

Constraint-Based Analysis of Reasoning Shortcuts in Neurosymbolic Learning

Akihiro Takemura¹, Katsumi Inoue¹, Masaaki Nishino²

¹National Institute of Informatics, Tokyo, Japan

²Communication Science Laboratories, NTT, Inc., Japan
{atakemura, inoue}@nii.ac.jp, masaaki.nishino@ntt.com

Abstract

Neurosymbolic systems can satisfy logical constraints during learning without achieving the intended concept-label correspondence; this is a problem known as reasoning shortcuts. We formalize reasoning shortcuts as a constraint satisfaction problem and investigate under which conditions concept mappings are uniquely determined by the constraints. We prove that a discrimination property (requiring that no valid concept mapping can be transformed into another valid mapping by swapping two concept values) is necessary for shortcut-freeness under bijective mappings, but demonstrate via a counterexample that it is insufficient even when the constraint graph is connected. We develop an ASP-based algorithm that verifies whether a given constraint set uniquely determines the intended concept mapping, with proven soundness and completeness. When shortcuts are detected, a greedy repair algorithm eliminates them by augmenting the constraint set, converging in at most k iterations, where k is the number of alternative valid mappings. We further provide a complexity classification: deciding shortcut-freeness is coNP-complete, counting shortcuts is #P-complete, and finding minimal repairs is NP-hard. We also establish sample complexity bounds showing that logarithmically many label queries suffice for disambiguation in favorable cases, while querying all ambiguous positions suffices in the worst case. Experiments across eight benchmark domains validate our approach.

1 Introduction

Neurosymbolic AI systems integrate neural perception with symbolic reasoning, combining the pattern recognition capabilities of deep learning with the reliability and explainability of symbolic logic (Besold et al. 2021; Manhaeve et al. 2021; Yang, Ishay, and Lee 2020; Badreddine et al. 2022). However, recent work has identified a critical problem called *reasoning shortcuts*, where models satisfy constraints while learning concept mappings that differ from the intended correspondence (Marconato et al. 2023b; Yang et al. 2024).

Consider a neurosymbolic system for visual arithmetic: neural networks predict digit concepts from images, and constraints enforce arithmetic relationships (e.g., the labels assigned to two images must sum to 3). While the system may achieve high accuracy on training tasks, the learned concepts might be *swapped*, predicting 1 as 2 and vice

versa, yet still satisfy all constraints. This reasoning shortcut undermines both correctness (concepts do not correspond to their intended labels) and robustness (fails on out-of-distribution tasks) (Marconato et al. 2024; van Krieken et al. 2025). We focus on the class of constraint-based neurosymbolic systems where logical rules over concept mappings are explicit (Definition 1); our results characterize when such a constraint set uniquely determines the intended concept mapping, and at what computational cost.

Recent empirical work has developed mitigation strategies including uncertainty-aware methods (Marconato et al. 2024), theoretical risk analysis (Yang et al. 2024), prototypical neurosymbolic architectures that encourage models to rely on correct internal evidence (Andolfi and Giunchiglia 2025), and comprehensive benchmarks (Bortolotti et al. 2024). However, it remains unclear *whether and when constraints uniquely determine concept mappings*, which is essential for deploying neurosymbolic systems with confidence.

We approach this question by formalizing reasoning shortcuts as a *constraint satisfaction problem* (CSP). Given a set of constraints C over concept mappings $\phi : N \rightarrow S$, we ask, does C admit exactly one valid mapping (shortcut-free), or do multiple valid mappings exist (shortcuts present)? This CSP perspective enables us to:

- *Prove necessary conditions*: Discrimination (no valid mapping related to another by a concept transposition) is necessary for shortcut-freeness (Theorem 1), but insufficient in general. We construct an explicit counterexample where discrimination holds yet shortcuts persist via longer permutation cycles, even with a connected constraint graph (Example 4).
- *Classify complexity*: Deciding shortcut-freeness is coNP-complete, counting shortcuts is #P-complete, and finding minimal repairs is NP-hard (Theorems 4–6).
- *Bound disambiguation cost*: We define a shortcut multiplicity measure with connections to the M-unambiguity (Wang, Tsamoura, and Roth 2023), and prove that logarithmically many label queries suffice for disambiguation in favorable cases, while querying all ambiguous positions suffices in the worst case (Theorem 2).
- *Provide practical algorithms*: a verification algorithm based on *Answer Set Programming* (ASP) (Gebser, Kauf-

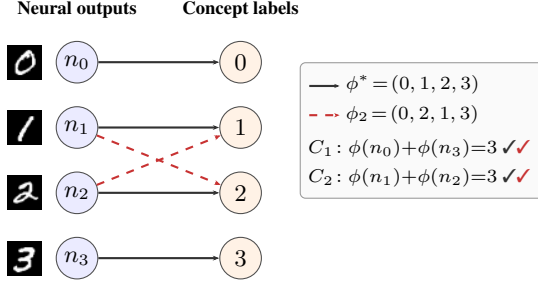


Figure 1: Reasoning shortcut in the 4-node addition problem (Example 1). Both the intended mapping ϕ^* (solid) and the shortcut ϕ_2 (dashed) satisfy all constraints, but ϕ_2 incorrectly swaps concepts 1 and 2.

mann, and Schaub 2012), whose native model enumeration makes it well-suited to exhaustive shortcut search, with correctness guarantees (Algorithm 1), greedy repair that eliminates shortcuts by augmenting the constraint set with convergence guarantees (Algorithm 2), and active learning strategies.

Our key observation is that reasoning shortcuts arise from *symmetries in the constraint structure*. Disconnected constraint graphs allow independent concept swaps within components (Example 1), while symmetric constraints such as modulo arithmetic admit rotational shortcuts even when the graph is connected (Example 4). This connects our work to classical symmetry-breaking in CSPs (Gent, Petrie, and Puget 2006), but extends it to the neurosymbolic setting where a specific ground-truth mapping must be uniquely determined.

Our work complements existing neurosymbolic approaches, e.g., shortcut detection methods (Marconato et al. 2023b; Yang et al. 2024), ASP-based systems (Yang, Ishay, and Lee 2020; Skryagin et al. 2024), Semantic Loss (Xu et al. 2018), and Concept Bottleneck Models (CBMs) (Koh et al. 2020), by providing the *formal foundations*: we prove when shortcuts can and cannot exist, characterize their computational complexity, and supply verification and repair algorithms with correctness guarantees.

Example 1 (Motivating Example: Simple Addition). *Consider learning a mapping from 4 neural outputs to 4 digit concepts using addition constraints (Figure 1):*

Problem setting:

- *Neural outputs:* $N = \{n_0, n_1, n_2, n_3\}$
- *Concept labels:* $S = \{0, 1, 2, 3\}$
- *Constraints:*
 $C_1 : \phi(n_0) + \phi(n_3) = 3$; $C_2 : \phi(n_1) + \phi(n_2) = 3$
- *Intended concept mapping*
 $\phi^* = \{n_0 \mapsto 0, n_1 \mapsto 1, n_2 \mapsto 2, n_3 \mapsto 3\}$
or equivalently in tuple notation, $\phi(n_i)$ as the i -th entry:
 $\phi^* = (0, 1, 2, 3)$

Issue: *There are multiple bijective mappings that satisfy all constraints. For example:*

- $\phi_1 = (0, 1, 2, 3)$ — *intended mapping*

- $\phi_2 = (0, 2, 1, 3)$ — *swaps $n_1 \leftrightarrow n_2$*
- $\phi_3 = (3, 1, 2, 0)$ — *swaps $n_0 \leftrightarrow n_3$*
- ... (8 total bijective valid mappings)

The network may learn any of these 8 mappings during training. If the learned mapping differs from ϕ^ , test-time predictions will be systematically incorrect.*

The constraint graph G_C (Definition 2; Figure 2) decomposes into two disconnected components $\{n_0, n_3\}$ and $\{n_1, n_2\}$, and this disconnection allows the two component-level transpositions to combine into 8 valid bijections.

This paper also provides a mechanism to: (1) detect such shortcuts via ASP-based verification, and (2) eliminate them by augmenting the constraint set (e.g., adding “ $\phi(n_1) = 1$ ” reduces the 8 valid mappings to 4).

Our framework provides immediately deployable tools for practitioners: ASP verification directly checks shortcut-freeness without requiring analytical proofs, repair algorithms systematically eliminate shortcuts with convergence guarantees, and active learning strategies achieve near-optimal label efficiency.

This paper is organized as follows: Section 2 surveys related work. Section 3 introduces the problem setting and core definitions. Section 4 develops theoretical foundations: necessary conditions, their limitations, symmetry analysis, sample complexity bounds, and practical label selection strategies. Section 5 presents verification and repair algorithms with complexity analysis. Section 6 provides experimental validation across 8 benchmark domains. Section 7 concludes with open problems.

2 Related Works

2.1 Reasoning Shortcuts in Neurosymbolic Systems

Marconato et al. (2023b) first characterized reasoning shortcuts in neurosymbolic learning, identifying four key conditions under which they arise. They demonstrated that models can achieve high task accuracy while learning concept mappings that differ from the intended correspondence, compromising both interpretability and out-of-distribution robustness. Building on this, Marconato et al. (2024) introduced *BEARS*, an uncertainty-aware framework for detecting and mitigating shortcuts through Bayesian estimation. Marconato et al. (2023a) further showed that reasoning shortcuts persist across continual learning scenarios, requiring explicit concept rehearsal strategies.

Recent theoretical work by Yang et al. (2024) provided the first formal analysis of shortcut risk, quantifying it through knowledge base complexity, sample size, and hypothesis space. They proved that Abductive Learning (ABL) algorithms reduce shortcut risk compared to standard neurosymbolic approaches by selecting appropriate distance functions. van Krieken et al. (2025) demonstrated that the independence assumption, commonly used in neurosymbolic frameworks to simplify inference, prevents models from correctly representing uncertainty over concept combinations, thereby hindering awareness of reasoning shortcuts. In a complementary direction, Takemura and Inoue

(2026) apply matrix-based differentiable logic programming to empirically mitigate shortcuts through one-to-one atom grounding; our work addresses the orthogonal question of when constraints formally determine uniqueness.

To facilitate systematic evaluation, Bortolotti et al. (2024) introduced *RSBench*, a comprehensive benchmark suite providing customizable tasks affected by reasoning shortcuts, along with formal verification procedures for assessing their presence. More recently, Bortolotti et al. (2025) formalized “joint reasoning shortcuts” (JRSs) in CBMs with learned inference layers, showing that reducing deterministic JRSs to zero provably prevents all JRSs and yields identifiability. Their identifiability result for CBMs with learned inference layers complements ours: they show *when* eliminating deterministic shortcuts suffices, while we characterize *whether* a given constraint set admits shortcuts and at what computational cost.

2.2 Neurosymbolic Integration and Constraint-Based Learning

Several frameworks integrate neural perception with symbolic constraints. DeepProbLog (Manhaeve et al. 2021) combines neural predicates with probabilistic logic programming; Logic Tensor Networks (Badreddine et al. 2022) embed logical formulas as differentiable tensor operations; and Semantic Loss (Xu et al. 2018) derives differentiable losses from logical constraints via weighted model counting, with recent extensions to autoregressive models (Ahmed, Chang, and den Broeck 2023). PiShield (Stoian et al. 2024) provides shield layers guaranteeing constraint satisfaction during inference regardless of input. Concept Bottleneck Models (CBMs) (Koh et al. 2020) take a complementary approach, routing predictions through human-interpretable concept layers to enable test-time interventions. Recent extensions include generative CBMs (Ismail et al. 2023) and analysis of joint reasoning shortcuts in CBMs with learned inference layers (Bortolotti et al. 2025). These approaches share an implicit assumption that constraints or bottleneck architectures should suffice to ensure the intended concept-label correspondence. Our work shows this assumption requires formal verification, i.e., constraints may admit multiple valid mappings even when all are satisfied. More broadly, constraint-based formulations have proven valuable for ML interpretability beyond the neurosymbolic setting. Ignatiev et al. (2021) survey reasoning and constraint-based approaches to learning inherently interpretable models such as decision trees and decision sets, while Shrotri et al. (2022) use Boolean constraints to guide explanation generation for black-box models.

