# CONSISTENCY IS NOT IDENTIFICATION: WHAT REUSE REVEALS ABOUT LATENT ACTIONS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Latent-action models learn action representations from unlabeled video by encoding observed transitions into codes. Many methods shape these representations using algebraic constraints or cycle-consistency checks that re-encode a model's own predictions. However, we prove that a model can fit the observed trajectories and satisfy these constraints yet predict the wrong effect when an action is reused in a new state. In this sense, consistency is not identification. We then show which data resolve this ambiguity. For reversible actions on a connected finite state graph, repeated-command trajectories and trials that re-execute a command after an intervening one together uniquely determine the set of true action operators. To evaluate learned actions in this setting, we further show that, for deterministic models with as many codes as commands, the distance between learned and true operators is bounded by local reconstruction error plus reuse error at independently sampled targets. In controlled experiments, training on re-execution trials learns the true state-transition operators on every restart, whereas training only on repeated-command data can yield consistent but incorrect operators. In LAPO models across 15 Procgen environments, codes inferred at the target select the correct outcome in $54.4\%$ of eligible pairs, compared with $33.0\%$ for codes transferred from another state on the same pairs. These results show that the gap between reconstructing observed transitions and reusing actions in new states also appears in latent-action models trained on video.

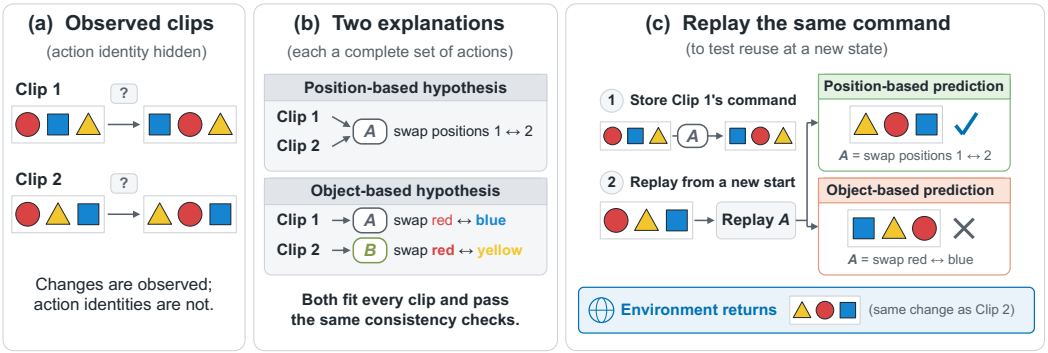



Figure 1: **Two explanations of the same clips, separated by reuse.** (a) Two observed transitions; which command produced each is hidden. (b) A position-based and an object-based family of six actions both reproduce every clip and pass the same consistency checks, yet disagree on whether the two clips share a command. (c) Replaying the command that produced Clip 1 from a new start state separates them: the environment returns the position-based prediction, the same change as Clip 2.

## 1 INTRODUCTION

Latent actions can be learned from observation alone (Edwards et al., 2019). Several recent latent-action models encode an observed transition into a code and condition a dynamics model on that code (Schmidt & Jiang, 2024; Bruce et al., 2024; Ye et al., 2025; Gao et al., 2025). To serve as



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.

actions, these codes must also produce the effect of the same environment command in another state. We call this *action reuse*.

Fitting observed transitions does not settle reuse, because a single transition can support different rules. For instance, suppose red, blue and yellow objects occupy positions 1, 2 and 3. Swapping positions 1 and 2 gives the same result as swapping the red and blue objects. If the action is reused from a state with blue in position 3, however, the two rules differ (Figure 1). One rule is tied to positions, the other to object identities (Appendix A.2), and only the effect of the action in another state separates them.

Beyond transition prediction, some methods train latent actions with compositional constraints (Rybkin et al., 2019; Wei et al., 2026a; Tang et al., 2026) or action-cycle objectives (Chen et al., 2026). Other studies evaluate latent actions through action decoding or downstream control (Nikulin et al., 2025; Nie et al., 2026; Bu et al., 2026). We focus on what internal consistency alone establishes. Algebraic and cycle checks, including the source–target–source cycle of Garrido et al. (2026) and Zhang et al. (2026), do not observe the environment's response when a code is reused in another state. Passing them therefore does not guarantee that the code reproduces the command's effect there. We prove this by constructing observationally equivalent operator families that pass the same algebraic and cycle checks yet disagree under reuse, as the two rules above do. Consequently, no *target-blind* check, one without the environment's outcome at the target state, can guarantee correct reuse. In this sense, **consistency is not identification**. Non-uniqueness under cycle consistency is known in image translation (Moriakov et al., 2020). Here we study the state-transition operators that latent actions represent.

We therefore ask which observations identify the action operators, and which measurements bound a trained model's reuse error. We study finite, fully observed systems with distinct command effects, which isolates cross-state correspondence from perception and local action ambiguity.

**Contributions.** First, we show that trajectories that repeat a command, combined with trials that re-execute it after an intervening one, identify the operators for reversible commands on a connected transition graph under full-support sampling, and that neither suffices alone (Figure 2). Second, to assess learned actions without labels, we prove that reuse error at an independently drawn target upper-bounds the risk of the best fixed command-to-code dictionary, provided the decoder reads only the target and the code. For deterministic models with as many codes as commands, reuse error plus local reconstruction error also bounds the distance between learned and true operators (Figure 3). Third, experiments show that reproducing observed transitions does not ensure correct reuse. In controlled systems, training on re-execution trials recovers the true operators on every restart, whereas training only on repeated-command data can return consistent but incorrect operators. In frozen LAPO models on 15 Procgen environments (Cobbe et al., 2020), codes carried from another state succeed on $33.0\%$ of eligible pairs, against $54.4\%$ for codes inferred at the target (Section 5).

## 2 Observation model and recovery criteria

**Commands, states and codes.** Let $S$ be a finite state space ($|S| = N$) and $A$ a set of $m$ *primitive commands*, where each command is a deterministic map $h_a \colon S \to S$. The true operator family is $H = (h_a)_{a \in A}$. We require commands to have distinct effects at every state, a property we call *pointwise distinguishability*:

$$a \neq b \implies h_a(x) \neq h_b(x) \quad \text{for every } x \in S. \tag{1}$$

When there are fewer commands than states, the set of *valid edges*

$$\mathcal{E}_H = \{(x, h_a(x)) : x \in S,\ a \in A\} \tag{2}$$

cannot cover all of $S \times S$. On any valid edge, pointwise distinguishability uniquely determines the command mapping $x$ to $y$, denoted $a_H(x, y)$. We do not assume reversibility unless noted (Table 6 summarizes the structural assumptions).

**Latent-action model.** A model encodes a transition and applies the inferred code to a new target state:

$$E \colon S \times S \to \mathcal{I}, \quad c = E(x, y), \qquad D \colon S \times \mathcal{I} \to S, \quad \widehat{w} = D(z, c). \tag{3}$$

Unless a result states otherwise, $\mathcal{I}$ has $m$ discrete codes and $E$ and $D$ are deterministic. We separate the latent code space $\mathcal{I}$ from the true command set $A$ because the model never observes the true

action labels. A *structured candidate* is a pointwise-distinguishable family of permutations $K = (k_c)_{c\in\mathcal{I}}$. For such a candidate, the code taking $x$ to $y$ is uniquely determined on its valid edges, denoted $c_K(x, y)$. This induces the model $E_K(x, y) = c_K(x, y)$ and $D_K(z, c) = k_c(z)$, which exactly reconstructs every valid edge: $D_K(x, E_K(x, y)) = y$. This alone does not guarantee correct action reuse at another state.

**Data-collection protocols.** The learner observes state trajectories without command labels and knows which steps share a command under each collection protocol. A *persistence run* samples a start state and a command uniformly, then holds that command fixed to produce $x, h_a(x), h_a^2(x), \ldots$. We denote the population law of its first $T$ transitions by $W_T(H)$. Holding an unnamed command fixed provides temporal structure that generic video lacks (Appendix B.2). *Native reuse* draws the start state and two commands independently and uniformly, the second possibly equal to the first, then repeats the first command. Formally, given $y = h_a(x)$, $z = h_b(y)$ and $w = h_a(z)$, the target $z$ is the state the intervening command reached, and we define its 0–1 risk as $r_{\text{nat}} = \Pr[\widehat{w} \neq w]$. The predictor receives only $(x, y, z)$, and $w$ serves as a training or evaluation target. *Independent-target reuse* instead samples $z$ independently of the source transition before applying the same command.

**The regular specialization.** We define the *regular specialization* as the case where $m = N$ and every $h_a$ is a permutation. Here, $\mathcal{E}_H = S \times S$ and $H$ becomes *sharply transitive*: every ordered state pair is connected by exactly one operator. While a composition-closed sharply transitive family forms a *regular subgroup*, sharp transitivity alone guarantees neither algebraic closure nor an identity element (Appendix A.3). Section 3.2 and the controlled experiments use this specialization.

**Recovery criteria.** Family recovery means equality between the learned and generating sets of state-transition maps, allowing a global permutation of code indices but keeping state labels fixed. Exact fit requires only that the model explain the observations.

## 3 What observations identify

This section asks which of the protocols in Section 2 identify the true family. Persistence runs fix the valid edges and a continuation rule, and algebraic consistency narrows the compatible families without always resolving them. For reversible commands on a connected transition graph, adding native reuse trials identifies the family.

### 3.1 Valid edges and the continuation rule

A persistence trajectory maintains one command across consecutive transitions. We define the *continuation rule* as the state reached by re-applying the command responsible for the initial transition $(x, y)$:

$$C_H(x, y) \;=\; h_{a_H(x,y)}(y), \qquad (x, y) \in \mathcal{E}_H. \tag{4}$$

Two operator families sharing identical valid edges and continuation rules are called *run-equivalent*.

**Proposition 3.1** (Sufficiency of two-transition persistence for run-equivalence)**.** *Let $H$ be a pointwise-distinguishable family of $m$ maps over $S$. Let $W_T(H)$ denote the population law of a persistence trajectory of $T$ transitions, where the initial state and the fixed command are sampled independently and uniformly. Then $W_1(H)$ is uniform over the valid edges $\mathcal{E}_H$, and $W_2(H)$ determines $C_H$ over those edges. Because every longer trajectory obeys the recurrence $X_{t+1} = C_H(X_{t-1}, X_t)$, the equality $W_2(H) = W_2(H')$ implies $W_T(H) = W_T(H')$ for all horizons $T \geq 2$.*

Intuitively, a run applies one command again and again, so each state is fixed by the two before it, and two-transition data already reveal that rule. Persistence data of any length therefore separate two families only through their valid edges and continuation rules. In the regular specialization, the valid edges are all state pairs. For the XOR translations of $\mathbb{Z}_2^3$, every command is self-inverse, so the continuation always returns to the preceding state, and every sharply transitive family of involutions shares this rule, so persistence alone cannot separate any of them.

### 3.2 Algebraic consistency and composition closure

We next investigate how algebraic composition closure constrains the set of observationally compatible families within the regular specialization.

**Proposition 3.2** (Consistency reduces to composition closure)**.** *For sharply transitive permutation families, imposing the combined requirements of an identity element, inverses, and algebraic composition is equivalent to enforcing closure of the learned operator family under composition. Identification is achieved only when this* composition-closed *run-equivalent class is a singleton.*

Group-structured representation learning motivates these constraints (Caselles-Dupré et al., 2019; Keurti et al., 2023). Table 1 counts the families compatible with the persistence law, $|\mathcal{V}_{\text{obs}}|$, and the subset closed under composition, $|\mathcal{V}_{\text{grp}}|$, enumerated exhaustively (Appendix D.1; definitions as in Dixon & Mortimer, 1996). Closure reduces the 96 candidates for $\mathbb{Z}_8$ to the truth alone. It leaves two of 98,560 for $D_4$ and 30 for $\mathbb{Z}_2^3$, and these survivors are themselves groups that pass every algebraic check. A learner that enforces consistency can therefore still return a wrong family without any sign of error. If the truth is uniform over the 30 closed families for $\mathbb{Z}_2^3$, no learner using persistence data recovers it with probability above $1/30$ (Proposition B.5). Consistency reduces ambiguity but, for four of the five groups, does not resolve it.

