordinate. The minimum number of auxiliary coordinates is exactly $1 + \tau(H)$, where $\tau(H)$ is the minimum vertex-cover size. The decision threshold k for Vertex Cover becomes $k + 1$ for the constructed instance. This is a polynomial-time many-one reduction and proves the stated NP-hardness. Replacing F by the identity makes every selected coordinate update by identity; the empty selector is then admissible because its semantics also update by identity and its response is constant.

For completeness, admissibility in this construction does not require expressive update formulas. Given a vertex cover C, write a retained state as $(a, q, z_0, (z_i)_{i\in C})$, where $q = a_B t$ and $z_i = f_i(a, t)$. For each edge e, choose an endpoint $c(e) \in C \cap e$. On the image of the extractor, define

$$a' = \Pi a,\ q' = a_A z_0,\ z'_0 = q + \sum_{e\in E_H} a_{P_e} z_{c(e)},\ z'_i = z_0 \sum_{e\in E_H : i\in e} a_{Q_e}\ (i \in C). \tag{13}$$

These are branch-free polynomials of degree at most two. On A, $z_0 = t$ and the formulas produce the semantic coordinate $q' = t$ at B. On B, $q = t$ and they produce $z'_0 = t$ at A. On P_e, the chosen endpoint feature equals t, producing $z'_0 = t$ at Q_e; all selected successor vertex features vanish. On Q_e, $z_0 = t$, producing exactly the selected endpoint features at P_e; the successor root feature vanishes. In every case the other displayed quantities have the required zero values. This verifies all transition identities on X; the readout is the constant zero polynomial. Thus the same reduction on the segment domain remains valid if admissibility is additionally required to have polynomial updates of degree at most two.

The polynomial hardness construction also extends to the full-dimensional rational box $\widetilde{X} = [0, 1]^{d+1}$, with exactly the same functions defined in (8), (9), and (10). The permutation transition preserves this box. The witnesses at $a = \mathbf{e}_A$ still force the root coordinate, and those at $a = \mathbf{e}_{P_e}$ still force the selection of an endpoint for every edge. Thus every admissible selector must have the form in (12). Conversely, suppose $0 \in J$ and $C = J \cap V_H$ is a vertex cover. Consider two states with the same extracted value. Their a coordinates agree because a is included in the semantics. If their t coordinates agree, the states are identical. If they differ, equality of the last semantic coordinate, the root coordinate, and every selected vertex coordinate gives

$$a_B = 0,\ a_A + \sum_{e\in E_H} a_{Q_e} = 0,\ \sum_{e\in E_H : i\in e} a_{P_e} = 0\ (i \in C). \tag{14}$$

All entries of a are nonnegative. The first two equalities therefore imply $a_A = a_B = a_{Q_e} = 0$ for every edge. Each edge has an endpoint in C, so the last family also implies $a_{P_e} = 0$ for every edge. Hence $a = 0$. At such states all semantic and candidate coordinates, before and after F, are zero, so every required successor value agrees. The response is constant. Theorem 2 proves admissibility and establishes (12) on $\widetilde{X}$, proving NP-hardness on the full-dimensional box. The root alone still satisfies every mandatory condition: if equal root values occur with different t, then $a_A + \sum_e a_{Q_e} = 0$, which implies that the successor semantic coordinate $a_A t$ agrees. Thus this extension preserves the distinction between the forced root and the combinatorial optional-update obligations.

The quadratic formulas in (13) use the one-hot domain and are not asserted on the full box. Deterministic high-level updates on the box can instead be given explicitly. Define $D(a) = a_B + a_A + \sum_e a_{Q_e} + \sum_{i\in C} \sum_{e\in E_H : i\in e} a_{P_e}$ and $U_C = q + z_0 + \sum_{i\in C} z_i$. On the extractor image, $U_C = tD(a)$. When $D(a) > 0$, recover $t = U_C/D(a)$ and substitute this value into the supplied successor functions. When $D(a) = 0$, nonnegativity and the cover property imply $a = 0$, so all successor semantic and retained auxiliary values are zero. These maps are well defined on the extractor image. This piecewise rational construction is used only to exhibit deterministic updates, not as a claim that these box instances are executable in a prescribed polynomial syntax.

To obtain a finite rational instance, replace $[0, 1]$ in (7) by $\{0, 1\}$. There are $2d = 4 + 4|E_H|$ states. The transition still permutes these states involutively, and every semantic, feature, and response value is zero or one. The preceding pair classification remains exact because each nontrivial pair is witnessed by the two endpoints of its segment. Explicitly writing the rational states and all function evaluation tables takes polynomial space and time. Hence the restricted finite-domain problem is NP-hard.

For membership in NP, consider any instance given by finite rational state lists and explicit transition, semantic, candidate, and response tables. A certificate specifies the selected indices at each type. Its

size is polynomial in the input, and its cardinality can be checked directly. Enumerate every source-state pair at every type. Whenever their selected extractors agree, compare all mandatory and selected optional successor coordinates for every outgoing transition, and compare every specified response. Exact equality of listed rational values is decidable in polynomial bit time, and the number of such comparisons is polynomial in the explicit input length. The factorization criterion in Theorem 2 makes these checks necessary and sufficient, proving membership in NP. This NP-membership argument is confined to the explicit finite representation; it does not assert membership in NP for arbitrary compact semialgebraic domains and polynomially represented functions.

Finally suppose an explicit collection of state-pair clauses has distinguishing sets of size at most one. A mandatory clause is either a positive unit b_j or the empty disjunction, which immediately proves infeasibility. An optional clause is either a negative unit $\neg b_k$ or an implication $b_k \Rightarrow b_j$. Let P be the indices appearing in positive units and Z those appearing in negative units. Form a directed graph with an arc $k \to j$ for each implication, and let R be the vertices reachable from P, including P. If $R \cap Z \neq \varnothing$, no satisfying selector exists because every satisfying selector must contain R. Otherwise the selector R satisfies every positive and negative unit and every implication, and it is contained in every satisfying selector. It is therefore the unique least admissible selector and in particular a minimum-cardinality one. Graph construction and reachability take time linear in the explicit clause representation. In the NP-hard construction described by (11), all mandatory clauses are units and all distinguishing sets have size at most two. This proves the claimed boundary. □

F Complexity of selecting auxiliary variables

Theorem 18 (Hardness of auxiliary-state selection)**.** *Deciding whether a library contains an admissible selector with at most k auxiliary coordinates is NP-hard, even with one type, an identity transition, a compact box domain, one nonconstant linear semantic coordinate, and one linear response. Under the same identity-transition restriction, with rational affine semantic maps, candidate features, and responses on a full-dimensional rational box, the decision problem with affine high-level formulas is NP-complete.*

Proof of Theorem 18. Consider an instance of Exact Cover by 3-Sets, with universe $U = \{1, \dots, 3k\}$ and triples $T_1, \dots, T_q \subseteq U$. The question is whether k of these triples are pairwise disjoint and cover U. We reduce from Exact Cover by 3-Sets.