In formal explainable AI research, constraint solvers have been used to derive provably correct explanations of black-box predictions (Darwiche 2023; Marques-Silva and Ignatiev 2022). These approaches share our use of constraint-based reasoning over learned models but address a different question: given a fixed model and input, what is a (minimal) sufficient reason for the prediction? Our setting reverses this perspective: given a constraint set used *during* learning, does it admit a unique concept-label correspondence? Both perspectives concern when symbolic structure provides

sufficient determination, but at different stages of the ML pipeline (explanation vs. pre-deployment verification). Our CSP-based analysis of reasoning shortcuts extends this line of work to the neurosymbolic setting, where the goal is not to explain predictions but to verify whether constraints uniquely determine concept-label correspondences.

2.3 ASP-Based Neurosymbolic Systems and Symmetry Breaking

ASP has emerged as a tool for integrating symbolic reasoning with neural perception. NeurASP (Yang, Ishay, and Lee 2020) treats neural outputs as probability distributions over ASP atoms, enabling symbolic rules to guide training. SLASH (Skryagin et al. 2022) scales this via pruning of stochastically insignificant groundings, and Answer Set Networks (Skryagin et al. 2024) translate ASP programs into graph neural networks for GPU-accelerated reasoning. These systems use ASP for *inference-time probabilistic reasoning*. Our use of ASP is different: we perform *compile-time verification* of structural properties, checking whether constraints uniquely determine concept mappings before deployment. This connects to classical symmetry breaking in CSPs (Gent, Petrie, and Puget 2006), but differs in that we must verify a specific ground-truth mapping ϕ^* is uniquely determined, not merely find any solution efficiently.

Existing work either detects shortcuts empirically (Marconato et al. 2023b; Marconato et al. 2024; Bortolotti et al. 2024), quantifies shortcut risk under distributional assumptions (Yang et al. 2024), or assumes constraints suffice for correctness (Xu et al. 2018; Koh et al. 2020). We provide the formal foundations: a CSP formalization with necessary conditions (Theorem 1), complexity classification (Theorems 4–6), ASP-based verification with correctness guarantees (Algorithm 1), and sample complexity bounds connecting shortcut multiplicity to disambiguation requirements (Theorem 2). This shifts the question from “how do we detect shortcuts?” to “when do constraints guarantee uniqueness?”

3 Preliminaries

Definition 1 (Constraint-Based Neurosymbolic Learning Problem). A constraint-based neurosymbolic learning (NSL) problem is a tuple $\mathcal{P} = (N, S, C, \phi^*, D)$ where:

- $N = \{n_1, \dots, n_m\}$ is a finite set of m neural outputs
- $S = \{s_1, \dots, s_r\}$ is a finite set of r concept labels
- C is a finite set of constraints over mappings $\phi : N \rightarrow S$
- $\phi^* : N \rightarrow S$ is the intended (ground-truth) concept mapping
- D is a dataset of observations

The triple (N, S, C) defines a constraint satisfaction problem (CSP) whose variables are the neural outputs in N , domain is S , and constraints are C .

This definition captures NeSy systems in which logical constraints over concept mappings are explicit. Frameworks such as DeepProbLog, NeurASP, and Logic Tensor Networks with declared inference rules can be analyzed in our

framework by extracting their constraint structure. Frameworks that operate without explicit symbolic constraints (e.g., end-to-end learned representations) fall outside our scope.

In practice, the intended mapping ϕ^* is unknown to the learner, and recovering it is the goal of neurosymbolic learning. We include ϕ^* in the problem definition because our aim is not to solve the learning task itself, but to *analyze* the constraint structure: given a hypothetical intended mapping, we ask whether the constraints uniquely determine the mapping or admit alternative valid mappings (reasoning shortcuts). This is analogous to verifying identifiability conditions before deployment, rather than during training.

Definition 2 (Constraint Graph). *The constraint graph of the CSP (N, S, C) is the undirected graph $G_C = (N, E)$ whose vertex set N is the set of neural outputs and whose edge set is $E = \{(n_i, n_j) \mid n_i \neq n_j \text{ and both } n_i, n_j \text{ appear in some constraint } c \in C\}$.*

Constraint graphs for the running examples are shown in Figure 2.

Example 2 (MNIST-Half as NeSy Learning Problem). *The MNIST-Half task instantiates Definition 1 as follows:*

- $N = \{n_0, n_1, n_2, n_3, n_4\}$ (neural outputs for digits 0-4)
- $S = \{0, 1, 2, 3, 4\}$ (digit concept labels)
- $C = \{C_1, C_2, C_3, C_4\}$ where:

$$\begin{aligned} C_1 : \phi(n_0) + \phi(n_0) &= 0 & C_2 : \phi(n_0) + \phi(n_1) &= 1 \\ C_3 : \phi(n_2) + \phi(n_3) &= 5 & C_4 : \phi(n_2) + \phi(n_4) &= 6 \end{aligned}$$

- $\phi^* = (0, 1, 2, 3, 4)$
- D : Dataset of digit pairs with sum labels

The constraint graph G_C has two components: $\{n_0, n_1\}$ (edge from C_2) and $\{n_2, n_3, n_4\}$ (edges from C_3, C_4); the unary constraint C_1 pins $\phi(n_0) = 0$ directly.

Definition 3 (Valid Mapping and Shortcut). *Let $\mathcal{P} = (N, S, C, \phi^*, D)$ be a constraint-based NSL problem.*

The set of valid mappings is:

$$\Phi_C = \{\phi : N \rightarrow S \mid \phi \text{ satisfies all constraints in } C\}$$

A valid mapping $\phi \in \Phi_C$ is a reasoning shortcut if $\phi \neq \phi^$.*

The problem is shortcut-free if $|\Phi_C| = 1$ and the unique element of Φ_C is ϕ^ .*

Standing assumption: Throughout this paper, we assume the intended mapping satisfies the constraints, i.e., $\phi^* \in \Phi_C$. This implies $\Phi_C \neq \emptyset$. When this assumption does not hold, we treat it as an error condition (see Algorithm 1, line 9: INTENDED-INVALID).

Remark 1 (Restriction to Bijections). In this work, we assume the intended mapping ϕ^* is bijective, which is natural for classification tasks where each neural output should correspond to exactly one concept and each concept should be represented exactly once. We then restrict the search space to bijective mappings when analyzing shortcuts. This restriction serves as a structural prior: if ϕ^* is the unique valid mapping (i.e., $|\Phi_C| = 1$), then non-bijective alternatives are

already excluded, so no generality is lost for the shortcut-freeness question. When $|\Phi_C| > 1$, restricting to bijections may exclude some non-bijective shortcuts, but this is intentional: our analysis focuses on the harder case where shortcuts preserve the bijectivity structure of ϕ^* . Algorithm 1 supports both bijective and non-bijective enumeration, and our experiments report both (Table 1).

Definition 4 (Shortcut Multiplicity). *The shortcut multiplicity of constraint set C is $SM(C) = |\Phi_C| - 1$. We denote $k = SM(C)$ throughout. An NSL problem is shortcut-free if $SM(C) = 0$ (equivalently, $k = 0$).*

Notation for bijective mappings. To avoid ambiguity between general mappings and bijective mappings, we introduce explicit notation:

- $\Phi_C^{\text{all}} = \{\phi : N \rightarrow S \mid \phi \text{ satisfies all constraints in } C\}$
- $\Phi_C^{\text{bij}} = \{\phi \in \Phi_C^{\text{all}} \mid \phi \text{ is bijective}\}$
- $SM^{\text{bij}}(C) = |\Phi_C^{\text{bij}}| - 1$

Unless stated otherwise, when we write “ Φ_C ” we mean Φ_C^{bij} (bijective mappings only), and similarly “ $SM(C)$ ” means $SM^{\text{bij}}(C)$. For example, Algorithm 1 explicitly enforces bijectivity through constraints (lines 4-5).

Wang et al. (2023) study the learnability of multi-instance partial-label learning under a transition function σ that maps a tuple of gold labels to a weak supervision signal. Their *M-unambiguity* property characterizes when the gold labels can be uniquely recovered from the weak signal. Our setting is complementary: rather than inferring unobserved labels from weak supervision, we ask whether a set of logical constraints uniquely determines a concept mapping. The following measure adapts their notion to our CSP setting and clarifies the structural relationship between our shortcut multiplicity $SM(C)$, the per-pair disagreement $A(C)$, and the set of disagreement positions Δ_C .

Definition 5 (M-Ambiguity). *Adapting the notion of M-unambiguity from (Wang, Tsamoura, and Roth 2023) to our setting, we define the ambiguity of a constraint set C as:*

$$A(C) = \max_{\phi, \phi' \in \Phi_C} |\{n \in N : \phi(n) \neq \phi'(n)\}|$$

This measures the maximum number of neural outputs on which any two valid mappings disagree.

Proposition 1 (Relationship Between Measures). *Define the disagreement set of constraint set C as*

$$\Delta_C = \{n \in N : \exists \phi, \phi' \in \Phi_C \text{ with } \phi(n) \neq \phi'(n)\},$$

i.e., the set of positions where valid mappings disagree. Then:

1. $SM(C) = 0 \Leftrightarrow |\Phi_C| = 1$
2. If $SM(C) = 0$, then $A(C) = 0$ and $|\Delta_C| = 0$
3. If $A(C) = 0$, then $SM(C) = 0$
4. If $SM(C) > 0$, then $A(C) \geq 1$ and $|\Delta_C| \geq 1$
5. $A(C) \leq |\Delta_C| \leq |N|$ (with equality $|\Delta_C| = |N|$ only when mappings disagree everywhere)

Proof. Parts (1)-(4) follow directly from definitions. For part (5): By definition, $A(C)$ measures the maximum disagreement between any single pair of mappings, while Δ_C is the union over all pairs. The maximum pairwise disagreement cannot exceed the union: $A(C) \leq |\Delta_C|$. Furthermore, by definition $\Delta_C \subseteq N$, so $|\Delta_C| \leq |N|$. Equality $|\Delta_C| = |N|$ occurs when for every position $n \in N$, there exist $\phi, \phi' \in \Phi_C$ with $\phi(n) \neq \phi'(n)$. \square

4 Theoretical Framework

We now investigate under what conditions a constraint set uniquely determines the intended mapping. We begin by identifying a necessary condition, the discrimination property, then show its limitations, examine the symmetry structure of the solution space, analyze the sample complexity of disambiguation when constraints alone are insufficient, and describe a practical label selection strategy with a formal bound.

4.1 Discrimination is Necessary

Definition 6 (Discrimination Property). *Let $\phi : N \rightarrow S$ be a concept mapping. For distinct concepts $s_i, s_j \in S$, define the transposed mapping $\phi_{s_i \leftrightarrow s_j} : N \rightarrow S$ by:*

$$\phi_{s_i \leftrightarrow s_j}(n) = \begin{cases} s_j & \text{if } \phi(n) = s_i \\ s_i & \text{if } \phi(n) = s_j \\ \phi(n) & \text{otherwise} \end{cases}$$

A constraint set C is discriminative if:

$$\forall s_i, s_j \in S \text{ with } s_i \neq s_j, \forall \phi \in \Phi_C : \phi_{s_i \leftrightarrow s_j} \notin \Phi_C$$

This definition quantifies over all $\phi \in \Phi_C$ for clarity. Because shortcut-freeness implies there is only one solution, i.e., $\Phi_C = \{\phi^*\}$, under shortcut-freeness this reduces to discrimination at ϕ^* . Informally, discrimination forbids all 2-cycles (transpositions) in the automorphism group of valid mappings (see Definition 7 and Proposition 2).