Table 1: **The five order-8 groups used throughout.**

| $G$ | $\lvert\mathcal{V}_{\text{obs}}\rvert$ | $\lvert\mathcal{V}_{\text{grp}}\rvert$ |
|---|---|---|
| $\mathbb{Z}_8$ | 96 | **1** |
| $Z_2 \times Z_4$ | 5,632 | 2 |
| $D_4$ | 98,560 | 2 |
| $Q_8$ | 2,560 | 2 |
| $\mathbb{Z}_2^3$ | $> 6{,}240$ | 30 |

### 3.3 Re-executing a command

Two constructions show that neither protocol suffices on its own. In both, the wrong family shares its valid-edge graph with the truth (Figure 2).

**Native reuse alone can fail.** On the two-command cycle $h_{\pm}(x) = x \pm 1$ over $\mathbb{Z}_8$, let the candidate reverse the step at odd states. The two families have identical native reuse laws but different two-transition persistence laws. Their operators disagree on half of the state-command pairs under either global matching.

**Persistence alone can fail.** Consider $2L$ states in two layers, with commands that rotate within a layer $(+)$, rotate back $(-)$, swap layers $(v)$, and swap layers with a reflection $(v')$. Conjugating all four commands by a reflection of the second layer preserves every persistence law but changes the native reuse law. The two families also satisfy the same command relations, so no algebraic check separates them either. The layer-changing commands keep the graph connected, and the best global matching has error $1/4$ for every even $L \geq 4$ (256 states at $L = 128$). Appendix A.5 gives the construction, and Appendix B.2 extends it to repeat-or-resample sampling.

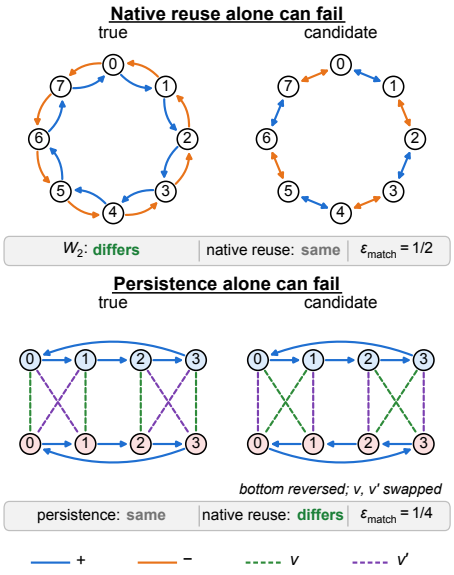



Figure 2: **Each protocol alone misses one compatible family.** The lower row is drawn at $L = 4$ and omits $-$, the reverse of $+$.

**Theorem 3.3** (Identification from persistence and native reuse)**.** *Let $H$ and $K$ be pointwise-distinguishable families of $m$ permutations over $S$ whose valid-edge graph has a connected underlying undirected graph, and assume that each family's native reuse trials give positive probability to every source state $x$ and every pair $(a, b)$ of its first and intervening commands. If $W_2(H) = W_2(K)$ and their native reuse laws agree, then $k_{\pi(a)} = h_a$ for a single bijection $\pi\colon A \to \mathcal{I}$.*

Intuitively, equal one-step laws give a local command-to-code matching at each state. Persistence makes this matching agree along repeated-command transitions, and native reuse makes it agree after an intervening transition. Invertibility and connectivity then make the matching global (Appendix A.4).

**Corollary 3.4** (The regular specialization)**.** *When $m = N$, native reuse with full support alone identifies the family up to a global permutation of indices.*

In this regime every ordered pair is a valid edge, so the intervening command reaches every target and the response at each source edge traces a whole operator. Neither result uses closure or a group. Identification therefore comes from the protocol, not from algebraic structure. Both results hold under state-dependent sampling with full prefix support. Reuse trials never name the commands (Proposition B.4), so the operators are fixed up to a global relabeling.

## 4 What reuse measurements guarantee

This section asks what to measure to confirm that a learned action acts as the same command in another state. Checks that never see the environment's response there cannot confirm it. Reuse at an independently chosen target can.

### 4.1 Limits of target-blind checks

Without the environment's response at the reuse state, no check can confirm correct reuse while observationally equivalent candidates remain. Cycle checks are no exception. A cycle and reuse start from the same prediction $\widehat{w} = D(z, E(x, y))$ and differ only in the outcome they compare it with (Figure 3). *Target-environment reuse* compares $\widehat{w}$ with the environment's response to the same command at $z$. The *source-scored cycle* re-encodes $(z, \widehat{w})$, returns to $x$, and compares $\widehat{y}_{\mathrm{cyc}} = D(x, E(z, \widehat{w}))$ with the observed source outcome $y$. We call a check *target-blind* when it does not observe the corresponding environment outcome at $z$. Persistence likelihoods, the algebraic checks of Section 3.2 and source-scored cycles are target-blind. For each candidate family, the source transition determines a command or code, and hence the state reached at the target:

$$F_H(x, y, z) \;=\; h_{a_H(x,y)}(z), \qquad F_K(x, y, z) \;=\; k_{c_K(x,y)}(z). \tag{5}$$

**Proposition 4.1** (Limit of target-blind checks)**.** *Let $\mathcal{O}$ collect everything available for training, model selection and scoring, and suppose $\mathcal{O}$ has the same law under $H$ and $K$. Fix a common query law $Q$, independent of the training and selection procedure, on triples whose source edge is valid for both families, so that both predictions are defined. For a randomized procedure, the risks below also average over the data and the algorithmic randomness. Write $\Delta = \Pr_Q[F_H(X, Y, Z) \neq F_K(X, Y, Z)]$. Then every model $M$ obtained from $\mathcal{O}$ satisfies $R_H(M) + R_K(M) \geq \Delta$, and hence $\max\{R_H(M), R_K(M)\} \geq \Delta/2$, where $R_W$ is its reuse risk on $Q$ when the world is $W$. Any target-blind score is a function of $\mathcal{O}$ and $M$, so it has the same law in both worlds and no threshold on it changes the bound.*

The proof is Le Cam's two-point argument (Yu, 1997). Both bounds are tight: each induced candidate has risk 0 in its own world and $\Delta$ in the other, and randomizing between them attains $\Delta/2$. The XOR family on $\mathbb{Z}_2^3$ and its conjugate by a transposition of states give $\Delta = 3/8$ under the independent uniform law on triples (Appendix A.6).

The *action cycle* of Garrido et al. (2026) and Zhang et al. (2026) is target-blind and, in the finite setting, also degenerate. Evaluated through its induced encoder, a structured candidate returns $c' = c$ and $\widehat{y} = y$ *identically* whenever it reconstructs the evaluated source edge, whether or not it is the true family. Incorrect structured candidates can therefore close the cycle exactly, and Section 5.2 gives learned examples.

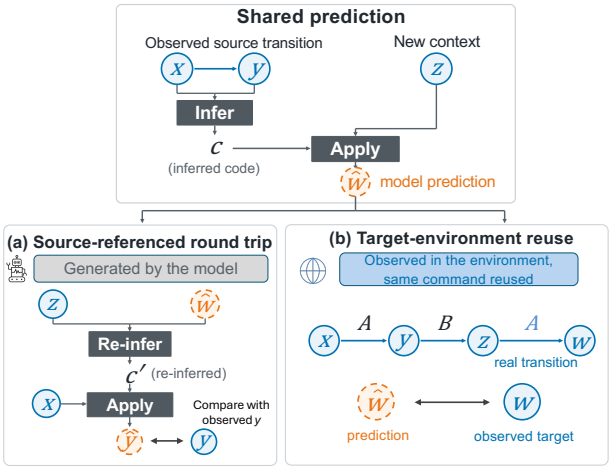



Figure 3: **The cycle is scored at the source; reuse is scored by the environment at the target.**

### 4.2 INDEPENDENT-TARGET REUSE

Reuse error at an independently chosen target bounds the risk of the best fixed command-to-code dictionary (Proposition 4.2). Together with local reconstruction error, it also bounds the distance between the learned and true operators (Theorem 4.3).

**Proposition 4.2** (Independent-target reuse bounds fixed-dictionary risk)**.** *Consider a fixed model. Let $V$ denote the target-side record used to evaluate a prediction, and let $L_a(V, c) \geq 0$ represent the loss of applying code $c$ at that target given the executed command $a$. We define the risk $r_{\mathrm{ind}}$ of the inferred code $C$ and the infimum risk $\varepsilon_{\mathrm{dict}}$ over fixed dictionaries (which assign one code to each command for use across all targets) as follows:*

$$r_{\mathrm{ind}} = \mathbb{E}\, L_A(V, C), \qquad \varepsilon_{\mathrm{dict}} = \inf_{\phi\colon A \to \mathcal{I}} \mathbb{E}\, L_A\big(V, \phi(A)\big). \tag{6}$$

*If the decoder's output distribution depends only on the target and the code, and $C \perp V \mid A$, then $\varepsilon_{\mathrm{dict}} \leq r_{\mathrm{ind}}$.*

For each command, the expected target loss over emitted codes is at least the infimum over a fixed code, even when the encoder emits different codes at different sources (Appendix A.6). A fixed dictionary selects one code for each command and uses it across all target states. Independently sampling the target record given the command ensures the required conditional independence, and an experimenter can arrange it without access to the encoder.

**The finite case.** For $m$ codes, the 0–1 loss and independent uniform $X$, $Z$ and $A$, so that $V$ is the target state, Equation 6 reduces to the reuse error $r_{\mathrm{ind}}$, and we define local reconstruction error and best-matching error beside it:

$$\delta_{\mathrm{loc}}(E, D) = \Pr\big[D(X, E(X, h_A(X))) \neq h_A(X)\big], \tag{7}$$

$$r_{\mathrm{ind}}(E, D) = \Pr\big[D(Z, E(X, h_A(X))) \neq h_A(Z)\big], \tag{8}$$

$$\varepsilon_{\mathrm{match}}(H, D) = \min_{\pi \in \mathrm{Bij}(A, \mathcal{I})} \Pr\big[D(Z, \pi(A)) \neq h_A(Z)\big]. \tag{9}$$

Since a dictionary need not be injective, $\varepsilon_{\mathrm{dict}} \leq \varepsilon_{\mathrm{match}}$.

**Theorem 4.3** (Reuse and local errors bound operator disagreement)**.** *Assume the true commands yield distinct effects at every source state, and consider arbitrary deterministic maps $E$ and $D$ using $m$ discrete codes. Then the error of the best global matching is at most the sum of the reuse error and the local reconstruction error:*

$$\varepsilon_{\mathrm{match}}(H, D) \;\leq\; r_{\mathrm{ind}}(E, D) + \delta_{\mathrm{loc}}(E, D). \tag{10}$$

At each source, locally correct commands must receive distinct codes. Extending this partial assignment to a bijection changes predictions only on locally incorrect commands, costing at most $\delta_{\mathrm{loc}}$, and the best global matching is no worse than the average of these per-source bijections. Both terms are computed from observed transitions without external command labels. Measuring $r_{\mathrm{ind}}$ requires same-command executions at independently sampled endpoints, which native reuse does not ensure: the two-command cycle of Section 3.3 has $r_{\mathrm{nat}} = \delta_{\mathrm{loc}} = 0$ but $\varepsilon_{\mathrm{match}} = 1/2$.

When evaluated through the encoder induced by a structured candidate sharing the true valid edges, this relationship becomes two-sided under independent uniform sampling:

$$\varepsilon_{\mathrm{match}} \;\leq\; r_{\mathrm{ind}} \;\leq\; 2\varepsilon_{\mathrm{match}} - \varepsilon_{\mathrm{match}}^2. \tag{11}$$

Appendix A.6 shows that the lower side is attained, that the factor 2 is asymptotically sharp and that shared valid edges are needed, and it extends the bounds to low error and dependent sampling.

## 5 EXPERIMENTS

Sections 3 and 4 prove that consistent but wrong action families can remain and that target-blind checks cannot reveal them. This section tests both claims in finite systems where family recovery can be checked exactly, and measures reuse and the cycle in frozen LAPO models on Procgen. The Q&As discuss why a learner reaches a wrong family and what the reuse and cycle measurements show.