The constructed instance has one type with domain $X = [-1, 1]^{3k+1}$ and the identity transition. Its semantic map, candidate features, and response are

$$s(x) = x_0,\ f_j(x) = \sum_{i \in T_j} x_i,\ r(x) = x_0 + \sum_{i=1}^{3k} x_i. \tag{15}$$

All coefficients are integers of bounded size, and the construction has size polynomial in the instance. Suppose a selector J of size at most k admits a deterministic output map. If an index $i \in U$ belongs to none of the selected triples, then x_i can vary while every other coordinate is held fixed. The semantic variable and every selected feature remain unchanged, but r changes. This contradicts output factorization. Thus the selected triples cover all $3k$ elements. Since each selected triple has size three, at most k selected triples can cover all $3k$ elements only when exactly k triples are selected and they are pairwise disjoint. Thus they form an exact cover.

Conversely, for an exact cover J, the extracted state has the identity update and the affine response

$$o(z) = z_0 + \sum_{j \in J} z_j, \tag{16}$$

where the optional coordinates are indexed by their chosen triples for this display. This reproduces r exactly. Therefore, the existence of a selector of size at most k is equivalent to the existence of an exact cover. The necessity argument allows arbitrary deterministic readouts, so the hardness does not arise from a restriction on the degree of the output formula.

For membership in NP in the affine rational-input subcase, a certificate consists of the selected indices. Every required affine identity on a full-dimensional rational box is an equality of finitely

many rational coefficients. Gaussian elimination decides feasibility in polynomial time. Equivalently, a consistent rational linear system has a rational basic solution with polynomial encoding length. Thus the restricted decision problem is in NP and is NP-complete. □

The input to this complexity statement is the library and the permitted affine formula class, not an explicit list of all 2^q selectors. This does not conflict with an enumeration procedure that is polynomial in the length of an explicitly listed template family. Such a family may itself be exponentially large in the library representation used by the hardness result. No lower bound on the runtime of every individual Transformer instance is asserted.

G Finite numerical synthesis with a positive margin

G.1 Effective assumptions

Fix one template and omit its subscript when this causes no ambiguity. The coefficient set is $K = [-B, B]^p$, with positive rational B. For each constraint, Ω_ν is a compact rational box. Evaluation of $h_\nu(t;\theta)$ at rational t, θ returns rational lower and upper bounds with arbitrarily prescribed width. A supplied effective uniform modulus allows construction of a finite net on which the variation in h_ν is at most any prescribed positive rational tolerance, uniformly over $\theta \in K$. The finite family of moduli can be replaced by their maximum.

The coefficient-sensitivity condition is

$$|h_\nu(t;\theta) - h_\nu(t;\theta')| \le L \left\|\theta - \theta'\right\|_\infty ,\ L \ge \sup_{\nu,t} \left\|a_\nu(t)\right\|_1 ,\ L > 0, \tag{17}$$

where a positive rational L is chosen; it may strictly exceed the supremum, including when every coefficient row is zero. For general compact domains, the same construction requires finite nets of computable points lying in the domain together with certified covering radii. Compactness without these effective data is not the algorithmic assumption.

Polynomial expressions on boxes satisfy these requirements with coefficient-based derivative bounds. Fixed compositions of computable smooth primitives also do so when range and derivative bounds are supplied and all divisions, logarithms, and square roots stay in certified valid domains. Positive normalization stabilizers and positive softmax denominators are relevant examples, but a bound on the whole composition must still be obtained. Merely observing bounded values at finitely many activations does not establish this condition.

G.2 A terminating sample solver

Given a finite sample set, construct a rational tensor grid $G \subseteq K$ with covering radius at most $\delta/(8L)$ in the maximum norm. Subdividing each coordinate interval into pieces of width at most $\delta/(4L)$ suffices. At each grid point, evaluate every sampled constraint in a rational interval of width at most $\delta/16$. Return any grid point for which every upper endpoint is at most $\delta/4$. If no such point exists, return an infeasibility certificate for the original finite sample system.

This procedure is sound because each accepted upper endpoint bounds the corresponding exact value. To prove the negative claim, suppose a coefficient vector θ_* satisfies every sampled inequality with right-hand side zero. Some grid point q has $\left\|q - \theta_*\right\|_\infty \le \delta/(8L)$. By (17), every exact sampled value at q is at most $\delta/8$. Its interval upper endpoint is therefore at most $3\delta/16 < \delta/4$, so the grid point would have been accepted. This is a contradiction.

There is also an explicit finite certificate for rejection. For every grid point choose a sampled constraint whose upper endpoint exceeds $\delta/4$. Its lower endpoint exceeds $3\delta/16$. Throughout the coefficient ball of radius $\delta/(8L)$ around that grid point, the same constraint exceeds $\delta/16 > 0$. These balls cover K. Their interval bounds and the coefficient Lipschitz bound therefore exclude every coefficient vector satisfying the original constraints. No exact comparison between transcendental constants is used.

The grid is used to establish completeness, not a recommended exhaustive numerical implementation. A certified linear-inequality solver or a branch-and-bound procedure can replace it if it satisfies

the same soundness and completeness requirements at the stated tolerances. A heuristic optimizer without those guarantees can propose candidates, but cannot justify rejection or the complete cost comparison.

G.3 A TERMINATING VERIFIER

An empty constraint family is accepted immediately. Otherwise, for a fixed rational proposal θ, use the effective modulus to construct finite nets on all constraint domains such that every domain point is within variation $\delta/16$ of some net point. Evaluate each constraint at its net points in intervals of width at most $\delta/16$. Let U_* be the largest upper endpoint across this finite collection. If $U_* + \delta/16 \le \delta$, accept: every full-domain constraint is then at most δ.