Theorem 1 (Discrimination is necessary). *Let $\mathcal{P} = (N, S, C, \phi^*, D)$ with $|N| = |S|$. If \mathcal{P} is shortcut-free, then C is discriminative.*

Proof. Assume \mathcal{P} is shortcut-free, so $\Phi_C = \{\phi^*\}$. Consider any distinct $s_i, s_j \in S$. Since ϕ^* is bijective, there exist distinct $n_i, n_j \in N$ with $\phi^*(n_i) = s_i$ and $\phi^*(n_j) = s_j$. The transposition swaps these assignments, so $\phi_{s_i \leftrightarrow s_j}^* \neq \phi^*$. Since $|\Phi_C| = 1$, we conclude $\phi_{s_i \leftrightarrow s_j}^* \notin \Phi_C$. As this holds for all distinct pairs, C is discriminative. \square

Example 3 (Discrimination in MNIST-Half). *In MNIST-Half (Example 2), consider the intended mapping $\phi^* = (0, 1, 2, 3, 4)$.*

Attempt the transposition $2 \leftrightarrow 3$:

$$\phi_{2 \leftrightarrow 3}^* = (0, 1, 3, 2, 4)$$

Check constraints:

- C_3 : $\phi(n_2) + \phi(n_3) = 3 + 2 = 5$ (satisfied)
- C_4 : $\phi(n_2) + \phi(n_4) = 3 + 4 = 7 \neq 6$ (violated)

The transposition violates C_4 , so $\phi_{2 \leftrightarrow 3}^ \notin \Phi_C$. Thus C discriminates the pair $(2, 3)$ at ϕ^* .*

4.2 Discrimination Can Fail in Practice

Theorem 1 shows discrimination is necessary, but the following example shows that it is not sufficient.

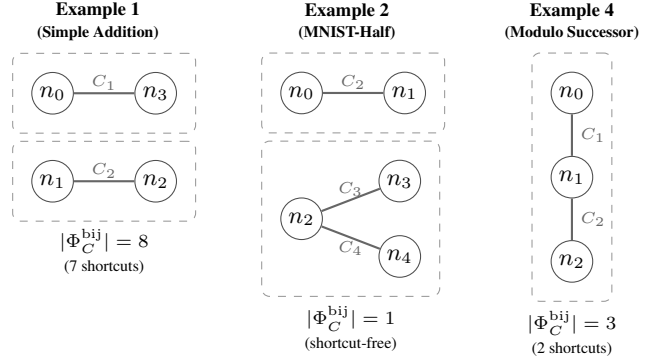


Figure 2: Constraint graphs for the three running examples. Example 1 has two disconnected components, admitting 7 bijective shortcuts. Example 2 (MNIST-Half) also has disconnected components, but stronger arithmetic constraints achieve uniqueness under bijectivity (a unary constraint is omitted from the figure). Example 4 has a connected graph, yet admits 2 shortcuts via rotational symmetry (3-cycles), showing that connectivity alone does not prevent shortcuts.

Example 4 (Discrimination is necessary but not sufficient). *Consider a 3-node problem with modulo successor constraints:*

- $N = \{n_0, n_1, n_2\}$
- $S = \{0, 1, 2\}$
- *Constraints:*
 $C_1 : \phi(n_1) \equiv \phi(n_0) + 1 \pmod{3}$; $C_2 : \phi(n_2) \equiv \phi(n_1) + 1 \pmod{3}$
- *Intended:* $\phi^* = (0, 1, 2)$

The constraint graph is connected ($n_0 - n_1 - n_2$), yet three bijective valid mappings exist:

- $\phi_1 = (0, 1, 2)$ (intended, ϕ^*)
- $\phi_2 = (1, 2, 0)$ (rotation by 1)
- $\phi_3 = (2, 0, 1)$ (rotation by 2)

Each ϕ_i assigns concepts so that consecutive outputs differ by 1 modulo 3, satisfying both C_1 and C_2 .

Discrimination holds: Consider $\phi_1 = (0, 1, 2)$. Swapping $0 \leftrightarrow 1$ gives $(1, 0, 2)$: check C_1 : $0 \equiv 1 + 1 \pmod{3}$? No. Similarly, every transposition applied to any valid mapping violates at least one constraint. Thus C satisfies the discrimination property.

Shortcuts exist via 3-cycles: The cyclic permutation $\sigma : 0 \mapsto 1, 1 \mapsto 2, 2 \mapsto 0$ acts on solutions by composition $(\sigma \circ \phi)(n) = \sigma(\phi(n))$:

$$\begin{aligned} \sigma \circ \phi_1 &= (1, 2, 0) = \phi_2, \\ \sigma \circ \phi_2 &= (2, 0, 1) = \phi_3, \\ \sigma \circ \phi_3 &= (0, 1, 2) = \phi_1. \end{aligned}$$

So $\sigma \in \text{Aut}(\mathcal{X})$. Together with σ^2 and the identity, $\text{Aut}(\mathcal{X}) = \{\text{id}, \sigma, \sigma^2\} \cong \mathbb{Z}/3$ is non-trivial, and acts transitively on Φ_C .

Discrimination eliminates 2-cycles from $\text{Aut}(\mathcal{X})$, but cannot detect 3-cycles, or longer cycles in general. This shows that discrimination, while necessary (Theorem 1), is not sufficient for shortcut-freeness, even when the constraint graph is connected.

Example 4 shows that the gap between discrimination and shortcut-freeness is not merely a matter of constraint graph connectivity: the modulo successor constraints form a connected graph, yet shortcuts persist through 3-cycles that discrimination cannot detect (Figure 2). This contrasts with Example 1, where shortcuts arise from a different mechanism, i.e., disconnected components allowing independent transpositions. Together, these examples show that discrimination is not a practical guardrail on its own, and the question of sufficient conditions for uniqueness leads naturally to analyzing value symmetries.

4.3 Value-Symmetry Elimination

Having established the limits of discrimination, we turn to sufficient conditions by examining the symmetry structure of the solution space.

Definition 7 (Symmetric Group and Automorphism Group). Let $\mathcal{X} = (N, S, C)$ be the CSP instance restricted to bijections $\phi : N \rightarrow S$. A permutation of S is a bijection $\sigma : S \rightarrow S$. The set of all permutations of S forms the symmetric group $\text{Sym}(S)$. Each $\sigma \in \text{Sym}(S)$ acts on solutions by composition: $(\sigma \circ \phi)(n) = \sigma(\phi(n))$ for all $n \in N$.

The automorphism group of \mathcal{X} is:

$$\text{Aut}(\mathcal{X}) = \{\sigma \in \text{Sym}(S) : \sigma \circ \phi \in \Phi_C \text{ for all } \phi \in \Phi_C\}$$

Proposition 2 (Value-Symmetry Elimination). If $\text{Aut}(\mathcal{X})$ is trivial (i.e., $\text{Aut}(\mathcal{X}) = \{\text{id}\}$), then no two distinct solutions in Φ_C are related by a permutation in $\text{Aut}(\mathcal{X})$.

Proof. If $\phi' = \sigma \circ \phi$ for some $\sigma \in \text{Aut}(\mathcal{X})$, triviality forces $\sigma = \text{id}$, hence $\phi' = \phi$. \square

Proposition 2 shows that a trivial automorphism group eliminates the multiplicity arising from value-permutation symmetries, but does not imply $|\Phi_C| = 1$: a permutation $\sigma \notin \text{Aut}(\mathcal{X})$ might still map one particular solution to another. The gap between trivial $\text{Aut}(\mathcal{X})$ and uniqueness arises from symmetries that preserve some valid mappings but not all. We illustrate two distinct mechanisms by which shortcuts arise, and note that they have different repair implications.

Single-orbit shortcuts (Example 4). Here $\text{Aut}(\mathcal{X}) = \{\text{id}, \sigma, \sigma^2\} \cong \mathbb{Z}/3$, where σ is the 3-cycle $0 \mapsto 1 \mapsto 2 \mapsto 0$. The group $\text{Aut}(\mathcal{X})$ acts transitively on Φ_C : every valid mapping is reachable from any other via a single rotation. Repair is correspondingly cheap: pinning a single position $\phi(n^*) = \phi^*(n^*)$ breaks the orbit and identifies ϕ^* uniquely (one constraint suffices, see Theorem 3).

Component-decomposable shortcuts (Example 1). Here disconnected components of the constraint graph admit independent rearrangements. The two components $\{n_0, n_3\}$

and $\{n_1, n_2\}$ each support an independent transposition, yielding $4 \times 2 = 8$ valid bijections. Repair requires at least one constraint per component admitting non-trivial rearrangement, since pinning one component leaves the other free.

This contrast motivates Section 5: verification via Algorithm 1 detects shortcuts regardless of mechanism, while greedy repair via Algorithm 2 resolves them at a cost determined jointly by the orbit structure of $\text{Aut}(\mathcal{X})$ and the connectivity of G_C .

Discrimination (Definition 6) eliminates all 2-cycles from the value-symmetry group, but does not eliminate longer cycles or non-symmetry-based multiplicity. Two structural properties appear relevant but remain unproven as sufficient conditions. First, *constraint graph connectivity* (Definition 2) prevents independent component rearrangements as in Example 1, but connected graphs may still admit shortcuts through longer permutation cycles (Example 4; see also Figure 2). Second, *constraint strength*: in MNIST-Half, arithmetic constraints like $\phi(n_2) + \phi(n_4) = 6$ fully determine values when combined with bijectivity, while the modulo successor constraints in Example 4 do not. Formalizing when constraints are “strong enough” and whether connectivity and strength jointly suffice for uniqueness remains an open problem.

4.4 Sample Complexity for Disambiguation

Since complete sufficient conditions for shortcut-freeness remain open, a natural fallback is *disambiguation through labeled data*. Given that $|\Phi_C| > 1$, how many concept labels are necessary to identify ϕ^* among the valid mappings? This question is of practical importance because even when constraints alone are insufficient, a small number of strategically chosen labels may resolve the remaining ambiguity. We formalize this as a query complexity problem over the candidate set Φ_C .

Theorem 2 (Sample Complexity of Label Queries). Let Φ_C contain $k + 1$ valid bijective mappings with $SM(C) = k \geq 1$. Let $\Delta_C = \{n \in N : \exists \phi, \phi' \in \Phi_C, \phi(n) \neq \phi'(n)\}$ and let $r = |S|$. Suppose a learner can make label queries: given $n \in N$, an oracle returns $\phi^*(n) \in S$. The learner may choose each query adaptively based on previous answers. Then we have:

1. Lower bound: Any learner requires at least $\lceil \log_r(k + 1) \rceil$ queries to identify ϕ^* .
2. Upper bound: Querying all positions in Δ_C identifies ϕ^* using at most $|\Delta_C| \leq r$ queries.

The lower bound is achieved when Φ_C admits balanced partitions at each query step; the upper bound is achieved when each query eliminates only one candidate.

Proof. Lower bound: Each label query returns a value in S , providing at most $\log_2 |S|$ bits of information. Distinguishing among $k + 1$ candidates requires at least $\log_2(k + 1)$ bits. Therefore at least $\lceil \log_2(k + 1) / \log_2 |S| \rceil = \lceil \log_{|S|}(k + 1) \rceil$ queries are necessary, regardless of the adaptive strategy.

Upper bound: Query $\phi^*(n)$ for each $n \in \Delta_C$. After all queries, consider any remaining candidate $\phi \in \Phi_C$:

- For $n \in \Delta_C$: the query at n eliminates all ϕ with $\phi(n) \neq \phi^*(n)$, so $\phi(n) = \phi^*(n)$.
- For $n \in N \setminus \Delta_C$: by definition of Δ_C , all valid mappings agree at n , so $\phi(n) = \phi^*(n)$.

Thus $\phi = \phi^*$, and $|\Delta_C|$ queries suffice. Note that ϕ^* is never eliminated since it always agrees with its own labels.

Bound on $|\Delta_C|$: The $k + 1$ valid mappings are pairwise distinct when restricted to Δ_C (they agree on $N \setminus \Delta_C$), so $k + 1 \leq r^{|\Delta_C|}$, giving $|\Delta_C| \geq \lceil \log_r(k + 1) \rceil$. \square

The gap between the lower and upper bounds depends on the structure of Φ_C : favorable structure (as in Example 5 below) allows the lower bound to be achieved, while adversarial cases may require all $|\Delta_C|$ queries. In practice, several strategies approximate the oracle:

- *Random sampling* requires $O(|N|/|\Delta_C| \cdot k \log_2 k)$ expected queries
- *Uncertainty sampling* achieves $O(\min\{|\Delta_C|, k\})$ queries (Proposition 3 in Section 4.5)
- *Greedy disambiguation* approximates the lower bound

These practical strategies are discussed in Section 4.5, with additional details in the supplementary material.