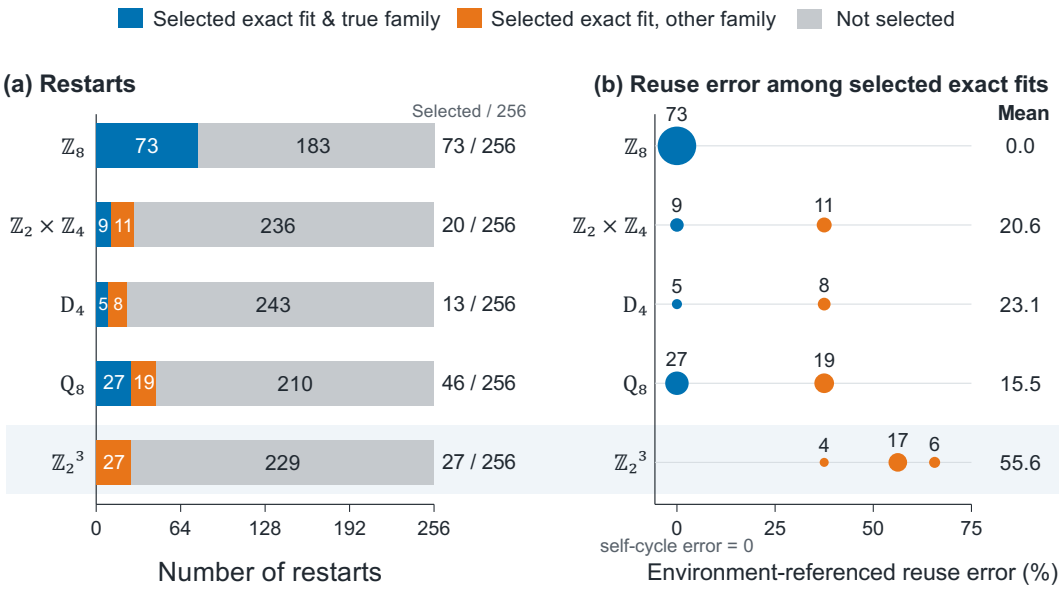



Figure 4: **Selected exact fits and their reuse error.**

### 5.1 Setup

**Controlled systems.** The synthetic suite instantiates the regular specialization (Section 2) across the five groups of order 8 (Table 1). The learner observes unlabeled state trajectories, with persistence sequences spanning 14 transitions. It represents each of the eight operators as a matrix of transition probabilities, a relaxation of a permutation, and fits them by maximum likelihood with consistency penalties, over 32 datasets $\times$ 8 restarts per condition (Appendix D.2). A *selected exact fit* meets the likelihood criterion and, once each row is rounded to its largest entry, reproduces every training run with exactly one operator.

**Video models.** We use one frozen stage-1 LAPO (Schmidt & Jiang, 2024) checkpoint per Procgen (Cobbe et al., 2020) environment, a transition encoder and a forward predictor. We pair source and target states from separately simulated rollouts, and at both states we execute each of the 15 actions from the same saved state to record its outcome. A prediction at the target is correct when the nearest of these outcomes under pixel mean squared error is that of the executed action. We keep pairs whose executed command has a unique successor at both endpoints, 18.8% of those sampled (Appendix E.1).

**Metrics and uncertainty.** We evaluate *local* reconstruction at the source transition, *reuse* at the target with the source's code for the executed command (the *matched* code), and the *source-scored cycle* (Section 4.1). Reuse controls replace the matched code with a zero latent or the code of a different command at the same source (the *mismatched-command* code). The cycle's control replaces its re-inferred code with a *shuffled* code, the re-inferred code of another scored pair in the same environment. We report 95% bootstrap confidence intervals over dataset seeds (controlled) or source episodes (Procgen), with a crossed-bootstrap sensitivity analysis in Appendix E.1.

### 5.2 Controlled identification

**Fit and family recovery.** On $\mathbb{Z}_2^3$, 27 of 256 restarts are selected exact fits, all satisfying identity, inverse and associativity. They represent 18 distinct families, none equal to the generating family. All have zero cycle error under their induced encoders, yet their reuse errors are 37.5%, 56.25% or 65.625%, with a mean of 55.6%. Consistency and a closed cycle thus coexist with a wrong family. For the regular $\mathbb{Z}_8$ action, whose compatible closed class is a singleton, all 73 selected fits recover the family with zero reuse error. Figure 4 shows all five groups.

**Learning from reuse trials.** We replace persistence data with native reuse trials while keeping the learner, optimizer, penalties, transition budget and seeds fixed. All 256 restarts recover the generating family in each of the five systems (Table 2). A short-persistence control matches the reuse condition's sequence count and length but holds one command throughout. On $\mathbb{Z}_2^3$ it yields 42 selected fits and no recovery. Sequence count and length alone therefore do not explain the change (Appendix D.4). That reuse trials help is expected, since they tell the learner which transitions share a command. What Section 3 adds is that persistence data alone cannot remove the ambiguity, and no consistency check reveals it.

Table 2: Restarts that recover the family, of those selected.

| $G$ | Persistence | Short | Reuse |
|---|---|---|---|
| $\mathbb{Z}_8$ | 73/73 | 227/227 | 256/256 |
| $\mathbb{Z}_2 \times \mathbb{Z}_4$ | 9/20 | 60/133 | 256/256 |
| $\mathbb{Z}_2^3$ | **0**/27 | **0**/42 | **256**/256 |
| $D_4$ | 5/13 | 32/67 | 256/256 |
| $Q_8$ | 27/46 | 98/188 | 256/256 |

### 5.3 Functional reuse in video models

**Cross-state transfer.** On 6,027 eligible pairs, matched codes achieve 33.0% accuracy, compared with 7.0% for mismatched-command codes. The paired difference is 25.9 percentage points $[23.8, 28.1]$ and is positive in 14 of the 15 environments with eligible pairs. Matched codes thus carry command information to other states, yet reuse is far less accurate than local reconstruction at the source, 53.4% (Table 3). Section 5.4 compares transfer with codes inferred at the target itself. Figure 5 shows examples.

Table 3: **Local reconstruction and cross-state reuse on** 6,027 **pairs.**

| Evaluation condition | Accuracy |
|---|---|
| Target-local | 0.544 |
| Matched source code | 0.330 |
| Mismatched-command code | 0.070 |
| Zero latent | 0.093 |
| Copy last frame | 0.041 |
| Source-local | 0.534 |

**Interpreting the dictionary bound.** Collection-order pairing does not enforce the conditional independence of Proposition 4.2, and action-effect collisions fall outside Theorem 4.3, so the observed error is a measurement, not a verified bound (Appendix E.1).

### 5.4 Q&As

**Q: Why can a consistent learner return the wrong action family?**

**A:** Because the observations do not distinguish the true family from the wrong ones (Section 3.2), initialization changes which family optimization reaches. On 32 new datasets, biasing initialization toward a chosen family increases its recovery rate across all 256 restarts by 12.1 percentage points $[8.2, 16.0]$ for the true family and 9.8 $[6.6, 13.3]$ for a designated incorrect family (Table 4). Selection also becomes more frequent, so these changes combine effects on fitting and family choice (Appendix D.5).

Table 4: Planted initialization, $\mathbb{Z}_2^3$.

| | | Recovered family | |
|---|---|---|---|
| Initialization | Selected | True | Designated |
| None | 31 | 1 | 2 |
| Toward true | 63 | **32** | 4 |
| Toward wrong | 42 | 4 | **27** |

**Q: Does the transfer gap remain on the same targets?**

**A:** Yes, so the transfer gap does not come from harder target states. On the same 6,027 evaluation pairs, codes inferred from the target transition achieve 54.4% accuracy, compared with 33.0% for codes inferred from the source transition (Table 3). The paired difference is 21.4 percentage points (95% CI $[19.3, 23.7]$; crossed-bootstrap CI $[17.4, 25.7]$). Target-local accuracy is higher in 13 of the 15 environments with eligible pairs (Appendix E.1). On bigfish, two training runs with similar source-local accuracy (69.7% and 69.1%) reach matched accuracies of 47.8% and 31.8% on the same 851 pairs (Appendix E.1). Reconstructing a change where it happens therefore does not imply reusing a code from elsewhere.

**Q: What does re-encoding add to the source-scored cycle?**

**A:** Re-encoding adds 0.6 percentage points over the shuffled code, 11.1% against 10.5% source-return accuracy on the same 5,226 pairs (Table 5). The 95% interval is $[0.1, 1.1]$ under source-episode resampling and $[-0.3, 1.5]$ under crossed resampling, and only the former excludes zero (Appendix E.2). The shuffled code comes from an unrelated pair and carries this pair's command only by chance, yet it reaches almost all of the cycle's accuracy. A closed cycle is therefore weak evidence that the re-inferred code carries the command.

Table 5: **The source-scored round trip.**

| Code returned | Source-return |
|---|---|
| Re-inferred | 0.111 |
| Shuffled | 0.105 |
| Mismatched command | 0.067 |
| Zero latent | 0.084 |

## 6 Related work

**Internal consistency in latent-action models.** Rybkin et al. (2019) add a composability loss and evaluate cross-sequence transfer, Menapace et al. (2021) extract discrete action labels designed for cross-state consistency, and Keurti et al. (2023) learn group-structured representations from transitions paired with the observed actions. AC-LAM (Wei et al., 2026a) regularizes latent actions toward additive composition, and ALAM (Tang et al., 2026) also toward temporal reversal. These soft objectives motivate our analysis of exact algebraic constraints in a finite operator class, where consistency can leave several compatible families (Proposition 3.2). CycleMimic (Chen et al., 2026) trains its action encoder to recover a sampled code from a frame and the frame generated with that code. In our finite setting, the exact form of this objective, $E_K(x, D_K(x, c)) = c$, holds for every structured candidate, including incorrect ones. Garrido et al. (2026) and Zhang et al. (2026) evaluate source–target–source cycles, which Section 4 shows to be target-blind.

**Identifying latent actions.** Lachapelle (2025) identifies state-independent injective action representations using continuity, distinguishable effects, connectedness and overlap, and Schur (2026) identifies per-state factors from demonstrator diversity and aligns them through continuity. Zhang et al. (2025) analyze linear latent-action models and show how augmentation can remove observation-dependent components from the latent code. Nam et al. (2026) argue that agent ambiguity calls for state-conditioned summaries of transition effects. We instead relate specified observation protocols to the recovery of finite operator families, and reuse measurements to dictionary and best-matching error. Appendix C compares further designs and separates topological from graph connectedness.

**Passive and interventional identifiability.** Unsupervised disentanglement is not identifiable without further assumptions (Locatello et al., 2019). Identifiability can follow from auxiliary variables (Khemakhem et al., 2020), temporal sequences with observed intervention targets (Lippe et al., 2022), or interventional data under structural assumptions (Ahuja et al., 2023). Weakly supervised methods also use paired observations, with restrictions on how their latent factors or interventions differ (Locatello et al., 2020; Shu et al., 2020; Brehmer et al., 2022). Our re-execution trials instead pair executions of the same command, and we study the operators that latent actions represent.

## 7 Conclusion

Internal consistency alone does not guarantee that learned action operators are correct. Under our assumptions, repeated-command trajectories and re-execution trials resolve this ambiguity, while reconstruction and independent-target reuse errors together bound the distance to the true operators. In frozen LAPO models, the gap of 21.4 percentage points shows that reconstruction accuracy alone is insufficient for assessing action reuse.

**What to report.** When target-side outcomes are available, sample source and target independently given the command, state how same-command pairs were obtained, and report reuse alongside local reconstruction. Otherwise, compare the source-scored cycle with a shuffled-code control under the same scoring rule. Neither replaces target-side evaluation.

**Limitations.** Identification is proved for finite, fully observed systems, and continuous or partially observed states are the natural next step. The controlled experiments use one tabular optimizer, and the Procgen study examines LAPO in depth, with one main checkpoint per environment.

## AI USE STATEMENT

AI assistants (e.g., Claude, ChatGPT) were used to polish the text for clarity, to refine the mathematical proofs, to verify coverage of related literature, and for routine code review. All conceptual ideas and theoretical frameworks were developed by the author, who verified the final proofs and assumes full responsibility for all claims.