Otherwise choose the point attaining U_*. Its lower endpoint is strictly greater than $14\delta/16$, hence greater than $\delta/2$. It is a certified counterexample of the required kind. All comparisons involve rational interval endpoints, so they terminate even when an exact function value happens to lie on a threshold. The construction uses one sufficiently fine finite cover; adaptive subdivision may be more economical but is not needed for the existence proof.

G.4 PROOF OF THEOREM 9

Proof. The preceding constructions establish the existence of finite subroutines with the stated specifications. Every proposed coefficient vector satisfies all previous sampled constraints with value at most $\delta/4$. If the verifier rejects proposal $\theta^{(i)}$, it supplies (ν_i, t_i) with

$$h_{\nu_i}(t_i; \theta^{(i)}) > \delta/2,\ h_{\nu_i}(t_i; \theta^{(j)}) \le \delta/4\ (j > i). \tag{18}$$

Equation (17) implies that any two distinct proposals have maximum-norm distance greater than $\delta/(4L)$. This includes the final accepted proposal, if any.

Let $\ell_{\mathrm{pack}} = \delta/(4L)$. The open cubes of maximum-norm radius $\ell_{\mathrm{pack}}/2$ centered at the proposals are disjoint. Each has volume ℓ_{pack}^p and lies in $[-B - \ell_{\mathrm{pack}}/2, B + \ell_{\mathrm{pack}}/2]^p$. If there are q_* proposals, comparison of volumes gives

$$q_* \ell_{\mathrm{pack}}^p \le (2B + \ell_{\mathrm{pack}})^p,\ q_* \le \left(1 + \frac{8BL}{\delta}\right)^p. \tag{19}$$

An infinite run would contain finite initial segments exceeding this bound, which is impossible. The loop must therefore reach verifier acceptance or sample-solver rejection.

Acceptance certifies the full relaxed inequalities by the verifier specification. Sample-solver rejection proves that no coefficient vector satisfies all sampled inequalities before relaxation and hence no coefficient vector satisfies every original requirement. Conversely, any coefficient vector satisfying all original constraints also satisfies every sampled system before relaxation, so the sample solver cannot reject while such a vector exists. Termination must then yield a program satisfying the relaxed constraints. This proves all alternatives and the verification-call bound. When $p = 0$, the empty coefficient vector is the only proposal. An accepting verifier certifies it; a returned constraint exceeding $\delta/2$ immediately refutes feasibility for the original constraints. □

The bound concerns coefficient dimension, not the input-state dimension. The latter affects the size of the verification covers and can dominate all computation. The finite iteration bound is therefore not a practical runtime estimate.

G.5 TWO-TOLERANCE COST COMPARISON

Proposition 19. *For a finite cost-ordered template family satisfying Theorem 9, let S_* be a least-cost template with a coefficient vector satisfying every original constraint. The numerical outer loop returns a program satisfying the relaxed constraints uniformly at cost at most $c(S_*)$. If it rejects all templates, none has a coefficient vector satisfying the original constraints.*

Proof. Each fixed-template loop terminates. It cannot reject a template feasible for the original constraints. Consequently, if no earlier template is accepted, the loop for S_* must accept a program

satisfying the relaxed constraints. Every earlier accepted template has no greater cost. If all loops reject, their sound negative statements cover every member of the finite family. □

A template feasible only after relaxation may be rejected. Therefore the theorem does not compare the returned cost with the minimum cost under the relaxed constraints, and exhaustive numerical rejection does not establish nonexistence at the accepted error tolerance.

The exact greatest-selector preprocessing is not used automatically in this numerical outer loop. Its deletions are necessary for exact factorization, and a program with small nonzero approximation errors may legitimately retain a coordinate deleted by the exact procedure. Sound quantitative pair clauses from Proposition 10 can be used instead when their thresholds match the accepted error bounds.

H Domain certificates and finite execution

H.1 Proof of Corollary 11

Proof. For a scalar transition residual $R_{e,j}$, the pre-relaxation constraints are $R_{e,j} - \varepsilon_e \le 0$ and $-R_{e,j} - \varepsilon_e \le 0$. Relaxed feasibility gives $|R_{e,j}| \le \varepsilon_e + \delta$. Taking the maximum over coordinates gives $\left\| \alpha^J_{\mathrm{dst}(e)}(F_e(x)) - g^S_e(\alpha^J_{\mathrm{src}(e)}(x); \theta) \right\|_\infty \le \varepsilon_e + \delta$. The same argument yields the corresponding terminal-response bounds.

For domain invariance, relaxed feasibility of the two inequalities in (2) gives

$$\underline{z}_{\mathrm{dst}(e),j} \le g^S_{e,j}(z;\theta) \le \overline{z}_{\mathrm{dst}(e),j} \ (z \in Z_{\mathrm{src}(e)}). \tag{20}$$

All successor coordinates therefore belong to the destination box. Independently certified extraction containment places the initial abstract state in its declared box, and induction then shows that every subsequent high-level state remains in the corresponding declared box. Applying Proposition 19 to this combined constraint family gives the stated two-tolerance cost conclusion. □

The upper and lower box faces are part of the supplied template specification. They can be chosen separately for different selectors, since their dimensions differ. Discovery may enumerate a finite family of proposed boxes as part of its syntax. It does not synthesize arbitrary invariant sets without a specified representation and verification procedure. A feasibility comparison before relaxation then includes the choice of box as well as the choice of program.

Only extraction containment and low-level domain preservation are prerequisites. For a polynomial instance they can themselves be checked semialgebraically. For a smooth decoder on compact boxes, they can be certified using verified bounds on the output ranges when those bounds are sufficiently tight. A failed conservative enclosure is not a proof that actual containment fails. It should prevent a claimed certificate, not silently change the domain.