Example 5 (Sample Complexity Demonstration). *For the 4-node addition problem (Example 1) with $k = 7$, $|S| = 4$, and $\Delta_C = \{n_0, n_1, n_2, n_3\}$:*

Bounds: *Lower* = $\lceil \log_4(8) \rceil = 2$; *Upper* = $|\Delta_C| = 4$.
Achieved in 2 label queries (*matching the lower bound*):

- *Query 1: “What is $\phi^*(n_0)$?” Oracle returns 0. Among the 8 valid bijections, only those with $\phi(n_0) = 0$ remain. Direct enumeration gives exactly two: $\phi_1 = (0, 1, 2, 3)$ and $\phi_2 = (0, 2, 1, 3)$. The first query eliminates 6 of 8 candidates because the constraint $\phi(n_0) + \phi(n_3) = 3$, combined with bijectivity, forces $\phi(n_3) = 3$ once $\phi(n_0) = 0$.*
- *Query 2: “What is $\phi^*(n_1)$?” Oracle returns 1. This eliminates ϕ_2 (which has $\phi_2(n_1) = 2$). Only $\phi_1 = \phi^*$ remains. Identification complete.*

In adversarial cases (e.g., when each query removes only one candidate), all $|\Delta_C| = 4$ queries would be needed, matching the upper bound.

4.5 Practical Label Selection Strategies

While Theorem 2 provides information-theoretic bounds assuming an oracle, we now present a practical strategy with a formal bound. The strategy assumes access to (or maintenance of) the candidate set $\Psi \subseteq \Phi_C$, which can be approximated via bounded model enumeration (e.g., `clingo --models=100`), diversified sampling, or incremental verification.

Uncertainty Sampling. Maintain candidate valid mappings $\Psi \subseteq \Phi_C$ (discovered via Algorithm 1). At each step:

1. For each $n \in N$, compute disagreement: $d(n) = |\{s \in S : \exists \phi \in \Psi, \phi(n) = s\}|$
2. Select $n^* = \arg \max_{n \in N} d(n)$ (output with most disagreement)

3. Query label $(n^*, \phi^*(n^*))$

4. Remove all $\phi \in \Psi$ with $\phi(n^*) \neq \phi^*(n^*)$

Proposition 3 (Uncertainty Sampling Bound). *Assume oracle access to the full candidate set $\Psi = \Phi_C$. Under uncertainty sampling, at most $\min\{k, |\Delta_C|\}$ labeled examples are needed to identify ϕ^* , where Δ_C is the set of disagreement positions.*

Proof. Each query selects position n^* with maximum disagreement and obtains label $\phi^*(n^*)$. This eliminates all candidates $\phi \in \Psi$ with $\phi(n^*) \neq \phi^*(n^*)$, which is at least one candidate (since $d(n^*) \geq 2$ when $|\Psi| \geq 2$). Once we query position n , all remaining candidates agree with ϕ^* at n , so each query uses a distinct position. In the best case, each query eliminates approximately half the remaining candidates, requiring $\lceil \log_2(k + 1) \rceil$ queries. In the worst case, each query eliminates exactly one candidate, requiring k queries but bounded by the number of disagreement positions: $\min(k, |\Delta_C|)$. \square

This is an information-theoretic bound assuming perfect knowledge of Ψ , not a computational guarantee.

5 Algorithms and Complexity

5.1 Algorithmic Verification

We now translate the theoretical framework into a practical verification algorithm using Answer Set Programming (ASP). Our use of ASP is not for solving the task instance, but for *reasoning* about the constraint theory itself: verifying uniqueness, counting alternatives, and guiding repair.

Algorithm 1: Shortcut Verification (Pseudocode)

Require: Constraint set C , concepts S , neural outputs N with $|N| = |S|$, intended mapping ϕ^*
Ensure: SHORTCUT-FREE, INTENDED-INVALID, or list of shortcuts

- 1: // Initialize verification problem
- 2: Initialize search space: all mappings $\phi : N \rightarrow S$
- 3: Add all constraints from C
- 4: Add bijectivity constraint:
- 5: each output (resp. concept) maps to exactly one concept (resp. output)
- 6: // Step 1: Verify intended mapping
- 7: **for** each constraint $c \in C$ **do**
- 8: **if** ϕ^* violates c **then**
- 9: **return** INTENDED-INVALID
- 10: // Step 2: Search for alternative valid mappings
- 11: Exclude ϕ^* from search space
- 12: $\Psi \leftarrow$ Find all valid bijective mappings
- 13: **if** $\Psi = \emptyset$ **then**
- 14: **return** SHORTCUT-FREE
- 15: **else**
- 16: **return** Ψ (each $\phi \in \Psi$ is a shortcut)

The ASP encoding follows a uniform pattern. For a given constraint-based neurosymbolic learning problem

$P = (N, S, C, \phi^*, D)$: (i) declare neural outputs and concepts as domain atoms, (ii) use choice rules with cardinality constraints to generate candidate mappings, optionally enforcing bijectivity via cardinality constraints (one per neural output and one per concept), (iii) encode each task constraint as an integrity constraint over value atoms, and (iv) exclude ϕ^* via a joint integrity constraint to enumerate only alternative valid mappings. For instance, the arithmetic constraint $\phi(n_0) + \phi(n_3) = 3$ becomes $:- \text{value}(n_0, V_0), \text{value}(n_3, V_3), V_0 + V_3 \neq 3$. This encoding is directly executable in *clingo* with `--models=0` for full enumeration. Complete ASP encodings for all examples are available in the supplementary material.

Proposition 4 (Algorithm Correctness). *Algorithm 1 is correct:*

1. If it returns *INTENDED-INVALID*, then $\phi^* \notin \Phi_C$
2. If it returns *SHORTCUT-FREE*, then $\phi^* \in \Phi_C$ and no other bijective mapping satisfies C
3. If it returns *shortcuts*, then $\phi^* \in \Phi_C$ and there exist other bijective mappings satisfying C

Proof. Part 1: Lines 7-9 explicitly check if ϕ^* satisfies all constraints in C . If any constraint is violated, the algorithm returns *INTENDED-INVALID* (line 9), so $\phi^* \notin \Phi_C$.

Part 2 (Soundness): If the algorithm returns *SHORTCUT-FREE*, then ϕ^* passed all constraint checks (lines 7-9), so $\phi^* \in \Phi_C$. After excluding ϕ^* (line 11), the search space consists of all bijective mappings satisfying C except ϕ^* (lines 2-5, 12). Since $\Psi = \emptyset$ (line 13), no other bijective mapping satisfies C .

Part 3 (Completeness): If $\phi^* \in \Phi_C$ and other bijective mappings satisfy C , then Step 1 succeeds, and any bijective $\phi' \neq \phi^*$ satisfying C is found and included in Ψ (line 12), which is returned (line 16). \square

5.2 Constraint Repair

When verification reveals shortcuts, the next step is to eliminate them by augmenting the constraint set. Algorithm 2 adds *unary pinning constraints* of the form $\phi(n) = \phi^*(n)$ for some $n \in N$. The NP-hardness result in Theorem 6 is stated for the more general decision problem in which candidate constraints are drawn from a fixed library of pinning constraints, where each candidate may pin one or more positions.

Theorem 3 (Repair Algorithm Correctness). *Let $C_0 = C$ and C_i be the constraint set after iteration i of Algorithm 2. Assume $\phi^* \in \Phi_{C_0}$, $|\Phi_{C_0}| < \infty$ (finite number of bijections from N to S with $|N| = |S|$), and $T \geq k$ where $k = SM(C_0)$. Then:*

1. Monotonic decrease: $SM(C_{i+1}) < SM(C_i)$ for each iteration that adds a constraint
2. Preserves intended: $\phi^* \in \Phi_{C_i}$ for all i
3. Termination bound: Algorithm terminates in at most k iterations with $SM(C') = 0$
4. Suboptimality: The algorithm does not minimize $|C' \setminus C|$

Algorithm 2: Greedy Shortcut Repair

Require: Constraint set C , intended mapping ϕ^* , maximum iterations T

Require: $\phi^* \in \Phi_C$ (intended mapping must be initially valid)

Ensure: Either *SHORTCUT-FREE* constraint set C' or *TIMEOUT* with current C'

```

1:  $C' \leftarrow C, t \leftarrow 0$ 
2: repeat
3:    $\Psi \leftarrow \text{Run Algorithm 1 with } (C', \phi^*)$ 
4:   if  $\Psi = \text{SHORTCUT-FREE}$  then
5:     return  $C'$ 
6:   else if  $\Psi = \text{INTENDED-INVALID}$  then
7:     return ERROR:  $\phi^*$  inconsistent with constraints
8:   Select  $\phi_{sc} \in \Psi$  (any detected shortcut)
9:    $D_{sc} \leftarrow \{n \in N : \phi^*(n) \neq \phi_{sc}(n)\}$ 
10:  Select  $n^* \in D_{sc}$  (arbitrary or by heuristic)
11:   $c_{new} \leftarrow \text{constraint } "\phi(n^*) = \phi^*(n^*)"$ 
12:   $C' \leftarrow C' \cup \{c_{new}\}$ 
13:   $t \leftarrow t + 1$ 
14: until  $t \geq T$ 
15: return TIMEOUT with current  $C'$ 

```

Proof. (1) Monotonic decrease:

At iteration i , let $\phi_{sc} \in \Phi_{C_i}$ be the detected shortcut. The algorithm adds constraint $c_{new} : \phi(n^*) = \phi^*(n^*)$ where $\phi^*(n^*) \neq \phi_{sc}(n^*)$.

Clearly ϕ_{sc} violates c_{new} , so $\phi_{sc} \notin \Phi_{C_{i+1}}$.

Thus $\Phi_{C_{i+1}} \subset \Phi_{C_i}$ (strict subset), hence $|\Phi_{C_{i+1}}| < |\Phi_{C_i}|$, so $SM(C_{i+1}) < SM(C_i)$.

(2) Preserves intended:

The constraint $c_{new} : \phi(n^*) = \phi^*(n^*)$ is satisfied by ϕ^* by construction. Thus if $\phi^* \in \Phi_{C_i}$, then $\phi^* \in \Phi_{C_{i+1}}$.

By induction: $\phi^* \in \Phi_{C_0}$ by assumption, hence $\phi^* \in \Phi_{C_i}$ for all i .

(3) Termination:

Each iteration strictly decreases $|\Phi_{C_i}|$ by at least 1 (part 1).

Initially: $|\Phi_{C_0}| = SM(C) + 1 = k + 1 < \infty$

After iteration i : $|\Phi_{C_i}| \leq k + 1 - i$

To reach $|\Phi_{C_k}| = 1$, we need at most k iterations.

By part (2), $\phi^* \in \Phi_{C_k}$, so $\Phi_{C_k} = \{\phi^*\}$, thus $SM(C_k) = 0$.

Since the timeout assumption $T \geq k$ ensures the algorithm does not exit before iteration k , and the decreasing chain terminates (part 1), Algorithm 2 returns C_k with $SM(C_k) = 0$.

(4) Suboptimality:

The algorithm is greedy and therefore does not minimize $|C' \setminus C|$. Moreover, the more general candidate-library repair problem is NP-hard (Theorem 6), indicating that optimal repair is computationally difficult in general. \square

5.3 Application to Existing Frameworks

The verification and repair workflow applies directly to NeSy frameworks such as DeepProbLog, NeurASP, and Logic Tensor Networks. First, export the logical rules as

ASP constraints, define ϕ^* and the neural output/concept sets with $|N| = |S|$, then run Algorithm 1 to check shortcut-freeness. If shortcuts are detected, Algorithm 2 eliminates them with convergence guarantees, or uncertainty sampling (Section 4.5) can resolve ambiguity via labeled examples (see supplementary material for a step-by-step guide).

5.4 Complexity Results

Given a constraint-based NSL problem with intended mapping ϕ^* : (i) deciding if the problem is shortcut-free is coNP-complete, (ii) counting shortcuts is #P-complete, and (iii) finding minimal repair is NP-hard.