## REPRODUCIBILITY STATEMENT

Every theoretical claim in the main text is stated with its hypotheses and proved in Appendix A or B, and the two constructions of Section 3.3 are given explicitly in Appendix A.5. Appendix D.2 specifies the synthetic learner exactly, giving the objective, the four penalties, the annealing schedule, the discretization and the two fit criteria, with Algorithm 1 the procedure one restart runs. Appendix D.1 gives the enumeration behind the candidate counts, whose observational entry for $\mathbb{Z}_2^3$ is a strict lower bound. Appendix E.1 states the frozen-checkpoint protocol, the evaluation conditions, the second bigfish checkpoint, the exclusion rules and their counts, the scoring rule and the definition of every interval, Appendix E.2 the return-code controls, and Table 7 lists the evaluation samples. Code for all experiments is available at `https://anonymous.4open.science/r/consistency-is-not-identification-9DB2` and in the supplementary material.

## REFERENCES

Kartik Ahuja, Divyat Mahajan, Yixin Wang, and Yoshua Bengio. Interventional causal representation learning. In *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 372–407. PMLR, 2023.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Johann Brehmer, Pim de Haan, Phillip Lippe, and Taco Cohen. Weakly supervised causal representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal M. P. Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott E. Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR, 2024.

Xizhou Bu, Qingda Hu, Lei Zhou, Lingfeng Zhang, Yingbo Tang, Zihao Liu, Xinyi Tao, Zhiqiang Ma, Qingqiu Huang, Chufeng Tang, Hongbo Wang, Jing Zhang, Jiayi Ma, Hangjun Ye, Wei Li, and Xiaoshuai Hao. What matters for latent actions in robot learning. *arXiv preprint arXiv:2608.19613*, 2026.

Jin Cao, Zian Meng, and Kaipeng Zhang. ShadowDancer: Teaching video world models any action by learning unified dynamics representations from a video and its shadow. *arXiv preprint arXiv:2607.28362*, 2026.

Hugo Caselles-Dupré, Michael Garcia-Ortiz, and David Filliat. Symmetry-based disentangled representation learning requires interaction with environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Guangyan Chen, Qi Shao, Te Cui, Zichen Zhou, Weixin Mao, Luojie Yang, Meiling Wang, Yi Yang, Hua Chen, and Yufeng Yue. Learning a unified latent action space from videos with action-centric cycle consistency. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12871–12880, 2026.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2048–2056. PMLR, 2020.

John D. Dixon and Brian Mortimer. *Permutation Groups*, volume 163 of *Graduate Texts in Mathematics*. Springer, 1996.

Ashley D. Edwards, Himanshu Sahni, Yannick Schroecker, and Charles L. Isbell Jr. Imitating latent policies from observation. In *International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1755–1763. PMLR, 2019.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. AdaWorld: Learning adaptable world models with latent actions. In *International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 18744–18771. PMLR, 2025.

Quentin Garrido, Tushar Nagarajan, Basile Terver, Nicolas Ballas, Yann LeCun, and Michael Rabbat. Learning latent action world models in the wild. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2601.05230.

William H. Gustafson. What is the probability that two group elements commute? *The American Mathematical Monthly*, 80(9):1031–1034, 1973.

Hamza Keurti, Hsiao-Ru Pan, Michel Besserve, Benjamin F. Grewe, and Bernhard Schölkopf. Homomorphism AutoEncoder: Learning group structured representations from observed transitions. In *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 16190–16215, 2023.

Ilyes Khemakhem, Diederik P. Kingma, Ricardo Pio Monti, and Aapo Hyvärinen. Variational autoencoders and nonlinear ICA: A unifying framework. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2207–2217. PMLR, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Sébastien Lachapelle. On the identifiability of latent action policies. *arXiv preprint arXiv:2510.01337*, 2025.

Sizhe Lester Li, Evan Kim, Xingjian Bai, Tong Zhao, Tao Pang, Max Simchowitz, and Vincent Sitzmann. Turning video models into generalist robot policies. *arXiv preprint arXiv:2605.27817*, 2026.

Phillip Lippe, Sara Magliacane, Sindy Löwe, Yuki M. Asano, Taco Cohen, and Efstratios Gavves. CITRIS: Causal identifiability from temporal intervened sequences. In *International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 13557–13603. PMLR, 2022.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4114–4124. PMLR, 2019.

Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-supervised disentanglement without compromises. In *International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6348–6359. PMLR, 2020.

Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

Willi Menapace, Stéphane Lathuilière, Sergey Tulyakov, Aliaksandr Siarohin, and Elisa Ricci. Playable video generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10061–10070, 2021.

Nikita Moriakov, Jonas Adler, and Jonas Teuwen. Kernel of CycleGAN as a principal homogeneous space. In *International Conference on Learning Representations (ICLR)*, 2020.

Heejeong Nam, Chandradithya S. Jonnalagadda, Harshit Aggarwal, Eric Xu, and Randall Balestriero. Latent actions from factorized transition effects under agent ambiguity. *arXiv preprint arXiv:2606.30544*, 2026.

Dujun Nie, Fengjiao Chen, Qi Lv, Jun Kuang, Xiaoyu Li, Xuezhi Cao, and Xunliang Cai. LARY: A latent action representation yielding benchmark for generalizable vision-to-action alignment. In *European Conference on Computer Vision (ECCV)*, 2026. arXiv:2604.11689.

Alexander Nikulin, Ilya Zisman, Denis Tarasov, Nikita Lyubaykin, Andrei Polubarov, Igor Kiselev, and Vladislav Kurenkov. Latent action learning requires supervision in the presence of distractors. In *International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 46427–46447. PMLR, 2025.

Art B. Owen. The pigeonhole bootstrap. *The Annals of Applied Statistics*, 1(2):386–411, 2007.

Oleh Rybkin, Karl Pertsch, Konstantinos G. Derpanis, Kostas Daniilidis, and Andrew Jaegle. Learning what you can do before doing anything. In *International Conference on Learning Representations (ICLR)*, 2019.

Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *International Conference on Learning Representations (ICLR)*, 2024.

Felix Schur. Identifying latent actions and dynamics from offline data via demonstrator diversity. *arXiv preprint arXiv:2603.17577*, 2026.

Rui Shu, Yining Chen, Abhishek Kumar, Stefano Ermon, and Ben Poole. Weakly supervised disentanglement with guarantees. In *International Conference on Learning Representations (ICLR)*, 2020.

Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

Zuojin Tang, Haoyun Liu, Xinyuan Chang, Changjie Wu, Dongjie Huo, Yandan Yang, Bin Liu, Zhejia Cai, Feng Xiong, Mu Xu, Jiachen Luo, De Ma, Zhiheng Ma, and Gang Pan. ALAM: Algebraically consistent latent action model for vision-language-action models. *arXiv preprint arXiv:2605.10819*, 2026.

Hangxing Wei, Xiaoyu Chen, Chuheng Zhang, Tim Pearce, Jianyu Chen, Alex Lamb, Li Zhao, and Jiang Bian. Learning additively compositional latent actions for embodied AI. *arXiv preprint arXiv:2604.03340*, 2026a.

Yufan Wei, Kun Zhou, Lingjun Mao, Zijun Zhang, Ziming Xu, Ziqiao Xi, Shuang Liang, Ruobing Han, Yuchen Yan, Xinyue Wang, Fan Feng, and Biwei Huang. Causally debiased latent action model for embodied action conditioned world models. *arXiv preprint arXiv:2607.09185*, 2026b.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *International Conference on Learning Representations (ICLR)*, 2025.

Bin Yu. Assouad, Fano, and Le Cam. In *Festschrift for Lucien Le Cam*, pp. 423–435. Springer, 1997.

Chuheng Zhang, Tim Pearce, Pushi Zhang, Kaixin Wang, Xiaoyu Chen, Wei Shen, Li Zhao, and Jiang Bian. What do latent action models actually learn? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Tianqiu Zhang, Muyang Lyu, Yufan Zhang, Fang Fang, and Si Wu. DiLA: Disentangled latent action world models. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2605.15725.

# A PROOFS

Appendix A proves the main results and gives the counterexamples. Appendix B extends the observation model; Appendix C details related work; Appendix D specifies the controlled experiments; and Appendix E details the Procgen evaluation. Table 6 summarizes the structural assumptions of the stated results.

We use the observation model of Section 2. For the regular group specialization, let $S = G$ for a finite group $G$ of order $n$, so that $N = m = n$, and let $h_a(x) = ax$.

## A.1 PROPOSITION 3.1: TWO TRANSITIONS DETERMINE THE PERSISTENCE LAW

*Proof.* Let $H$ be a pointwise-distinguishable family of $m$ maps of $S$, with $|S| = N$, and draw the start state and the held command independently and uniformly. By equation 1, exactly one command produces each valid edge, giving every edge in $\mathcal{E}_H$ a mass of $1/(Nm)$. On a persistence run, the held command is $a_H(x_{t-1}, x_t)$, making $x_{t+1} = C_H(x_{t-1}, x_t)$ for every $t \geq 1$. Conversely, any sequence obeying this recurrence follows a single command at every step. Hence, for $T \geq 1$,

$$W_T(H)(x_{0:T}) \;=\; \frac{1}{Nm}\,\mathbf{1}\{(x_0, x_1) \in \mathcal{E}_H\} \prod_{t=1}^{T-1} \mathbf{1}\{x_{t+1} = C_H(x_{t-1}, x_t)\},$$

the product being empty at $T = 1$. The two-transition law therefore comprises the valid-edge set and $C_H$ on it. Since $C_H$ determines every longer law through this formula, $W_2(H) = W_2(H')$ implies $W_T(H) = W_T(H')$ for all $T \geq 2$. □

**Lemma A.1** (State relabeling and the persistence law)**.** *For a permutation $\tau$ of $S$ and any family $H$, conjugation $h \mapsto \tau h \tau^{-1}$ gives $W_T(\tau H \tau^{-1}) = \tau_* W_T(H)$ at every horizon, where $\tau_*$ relabels every state of a run by $\tau$. If $\tau_* W_2(H) = W_2(H)$, then $\tau H \tau^{-1}$ and $H$ induce the same law on persistence runs of every length. For permutation families, conjugate families have the same multiset of operator orders.*

*Proof.* $(\tau h_a \tau^{-1})^t(x) = \tau h_a^t(\tau^{-1} x)$ for every $t$, and $\tau$ preserves the uniform law on start states. Thus, relabeling the states by $\tau$ maps the run law of $H$ to that of $\tau H \tau^{-1}$. The all-horizon conclusion follows from Proposition 3.1. □

**The group case.** For $h_a(x) = ax$, the command taking $x$ to $y$ is $yx^{-1}$, so $C_H(x, y) = yx^{-1}y$ and the transfer map of Section 4.1 is $F_H(x, y, z) = yx^{-1}z$. The XOR pair in Appendix A.6 shares $C_H$ but disagrees under reuse.