H.2 Proof of Proposition 12

Proof. Let $a_t = \alpha^J_{v_t}(x_t)$ denote the extracted state along the low-level run. By the certified extraction containment, $a_t \in Z_{v_t}$. Each abstract update maps its source box into the corresponding destination box, so every suffix function Q_t is well defined on Z_{v_t}. By definition, $Q_t = Q_{t+1} \circ g^S_{e_t}$. Therefore

$$\begin{aligned} r_{v_T}(x_T) - Q_0(a_0) &= r_{v_T}(x_T) - o^S_{v_T}(a_T) \\ &\quad + \sum_{t=0}^{T-1} \big[Q_{t+1}(a_{t+1}) - Q_{t+1}(g^S_{e_t}(a_t)) \big]. \end{aligned} \tag{21}$$

The equality follows by canceling consecutive $Q_t(a_t)$ terms. Each bracket has magnitude at most $\Lambda_{t+1}(\varepsilon_{e_t} + \delta)$ by the transition error and the Lipschitz bound on the destination box. The first term is bounded by $\varepsilon^{\mathrm{out}}_{v_T} + \delta$. Since $z_0 = a_0$, the abstract run gives $Q_0(a_0) = o^S_{v_T}(z_T)$, proving $|r_{v_T}(x_T) - o^S_{v_T}(z_T)| \le \varepsilon^{\mathrm{out}}_{v_T} + \delta + \sum_{t=0}^{T-1} \Lambda_{t+1}(\varepsilon_{e_t} + \delta)$.

If the initialized high-level state is another point of Z_{v_0} at distance at most η_0 from a_0, its suffix response differs by at most $\Lambda_0\eta_0$. This contributes an additional error of at most $\Lambda_0\eta_0$. For two runs, the triangle inequality together with the two individual response bounds gives the stated bound on their response difference. □

An initialization error bound requires the perturbed initial state to remain in the supplied box. If an implementation introduces further numerical errors in evaluating a high-level update, its executed map, rather than its exact real-arithmetic formula, must satisfy the invariance and error specification. An additional verified rounding-error allowance can be included, with the invariance requirements tightened accordingly. The current formulas do not certify floating-point execution merely because the mathematical program has been certified.

H.3 What a causal statement additionally requires

A semantic intervention is not certified merely because an executable state abstraction has been discovered. For an abstract intervention ι, a corresponding low-level implementation I must be supplied and separately verified to satisfy $\alpha^J \circ I = \iota \circ \alpha^J$, or an appropriate approximate analogue. When such an implementation is available, Proposition 12 can be applied before and after the intervention to bound the resulting response difference. The discovery procedure itself does not establish the existence of arbitrary semantic interventions or assign unique causal responsibility to individual retained variables.

I Worked algebraic examples

I.1 A selected program without linear closure of the full feature set

On $\mathbb{R}^3$, let $F(x, y, w) = (x + y^2, 2y, w^2)$, $s(x, y, w) = x$, and $r = x$. Consider optional functions $f_1 = y^2$ and $f_2 = \exp(w)$. For $J = \{1\}$,

$$\alpha^J(x, y, w) = (x, y^2),\ g(z_1, z_2) = (z_1 + z_2, 4z_2),\ o(z_1, z_2) = z_1. \tag{22}$$

Direct substitution verifies the two exact identities. The entire span of $1, x, y^2, \exp(w)$ is not invariant under composition with F. Otherwise $\exp(w^2)$ would be a linear combination of these four functions. Setting $x = y = 0$ would give $\exp(w^2) = a + b\exp(w)$ for all real w, contradicted by letting w tend to negative infinity. Thus a matrix representation that propagates every candidate feature does not exist, even though a subset of the features admits an exact executable representation.

The greatest admissible selector in this example can be larger than $\{1\}$. Indeed, $\exp(w)$ determines w and hence its own successor value. This does not invalidate the example. The first synthesis stage concerns information sufficiency; the second asks which selected state has a formula in the prescribed language and at what cost. It need not retain every coordinate that survives deletion.

I.2 A nonlinear update with no finite linear invariant lift

Let $X = [0, 1]$ and $F(x) = x^2$, with semantic map and response both equal to x. The extractor $\alpha(x) = x$ has the polynomial update $g(z) = z^2$ and identity readout. Suppose a finite-dimensional vector space W of functions on $[0, 1]$ contains x and is invariant under $f \mapsto f \circ F$. Invariance then implies by induction that $x^{2^t} \in W$ for every nonnegative integer t. These functions are linearly independent: a finite linear combination is a polynomial with distinct degrees, and a polynomial vanishing on an interval has all coefficients zero. This contradicts finite dimension.

The conclusion applies to a single reused transition and a time-homogeneous invariant linear function space. For a prescribed finite horizon, a stage-dependent lift may contain the finitely many powers required along that horizon. The example therefore distinguishes the present nonlinear program class from time-homogeneous finite-dimensional linear lifting, rather than from every finite-horizon representation.

I.3 Information sufficiency without polynomial expressibility

On $[-1, 1]$ with identity transition, let $s(x) = x^3$ and $r(x) = x$. Since $x \mapsto x^3$ is injective, there is no state-pair obstruction to a semantic-only abstraction. Its response is the real cube-root function. Suppose a polynomial P satisfied $P(x^3) = x$ on the interval. Expanding $P(x^3)$ yields only powers whose degrees are divisible by three, so the coefficient of x is zero. Equality on an interval would imply equality as polynomials, contradicting the unit coefficient of x on the right-hand side. This holds at every finite polynomial degree.

J Limits of finite observations and of syntax bounds

J.1 Sampling alone is not exact verification

For a nonempty open subset of $\mathbb{R}^n$, let $x^{(1)}, \ldots, x^{(q)}$ be any finite set of evaluation points and let d_* be any finite derivative order. The nonzero polynomial

$$P_*(x) = \prod_{j=1}^{q} \left\| x - x^{(j)} \right\|_2^{2(d_*+1)} \tag{23}$$

vanishes with all derivatives through order d_* at those points. It is not identically zero on the open set. For an adaptive finite observation procedure, run it on the zero function and use its resulting finite query set in this construction. The same responses cause the procedure to follow the same query sequence on P_*. Consequently, finite observations and unrestricted analyticity alone cannot implement the exact identity verifier.

This argument does not apply when a degree bound and a grid that uniquely determines polynomials of that degree are supplied, as in Proposition 15, or when the complete expression is available to a sound symbolic verifier. It also does not prevent approximate certification with a known modulus and a finite cover. The three kinds of evidence answer different questions.

J.2 A finite syntax is a substantive restriction

A finite syntax requires a finite primitive and extractor library together with finite structural bounds such as expression size, degree, and coefficient-sharing patterns. The $p_S + 1$ synthesis bound further requires affine dependence on the unknown coefficients, and every minimum-cost claim remains relative to this declared syntax.

J.3 Relation to empirical program discovery

Observed trajectories may propose selectors or programs, but all claims here rely on subsequent exact or certified numerical verification; heuristic failure does not establish infeasibility or optimality.