Theorem 4 (Shortcut-Free verification is coNP-complete). *Given a constraint-based NSL problem $\mathcal{P} = (N, S, C, \phi^*, D)$ with $|N| = |S|$, deciding if $SM(C) = 0$ (given ϕ^*) is coNP-complete.*

Proof. (Proof sketch) *Membership in coNP*: a certificate that $SM(C) > 0$ is any valid mapping $\phi \neq \phi^*$, verifiable in polynomial time. *coNP-Hardness*: we reduce UNSAT to $SM(C) = 0$. Given a CNF formula ψ over Boolean variables $\{x_1, \dots, x_m\}$, construct a constraint-based NSL problem with $2m$ neural outputs (one per literal) and $2m$ concepts (paired truth values T_i, F_i). Add constraints enforcing neural outputs $\{n_i, \bar{n}_i\}$ must map to concepts from $\{T_i, F_i\}$. The intended mapping ϕ^* encodes the all-true assignment over $\{x_1, \dots, x_m, y\}$ for an auxiliary formula $\psi' = \psi \wedge \neg y$ with fresh variable y , ensuring ϕ^* is never a satisfying assignment of ψ' . Constraints use an auxiliary atom *alt* that activates iff $\phi \neq \phi^*$. When active, clause satisfaction constraints enforce that the encoded assignment satisfies ψ . By construction, $\phi^* \in \Phi_C$, and any $\phi \neq \phi^*$ is valid iff it encodes a satisfying assignment of ψ' . Since $\#SAT(\psi') = \#SAT(\psi)$ and ψ' is satisfiable iff ψ is, the reduction preserves both UNSAT and counting. Thus $SM(C) = 0$ iff ψ is unsatisfiable. \square

In this reduction, we do *not* exclude ϕ^* from the constraint encoding to ensure $\Phi_C \neq \emptyset$. This differs from the algorithmic enumeration (Algorithm 1) where we *do* exclude ϕ^* to enumerate alternative shortcut solutions.

Theorem 5 (Shortcut Counting is #P-complete). *Given a constraint-based NSL problem $\mathcal{P} = (N, S, C, \phi^*, D)$ with $|N| = |S|$, computing $SM(C) = |\Phi_C| - 1$ is #P-complete.*

Proof. (Proof sketch) *Membership in #P*: candidate mappings are verifiable in polynomial-time, so counting them is in #P. *#P-Hardness*: using the same reduction as Theorem 4, each $\phi \neq \phi^*$ in Φ_C corresponds bijectively to a satisfying assignment of ψ' , so $SM(C) = \#SAT(\psi') = \#SAT(\psi)$. Since #SAT is #P-complete (Valiant 1979), counting $SM(C)$ is #P-hard. \square

Theorem 6 (Minimal Repair over Candidate Pinning Constraints is NP-hard). *Given a constraint-based NSL problem $\mathcal{P} = (N, S, C, \phi^*, D)$ and a library \mathcal{L} of candidate pinning constraints, finding the smallest subset of \mathcal{L} whose addition to C achieves $SM^{\text{all}}(C') = 0$ is NP-hard.*

Proof. (Proof sketch) We reduce Set Cover to the minimal repair problem. Given universe $U = \{u_1, \dots, u_n\}$ and collection \mathcal{S} , construct a constraint-based NSL problem with n neural outputs, identity intended mapping, and no initial constraints (so $|\Phi_C| = n^n$, treating mappings as arbitrary functions rather than bijections). Each set $S_j \in \mathcal{S}$ corresponds to a candidate constraint that pins $\phi(n_i) = \phi^*(n_i)$ for all $u_i \in S_j$. Achieving $SM^{\text{all}}(C') = 0$ with $\leq \ell$ constraints requires pinning every position, which corresponds exactly to covering U with $\leq \ell$ sets. \square

This hardness result explains why Algorithm 2 cannot guarantee optimality (Theorem 3, Part 4), since finding the minimum number of repair constraints would itself require solving an NP-hard problem.

6 Experiments

We validate Algorithm 1 (verification) and Algorithm 2 (greedy repair) on 8 domains from the *RSBench* benchmark suite (Bortolotti et al. 2024), covering visual arithmetic (MNIST-Half, MNIST-XOR, MNIST-EvenOdd, MNIST-Math), visual logic (CLE4EVR, Kandinsky), and autonomous driving (BDD-OIA, SDD-OIA). Each domain defines a constraint-based NSL problem $\mathcal{P} = (N, S, C, \phi^*, D)$ with varying numbers of neural outputs ($|N|$ from 2 to 21), concepts, and constraint structures. All experiments use *clingo* with `--models=10,000` for bounded enumeration, averaged over 10 runs. In deployment, exact enumeration is often unnecessary: shortcut-freeness can be checked by searching for *any* alternative mapping, and shortcut analysis can be approximated via bounded enumeration (e.g., `--models=M`), randomized restarts, or incremental verification (Section 4.5). We report results under two encodings. The *base* encoding enforces only that each neural output maps to exactly one concept (i.e., ϕ is a function), but allows multiple outputs to share the same concept label; this corresponds to Φ_C^{all} in our notation. The *bijective* encoding additionally requires that each concept is used exactly once, corresponding to Φ_C^{bij} . Our theoretical results assume bijectivity, but we include the base encoding to show how much ambiguity bijectivity alone resolves. Table 1 summarizes the results.

The experiments paint an empirical *difficulty spectrum* that reflects our complexity classification. Verification (Algorithm 1) is practical: even on the largest domains (21 neural outputs), bounded enumeration completes in under a second. Repair difficulty, in contrast, varies sharply by domain: MNIST tasks repair in 1–2 iterations, BDD-OIA requires 21 pinning constraints, and SDD-OIA resists all strategies tested, which is consistent with the NP-hardness of minimal repair (Theorem 6). The complementary success of greedy and random strategies (greedy wins on BDD-OIA, random wins on CLE4EVR) suggests that hybrid strategies are a promising direction for future work.

The approximate shortcut counting method employed by *RSBench* (Bortolotti et al. 2024) enumerates reasoning shortcuts by generating all possible ground-truth concept vectors exhaustively and encoding the problem as SAT. This is tractable for small domains (e.g., MNIST-Half: $5^5 =$

3,125 vectors) but infeasible for larger ones, e.g., Kandinsky’s 18 ternary concepts yield $3^{18} \approx 387$ million vectors. We instead formulate shortcut enumeration as an ASP problem over minimal constraint sets. For MNIST domains, arithmetic constraints directly encode the task rules without requiring training samples. For non-MNIST domains (marked * in Tables 1 and 2), we use 2–8 synthetic samples targeting distinct aspects of each classification rule; in Kandinsky we reduced the concept domain from 18 to 8 (details in supplementary material). Since $C \subseteq C'$ implies $\Phi_{C'} \subseteq \Phi_C$ (monotonicity), our counts are upper bounds: $SM(C) = 0$ under minimal constraints guarantees shortcut-freeness under any superset, including the full training set.

Table 1: Reasoning shortcut detection: number of shortcuts and analysis time. *: Minimal synthetic samples, †: Reduced concept domain (from 18 to 8), N: number of neural outputs, Bij.: bijective encoding, Time represents average over 10 runs, 0 indicates UNSAT (no shortcuts exist).

Task	N	Shortcuts		Time (ms)	
		Base	Bij.	Base	Bij.
MNIST-XOR	2	1	1	353.3	301.7
MNIST-Half	5	2	0	291.2	290.7
MNIST-EvenOdd	10	35	1	299.9	299.4
MNIST-Math	4	4	0	302.5	301.4
BDD-OIA*	21	$\geq 10^4$	$\geq 10^4$	528.4	521.0
SDD-OIA*	21	$\geq 10^4$	$\geq 10^4$	527.5	523.2
CLE4EVR*	8	$\geq 10^4$	5,759	471.0	421.1
Kandinsky*†	8	$\geq 10^4$	3,455	470.7	377.3

Algorithm 1 successfully detects shortcuts across all 8 domains and fully enumerates valid mappings whenever the number of alternatives is below the 10,000-model cap; for the 4 domains where the count saturates at $\geq 10^4$, the reported value is a lower bound rather than an exhaustive count. Enforcing bijectivity eliminates all shortcuts in 2 of 8 domains ($SM^{\text{bij}} = 0$). This also shows that bijectivity combined with sufficiently strong constraints can achieve uniqueness, but is not universally sufficient.

Table 2: Repair comparison (base encoding): greedy vs. random repair. I.: iterations, C.: constraints added, S.: success rate (%), “–” indicates failure to repair within $T = 100$ iterations.

Task	Greedy Repair			Random Repair		
	I.	C.	S.	I.	C.	S.
MNIST-XOR	2.0	1.0	100%	2.0	1.0	100%
MNIST-Half	2.0	1.0	100%	2.0	1.0	100%
MNIST-EvenOdd	3.0	2.0	100%	3.0	2.0	100%
MNIST-Math	2.0	1.0	100%	2.0	1.0	100%
BDD-OIA*	22.0	21.0	100%	–	–	0%
SDD-OIA*	–	–	0%	–	–	0%
CLE4EVR*	–	–	0%	8.1	7.1	80%
Kandinsky*†	9.0	8.0	100%	9.0	8.0	100%

Table 2 compares greedy and random repair on the base

(non-bijective) encoding. Greedy repair (Algorithm 2) achieves full shortcut elimination in 6 of 8 domains, always terminating within the k -iteration bound of Theorem 3. The two strategies show no statistically significant difference overall, though they exhibit complementary strengths: greedy repair succeeds on BDD-OIA where random fails, while random repair succeeds on CLE4EVR where greedy does not. SDD-OIA alone resists all strategies, retaining $\geq 10^4$ shortcuts even with bijectivity enforced. Its 21 neural outputs with weak binary constraints yield a nearly unconstrained solution space where single-position pinning eliminates too few mappings per iteration, which is consistent with the NP-hardness of minimal repair (Theorem 6).

7 Conclusion

We formalized reasoning shortcuts in neurosymbolic learning as a constraint satisfaction problem and established both theoretical foundations and practical tools. On the theoretical side, we proved that discrimination is necessary for shortcut-freeness under bijective mappings (Theorem 1), but demonstrated via explicit counterexample that it is insufficient even when the constraint graph is connected (Example 4). We provided a complexity classification: deciding shortcut-freeness is coNP-complete, counting shortcuts is #P-complete, and minimal repair is NP-hard (Theorems 4–6). Sample complexity for disambiguation lies between $\lceil \log_r(k+1) \rceil$ and $|\Delta_C|$ label queries (Theorem 2). Our analysis applies to the constraint-based class of NeSy systems captured by Definition 1; characterizing analogous shortcut phenomena in systems without explicit logical constraints remains an interesting question for future work.

On the practical side, our ASP-based verification algorithm correctly detects all bijective shortcuts (Proposition 4), and greedy repair converges in at most k iterations (Theorem 3). Experiments across 8 RSBench domains validate both algorithms: bijectivity alone eliminates shortcuts in 2 of 8 domains, and greedy and random repair exhibit complementary strengths on the remainder, jointly resolving 7 of 8 domains.

Several directions remain open. Most pressing is a complete characterization of sufficient conditions for uniqueness beyond trivial automorphism groups: formalizing when constraint connectivity and strength jointly guarantee shortcut-freeness. Our bijective restriction ($|N| = |S|$, each concept used exactly once), while natural for balanced classification, excludes many practical scenarios; extending to non-bijective mappings and developing approximation algorithms for minimal repair are natural next steps.

Acknowledgements

This work has been supported by JSPS KAKENHI Grant Number JP25K03190 and JST CREST Grant Number JPMJCR22D3.

AI Declaration

The authors used a large language model (Claude) for improving the clarity of the writing. All technical content was developed and verified by the authors.