## A.2 LEFT AND RIGHT ACTIONS

The two rules of Figure 1, one tied to positions and one to object identities, are the left and right actions of a group. Let $\mathcal{A} \subseteq G$ be nonempty, closed under conjugation and under inverses, and compare $h_a(x) = ax$ with $k_a(x) = xa$ on $S = G$. The command taking $x$ to $y$ is $yx^{-1}$ in the first family and $x^{-1}y$ in the second, both in $\mathcal{A}$, so the valid edges agree, and both continuations equal $yx^{-1}y$, so the persistence laws agree at every horizon (Proposition 3.1). With $\tau(x) = x^{-1}$, $\tau h_a \tau^{-1} = k_{a^{-1}}$, so after reindexing codes by inversion the two families satisfy the same labeled relations; neither is closed under composition unless $\mathcal{A}$ is a subgroup. For $y = ax$, the reuse predictions at $z$ are $az$ and $zx^{-1}ax$, which agree exactly when $z^{-1}az = x^{-1}ax$. For independent uniform $X$ and $Z$, both conjugates are uniform on the class $a^G$, so under independent uniform $X$, $Z$ and $A$ the predictions disagree with probability

$$\Delta = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \Big(1 - \frac{1}{|a^G|}\Big).$$

Table 6: **Selected structural assumptions of the stated results.** The marks are the hypotheses of the stated versions, which for some results are stronger than their proofs need. Each statement also gives its sampling and decoder conditions. *Independent target* means the target is drawn independently of the source given the command.

| Result | Reversible | Ptwise distinct | Connected | $\lvert\mathcal{I}\rvert = m$ | Indep. target |
|---|---|---|---|---|---|
| Two-transition persistence (Prop. 3.1) | — | ✓ | — | — | — |
| Consistency and closure (Prop. 3.2) | ✓ | ✓ | — | — | — |
| Limit of target-blind checks (Prop. 4.1) | — | — | — | — | — |
| Joint identification (Thm. 3.3) | ✓ | ✓ | ✓ | ✓ | — |
| Dictionary bound (Prop. 4.2) | — | — | — | — | ✓ |
| Matching bound (Thm. 4.3) | — | ✓ | — | ✓ | ✓ |
| Low-error matching (Cor. A.2) | — | ✓ | — | ✓ | ✓ |
| Two-sided bound (Eq. 11)[†] | ✓ | ✓ | — | ✓ | ✓ |

[†]Equation 11 also requires the candidate to share the true valid edges, $\mathcal{E}_K = \mathcal{E}_H$; without it the left inequality need not hold.

For $\mathcal{A} = G$ this is $1 - k(G)/|G|$, where $k(G)$ counts conjugacy classes, that is, one minus the probability that two uniform elements commute. It is at least $3/8$ for every non-abelian group (Gustafson, 1973), with equality for $D_4$ and $Q_8$, whose two closed families in Table 1 are these two actions. In Figure 1 each family holds the six permutations of $S_3$; a swap then disagrees with probability $2/3$, and the whole family with probability $1/2$. Left and right multiplication are the space- and body-frame conventions for rigid-body motions (Lynch & Park, 2017).

### A.3 Proposition 3.2: equivalence of the three constraints and composition closure

*Proof.* Let $K$ be a nonempty finite family of permutations closed under composition. Since every $h \in K$ has finite order $q$, closure gives $h^q = \mathrm{id} \in K$ and $h^{-1} = h^{q-1} \in K$. Conversely, enforcing the identity, inverse and composition requirements implies composition closure, reducing the three constraints to it. A composition-closed sharply transitive family is a *regular subgroup* (Dixon & Mortimer, 1996). Thus, the surviving candidates are precisely the regular subgroups within the run-equivalence class, and identification follows only when this set is a singleton. □

**Why the $\mathbb{Z}_2^3$ entry is a lower bound.** The enumeration of Appendix D.1 yields 6,240 sharply transitive families of involutions on eight states that *contain the identity*. The class of Section 3.2 requires no identity and is strictly larger. Replacing the identity and the translation by 1 among the XOR translations of $\mathbb{F}_2^3$ with

$$k_0 = (0\ 1), \qquad k_1 = (2\ 3)(4\ 5)(6\ 7),$$

and keeping the rest yields a sharply transitive family of involutions without an identity that induces the same persistence law. Thus, 6,240 is a strict lower bound on the class without an identity requirement.

**Closed XOR families.** For the XOR translations of $\mathbb{F}_2^d$ on $n = 2^d$ states, a compatible composition-closed family is a regular subgroup of involutions. Every such family is a relabeling of the XOR family, and exactly the $2^d|\mathrm{GL}(d,2)|$ affine relabelings of $\mathbb{F}_2^d$ leave it unchanged, so there are $q_d = (2^d)!/\big(2^d|\mathrm{GL}(d,2)|\big)$ of them: $q_3 = 30$, the entry of Table 1, and $q_4 = 64{,}864{,}800$ for sixteen states and sixteen commands.

### A.4 Proofs of joint identification and the regular specialization

*Proof of Theorem 3.3.* Equal two-transition persistence laws imply equal one-transition marginals, yielding $\mathcal{E}_H = \mathcal{E}_K$. Under pointwise distinguishability, each state $x$ admits a unique bijection $\sigma_x \colon A \to \mathcal{I}$ satisfying

$$k_{\sigma_x(a)}(x) = h_a(x). \tag{12}$$

*Step 1: propagation along the repeated command.* Condition on an observed edge $(x, y)$ where $y = h_a(x)$. Since both continuations are deterministic, the equal two-transition laws imply $k_{\sigma_x(a)}(y) = h_a(y)$. Pointwise distinguishability at $y$ then yields

$$\sigma_{h_a(x)}(a) = \sigma_x(a). \tag{P}$$

*Step 2: propagation through an intervening command.* Fix $a, b, x$, setting $y = h_a(x)$ and $z = h_b(y)$. Full support guarantees the prefix $(x, y, z)$ is observable, which determines the candidate's first code, $\sigma_x(a)$. Equal native reuse laws on this prefix imply $k_{\sigma_x(a)}(z) = h_a(z)$, yielding

$$\sigma_{h_b h_a(x)}(a) = \sigma_x(a). \tag{R}$$

*Step 3: one correspondence over the connected graph.* Fix $a$. Since $h_a$ is surjective, $y = h_a(x)$ spans all of $S$. Combining equation P with equation R gives $\sigma_{h_b(y)}(a) = \sigma_y(a)$ for all $y \in S$ and $b \in A$. Thus, $\sigma_\cdot(a)$ remains constant across every edge of the valid-edge graph. The connectivity of the underlying undirected graph then forces $\sigma_\cdot(a)$ to be constant over $S$. Because each $\sigma_y$ is a bijection, its constant value $\pi(a)$ defines a global bijection $\pi\colon A \to \mathcal{I}$ satisfying $k_{\pi(a)} = h_a$ for all $a$. □

*Proof of Corollary 3.4.* When $m = N$, every ordered pair is a valid edge. Given a fixed source edge $(x, y)$, the intervening command reaches every possible target $z$. Consequently, the conditional final outcome of a native reuse trial reveals the entire operator mapping $x$ to $y$. Fixing $x$ and varying $y$ recovers the family up to a global permutation of codes. This argument relies solely on deterministic transitions and full prefix support, not on the persistence law. It holds even when the first two commands are drawn from a state-dependent distribution, provided every prefix has positive probability. □

**Connectedness in Theorem 3.3 cannot be omitted in general.** If the valid-edge graph splits into disconnected components, one can permute the codes independently within each: set $k_{\sigma_j(a)}(x) = h_a(x)$ for $x$ in component $C_j$, where $\sigma_j$ is a component-specific permutation of commands. Because persistence runs and native reuse trials remain confined within a single component, and the induced self-cycle merely compares the candidate to itself, all three agree with the truth exactly. For instance, given two four-cycles and two commands, swapping the two codes on the second component alone leaves persistence, native reuse and the cycle identical, yet makes all 64 cross-component reuse pairs wrong.

### A.5 Counterexamples for Section 3.3

We evaluate each candidate using its induced encoder. Because the true and candidate edge sets agree, both models reconstruct every observed transition exactly, yielding zero self-cycle error.

**Two commands on an eight-state cycle.** Let $S = \mathbb{Z}_8$ and $A = \mathcal{I} = \{+, -\}$, with arithmetic modulo eight. We set

$$h_+(x) = x + 1, \quad h_-(x) = x - 1, \qquad k_+(x) = x + (-1)^x, \quad k_-(x) = x - (-1)^x,$$

yielding $k_+ = (0\,1)(2\,3)(4\,5)(6\,7)$ and $k_- = (1\,2)(3\,4)(5\,6)(7\,0)$. Both families share the identical connected valid-edge graph and maintain distinct successors at every state.

*Persistence separates them.* Repeating a true command twice reaches $x \pm 2$, whereas both candidate commands are involutions. Thus, $W_2(H) \neq W_2(K)$. Even under a global permutation of codes, the candidate is not a state conjugate of the truth: conjugation preserves operator orders, which are $(8, 8)$ for the truth versus $(2, 2)$ for the candidate.

*The native reuse laws are identical.* The local matching $\sigma_x$ acts as the identity on even states and exchanges $+$ and $-$ on odd states. Because two steps along the common graph preserve parity, the source $x$ and the native target $z = h_b h_a(x)$ share the same matching. Consequently, the reused prediction agrees on every prefix. Each prefix has probability $1/(8 \cdot 2^2)$ under either family, so the native reuse laws are equal, not only their average errors.

*The three risks.* An independently drawn target has the opposite parity with probability $1/2$, giving $r_{\mathrm{ind}} = 1/2$, whereas $r_{\mathrm{nat}} = 0$. Because each global matching fails on one parity class, $\varepsilon_{\mathrm{match}} = 1/2$.

**Four commands on a connected two-layer cycle.** Let $L \geq 4$ be even, $S = \mathbb{Z}_L \times \{0,1\}$, and $A = \mathcal{I} = \{+, -, v, v'\}$, with

$$h_\pm(x,b) = (x \pm 1, b), \qquad h_v(x,b) = (x, 1-b), \qquad h_{v'}(x,b) = (1-x, 1-b).$$

Distinctness holds because $2x = 1 \pmod{L}$ has no solution for even $L$. We define

$$\tau(x,b) = \begin{cases} (x,0), & b = 0, \\ (1-x,1), & b = 1, \end{cases} \qquad k_a = \tau h_a \tau^{-1},$$

yielding $\tau^2 = \mathrm{id}$, $k_\pm(x,b) = (x \pm (-1)^b, b)$, $k_v = h_{v'}$, and $k_{v'} = h_v$.

*Persistence cannot separate them.* The matching acts as the identity for horizontal commands in layer zero, and exchanges $+$ and $-$ in layer one. This matching remains consistent along any horizontal orbit since horizontal commands preserve the layer, while the two vertical commands are exchanged globally. Consequently, each source induces a bijection of the four uniformly drawn commands that pairs equal persistent trajectories, yielding $W_T(H) = W_T(K)$ for all $T \geq 1$.

*The labeled relations agree.* For every word $q$ in the commands and their inverses, $k_q = \tau h_q \tau^{-1}$. Thus, $h_q = h_{q'}$ if and only if $k_q = k_{q'}$. The two families satisfy identical labeled relations despite being distinct concrete families.

*Native reuse separates them, and the three risks coincide.* Independent-target reuse fails exactly when the command is horizontal and the target lies in the opposite layer. Native reuse fails exactly when the first command is horizontal and the intervening command changes the layer. Thus, $r_{\mathrm{ind}} = r_{\mathrm{nat}} = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. Using the order $(+, -, v, v')$, the normalized agreement matrix from Appendix A.6 is

$$B = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Its optimal assignment sums to three, giving $\varepsilon_{\mathrm{match}} = 1 - 3/4 = 1/4$. Both counterexamples attain the lower side of Eq. 11; the XOR sequence in Appendix A.6 establishes the asymptotic sharpness of its leading factor 2.

### A.6 Proofs of the evaluation guarantees

**The XOR pair.** Let $G = \mathbb{F}_2^d$ with $n = 2^d \geq 8$. Let $h_a(x) = x \oplus a$ define the true family, let $\tau$ transpose states 0 and 1, and set $k_a = \tau h_a \tau$. Because $K$ is sharply transitive and satisfies $k_a k_b = k_{a \oplus b}$, it is composition-closed and satisfies identity, inverse and associativity exactly. Both families consist of the identity and fixed-point-free involutions. Consequently, a persistence run landing on $y$ at the first step oscillates as $x, y, x, y, \ldots$ in both models, inducing identical persistence laws at every horizon under uniform sampling. At $d = 3$, they disagree on 192 of the 512 triples, yielding $\Delta = 3/8$ under the independent uniform law.