K Experimental setup and complete results

K.1 Hardware, arithmetic, and reporting

Experiment 3 used an NVIDIA GeForce RTX 4090 GPU for training; all certification, combinatorial search, and the remaining experiments ran on the CPU. Table 2 specifies the numerical representations; exact certification is not implemented by replacing FP32 with FP64. All reported results are taken from the recorded CSV outputs. Experiment 1 reports one run per algebraic instance, not a timing average over training seeds. Experiment 3 reports ten separately seeded tasks, each with one trained network. Means and sample standard deviations use the stated population of instances. A failed admissibility test has no greatest admissible selector: the remaining candidates in a failed run are an intermediate selection and are not included in statistics for greatest admissible selections. Solver decisions, symbolic identities, and numerical prediction errors are reported separately.

Table 2: Arithmetic used by the experiments. Decimal summaries and timing measurements are not inputs to exact acceptance or rejection decisions.

Experiment	Arithmetic and execution
1	Arbitrary-precision rational polynomial coefficients and linear systems (Python `Fraction`, SymPy `Rational`); Z3 nonlinear real-arithmetic queries. CPU execution, with no FP32/FP64 approximation in the logical tests.
2	Exact rational tolerances and grid spacings; arbitrary-precision integer point counts. Python `Decimal` with 80-digit precision for auxiliary time/storage estimates; floating-point logarithms and decimal CSV fields are summaries only. CPU execution.
3	FP64 training in PyTorch (`torch.float64`) on the RTX 4090; FP64 least-squares comparison in NumPy. Frozen weights are rounded to rational values, after which polynomial dependence, coefficient matching, and zero-residual tests use exact rational arithmetic on the CPU.
4	Finite tables contain only exactly representable zero–one values. Clause generation, selector verification, and combinatorial optimization use exact discrete comparisons; FP64 values are not used with numerical tolerances. CPU execution.
5	Rationalized Experiment 3 weights, exact rational polynomial arithmetic, Z3 nonlinear real-arithmetic fiber queries, and SymPy rational linear algebra. No retraining is performed. CPU execution.

K.2 Experiment 1: instances, synthesis, and query counts

The five instances have one supplied semantic coordinate and one scalar response. Candidate libraries and degree bounds were fixed before running the search. All transition maps have degree at most two, although candidate coordinates and substituted template residuals can have higher degree.

Square update. On $X = [0, 1]$, use $F(x) = x^2$, $s(x) = r(x) = x$, and an empty auxiliary library. The allowed update and output degrees are one, two, and three. The degree-one template is rejected and the degree-two template is accepted.

Affine readout. On $X = [-1, 1]$, use $F(x) = x$, $s(x) = x^3$, $r(x) = x$, and an empty auxiliary library. This instance permits only degree-one formulas in the extracted coordinate. The selector is admissible because x^3 is injective, but the affine response template is rejected. This is formula-class infeasibility, not missing state information.

Feature deletion. On $X = [-1, 1]^3$, use $F(x, y, w) = (x, w, w)$, $s(x, y, w) = r(x, y, w) = x$, and the single candidate y. Degrees one and two are permitted. Z3 returns the source pair $(0, 0, 1/2)$ and $(0, 0, 0)$: their extracted states agree when y is retained, but their successor y values differ. The algorithm deletes y and accepts the semantic-only identity program.

Greatest versus minimum selection. On $X = [-1, 1]^2$, use identity dynamics, $s(x, y) = x$, $r(x, y) = y$, and candidates y and y^3, with degrees one through three. The greatest admissible selection is $\{y, y^3\}$. All tested semantic-only templates fail; the first accepted template retains only y and has degree one.

Auxiliary update. Use the polynomial map $F(x, y) = (x + y^2, 2y)$, $s(x, y) = r(x, y) = x$, and the candidate y^2, with degrees one through three. Verification queries use the test box $[-1, 1]^2$. The accepted program is $(z_1, z_2) \mapsto (z_1 + z_2, 4z_2)$ with response z_1. The test box is not invariant. Since the residual polynomials vanish on a full-dimensional box, the identities extend to $\mathbb{R}^2$, where F is a valid self-loop. The run therefore certifies the global polynomial identities, not execution invariance of the test box or numerical multi-step error bounds.

Exact synthesis and certificates. For each selector, every scalar formula is a linear combination of all monomials up to its declared degree, including a constant term. Separate coefficient blocks are used for the semantic successor, each retained auxiliary successor, and the response; p_S counts all these unknown coefficients. Selectors are searched by auxiliary-coordinate count, with library order breaking selector ties and increasing degree within each selector. The returned programs minimize

the auxiliary-coordinate count among the declared templates; their reported degrees are the first feasible degrees tested for their selectors.

The coefficient solver uses exact rational linear algebra. Each counterexample-guided proposal satisfies the accumulated state equations. Nonzero polynomial residuals are checked with Z3 `nlsat`, with a 60-second timeout per scalar solver query, and rational witnesses are obtained from degree-dependent tensor grids as in Proposition 15. An identically zero coefficient representation is accepted directly. Fiber queries test the existence of two states agreeing under the current extractor but disagreeing on a required target. No timeout or unknown result is counted as a proof of infeasibility. Inconsistent coefficient systems are reduced by deleting unnecessary equations; the reported certificate size is the number of equations in the resulting subsystem. These are solver and arithmetic checks, not proof-assistant verification of the general theorems.

The CSV fiber counter records one call per scalar target. Theorem 3 instead counts a complete round that checks the current selection until a counterexample is returned or all obligations hold. Table 3 separates these quantities. Likewise, one identity-verifier call in Table 4 checks a complete coefficient proposal and may inspect several scalar residuals. It is this proposal count that is compared with $p_S + 1$.

Table 3: State-selection counts in Experiment 1. Empty sets denote semantic-only selections.

Instance	Candidates N	Rounds	Scalar queries	Greatest	Returned
Square update	0	1	2	$\varnothing$	$\varnothing$
Affine readout	0	1	2	$\varnothing$	None in syntax
Feature deletion	1	2	5	$\varnothing$	$\varnothing$
Greatest vs. minimum	2	1	4	$\{y, y^3\}$	$\{y\}$
Auxiliary update	1	1	3	$\{y^2\}$	$\{y^2\}$

Table 4: Every tested template in Experiment 1. Calls are identity-verifier proposal checks; bound is $p_S + 1$. Certificate size is zero for an accepted template and otherwise counts scalar equations in a reduced inconsistent subsystem.