References

- Ahmed, K.; Chang, K.-W.; and den Broeck, G. V. 2023. A Pseudo-Semantic Loss for Autoregressive Models with Logical Constraints. In *37th Conference on Neural Information Processing Systems*.
- Andolfi, L., and Giunchiglia, E. 2025. Right for the Right Reasons: Avoiding Reasoning Shortcuts via Prototypical Neurosymbolic AI. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Badreddine, S.; d’Avila Garcez, A.; Serafini, L.; and Spranger, M. 2022. Logic Tensor Networks. *Artificial Intelligence* 303:103649.
- Besold, T. R.; d’Avila Garcez, A. S.; Bader, S.; Bowman, H.; Domingos, P. M.; Hitzler, P.; Kühnberger, K.-U.; Lamb, L. C.; Lima, P. M. V.; de Penning, L.; Pinkas, G.; Poon, H.; and Zaverucha, G. 2021. Neural-symbolic learning and reasoning: A survey and interpretation. In Hitzler, P., and Sarker, M. K., eds., *Neuro-Symbolic Artificial Intelligence: The State of the Art*, volume 342 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. 1–51.
- Bortolotti, S.; Marconato, E.; Carraro, T.; Morettin, P.; van Krieken, E.; Vergari, A.; Teso, S.; and Passerini, A. 2024. A Neuro-Symbolic Benchmark Suite for Concept Quality and Reasoning Shortcuts. In *Advances in Neural Information Processing Systems*, volume 37, 115861–115905.
- Bortolotti, S.; Marconato, E.; Morettin, P.; Passerini, A.; and Teso, S. 2025. Shortcuts and Identifiability in Concept-based Models from a Neuro-Symbolic Lens.
- Darwiche, A. 2023. Logic for Explainable AI. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 1–11. IEEE Computer Society.
- Gebser, M.; Kaufmann, B.; and Schaub, T. 2012. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence* 187–188:52–89.
- Gent, I.; Petrie, K.; and Puget, J.-F. 2006. Symmetry in Constraint Programming. In *Handbook of Constraint Programming*. Elsevier. 329–376.
- Ignatiev, A.; Marques-Silva, J.; Narodytska, N.; and Stuckey, P. J. 2021. Reasoning-Based Learning of Interpretable ML Models. In *Twenty-Ninth International Joint Conference on Artificial Intelligence*, volume 5, 4458–4465.
- Ismail, A. A.; Adebayo, J.; Bravo, H. C.; Ra, S.; and Cho, K. 2023. Concept Bottleneck Generative Models. In *The Twelfth International Conference on Learning Representations*.
- Koh, P. W.; Nguyen, T.; Tang, Y. S.; Mussmann, S.; Piereson, E.; Kim, B.; and Liang, P. 2020. Concept Bottleneck Models.
- Manhaeve, R.; Dumančić, S.; Kimmig, A.; Demeester, T.; and De Raedt, L. 2021. Neural probabilistic logic programming in DeepProbLog. *Artificial Intelligence* 298:103504.
- Marconato, E.; Bontempo, G.; Ficarra, E.; Calderara, S.; Passerini, A.; and Teso, S. 2023a. Neuro-Symbolic Continual Learning: Knowledge, Reasoning Shortcuts and Concept Rehearsal. In *Proceedings of the 40th International Conference on Machine Learning*, 23915–23936. PMLR.
- Marconato, E.; Teso, S.; Vergari, A.; and Passerini, A. 2023b. Not All Neuro-Symbolic Concepts Are Created Equal: Analysis and Mitigation of Reasoning Shortcuts. In *NeurIPS*, 72507–72539.
- Marconato, E.; Bortolotti, S.; van Krieken, E.; Vergari, A.; Passerini, A.; and Teso, S. 2024. BEARS Make Neuro-Symbolic Models Aware of their Reasoning Shortcuts. In *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, 2399–2433. PMLR.
- Marques-Silva, J., and Ignatiev, A. 2022. Delivering Trustworthy AI through Formal XAI. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, 12342–12350. AAAI Press.
- Shrotri, A. A.; Narodytska, N.; Ignatiev, A.; Meel, K. S.; Marques-Silva, J.; and Vardi, M. Y. 2022. Constraint-Driven Explanations for Black-Box ML Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 8304–8314.
- Skryagin, A.; Stammer, W.; Ochs, D.; Dhimi, D. S.; and Kersting, K. 2022. Neural-Probabilistic Answer Set Programming. In *Proceedings of the Nineteenth International Conference on Principles of Knowledge Representation and Reasoning*, 463–473.
- Skryagin, A.; Ochs, D.; Deibert, P.; Kohaut, S.; Dhimi, D. S.; and Kersting, K. 2024. Answer Set Networks: Casting Answer Set Programming into Deep Learning.
- Stoian, M. C.; Tatomir, A.; Lukasiewicz, T.; and Giunchiglia, E. 2024. PiShield: A PyTorch Package for Learning with Requirements. In *Thirty-Third International Joint Conference on Artificial Intelligence*, volume 9, 8805–8809.
- Takemura, A., and Inoue, K. 2026. Differentiable logic programming to mitigate reasoning shortcuts in neurosymbolic systems. In *Proceedings of the 42nd International Conference on Logic Programming (ICLP 2026)*, Electronic Proceedings in Theoretical Computer Science (EPTCS). To appear.
- Valiant, L. G. 1979. The complexity of computing the permanent. *Theoretical Computer Science* 8(2):189–201.
- van Krieken, E.; Minervini, P.; Ponti, E.; and Vergari, A. 2025. Neurosymbolic Reasoning Shortcuts under the Independence Assumption. In *Proceedings of The 19th International Conference on Neurosymbolic Learning and Reasoning*, 285–302. PMLR.
- Wang, K.; Tsamoura, E.; and Roth, D. 2023. On Learning Latent Models with Multi-Instance Weak Supervision. In *NeurIPS*, 9661–9694.
- Xu, J.; Zhang, Z.; Friedman, T.; Liang, Y.; and den Broeck, G. V. 2018. A semantic loss function for deep learning with symbolic knowledge. In Dy, J. G., and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 5498–5507. PMLR.

Yang, X.-W.; Wei, W.-D.; Shao, J.-J.; Li, Y.-F.; and Zhou, Z.-H. 2024. Analysis for Abductive Learning and Neural-Symbolic Reasoning Shortcuts. In *ICML*, 56524–56541.

Yang, Z.; Ishay, A.; and Lee, J. 2020. NeurASP: Embracing Neural Networks into Answer Set Programming. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 1755–1762.

A Worked Examples: Repair and Bijectivity

Example 6 (Repair Demonstration). We demonstrate greedy shortcut repair (Algorithm 2) on the 4-node problem (Example 1).

Initial State:

- Constraints: $C = \{C_1 : \phi(n_0) + \phi(n_3) = 3, C_2 : \phi(n_1) + \phi(n_2) = 3\}$
- Intended: $\phi^* = (0, 1, 2, 3)$
- Verification (Algorithm 1): $SM(C) = 7$ shortcuts

Detected shortcuts include:

- $\phi_2 = (0, 2, 1, 3)$ swaps component 2
- $\phi_7 = (3, 1, 2, 0)$ swaps component 1
- ... (5 more)

Running Algorithm 2 (greedy shortcut repair):

Iteration 1: Detect $\phi_2 = (0, 2, 1, 3)$

- Disagreement: $D_{sc} = \{n_1, n_2\}$ (since $\phi^*(n_1) = 1 \neq 2 = \phi_2(n_1)$)
- Add: $c_3 : \phi(n_1) = 1$
- This forces $\phi(n_2) = 2$ (from $C_2 : \phi(n_1) + \phi(n_2) = 3$)
- Component 2 is now fully determined: $\{n_1 \mapsto 1, n_2 \mapsto 2\}$
- Component 1 remains free: $\{n_0, n_3\}$ can be $(0, 3)$ or $(3, 0)$
- New: $SM(C') = 1$ (2 valid bijections remain: ϕ^* and $\phi_7 = (3, 1, 2, 0)$)

Iteration 2: Detect $\phi_7 = (3, 1, 2, 0)$

- Disagreement: $D_{sc} = \{n_0, n_3\}$
- Add: $c_4 : \phi(n_0) = 0$
- This forces $\phi(n_3) = 3$ (from C_1)
- Component 1 is now fully determined: $\{n_0 \mapsto 0, n_3 \mapsto 3\}$
- New: $SM(C'') = 0$ (shortcut-free)

Converged in 2 iterations (≤ 7 as in Theorem 3).

Example 7 (MNIST-Half: Bijectivity Ensures Uniqueness). We demonstrate that the MNIST-Half constraints from Example 2, when combined with bijectivity enforcement, actually achieve uniqueness.

Setup:

- Constraints: $C = \{C_1, C_2, C_3, C_4\}$ (as in Example 2)
- Intended: $\phi^* = (0, 1, 2, 3, 4)$

Analysis WITHOUT bijectivity:

From C_1, C_2 : $\phi(n_0) = 0, \phi(n_1) = 1$ (forced).

For $\{n_2, n_3, n_4\}$, allowing repetition:

- $\phi^* = (0, 1, 2, 3, 4)$ — bijective (valid)
- $\phi_1 = (0, 1, 3, 2, 3)$ — overloads 3: $C_3 : 3 + 2 = 5$, $C_4 : 3 + 3 = 6$ (valid)
- $\phi_2 = (0, 1, 4, 1, 2)$ — overloads 1: $C_3 : 4 + 1 = 5$, $C_4 : 4 + 2 = 6$ (valid)

Without bijectivity: $SM^{\text{all}}(C) \geq 2$ (multiple non-bijective shortcuts).

Analysis WITH bijectivity (Algorithm 1 enforcement):

Must use $\{0, 1, 2, 3, 4\}$ exactly once. With $\phi(n_0) = 0, \phi(n_1) = 1$ forced, $\{n_2, n_3, n_4\}$ must partition $\{2, 3, 4\}$.

For C_3 : $\phi(n_2) + \phi(n_3) = 5$ from $\{2, 3, 4\}$:

- Only possibility: $\{\phi(n_2), \phi(n_3)\} = \{2, 3\}$ (since $4 + 1 = 5$ but $1 \notin \{2, 3, 4\}$)

For C_4 : $\phi(n_2) + \phi(n_4) = 6$:

- If $\phi(n_2) = 2$: then $\phi(n_4) = 4$ and $\phi(n_3) = 3$ (from C_3), which recovers ϕ^*
- If $\phi(n_2) = 3$: then $\phi(n_4) = 3$, violating bijectivity

With bijectivity: $|\Phi_C^{\text{bij}}| = 1$, $SM^{\text{bij}}(C) = 0$, and it becomes shortcut-free.

In MNIST-Half, bijectivity + 4 arithmetic constraints suffice for uniqueness. However, bijectivity doesn't always eliminate shortcuts, see Example 1 where 8 bijective mappings exist, out of which 7 are shortcuts, even with bijectivity enforced. The difference is constraint strength: MNIST-Half has coupled arithmetic constraints linking all neural outputs in component 2, while Example 1 has weak sum constraints allowing independent permutations within disconnected components. This demonstrates that disconnection alone doesn't guarantee shortcuts when constraints are sufficiently restrictive.

B Non-Bijective vs. Bijective Shortcuts

Example 8 (Bijectivity Eliminates Overloading, Not All Shortcuts). We compare two scenarios to show that bijectivity can eliminate overloading shortcuts, but is not sufficient for shortcut-freeness.

Scenario 1: Bijectivity Eliminates Shortcuts (MNIST-Half)

From Example 7: With 4 constraints and bijectivity, $SM^{\text{bij}}(C) = 0$ (unique). Without bijectivity, $SM^{\text{all}}(C) \geq 2$ (overloading allows shortcuts).

Scenario 2: Bijectivity is Insufficient (4-Node Example)

From Example 1: With 2 constraints and bijectivity, $SM^{\text{bij}}(C) = 7$ (8 valid bijections). All shortcuts are bijective, and the disconnected constraint graph allows independent permutations.

Therefore, bijectivity can help eliminate certain non-bijective shortcuts, but does not by itself guarantee uniqueness. Whether bijectivity suffices for a given constraint set depends on constraint strength: MNIST-Half achieves uniqueness despite disconnected components because its arithmetic constraints, together with bijectivity, fully determine each component; the 4-node example leaves independent transpositions unresolved.