*Proof of Proposition 4.1.* Let $\widehat{F}(x,y,z)$ denote the prediction of an arbitrary model. Since a prediction targets a single state, every triple satisfies

$$\mathbf{1}\{\widehat{F} \neq F_H\} + \mathbf{1}\{\widehat{F} \neq F_K\} \;\geq\; \mathbf{1}\{F_H \neq F_K\}.$$

Averaging over the fixed query law $Q$ and the common law of the prediction computed from $\mathcal{O}$ yields $R_H + R_K \geq \Delta$, the standard two-point argument (Yu, 1997). Thus, $\max\{R_H, R_K\} \geq \Delta/2$. Any score computed from $\mathcal{O}$ also has the same law in both worlds. The query law $Q$ must be fixed in advance; otherwise, a score could selectively query triples where the worlds agree. This bound is tight: the induced predictors for $H$ and $K$ achieve risks $(0, \Delta)$ and $(\Delta, 0)$, respectively, and selecting either with probability $1/2$ attains exactly $\Delta/2$ in both worlds. □

*Proof of Proposition 4.2.* Let $V$ denote the target-side record for scoring, and let $L_a(V,c) \geq 0$ be any nonnegative loss for applying code $c$ at that target given command $a$. Because the decoder's

output distribution depends only on $(Z, c)$, its randomness can be absorbed into $L_a$; define $g_a(c) = \mathbb{E}\big[L_a(V, c) \mid A = a\big]$. The encoder's randomness is carried by the law of $C$. Since $C \perp V \mid A$,

$$r_{\text{ind}} = \mathbb{E}_{A,C}\, g_A(C) \;\geq\; \mathbb{E}_A \inf_c g_A(c) = \inf_{\phi\colon A \to \mathcal{I}} \mathbb{E}_A\, g_A(\phi(A)) = \varepsilon_{\text{dict}}.$$

This holds because an expected value never falls below its infimum, and $\phi$ can be chosen separately for each of the finitely many commands. We write an infimum rather than a minimum because it might not be attained on a continuous alphabet. The decoder condition cannot be dropped: if the encoder and decoder share a fair bit $U$ independent of the code, setting $C = A \oplus U$ with a decoder applying $C \oplus U$ yields $r_{\text{ind}} = 0$, whereas every fixed dictionary errs at $1/2$. Setting $V = Z$ and $L_a(z, c) = \mathbf{1}\{D(z, c) \neq h_a(z)\}$ recovers equation 8. The fifteen-way choice loss in Appendix E.1 provides another instance. □

**Reuse error evaluates the complete model.** It scores the encoder and decoder together, not the operator family in isolation. In the regular specialization, an exactly recovered family paired with an encoder that emits a single constant code applies one operator at every state. By pointwise distinguishability, this prediction is correct only for its own command. Thus, $r_{\text{ind}} = 1 - 1/n$ even when $\varepsilon_{\text{match}} = 0$.

*Proof of Theorem 4.3.* All probabilities assume independent uniform $X, Z \in S$ and $A \in A$. Fix a source $x$ and call a command $a$ *locally correct at* $x$ when $D(x, E(x, h_a(x))) = h_a(x)$. Write $L_x \subseteq A$ for the locally correct commands and $c_x(a) = E(x, h_a(x))$ for the code the model infers there.

*The inferred codes are injective on $L_x$.* If $a, b \in L_x$ and $c_x(a) = c_x(b)$, applying $D$ at $x$ to this single code returns both $h_a(x)$ and $h_b(x)$. Thus, $h_a(x) = h_b(x)$, which implies $a = b$ by the distinctness of the true effects at $x$. Since $|A| = |\mathcal{I}| = m$, the partial injection $c_x|_{L_x}$ extends to a bijection $\pi_x\colon A \to \mathcal{I}$; fix any such extension.

*Replacing the inferred code by $\pi_x$ costs only the locally incorrect commands.* For $a \in L_x$, $\pi_x(a) = c_x(a)$, so the prediction $D(z, \pi_x(a))$ is the prediction the model actually makes at every target $z$. Writing $\text{err}(\pi) = \Pr[D(Z, \pi(A)) \neq h_A(Z)]$ and splitting on whether $A \in L_X$,

$$\mathbb{E}_X \,\text{err}(\pi_X) \;\leq\; \Pr\big[D(Z, c_X(A)) \neq h_A(Z),\; A \in L_X\big] + \Pr[A \notin L_X] \;\leq\; r_{\text{ind}} + \delta_{\text{loc}},$$

because $\Pr[A \notin L_X] = \delta_{\text{loc}}$ by equation 7 and the first term is at most $r_{\text{ind}}$ by equation 8.

*One global matching is at least as good as the average of the per-source ones.* The left side of equation 9 minimizes err over all bijections; therefore, $\varepsilon_{\text{match}} = \min_\pi \text{err}(\pi) \leq \mathbb{E}_X \,\text{err}(\pi_X)$, yielding equation 10. □

The argument uses only the $m$ codes and the distinctness of true effects, equation 1; it places no structural assumption on $D$.

**Corollary A.2** (Bijective correspondence at low reuse error)**.** *Under the conditions of Theorem 4.3 with $|\mathcal{I}| = m$, with the target drawn from a law not depending on the command and the commands uniform under the* 0–1 *loss, if $r_{\text{ind}} < 1/m$ then $\varepsilon_{\text{match}} \leq r_{\text{ind}}$.*

*Proof of Corollary A.2.* If $\phi(a) = \phi(b)$ for $a \neq b$, the decoded state at each target matches at most one of $h_a(z)$ and $h_b(z)$. Consequently, under uniform commands, this dictionary incurs a risk of at least $1/m$. Since $\varepsilon_{\text{dict}} \leq r_{\text{ind}} < 1/m$ by Proposition 4.2, every minimizing dictionary must be injective, making it a bijection when $|\mathcal{I}| = m$. Thus, $\varepsilon_{\text{match}} = \varepsilon_{\text{dict}} \leq r_{\text{ind}}$. □

The four-state construction below attains $r_{\text{ind}} = \varepsilon_{\text{dict}} = 1/m$ with $\varepsilon_{\text{match}} > r_{\text{ind}}$, so the strict threshold cannot be relaxed.

**The two-sided relation.** Let $K$ be a structured candidate satisfying $\mathcal{E}_K = \mathcal{E}_H$, evaluated via $E_K$ under independent uniform sampling. Pointwise distinguishability and the shared valid edges give a unique bijection $\sigma_x\colon A \to \mathcal{I}$ at each state $x$, satisfying $k_{\sigma_x(a)}(x) = h_a(x)$. This yields

$$r_{\text{ind}} \;=\; \Pr_{x,z,a}\big[\sigma_x(a) \neq \sigma_z(a)\big], \tag{13}$$

the probability that the local command-to-code matchings disagree across two independently sampled states.

*Proof of the bound in Eq. 11.* Let $\Pi_x[a, c] = \mathbf{1}\{\sigma_x(a) = c\}$ be the $m \times m$ permutation matrix of $\sigma_x$, and define

$$M_{ac} = \sum_{x \in S} \mathbf{1}\{\sigma_x(a) = c\}, \qquad B = \frac{M}{N} = \frac{1}{N} \sum_{x \in S} \Pi_x.$$

As an average of permutation matrices, $B$ is doubly stochastic. Write $\mathcal{P}_m$ for the $m \times m$ permutation matrices and $\langle U, V \rangle_F = \sum_{a,c} U_{ac} V_{ac}$.

*Step 1: express both errors through $B$.* Reuse succeeds on $(x, z, a)$ if and only if $\sigma_x(a) = \sigma_z(a)$. Averaging Eq. 13 over independent uniform $x, z$ and uniform $a$ yields

$$1 - r_{\text{ind}}(H, K) = \frac{1}{m} \sum_{a,c} \Big( \frac{M_{ac}}{N} \Big)^2 = \frac{\|B\|_F^2}{m},$$

$$1 - \varepsilon_{\text{match}}(H, K) = \frac{1}{m} \max_{\Pi \in \mathcal{P}_m} \langle B, \Pi \rangle_F.$$

*Step 2: the lower bound.* Since $B$ averages the matrices $\Pi_x$,

$$\|B\|_F^2 = \frac{1}{N} \sum_{x \in S} \langle B, \Pi_x \rangle_F \leq \max_{\Pi \in \mathcal{P}_m} \langle B, \Pi \rangle_F.$$

Combined with Step 1, this gives $\varepsilon_{\text{match}} \leq r_{\text{ind}}$.

*Step 3: the upper bound.* Let $\pi_\star$ attain the maximum and define $\rho = 1 - \varepsilon_{\text{match}} = \frac{1}{m} \sum_a B_{a,\pi_\star(a)}$. Applying the Cauchy–Schwarz inequality and noting $0 \leq B_{a,\pi_\star(a)} \leq 1$ yields

$$\rho^2 \leq \frac{1}{m} \sum_a B_{a,\pi_\star(a)}^2 \leq \frac{\|B\|_F^2}{m} = 1 - r_{\text{ind}}.$$

Consequently, $r_{\text{ind}} \leq 1 - \rho^2 = 2\varepsilon_{\text{match}} - \varepsilon_{\text{match}}^2$. □

**Sharpness of the threshold and necessity of shared valid edges.** The bijections $\sigma_x$ exist only where both families produce the edge. Elsewhere, the candidate's induced encoder is undefined, and an extension of it to the true source edges need not satisfy the lower bound in equation 11. On four states, let

$$h_0 = \text{id}, \quad h_1 = (0\,1)(2\,3), \qquad k_0 = (2\,3), \quad k_1 = (0\,2)(1\,3).$$

Both families are pointwise distinguishable, but their valid-edge sets differ. Extend the induced encoder of $K$ by returning code 0 outside its valid edges; it then returns code 0 on every true source edge. Operator $k_0$ agrees with each true command on half the states, whereas $k_1$ agrees with neither on any state. Thus

$$\delta_{\text{loc}} = r_{\text{ind}} = \varepsilon_{\text{dict}} = \tfrac{1}{2} = \tfrac{1}{m}, \qquad \varepsilon_{\text{match}} = \tfrac{3}{4}.$$

This violates the lower bound in equation 11 without shared valid edges, and it shows why Corollary A.2 requires a strict inequality. Theorem 4.3 still holds ($3/4 \leq 1$).

**Asymptotic sharpness of the leading factor 2.** On the XOR pair, conjugation by $\tau$ moves only the four states $\{0, 1, a, a \oplus 1\}$, yielding

$$M_{aa} = n \ \ (a \in \{0, 1\}), \qquad M_{aa} = n - 4, \ \ M_{a,a\oplus 1} = 4 \ \text{ otherwise},$$

with all other entries zero. For $n \geq 8$, the diagonal entry of each row is at least every other entry in that row, so the identity matching is optimal. Thus,

$$\varepsilon_{\text{match}} = \frac{4(n-2)}{n^2}, \qquad r_{\text{ind}} = \frac{8(n-2)(n-4)}{n^3}, \qquad \frac{r_{\text{ind}}}{\varepsilon_{\text{match}}} = 2 - \frac{8}{n} \longrightarrow 2.$$

At $n = 8$, this ratio equals 1, an equality case of the lower bound. As $n$ grows, it approaches the leading factor 2.

**A minimal code on the XOR pair.** At $d = 3$, let $X$ and $Y$ be independent and uniform, and read the XOR pair as a latent-action model:

$$E(x, y) = \tau(x) \oplus \tau(y), \qquad D(z, c) = \tau\big(\tau(z) \oplus c\big), \qquad C = E(X, Y).$$

This model reconstructs $D(x, E(x, y)) = y$ exactly, closes the self-generated cycle exactly, uses composition-closed decoder operators, and matches the truth's persistence law. The code takes exactly $n = 8$ values, the minimum for exact deterministic one-step reconstruction over all state pairs, and

$$H(C) = H(C \mid X) = H(C \mid Y) = 3 \text{ bits}, \qquad I(C; X) = I(C; Y) = 0.$$

Thus, the code carries no information about either endpoint separately. The reuse error is still $3/8$. Shrinking the code, decorrelating it from the endpoints and forbidding literal copying of the target are therefore not sufficient, singly or together, to identify the true operator family.