Instance	Auxiliary set	Degree	p_S	Calls	Bound	Certificate	Outcome
Square update	$\varnothing$	1	4	3	5	3	Rejected
	$\varnothing$	2	6	6	7	0	Accepted
Affine readout	$\varnothing$	1	4	5	5	3	Rejected
Feature deletion	$\varnothing$	1	4	5	5	0	Accepted
Greatest vs. minimum	$\varnothing$	1	4	4	5	2	Rejected
	$\varnothing$	2	6	4	7	2	Rejected
	$\varnothing$	3	8	4	9	2	Rejected
	$\{y\}$	1	9	7	10	0	Accepted
Auxiliary update	$\varnothing$	1	4	2	5	2	Rejected
	$\varnothing$	2	6	2	7	2	Rejected
	$\varnothing$	3	8	2	9	2	Rejected
	$\{y^2\}$	1	9	8	10	0	Accepted

K.3 Experiment 2: exact cover-size calculation

This experiment calculates the size of a prescribed tensor grid; it does not run the numerical synthesis loop or evaluate a decoder on every grid point. We use the reference box $[-1, 1]^{d_{\mathrm{res}}+1}$ and the supplied reference modulus $\omega_{\mathrm{ref}}(h) = h$ in the maximum norm. The additional coordinate accounts for the extra state dimension suggested by the normalization example; the box is not asserted to be that example's physical domain. In particular, the reference modulus is not a bound derived from the actual denominator in (25).

To keep variation below $\delta/16$, a covering radius $\delta/16$ and grid spacing at most $\delta/8$ suffice under this reference modulus. The exact point count, denoted by N_{grid}, is

$$N_{\mathrm{grid}}(d_{\mathrm{res}}, \delta) = \left(\left\lceil \frac{16}{\delta} \right\rceil + 1 \right)^{d_{\mathrm{res}}+1}. \tag{24}$$

The two tolerances therefore require 1601 and 16001 points per coordinate, respectively. All powers and the comparison with the predeclared budget of 10^8 points are evaluated as arbitrary-precision integers. The scientific notation in Table 5 is rounded for display; full integer counts are retained in the CSV file. The first tested residual width exceeding the budget is four for both tolerances.

Table 5: Point counts for the specified reference grid. Every entry exceeds the declared 10^8-point budget.

d_{res}	State dimension	$\delta = 10^{-2}$	$\delta = 10^{-3}$
4	5	1.052×10^{16}	1.049×10^{21}
8	9	6.911×10^{28}	6.876×10^{37}
16	17	2.983×10^{54}	2.955×10^{71}
32	33	5.558×10^{105}	5.456×10^{138}

This is a limitation of the stated exhaustive grid and budget, not a hardware-measured threshold or a lower bound for all certified verifiers. For reference, the reported serial-time estimates assume 10^6 evaluations per second and the storage estimates assume eight bytes per stored scalar; neither quantity is a measured throughput or memory requirement for a streaming implementation. The result illustrates why finite termination alone does not imply a practical exhaustive implementation.

K.4 Experiment 3: training and discovery protocol

We use seeds 101, 211, 307, 401, 503, 601, 701, 809, 907, and 1009. Each seed determines a distinct masked teacher–student regression task, not another training replicate of a single fixed task. The state is $x = (x_0, \ldots, x_5) \in [-1, 1]^6$. Both teacher and student use two bias-free linear layers separated by the elementwise square, with six hidden units. A connection mask specifies which weights can be nonzero. The second-layer mask is diagonal. Each hidden unit receives one input with probability 0.75 and two distinct inputs with probability 0.25. These sets of input indices, called supports, are sampled without replacement; the semantic output's support is sampled from indices zero through four, while other supports may also include index five. Thus dependence on the unavailable coordinate can propagate through required auxiliary updates. The student shares the sampled masks with its teacher; the masks are not learned.

The task generator uses the instance seed plus 777. Its nonzero first-layer teacher weights are sampled from $\{\pm 1/4, \pm 3/8, \pm 1/2, \pm 5/8\}$; its diagonal output weights are sampled from $\{1/2, 5/8, 3/4\}$ and reduced if needed to respect the configured output bound $4/5$. No branch of the task generator conditions on a discovery result. The stored instance-structure CSV records all sampled supports.

For each instance, 4096 training states and 4096 validation states are sampled uniformly from the state box using the instance seed plus 10000. Targets are the teacher's six-dimensional outputs. The student is trained for 2500 full-batch Adam steps with learning rate 0.02, no weight decay, and mean squared error over all coordinates. First-layer weights are initialized from a zero-mean normal distribution with standard deviation 0.35; second-layer diagonal entries start at 0.5. The final iterate is used; validation data do not select the extractor, template, or training checkpoint.

Training on the RTX 4090 uses `torch.float64`, deterministic PyTorch algorithms, deterministic cuDNN settings, disabled cuDNN benchmarking, and disabled TF32. The cuBLAS workspace configuration is set to `:4096:8` before importing PyTorch. Exact certification is performed after training, with the effective masked weights rounded to multiples of $1/4096$. The implementation additionally applies its prescribed rational output-rescaling rule when the sum of the absolute polynomial coefficients for an output exceeds $4/5$. Discovery is performed on the resulting fixed rational-coefficient map. The coefficient-sum bound certifies that each output has absolute value at most $4/5$ throughout $[-1, 1]^6$, so this map preserves the declared state box.

The supplied semantic map is the coordinate projection $s(x) = x_0$, and the auxiliary candidates are the degree-one monomials x_1, x_2, x_3, x_4. Coordinate x_5 is not available. For a coordinate projection on the full box, a polynomial is constant on every extractor fiber exactly when its simplified polynomial expression uses only retained variables. The selector verifier uses this exact dependence test. When an omitted variable occurs, it constructs a pair witness on the rational grid with coordinate values $-1/2, 0, 1/2$; three nodes per variable suffice for the degree-two targets in this experiment. The deletion loop rechecks the current state after each removal. This procedure gives exact verification for the coordinate-projection and quadratic-polynomial setting used in the experiment.

The remaining selectors are searched by auxiliary-coordinate count, then degree (one before two), then library order. Each update coordinate has independent coefficients for all monomials of total degree at most the selected degree, including a constant. Feasibility is decided by exact polynomial coefficient matching. This step uses direct polynomial coefficient matching rather than the single-state CEGIS loop of Experiment 1, so the resulting coordinate identity checks do not measure the $p_S + 1$ proposal bound.