For systems with disconnected constraint graphs, additional constraints or labeled examples are needed (Algorithm 2, Section 4.5).

C Worked Example: Active Learning

Example 9 (Uncertainty Sampling on 4-Node Problem). Apply uncertainty sampling (Section 4.5) to the 4-node addition problem (Example 1) with $SM(C) = 7$ shortcuts.

Initial State: $|\Psi| = 8$ candidate bijections:

- $\phi_1 = (0, 1, 2, 3), \phi_2 = (0, 2, 1, 3)$ (component 1: $(0, 3)$)
- $\phi_3 = (1, 0, 3, 2), \phi_4 = (1, 3, 0, 2)$ (component 1: $(1, 2)$)
- $\phi_5 = (2, 0, 3, 1), \phi_6 = (2, 3, 0, 1)$ (component 1: $(2, 1)$)
- $\phi_7 = (3, 1, 2, 0), \phi_8 = (3, 2, 1, 0)$ (component 1: $(3, 0)$)

Step 1: Compute disagreement

- n_0 : Candidates use $\{0, 1, 2, 3\} \rightarrow d(n_0) = 4$
- n_1 : Candidates use $\{0, 1, 2, 3\} \rightarrow d(n_1) = 4$
- n_2 : Candidates use $\{0, 1, 2, 3\} \rightarrow d(n_2) = 4$
- n_3 : Candidates use $\{0, 1, 2, 3\} \rightarrow d(n_3) = 4$

All positions have maximum disagreement. Select $n^* = n_0$ (arbitrary tie-breaking).

Step 2: Query and eliminate

Query label: $\phi^*(n_0) = 0$

Eliminate all $\phi \in \Psi$ with $\phi(n_0) \neq 0$: Remove $\{\phi_3, \phi_4, \phi_5, \phi_6, \phi_7, \phi_8\}$

New $|\Psi| = 2$: $\{\phi_1, \phi_2\}$ (both have $\phi(n_0) = 0, \phi(n_3) = 3$)

Step 3: Second query

Compute disagreement:

- n_0 : All agree ($\phi(n_0) = 0$) $\rightarrow d(n_0) = 1$
- n_1 : Candidates use $\{1, 2\} \rightarrow d(n_1) = 2$
- n_2 : Candidates use $\{1, 2\} \rightarrow d(n_2) = 2$
- n_3 : All agree ($\phi(n_3) = 3$) $\rightarrow d(n_3) = 1$

Select $n^* = n_1$ (highest disagreement, or n_2).

Query label: $\phi^*(n_1) = 1$

New $|\Psi| = 1$: $\{\phi_1 = \phi^*\}$ (identified)

Uncertainty sampling identified ϕ^* in 2 queries, matching the lower bound $\lceil \log_4(8) \rceil = 2$ from Theorem 2. The favorable structure (component 1 choices fully determine component 2) enables this optimal performance.

D Concrete ASP Encoding

Listing 1: Example ASP encoding for Example 1

```

1 % Domain declaration for safety
2 val(0..3).
3
4 % Neural outputs
5 neural(n0). neural(n1). neural(n2).
   neural(n3).
6
7 % Concepts derived from value domain
8 concept(V) :- val(V).
9
10 % Bijectivity: exactly one neural output
   per concept
11 1 { maps_to(N,S) : concept(S) } 1 :-
   neural(N).
12 1 { maps_to(N,S) : neural(N) } 1 :-
   concept(S).
13
14 % Value extraction with domain guard
15 value(N, V) :- maps_to(N, V), val(V).
16
17 % Constraints: encode arithmetic with
   domain guards
18 :- value(n0, V0), value(n3, V3), val(V0)
   , val(V3), V0 + V3 != 3.
19 :- value(n1, V1), value(n2, V2), val(V1)
   , val(V2), V1 + V2 != 3.
20
21 % Exclude intended mapping phi* =
   (0,1,2,3)

```

```

22 % (used in Algorithm 1 for enumeration)
23 :- maps_to(n0,0), maps_to(n1,1),
24     maps_to(n2,2), maps_to(n3,3).
25
26 #show maps_to/2.
27 % When executed with enumeration mode,
   this will output 7 mappings.

```

Listing 2: ASP encoding for MNIST-Half (Example 2) without bijectivity constraint

```

1 % Domain declaration
2 val(0..4).
3
4 % Neural outputs
5 neural(n0). neural(n1). neural(n2).
   neural(n3). neural(n4).
6
7 % Concepts
8 concept(V) :- val(V).
9
10 % Mapping (allowing non-bijective)
11 1 { maps_to(N,S) : concept(S) } 1 :-
   neural(N).
12
13 % MNIST-Half constraints
14 :- maps_to(n0, V0), V0 + V0 != 0.
15 :- maps_to(n0, V0), maps_to(n1, V1), V0
   + V1 != 1.
16 :- maps_to(n2, V2), maps_to(n3, V3), V2
   + V3 != 5.
17 :- maps_to(n2, V2), maps_to(n4, V4), V2
   + V4 != 6.
18
19 % Exclude intended mapping phi* =
   (0,1,2,3,4)
20 :- maps_to(n0,0), maps_to(n1,1), maps_to
   (n2,2),
21     maps_to(n3,3), maps_to(n4,4).
22
23 #show maps_to/2.
24 % When executed with enumeration mode,
   this will output 2 mappings.

```

Listing 3: ASP encoding for MNIST-Half (Example 2) with bijectivity constraint

```

1 % Domain declaration
2 val(0..4).
3
4 % Neural outputs
5 neural(n0). neural(n1). neural(n2).
   neural(n3). neural(n4).
6
7 % Concepts
8 concept(V) :- val(V).
9
10 % Bijectivity: exactly one neural output
   per concept
11 1 { maps_to(N,S) : concept(S) } 1 :-
   neural(N).
12 1 { maps_to(N,S) : neural(N) } 1 :-
   concept(S).
13
14 % MNIST-Half constraints
15 :- maps_to(n0, V0), V0 + V0 != 0.
16 :- maps_to(n0, V0), maps_to(n1, V1), V0

```

```

+ V1 != 1.
17 :- maps_to(n2, V2), maps_to(n3, V3), V2
+ V3 != 5.
18 :- maps_to(n2, V2), maps_to(n4, V4), V2
+ V4 != 6.
19
20 % Exclude intended mapping phi* =
(0,1,2,3,4)
21 :- maps_to(n0,0), maps_to(n1,1), maps_to
(n2,2),
22 maps_to(n3,3), maps_to(n4,4).
23
24 #show maps_to/2.
25 % UNSATISFIABLE

```

Listing 4: ASP encoding for Example 4 (modulo successor)

```

1 % Domain declaration
2 val(0..2).
3
4 % Neural outputs
5 neural(n0). neural(n1). neural(n2).
6
7 % Concepts
8 concept(V) :- val(V).
9
10 % Bijectivity
11 1 { maps_to(N,S) : concept(S) } 1 :-
neural(N).
12 1 { maps_to(N,S) : neural(N) } 1 :-
concept(S).
13
14 % Value extraction
15 value(N, V) :- maps_to(N, V), val(V).
16
17 % Modulo successor constraints
18 :- value(n0, V0), value(n1, V1), val(V0)
, val(V1),
19 V1 != (V0 + 1) \ 3.
20 :- value(n1, V1), value(n2, V2), val(V1)
, val(V2),
21 V2 != (V1 + 1) \ 3.
22
23 % Exclude intended mapping phi* =
(0,1,2)
24 :- maps_to(n0,0), maps_to(n1,1), maps_to
(n2,2).
25
26 #show maps_to/2.
27 % When executed with enumeration mode,
this will output 2 mappings.

```

E Full Proofs

E.1 Shortcut-Freeness is coNP-complete

Proof. Membership in coNP: A certificate that $SM(C) > 0$ is a valid mapping $\phi \neq \phi^*$ with $\phi \in \Phi_C$.

Verification: (i) Checking $\phi \neq \phi^*$ takes $O(r)$ time. (ii) Checking $\phi \in \Phi_C$ requires evaluating constraints, which takes $O(|C| \cdot \text{poly}(r))$ time. Both are polynomial, so the complement problem is in NP, thus $SM(C) = 0$ is in coNP.

Hardness (coNP-hard). We reduce UNSAT to deciding $SM(C) = 0$ given ϕ^* .

Reduction construction: Let ψ be a CNF formula over Boolean variables $\{x_1, \dots, x_m\}$. We reduce from ψ via an

auxiliary formula $\psi' = \psi \wedge \neg y$, where y is a fresh variable not appearing in ψ . Note that $\#SAT(\psi') = \#SAT(\psi)$, and ψ' is satisfiable iff ψ is satisfiable. The fresh variable ensures that the all-true assignment never satisfies ψ' , which is what makes the bijection between alternative mappings and satisfying assignments work in the construction below.

Construct a constraint-based NSL problem with:

- *Neural outputs* represent literals over the variables of ψ' :
 $N = \{n_i, \bar{n}_i : i \in [m]\} \cup \{n_y, \bar{n}_y\}$.
– n_i represents positive literal x_i
– \bar{n}_i represents negative literal $\neg x_i$
Thus $|N| = 2(m+1)$.
- *Concepts* represent truth values: $S = \{T_i, F_i : i \in [m]\} \cup \{T_y, F_y\}$.
– T_i represents “variable x_i is true”
– F_i represents “variable x_i is false”
– T_i and F_i are mutually exclusive truth values for variable x_i
Thus $|S| = 2(m+1)$.
- *Intended mapping* corresponds to the all-true assignment over $\{x_1, \dots, x_m, y\}$:

$$\begin{aligned} \phi^*(n_i) &= T_i, & \phi^*(\bar{n}_i) &= F_i & \text{for } i \in [m] \\ \phi^*(n_y) &= T_y, & \phi^*(\bar{n}_y) &= F_y \end{aligned}$$

Interpretation: Any admissible bijection $\phi : N \rightarrow S$, i.e., any bijection satisfying the pair-consistency constraints below, encodes a truth assignment over $\{x_1, \dots, x_m, y\}$, where variable v is true iff $\phi(n_v) = T_v$.

Constraint encoding: Bijectivity is enforced by constraints (each neural output maps to exactly one concept, each concept used exactly once). We additionally enforce pair-consistency: for each variable $v \in \{x_1, \dots, x_m, y\}$, the outputs n_v and \bar{n}_v may only map to the concepts T_v and F_v . Together with bijectivity, this ensures that exactly one of the following holds: $\phi(n_v) = T_v, \phi(\bar{n}_v) = F_v$ or $\phi(n_v) = F_v, \phi(\bar{n}_v) = T_v$. Thus every admissible bijection corresponds uniquely to a Boolean assignment. To detect deviations from ϕ^* , introduce auxiliary atom *alt*:

For each variable $v \in \{x_1, \dots, x_m, y\}$:

$$alt \leftarrow \text{maps_to}(n_v, F_v)$$

$$alt \leftarrow \text{maps_to}(\bar{n}_v, T_v)$$

Note that *alt* is false iff $\phi = \phi^*$, and true otherwise.

For each clause D_j of ψ' (including the unit clause $\neg y$), encode clause satisfaction so that *satisfied*(D_j) holds exactly when the encoded assignment satisfies D_j . For each literal ℓ appearing in D_j , add the rule:

$$\text{satisfied}(D_j) \leftarrow \text{maps_to}(n_v, T_v) \quad \text{if } \ell = x_v$$

$$\text{satisfied}(D_j) \leftarrow \text{maps_to}(\bar{n}_v, T_v) \quad \text{if } \ell = \neg x_v$$

Both rules fire exactly when their literal is satisfied: x_v is true iff $\phi(n_v) = T_v$, and $\neg x_v$ is true iff $\phi(\bar{n}_v) = T_v$ (equivalently, $\phi(n_v) = F_v$).

Finally, enforce that whenever *alt* is true, every clause must be satisfied:

$$\perp \leftarrow alt, \text{not satisfied}(D_j) \quad \text{for each clause } D_j \text{ of } \psi'.$$

Translation: “If the mapping differs from ϕ^* (alt is true), then every clause of ψ' must be satisfied by the encoded assignment.”