**Guarantees under nonuniform coverage.** Full prefix support identifies $H$ in the regular specialization (Corollary 3.4) but does not bound the transport error. Consider the XOR pair at $n = 8$. Draw the start state and the first command uniformly, and reuse the first command in the third slot. Draw the intervening command from $(1 - \eta)\,\delta_e + \eta\,\mathrm{Unif}(G)$, where $\delta_e$ is the point mass at the identity and $\eta \in (0, 1]$. Every prefix has positive probability. When the second action is the identity, both families return to $x$, yielding $r_Q = \eta \cdot \frac{3}{8}$ while $\varepsilon_{\mathrm{match}} = \frac{3}{8}$. If the collection law $Q$ over triples satisfies $Q \geq \alpha U$ pointwise against the uniform law $U$, then $r_Q \geq \alpha\, r_{\mathrm{ind}} \geq \alpha\, \varepsilon_{\mathrm{match}}$ under the conditions of Eq. 11, giving $\varepsilon_{\mathrm{match}} \leq r_Q/\alpha$. In this example $\alpha = \eta$ and the inequality is tight. A reuse collection protocol must therefore specify its distribution over the intervening command.

# B Additional theory

This section analyzes how action frequencies, collection policies, and anonymous pairing affect identification.

## B.1 Action-frequency information

The enumeration of Section 3.2 assumes a uniform command prior; we ask whether non-uniform command frequencies resolve the ambiguity.

**Proposition B.1** (Recovery from state-independent distinct action frequencies)**.** *Assume sharp transitivity and full-support starts. Let the action be drawn independently of the start state with unknown probabilities $\pi(h) > 0$ whose $n$ values are* pairwise distinct. *If the population one-step law $J(s, t) = \Pr(x_1{=}t \mid x_0{=}s)$ is known, the unlabeled operator family is recovered exactly. Persistence is not required.*

*Proof.* Sharp transitivity guarantees that the operator taking $s$ to $t$ is unique, yielding $J(s, t) = \pi(h_{s,t})$. Because the values $\pi(h)$ are pairwise distinct, each defines a unique level set,

$$\{(s, t) \in S^2 \;:\; J(s, t) = \pi(h)\} \;=\; \{(s, h(s)) : s \in S\},$$

which is exactly the graph of $h$. These graphs uniquely determine $H$. □

Non-uniformity alone does not guarantee recovery. On $\mathbb{Z}_2^3$, assign the identity command probability $1/2$ and each of the remaining seven probability $1/14$. Conjugation by $\tau$ fixes the identity, while the other seven carry equal weight. Consequently, the relabeling in Lemma A.1 leaves the run law unchanged, and the XOR pair from Appendix A.6 induces the same persistence law at every horizon.

The level-set argument uses the same command probabilities at every state. If the start-state-dependent policy is unknown, an alternative family can absorb the probabilities into a different policy, as Proposition B.2 shows.

**Proposition B.2** (Unknown state-dependent action priors)**.** *Let $H$ and $H'$ be run-equivalent, and let $\pi(\cdot \mid s)$ be any start-state-dependent policy, where $h^H_{s,t}$ denotes the unique operator of $H$ taking $s$ to $t$. Define the matched policy*

$$\pi'(h' \mid s) \;=\; \pi\big(h^H_{s,h'(s)} \mid s\big).$$

*At each $s$, this mapping permutes the probabilities, so positivity, normalization and pairwise distinctness are preserved, and the matched operators generate identical persistence runs. Consequently, the laws of $T$-transition persistence runs under $(H, \pi)$ and $(H', \pi')$ match, $P^{(T)}_{H,\pi} = P^{(T)}_{H',\pi'}$, for every $T$.*

Frequencies therefore identify the family only through a *single action correspondence shared across all states*, which Schur (2026) obtains from continuity over a connected state space.

### B.2 Unknown history-dependent policies absorb a relabeling

**Proposition B.3** (Absorption of a relabeling by an unknown policy)**.** *Let $H$ and $K$ be pointwise-distinguishable families sharing identical valid edges, $\mathcal{E}_H = \mathcal{E}_K$. Every history-dependent policy generating trajectories under $H$ has an equivalent history-dependent policy under $K$ that generates the same state-trajectory distribution.*

*Proof.* Let $X_{0:t}$ denote the observed history up to $X_t$, and let $\pi_H$ be the policy under $H$. The next-state distribution is $P(X_{t+1} = s \mid X_{0:t}) = \sum_a \pi_H(a \mid X_{0:t}) \mathbf{1}\{h_a(X_t) = s\}$. Define $\pi_K(c \mid X_{0:t}) = P(X_{t+1} = k_c(X_t) \mid X_{0:t})$. Because pointwise distinguishability and the shared edge set make $c \mapsto k_c(X_t)$ a bijection onto the same successors enumerated by $a \mapsto h_a(X_t)$, $\pi_K$ is a valid distribution over codes. Consequently, the next-state law under $(K, \pi_K)$ equals that under $(H, \pi_H)$ at every history. By induction, the full trajectory laws agree. □

When nothing ties command identities across time, the unknown policy absorbs any reassignment of commands to edges. Persistence adds a known constraint on how command identity is shared over time: one unnamed command is held fixed across several transitions.

**Uniform single-transition observations.** In the regular specialization, the one-step distribution is uniform over ordered state pairs for every group operation (Proposition 3.1), so single transitions cannot distinguish group operations of the same order.

**Repeat-or-resample policies.** Assume that $A_{t+1} = A_t$ with probability $p$ and that $A_{t+1}$ is redrawn uniformly otherwise. The last observed edge $(x, y)$ uniquely determines the command that produced it, yielding

$$\Pr(X_{t+1} = z \mid X_{t-1} = x, X_t = y) = p\,\mathbf{1}\{z = C_H(x, y)\} + \frac{1-p}{m}\,\mathbf{1}\{(y, z) \in \mathcal{E}_H\}.$$

This transition probability depends on the operator family only through $\mathcal{E}_H$ and $C_H$. Consequently, families sharing these properties remain indistinguishable for any $p \in [0, 1]$, interpolating between uniform independent draws ($p = 0$) and pure persistence ($p = 1$). This result covers symmetric policies only, not arbitrary non-uniform or history-dependent ones. A state-independent frequency difference can separate the pair (Proposition B.1).

### B.3 Identification from same-command pairs

**Proposition B.4** (Identification from a complete same-action relation)**.** *Declare two transitions equivalent when produced by the same unknown action. Under sharp transitivity, this complete equivalence relation uniquely determines the operator family. Because this relation is invariant to any relabeling of the external command names, it resolves none of them.*

*Proof.* Under sharp transitivity, a unique operator $h_{s,t} \in H$ maps each ordered pair $(s, t)$. The relation defines $(s, t) \sim (s', t')$ if and only if $h_{s,t} = h_{s',t'}$. Consequently, its equivalence classes are the graphs of the operators in $H$, which uniquely determine $H$. Because this defining condition rests on operator equality rather than command names, any relabeling of the names leaves $\sim$ unchanged. □

More generally, any signal invariant to name relabeling cannot distinguish external command names. The frequency values analyzed in Proposition B.1 are exactly such a signal.

### B.4 Recovery under observational equivalence

**Proposition B.5** (Recovery probability under a uniform prior over an observationally equivalent class)**.** *Let $\mathcal{C} = \{H_1, \ldots, H_q\}$ be a set of composition-closed families inducing the* identical *observation law,*

$$P_{H_1} = \cdots = P_{H_q}.$$

*Draw the ground truth $H \sim \mathrm{Uniform}(\mathcal{C})$ and subsequently sample data $X \sim P_H$. Every $\mathcal{C}$-valued estimator $\hat{H}$, whether deterministic or randomized, achieves exactly $\Pr[\hat{H}(X) = H] = 1/q$. Any estimator permitted to abstain or predict outside $\mathcal{C}$ achieves $\Pr[\hat{H}(X) = H] \leq 1/q$.*

*Proof.* The identical observation laws make $X$ independent of the uniformly drawn truth $H$. The estimator, including any independent algorithmic randomness, therefore satisfies

$$\Pr[\hat{H}(X) = H] \;=\; \frac{1}{q}\sum_{i=1}^{q} \Pr[\hat{H}(X) = H_i].$$

The sum equals one for a $\mathcal{C}$-valued estimator and is at most one if the estimator can abstain or return a value outside $\mathcal{C}$. □

The probability in Proposition B.5 averages over the uniformly drawn truth and is a property of the observation model itself. In contrast, an empirical restart frequency measured against a single fixed truth also reflects the estimator's tie-breaking rules and optimization biases.

## C Extended comparison with the closest designs

**Four further designs.** CD-LAM (Wei et al., 2026b) also reuses codes in different visual contexts, but it evaluates transferred rollouts against reference videos using visual-fidelity and foreground-motion measures. We instead evaluate directly against the environment's response to a re-executed command. ShadowDancer (Cao et al., 2026) obtains the cross-state correspondence from constructed video pairs that replay the same dynamics under resampled appearance. VERA (Li et al., 2026) trains an embodiment-specific inverse dynamics model on recorded actions, and V-JEPA 2 (Assran et al., 2025) adds action-conditioned post-training on robot actions after action-free video pretraining. Our re-execution protocol instead uses only the relation that two executions share a command, not the command's label.

**Topological versus graph connectedness.** Lachapelle (2025) and Schur (2026) propagate cross-state correspondence using the continuity and topological connectedness of a metric state space. In contrast, Theorem 3.3 uses the connectedness of the discrete valid-edge graph. Neither property implies the other. A connected continuous state space carrying only identity dynamics has a transition graph joining no two states. Conversely, a finite cycle is connected as a transition graph while being totally disconnected in the discrete topology.

## D Additional experimental details

### D.1 How the candidate counts of Table 1 were enumerated

A candidate is a family of $n$ permutations over $n$ states. Two families are the same candidate if and only if they are equal as *sets* of permutations. Because the learner never observes external code identities, code indices do not define the candidate.

In the regular specialization, every ordered pair is a valid edge, so the edge set carries no distinguishing information, and the persistence law acts only through the continuation rule. Held across a persistence run, a command traces a directed cycle from its start state. The likelihood of a persistence run under a sharply transitive family therefore depends on the family only through its *cycle set*: the directed cycles its members trace, each taken up to rotation but not reversal. Thus, two sharply transitive families are run-equivalent if and only if their cycle sets match. The enumeration searches over these sets.

**Algorithm 1** Tabular latent-action learner on persistence runs

**Require:** Action-unlabeled runs $\mathcal{X}$ on $n$ states, an initialization seed, the associativity weight $\lambda_{\text{assoc}} = 80$

**Optimize the relaxed table**

Initialize $T_{asv} \sim \mathcal{N}(0, 0.02^2)$ from the seed
**for** $u = 0, \dots, 2999$ **do**
  $\tau_u \leftarrow 0.08^{\,u/2999}$, $b_u \leftarrow \min(1,\ u/1500)$ ▷ Temperature and warm-up
  $P_a \leftarrow \text{Sinkhorn}_{20}(T_a/\tau_u)$ for each action slice $a$
  $\mathcal{L}_u \leftarrow \mathcal{L}_{\text{run}}(P; \mathcal{X}) + \mathcal{L}_{\text{Latin}}(P) + b_u \lambda_{\text{assoc}} \mathcal{L}_{\text{assoc}}(P) + 1.5\,\mathcal{L}_{\text{id}}(P) + 0.2\,\mathcal{L}_{\text{inv}}(P)$
  Update $T$ with Adam (Kingma & Ba, 2015) (learning rate 0.08) to minimize $\mathcal{L}_u$

**Discretize and compute diagnostics**

$P^\star \leftarrow \text{Sinkhorn}_{20}(T/0.08)$
$\widehat{k}_a(s) \leftarrow \arg\max_v P^\star_a[s, v]$ for every $a, s$, giving $\widehat{K} = (\widehat{k}_a)_a$
$\ell_{\text{relax}} \leftarrow \mathcal{L}_{\text{run}}(P^\star; \mathcal{X})$
$c_r \leftarrow$ number of operators of $\widehat{K}$ reproducing every transition of run $r$, for each $r$
$g \leftarrow$ indicators of bijectivity, sharp transitivity and composition closure of $\widehat{K}$
$f_{\text{sel}} \leftarrow [\,\ell_{\text{relax}} - \log n \le 10^{-2}\,]$, $f_{\text{exact}} \leftarrow [\,c_r = 1 \text{ for every } r\,]$
**return** $\widehat{K}$, $\ell_{\text{relax}}$, $\{c_r\}_r$, $f_{\text{sel}}$, $f_{\text{exact}}$, $g$

The search fixes the true family's cycle set and covers it. Each cycle may be used once. By examining every cover of the cycle set without ever enumerating the full symmetric group $\text{Sym}(n)$, the search yields exhaustive counts rather than sampled ones. We then test composition closure on each enumerated family to determine $|\mathcal{V}_{\text{grp}}|$. For $\mathbb{Z}_2^3$, the reported observational count enumerates only those families containing the identity, and Table 1 marks this value as a lower bound.