Instance-level outcomes. Table 6 contains all ten outcomes. Seven instances admit exact programs and three admit no selector from the library. For each accepted program, the residual polynomial is identically zero in exact rational arithmetic. Three accepted programs require no auxiliary coordinate, one requires one, and three require two. All seven have degree two. On these seven feasible instances, the auxiliary count is 1.00 ± 1.00 and the number of templates rejected before acceptance is 7.57 ± 6.92. Templates removed by selector preprocessing are not included in that latter count. For example, seed 401 first loses x_1 because its successor depends on unavailable x_5; the semantic

Table 6: Experiment 3 outcomes. Sets list auxiliary coordinates only; $\varnothing$ means that the supplied semantic coordinate alone suffices. A dash denotes that no admissible selector exists, not an empty admissible selector. Rounds and queries distinguish complete selector-verification rounds from scalar dependence checks. Rejections count searched templates before acceptance.

Seed	Greatest auxiliary set	Minimum auxiliary set	Rounds	Queries	Rejections	Outcome
101	$\{x_1, x_2\}$	$\varnothing$	3	11	1	Exact
211	$\{x_1, x_2, x_4\}$	$\{x_1, x_4\}$	2	8	12	Exact
307	$\{x_1, x_3, x_4\}$	$\{x_3, x_4\}$	2	7	13	Exact
401	—	—	2	3	—	Infeasible
503	—	—	2	6	—	Infeasible
601	—	—	2	5	—	Infeasible
701	$\{x_1, x_2, x_3, x_4\}$	$\{x_2\}$	1	5	7	Exact
809	$\{x_1, x_3, x_4\}$	$\varnothing$	2	7	1	Exact
907	$\varnothing$	$\varnothing$	5	9	1	Exact
1009	$\{x_1, x_2, x_3, x_4\}$	$\{x_1, x_4\}$	1	5	18	Exact

successor then becomes undetermined because it depends on x_1. Seeds 503 and 601 fail analogously after removing x_4 and x_3, respectively. These are information-level obstructions, not optimizer failures. Conversely, seed 701 shows that the greatest admissible selection need not minimize storage: all four candidates can be updated, but only x_2 is needed by the returned program.

Numerical comparison and timing. For every accepted selector and template, an unconstrained NumPy FP64 least-squares fit uses 12000 independently sampled training states and 12000 test states, with random seed equal to the instance seed plus 1000003. Both sets are uniform on the same box, and the targets are the rationalized network's exact update polynomials evaluated numerically. Across the seven feasible instances, the aggregate test root-mean-square error is $(5.07 \pm 3.07) \times 10^{-17}$, and the mean per-instance maximum absolute test error is $(2.26 \pm 1.80) \times 10^{-16}$. These fits are numerically accurate but are not subjected to the exact identity checker. The experiment therefore does not show a predictive-accuracy advantage of exact synthesis.

Across all ten instances, validation mean squared error relative to the teacher has mean 8.21×10^{-10} and sample standard deviation 2.56×10^{-9} before rationalization. After rationalization it is $(1.38 \pm 2.93) \times 10^{-9}$; the largest latter error is 9.70×10^{-9}. The latter quantity is not a bound on the discrepancy between the original floating-point network and its rationalized counterpart. Recorded per-instance wall time through training, discovery, and the least-squares comparison is 10.23 ± 0.32

seconds, excluding subsequent CSV serialization. The existing log does not separate GPU training time from CPU discovery time, so no speedup between these components is inferred.

Scope of the trained-network experiment. The experiment tests exact state selection and executable formulas for controlled sparse polynomial networks with supplied coordinate semantics; it does not establish semantic discovery or verification scalability for pretrained Transformers. The observed $7/10$ feasibility rate applies only to these ten constructed instances.

K.5 Experiment 4: finite-table construction and controls

Instance generation. We generated simple undirected graphs with $n \in \{8, 12, 16, 20, 32, 48\}$ vertices and edge probability $\pi \in \{0.2, 0.5\}$, using instance seeds $1000, \ldots, 1009$ for each configuration. For instance seed σ, NumPy's `default_rng` was initialized with $\sigma + 100000n + \mathrm{int}(1000\pi)$. Unordered vertex pairs were visited in lexicographic order and included independently with probability π. Each graph was converted to the finite restriction of the construction in Appendix E, with $t \in \{0, 1\}$. This gave 120 systems, each with $4 + 4|E_H|$ states, $n + 1$ auxiliary candidates, one involutive transition, and a constant response. The implementation represented each one-hot label by its integer identifier. Candidate values and the remaining semantic coordinate were stored in FP64 but took only the exactly representable values zero and one; state comparisons therefore used exact equality without a numerical tolerance.

Selection and independent checks. The implementation grouped states by their semantic values and examined every unordered pair within each group. Differences in successor semantics generated mandatory clauses, while differences in optional successor coordinates generated conditional clauses. Duplicate clauses and tautologies were removed. Minimum-cardinality selection was solved directly from these generated clauses by branch-and-bound with unit propagation. Greedy feasible completions supplied upper bounds, and collections of pairwise disjoint active positive requirements supplied lower bounds. Independently, the minimum vertex-cover size was computed by an exact maximum-clique search on the complement graph, using greedy-coloring bounds. For the 40 instances with at most 12 vertices, exhaustive enumeration of auxiliary selections provided an additional check. Agreement required a valid returned selector and the same optimal cardinality; different optimal selections were allowed.

Controls and results. The mandatory-only control omitted optional-successor clauses while keeping the state table, semantics, and candidate library unchanged. Its selected coordinates were then checked against the original transition, including their own successor requirements. A second control replaced the transition by identity and regenerated the complete clause set. We also applied Algorithm 1 to obtain the greatest admissible selection.

All 120 original systems satisfied $\min |J| = 1 + \tau(H)$, and all 40 exhaustive checks agreed with the optimum. The mandatory-only control selected the single forced auxiliary coordinate in every instance, but its successor failed the original fiber condition. Identity transitions admitted the empty selector, whereas the greatest selectors of the original systems retained the full library. The saved records contain graph seeds, instance sizes, clause counts, selected coordinates, control outcomes, search-node counts, and computation times.