Observation: By construction, $\phi^* \in \Phi_C$: when $\phi = \phi^*$, the atom alt is false and the clause constraints fire vacuously. Note that ϕ^* encodes the all-true assignment, which fails the unit clause $\neg y$ of ψ' ; this is fine because the clause constraints only fire when alt is true.

Any admissible bijection $\phi \neq \phi^*$ makes alt true, and is valid iff its encoded assignment satisfies every clause of ψ' .

Correctness:

$$\begin{aligned} & \exists \phi \neq \phi^* \text{ with } \phi \in \Phi_C \\ & \Leftrightarrow \exists \text{ assignment over } \{x_1, \dots, x_m, y\} \text{ satisfying } \psi' \\ & \Leftrightarrow \psi' \text{ is satisfiable} \\ & \Leftrightarrow \psi \text{ is satisfiable.} \end{aligned}$$

Therefore, ψ is unsatisfiable $\Leftrightarrow SM(C) = 0$. The reduction produces a constraint set of size $O(|\psi|)$ (linear in the number of literal occurrences in ψ), and is computable in polynomial time. Thus the problem is coNP-hard. Combined with membership in coNP, deciding $SM(C) = 0$ is coNP-complete. \square

E.2 Shortcut Counting is #P-complete

Proof. Membership in #P: The function $SM(C)$ counts bijective mappings satisfying constraints in C that are distinct from ϕ^* .

For any candidate mapping $\phi : N \rightarrow S$, we can verify in polynomial time:

- **Bijectivity:** Check each concept used exactly once ($O(r)$ time)
- **Constraint satisfaction:** Evaluate all constraints under ϕ ($O(|C| \cdot \text{poly}(r))$ time)
- **Non-identity:** Check $\phi \neq \phi^*$

Since these witnesses are polynomial-time verifiable and we are counting them, $SM(C)$ is in #P.

Hardness (#P-hard): We reduce #SAT (counting satisfying assignments of Boolean formulas) to computing $SM(C)$.

Reduction construction: Given CNF formula ψ over variables $\{x_1, \dots, x_m\}$, we apply the same reduction as in Theorem 4, introduce a fresh variable y and set $\psi' = \psi \wedge \neg y$. Construct a constraint-based NSL problem with:

- **Neural outputs:** $N = \{n_i, \bar{n}_i : i \in [m]\} \cup \{n_y, \bar{n}_y\}$.
- **Concepts:** $S = \{T_i, F_i : i \in [m]\} \cup \{T_y, F_y\}$.
- **Intended mapping:** $\phi^*(n_v) = T_v$, $\phi^*(\bar{n}_v) = F_v$ (the all-true assignment over the extended variable set)
- **Constraints:** enforcing bijectivity, pair-consistency, and satisfaction of every clause of ψ' when alt is true.

Counting correspondence:

Each bijection $\phi \neq \phi^*$ in Φ_C corresponds to a satisfying truth assignment of ψ' :

- ϕ^* itself always satisfies C (as shown in Theorem 4; clause constraints fire vacuously when alt is false)

- Each $\phi \neq \phi^*$ encodes an assignment over $\{x_1, \dots, x_m, y\}$ where variable v is true iff $\phi(n_v) = T_v$
- $\phi \in \Phi_C$ iff the encoded assignment satisfies all clauses of ψ'

Therefore:

$$\begin{aligned} SM(C) &= |\Phi_C| - 1 \\ &= |\{\phi \in \Phi_C : \phi \neq \phi^*\}| \\ &= |\{\text{satisfying assignments of } \psi'\}| \\ &= \#SAT(\psi') = \#SAT(\psi). \end{aligned}$$

Since #SAT is #P-complete (Valiant 1979) and this reduction is polynomial in the size of the input formula ψ , computing $SM(C)$ is #P-hard. Combined with #P membership, computing $SM(C)$ is #P-complete. \square

E.3 Minimal Repair over Candidate Pinning Constraints is NP-hard

Proof. Problem: Given C , ϕ^* , budget ℓ , and a fixed library of candidate constraints, determine whether we can add $\leq \ell$ candidate constraints to achieve $SM^{\text{all}}(C') = 0$. Here SM^{all} denotes shortcut multiplicity over all mappings, without enforcing bijectivity.

Background: The classical SET COVER problem asks: given universe U and collection of subsets $\mathcal{S} = \{S_1, \dots, S_p\}$ with $S_i \subseteq U$, can we select $\leq \ell$ sets whose union equals U ? This problem is NP-complete.

Reduction from Set Cover:

Intuition: Each element of the universe U corresponds to a “degree of freedom” in our constraint-based NSL problem. Each set $S_j \in \mathcal{S}$ corresponds to a potential constraint that eliminates certain mappings. Covering all elements corresponds to eliminating all shortcuts.

Construction: Given universe $U = \{u_1, \dots, u_n\}$ with $n \geq 2$, and collection $\mathcal{S} = \{S_1, \dots, S_p\}$:

- **Neural outputs:** $N = \{n_1, \dots, n_n\}$ (one per universe element)
- **Concepts:** $S = \{0, 1, \dots, n-1\}$
- **Initial constraints:** $C = \{\}$ (empty); bijectivity is not enforced in this reduction.
 - Each $\phi : N \rightarrow S$ is unconstrained, so $|\Phi_C^{\text{all}}| = |S|^{|N|} = n^n$
 - Thus $SM^{\text{all}}(C) = n^n - 1$ (exponentially many shortcuts)
- **Intended mapping:** $\phi^* = \{n_i \mapsto i-1\}$ (identity: $n_1 \mapsto 0, n_2 \mapsto 1, \dots$)
- **Candidate constraints:** For each set $S_j \in \mathcal{S}$, define:

$$c_j : \bigwedge_{u_i \in S_j} \phi(n_i) = \phi^*(n_i)$$

This constraint pins each neural output n_i with $u_i \in S_j$ to its intended value $\phi^*(n_i)$.

Key correspondence:

Without bijectivity, an unpinned position n_i can independently take any value in S , so $\phi(n_i) \neq \phi^*(n_i)$ is available as a deviation. Hence $SM^{\text{all}}(C') = 0$ iff every position is pinned by some selected c_j :

- Adding constraint c_j eliminates all mappings that differ from ϕ^* on any neural output n_i where $u_i \in S_j$.
- For uncovered u_i , the mapping $\phi(n_i) = (\phi^*(n_i) + 1) \bmod n$ and $\phi(n_k) = \phi^*(n_k)$ for all $k \neq i$ is a valid shortcut, so $SM^{\text{all}}(C') > 0$.
- Therefore $SM^{\text{all}}(C') = 0$ iff selected sets cover U .

Correctness:

$$\begin{aligned} \exists R \subseteq \{c_1, \dots, c_p\} \text{ with } |R| \leq \ell \text{ and } SM^{\text{all}}(C \cup R) = 0 \\ \Leftrightarrow \exists \text{ selection of } \leq \ell \text{ constraints from } \{c_1, \dots, c_p\} \\ \text{forcing } \phi(n_i) = \phi^*(n_i) \text{ for all } i \\ \Leftrightarrow \exists \text{ selection of } \leq \ell \text{ sets from } \mathcal{S} \text{ covering } U \end{aligned}$$

The minimum number of candidate constraints needed to achieve $SM^{\text{all}}(C') = 0$ exactly equals the minimum set cover size. Since Set Cover is NP-complete, the decision version of minimal repair over candidate pinning constraints is NP-hard. \square

F Additional Label Selection Strategies and Practitioner’s Guide

F.1 Random Sampling

Observation 1 (Random Sampling – Empirical). If examples are selected uniformly at random from N , the expected number of labeled examples appears empirically to be approximately:

$$\mathbb{E}[m] \approx \frac{|N|}{|\Delta_C|} \cdot k \log_2 k$$

This assumes: (1) disagreement positions are hit with probability $|\Delta_C|/|N|$, and (2) once informative examples are found, coupon-collector analysis suggests $O(k \log_2 k)$ examples suffice.

F.2 Greedy Disambiguation

Strategy: At each step, select the example that eliminates the most shortcuts:

1. For each $n \in N$ and $s \in S$, compute: $e(n, s) = |\{\phi \in \Psi : \phi(n) \neq s\}|$
2. Select $(n^*, s^*) = \arg \max_{n,s} e(n, s)$
3. Query label (n^*, s^*) and verify $s^* = \phi^*(n^*)$
4. Remove all $\phi \in \Psi$ with $\phi(n^*) \neq s^*$

This strategy explicitly maximizes shortcuts eliminated per query, closely approximating the oracle.

Observation 2 (Greedy Performance – Information-Theoretic). Greedy disambiguation typically achieves $O(k \log_2 k)$ or better queries when shortcuts have diverse disagreement patterns (information-theoretic bound). However, we do not provide a formal worst-case guarantee, as performance depends on the structure of Φ_C .

F.3 Strategy Comparison

The oracle lower bound is $\lceil \log_r(k+1) \rceil$ from Theorem 2, with upper bound $|\Delta_C|$. Uncertainty sampling (Proposition 3) achieves $O(\min(k, |\Delta_C|))$ queries; greedy disambiguation typically achieves $O(k \log_2 k)$; and random sampling requires $O(|N|/|\Delta_C| \cdot k \log_2 k)$ expected queries. All bounds except uncertainty sampling are information-theoretic without formal worst-case guarantees.

F.4 Practitioner’s Guide

For practitioners using existing neurosymbolic frameworks (DeepProbLog, Logic Tensor Networks, NeurASP, etc.), we recommend the following workflow.

Step 1: Export Constraints. Convert logical rules into ASP format, define the intended mapping ϕ^* , and identify neural outputs N and concepts S with $|N| = |S|$.

Step 2: Verify Shortcut-Freeness. Run Algorithm 1. If the result is SHORTCUT-FREE, the unique valid mapping is confirmed. Otherwise, proceed to Step 3.

Step 3: Eliminate Shortcuts. Choose one of three approaches: (A) *Automated repair*: use Algorithm 2, which converges in $\leq k$ iterations; (B) *Active learning*: use uncertainty sampling (Proposition 3), requiring $\leq \min(k, |\Delta_C|)$ labels; (C) *Manual*: analyze detected shortcuts and add domain-specific constraints.

Step 4: Verify Improvement. Re-run Algorithm 1 to confirm shortcut elimination.

G Experimental Domain Details

Table 3 summarizes the encoding used for each experimental domain, including any simplifications relative to the original RSBench specification (Bortolotti et al. 2024).

Table 3: Domain encoding details. S.: Number of sample data, Org. $|N|$: original number of neural outputs in RSBench before reduction, “Constraint-only” means arithmetic rules are encoded directly without training samples.

Domain	$ N $	S.	Org. $ N $	Notes
MNIST-XOR	2	0	2	Constraint-only
MNIST-Half	5	0	5	Constraint-only
MNIST-EvenOdd	10	0	10	Constraint-only
MNIST-Math	4	0	4	Constraint-only
BDD-OIA*	21	8	21	Synthetic samples
SDD-OIA*	21	8	21	Synthetic samples
CLE4EVR*	8	4	8	Synthetic samples
Kandinsky*†	8	2	18	Reduced domain

MNIST domains. Arithmetic constraints (e.g., $\phi(n_0) + \phi(n_1) = 1$) capture all information relevant to shortcut detection. No training samples are needed and no domain reduction is applied.

BDD-OIA / SDD-OIA. The full 21 concepts are preserved. 8 hand-crafted symbolic samples cover key logical rules (traffic light states, obstacle presence, lane availability). This may miss constraints arising only from rare co-occurrences in the full dataset, potentially overestimating shortcut counts.

Kandinsky. The concept domain is reduced from 18 (3 figures \times 3 objects \times 2 attributes) to 8 (2 figures \times 2 objects \times 2 attributes) for ASP tractability. 2 synthetic samples encode the classification rule. Shortcut counts reflect the reduced domain and represent a lower bound on the full 18-concept problem.

CLE4EVR. All 8 concepts (2 objects \times 4 attributes) are preserved. 4 synthetic samples encode the classification rule:

```
And(Eq(color_1, color_2),  
    Eq(shape_1, shape_2),  
    Eq(material_1, material_2))
```