K.6 Experiment 5: alternative feature libraries and program classes

Frozen networks and libraries. We reconstructed the ten rationalized transitions from Experiment 3 using the saved weight numerators and denominators, without retraining or further modifying the weights. The state domain remained $[-1, 1]^6$, with semantic coordinate $s(x) = x_0$. We compared the base library $\{x_1, x_2, x_3, x_4\}$, the cubed library $\{x_1^3, x_2^3, x_3^3, x_4^3\}$, and their union. The first two libraries followed coordinate order; the union used the order $x_1, x_1^3, x_2, x_2^3, x_3, x_3^3, x_4, x_4^3$. Coordinate x_5 remained unavailable. Thus the comparison changed the supplied feature representation while preserving the underlying transition and semantics.

State sufficiency and structural recognition. For each network, we computed determining families for the semantic successor and the four base-coordinate successors. Each scalar fiber query used Z3

nonlinear real arithmetic with exact rational coefficients. The constraints required two states in the declared box to agree on the semantic coordinate and selected features while differing on the target. Unsatisfiability certified determinacy; satisfiability refuted it. Each query had a 60-second timeout, and an unknown result terminated the run without an infeasibility conclusion. Determining families were recovered by searching subsets in increasing cardinality and using upward closure to include supersets of established determining sets.

Because cubing is injective over the reals, equality of a cubed coordinate is equivalent to equality of its base coordinate. The same equivalence holds for their successor values. The implementation therefore reused the base determining families when evaluating the other libraries, associating each selector with the coordinates for which it retained at least one representation. It enumerated all 16, 16, and 256 selectors in the base, cubed, and union libraries, respectively. From the admissible families, it computed minimum cardinalities, all inclusion-minimal selections, the greatest selection, and whether the intersection of all admissible selections was itself admissible.

We separately ran the recognizer of Proposition 7 on each base-library target and compared its supportedness decision, intersection, and structural-condition decision with the enumerated determining family. All 50 target checks agreed, with at most six logical queries per target. These counts are distinct from the queries used to recover the complete determining families.

Polynomial programs and transformed constructions. For each library, admissible selectors were searched by auxiliary-coordinate count, then polynomial degree, then library order. The tested degrees were one and two. Each successor formula used all monomials up to the tested total degree in the retained state, with an independent coefficient block. Substitution of the extractor reduced feasibility to exact rational polynomial coefficient matching, implemented with Python `Fraction` and SymPy rational linear algebra.

For the transformed construction, we first fixed the coefficients of a verified degree-two base program. A retained cubed coordinate was converted to its base coordinate by the fixed real cube-root operation. The base program then computed the semantic and auxiliary successors, and each successor stored in cubed form was cubed again. Verification checked the base polynomial identities and the resulting auxiliary-successor identities using exact rational coefficients. The polynomial expressions after coordinate recovery have degree at most six. This is an upper bound for the constructed expressions, rather than a minimum-degree result or a claim that the update is polynomial in the cubed state. Both the polynomial search and the transformed construction were evaluated without a new single-state counterexample-guided synthesis run.

Results and interpretation. All three libraries admitted selectors for the same seven networks and gave the same minimum auxiliary-state sizes, with mean 1.00 ± 1.00 across those seven instances. The base-library search reproduced the accepted selectors and degrees from Experiment 3. Polynomial templates of degree at most two admitted programs for all seven feasible base-library instances, compared with three cubed-library instances; the latter were precisely the semantic-only cases. The cube-root constructions provided verified programs for the other four feasible cubed-library instances.

In the union library, seeds $211, 307, 701, 1009$ had respectively $4, 4, 2, 4$ inclusion-minimal admissible selectors and no least selector. Seeds $101, 809, 907$ retained the empty least selector even though some supported optional targets failed the intersection-closure condition. Those targets did not need to be retained by the semantic-only programs. The results distinguish a condition sufficient for a least selector from its necessity, and distinguish preservation of state information from expressibility in a fixed polynomial class. Saved outputs report the target checks, selector families, program outcomes, structural-condition comparisons, and agreement with the frozen-network results from Experiment 3.

L Full decoder-readout construction

Consider the terminal normalization of a fixed decoder with retained residual vector u_{res}, residual width d_{res}, and a separately retained squared-norm contribution $\rho \geq 0$. Let the supplied semantics be $s = C_{\mathrm{sem}} u_{\mathrm{res}} \in \mathbb{R}^m$ and let the vocabulary contrast have the form

$$r(u_{\mathrm{res}}, \rho) = \frac{a_{\mathrm{out}}^{\mathsf{T}} s}{\sqrt{(\|u_{\mathrm{res}}\|_2^2 + \rho)/d_{\mathrm{res}} + \epsilon_{\mathrm{norm}}}} + b_{\mathrm{out}}, \ \epsilon_{\mathrm{norm}} > 0, \tag{25}$$

with all displayed coefficients fixed by the decoder and the semantic map. The numerator assumption is substantive: the contrast must factor through the given semantics. This construction concerns a terminal response with one output type and no outgoing edges; it does not address closure under earlier attention or feed-forward transitions.

Let the source domain be a compact semialgebraic set containing two states with the same u_{res}, different admissible values of ρ, and $a_{\mathrm{out}}^{\mathsf{T}} s \neq 0$. They agree semantically and differ in response, so the mandatory pair clause rejects the empty auxiliary selection. Supply the single candidate

$$\chi(u_{\mathrm{res}}, \rho) = \|u_{\mathrm{res}}\|_2^2 + \rho. \tag{26}$$

The selected state (s, χ) supports the readout template

$$o(z; \theta) = \frac{\sum_{i=1}^{m} \theta_i z_i}{\sqrt{z_{m+1}/d_{\mathrm{res}} + \epsilon_{\mathrm{norm}}}} + \theta_{m+1}, \tag{27}$$

which is nonlinear in the state and affine in its $m + 1$ coefficients; $(\theta_1, \ldots, \theta_m) = a_{\mathrm{out}}$ and $\theta_{m+1} = b_{\mathrm{out}}$ is exact. When the constants and domains satisfy Corollary 14, the synthesis loop recovers a verified instance in at most $m + 2$ identity-verifier calls. The one-feature selector is minimal within this library because the only smaller selector carries the pair obstruction. The instance has a known solution by construction and verifies that the specification recovers the decoder operation with its normalization intact. The guarantee therefore depends on retaining the original normalization and vocabulary map; extending the interface to earlier transitions introduces additional identity obligations and may require further features.