### D.2 Exact specification of the synthetic learner

Algorithm 1 trains and evaluates a single model for one dataset and one initialization seed. Section 5.2 reports selected fits out of all 256 restarts per group. The twenty Sinkhorn iterations (Sinkhorn, 1964) alternate over the result and source indices, so each slice remains approximately doubly stochastic at this finite count.

**The four penalties.** Let $P \in [0,1]^{n\times n\times n}$ denote the relaxed table, where $P_{abx}$ represents the relaxed probability that code $a$ maps state $b$ to result $x$. Codes and states share a single index set, with $e = 0$ defining the identity index. Every mean denotes the uniform average over the specified indices, and every logarithm is natural. The four penalties are:

$$\mathcal{L}_{\text{Latin}} = \text{mean}_{ab}\big(\textstyle\sum_x P_{abx} - 1\big)^2 + \text{mean}_{bx}\big(\textstyle\sum_a P_{abx} - 1\big)^2 + \text{mean}_{ax}\big(\textstyle\sum_b P_{abx} - 1\big)^2 + 0.02\,\text{mean}_{ab}\, H(P_{ab\cdot}), \tag{14}$$

$$\mathcal{L}_{\text{assoc}} = \text{mean}_{abc} \textstyle\sum_x \Big(\sum_y P_{aby} P_{ycx} - \sum_y P_{bcy} P_{ayx}\Big)^2, \tag{15}$$

$$\mathcal{L}_{\text{id}} = \text{mean}_{bx}\,\big(P_{ebx} - \delta_{bx}\big)^2 + \text{mean}_{ax}\,\big(P_{aex} - \delta_{ax}\big)^2, \tag{16}$$

$$\mathcal{L}_{\text{inv}} = \text{mean}_a \min_b \Big(\text{mean}_x\,\big(P_{abx} - \delta_{xe}\big)^2 + \text{mean}_x\,\big(P_{bax} - \delta_{xe}\big)^2\Big), \tag{17}$$

where $H(P_{ab\cdot}) = -\sum_x P_{abx} \log P_{abx}$, with probabilities clipped away from zero for numerical stability. The associativity term sums over the result index $x$ and averages over $(a, b, c)$; averaging over $x$ as well would weaken it by a factor of $n$.

**Likelihood and budget.** The objective combines the persistence-run negative log-likelihood with the four penalties weighted as in Algorithm 1. The budget allocates $R = 768$ runs of 14 transitions per dataset and 3,000 optimization steps. We execute 8 restarts across 32 dataset seeds in each group, yielding 256 restarts per group and 1,280 in total. At $n = 8$, the likelihood uses all 14 transitions of each run. Because each run keeps a single latent code, we form its likelihood conditional on that code and marginalize the code afterwards. Writing $P_a[s, t]$ for the relaxed weight of $s \to t$ under

code $a$,

$$\mathcal{L}_{\text{run}} = -\frac{1}{R}\sum_{r}\log\Big[\tfrac{1}{n}\textstyle\sum_a \prod_t P_a\big[s_t^{(r)}, s_{t+1}^{(r)}\big]\Big]. \tag{18}$$

Because the likelihood conditions on the observed start state, a table uniquely explaining every run scores exactly $\log|G|$. This value serves as the selection reference of Section 5.1.

**Structural limits of the parameterization.** The parameterization represents a binary operation as a table and drives each slice toward a permutation. However, it does not make the family closed under composition. While a sharply transitive family forms a Latin square, a Latin square need not form a group. Eq. 15 therefore *promotes* closure, and the evaluation *verifies* it on the discretized table $\widehat{K}$.

**Selection and exact verification.** Selection uses the relaxed-likelihood criterion $f_{\text{sel}}$ of Algorithm 1, with tolerance $10^{-2}$. After discretization, exact verification requires one compatible held code for each persistence run ($f_{\text{exact}}$), or one compatible ordered code pair for each reuse trial (Appendix D.4). We report the structural indicators $g$ separately. All selected tables pass the exact check, and we call them *selected exact fits*. Every reported recovery is a property of the discretized family. Any tolerance between $10^{-3}$ and $10^{-2}$ selects the same restarts.

### D.3 The selected exact fits, per group

Figure 4 reports exact family recovery per group, with state labels held fixed. On $D_4$, the eight selected exact fits missing the true family return its left/right partner, the right regular action of Appendix A.2. This partner still disagrees with the truth on 192 of the 512 triples. We check identity, inverse and associativity independently of the exact fit criterion; 216 restarts pass this structural check.

**The closed class on $\mathbb{Z}_2^3$.** Against the truth under the independent uniform law, the 30 closed families compatible with the persistence law have exact reuse risk 0 (the truth), $3/8$ (seven families), $9/16$ (fourteen) and $21/32$ (eight); the 29 incorrect ones average $63/116 \approx 0.543$. Compared as sets of permutations, the 27 selected exact fits fall across 20 dataset seeds and realize 18 distinct families, none of which is the true one, and their risks take only these values (Section 5.2). The class fixes the possible errors; which member a restart reaches depends on initialization (Section 5.4).

### D.4 Supplying the reuse trials to the learner

In the regular specialization, Corollary 3.4 identifies the family from the population native-reuse law. We therefore train the same learner on finite samples drawn from this law. A three-slot trial $(x, y, z, w)$ applies an initial command, applies a second, and then reuses the first, with neither command named. Given $m$ codes and a uniform prior over the ordered pair, the likelihood is

$$p(y, z, w \mid x) \;=\; \frac{1}{m^2}\sum_a\sum_b P_a[x, y]\, P_b[y, z]\, P_a[z, w], \tag{19}$$

where the index $a$ in the first and third factors encodes the reuse. The training procedure of Appendix D.2 changes only in the data and the likelihood. For reuse trials, the selection reference is $2\log m$, and exact verification counts compatible ordered code pairs:

$$f_{\text{sel}}^{\text{reuse}} = \mathbf{1}\big\{\ell_{\text{relax}}^{\text{reuse}} - 2\log m \le 10^{-2}\big\}, \qquad f_{\text{exact}}^{\text{reuse}} = \mathbf{1}\big\{c_r^{\text{reuse}} = 1 \text{ for every trial } r\big\},$$

where $c_r^{\text{reuse}}$ is the number of ordered code pairs reproducing all three transitions of trial $r$. The reference $2\log m$ is the population cross-entropy of the correctly specified model under uniform commands. All three conditions contain 10,752 transitions. The reuse condition comprises 3,584 trials. The short-persistence condition matches this sequence count and length by holding a single command fixed across all three slots, so it lacks only a command re-executed *after* an intervening one. Each reuse-trial dataset covers 509 to 512 of the 512 possible prefixes $(x, a, b)$, 511.3 on average. All selected solutions pass their protocol's exact check. Table 2 reports the counts for all three conditions, and the persistence condition reproduces Figure 4. The conditions match the total transition count but differ in cross-state command information, and the comparison does not separate that information from its effect on optimization.

**Control with a conjugate generating family.** When trained on trials generated by the conjugate family $k_a = \tau h_a \tau^{-1}$, which persistence cannot distinguish from the truth, the learner recovers *that* family on all 4 restarts of this check. On $\mathbb{Z}_2^3$, the incorrect conjugate assigns probability zero to 192 of the 512 possible true reuse trials. Its exact negative log-likelihood under Eq. 19 is infinite, while the persistence likelihood scores both families at 2.079.

### D.5 Sensitivity to initialization bias

We compare an unmodified initialization with an additive logit bias of 0.01 toward either the true family or one fixed incorrect member of the same run-equivalent class of 30, leaving all other settings unchanged. Neither target is distinguishable from the data. The evaluation uses 32 new datasets. We ran 8 restarts per seed, so 256 per condition.

We measure recovery across *all* 256 restarts. This outcome requires both selection and exact equality to the designated family. Table 4 gives the counts, and Section 5.4 reports the paired contrasts, whose intervals resample paired dataset seeds. Because the selection rate also changes (Table 4), the contrasts combine effects on fitting and on family choice. Exact verification leaves these directional conclusions unchanged.

## E Supporting empirical diagnostics

This appendix tests whether an inferred code selects the correct outcome at a new state (Appendix E.1) and what re-encoding adds to the source-scored cycle (Appendix E.2).

### E.1 Functional reuse on frozen latent-action checkpoints

We evaluate a single frozen stage-1 checkpoint per Procgen environment. The sixteen checkpoints share an identical architecture and training length, each saved at step 50,000, the end of stage one. Because the number of scored evaluation pairs ranges from 1,058 per environment down to none, every pooled rate below is the accuracy pooled over eligible evaluation pairs rather than an average over environments. Tables 8 and 9 detail the main sample of Table 7.

**Endpoint sampling.** An evaluation pair combines two states drawn from distinct environment instances with independent random streams, so the target is never a descendant of the source. Each instance runs Procgen's easy distribution over levels 0 to 199 and is rolled forward under uniformly random buttons before each pair, 3 to 24 steps and then one more for the two-frame history. We pair the two instances in collection order, advancing the target instance only when the source state passes its screens.

**Scoring rule.** At each end, we save the environment state and branch all fifteen buttons from it, so both sets of successors are exact counterfactuals. We sample one command uniformly and execute it at the source. The encoder reads the resulting transition in its native three-frame format, and the world model receives two history frames at the target together with the quantized latent. We score a prediction by selecting the nearest of the fifteen counterfactual target successors under pixel mean squared error; the pair passes if the selected successor is that of the executed command. Ties for the nearest successor never include the executed command's successor, except in one starpilot target-local evaluation, which we score as a failure. Scoring it as a success raises starpilot's target-local accuracy from 0.610 to 0.611 and changes no pooled rate. The self-cycle re-encodes the world model's own target prediction, carries the re-inferred code back to the source, and checks whether it selects the observed source successor. This adapts the source–target–source round trip of Garrido et al. (2026) and Zhang et al. (2026), who compare perceptual prediction error on the source video before and after the cycle. Our fifteen-way pass or fail is not their metric.

**Compared conditions.** We score five conditions on identical evaluation pairs. We apply the code at the source it came from (*source-local*) and at the target (*matched*). Three controls supply a code inferred for a different command at the same source (*mismatched*), a *zero latent*, and the last observed frame with no model (*copy last*). A sixth condition, target-local, uses the same pairs and is described below.

Table 7: **The Procgen evaluation samples.** Every reported Procgen rate comes from one of these samples. The target-local condition uses the main sample's pairs, and the shuffled-code subset is part of the return-code sample.

| Evaluation set | Eligible | Reported |
|---|---|---|
| Replication sample | 5,992 | cycle-reuse association, 2.7 pt |
| Main sample | 6,027 | forward transfer, 25.9 pt |
| target-local condition | 6,027 | target-local gap, 21.4 pt |
| Return-code sample | 5,848 | forward contrast, 24.6 pt |
| shuffled-code subset | 5,226 | return-code conditions, 0.59 pt |

**Exclusions.** We exclude an evaluation pair if the executed command's successor is not unique at either end, if a branch ends the episode, or if the two-frame history would cross an episode boundary. Table 8 accounts for all 32,000 sampled pairs; 6,027 (18.8%) are scored. The screening order is source boundary, source branch reset, source collision, then the same three checks at the target, so each excluded pair is counted in exactly one category. Source collisions exclude the most pairs, 23,959 (74.9% of all sampled pairs). In these cases, the executed command is indistinguishable in one step from another button. The observable analogue of Eq. 1 therefore fails in every environment, although we cannot extend this conclusion to the hidden state. Theorem 4.3 is not tested here. Command effects collide in image space, and the code is a product quantization: four latent vectors, each quantized against two 64-entry codebooks. This yields eight indices rather than an alphabet the size of the command set. The model also reads finite frame histories, and the evaluated states follow the rollout distribution rather than the uniform draw.

**Cross-state transfer.** The matched code exceeds the mismatched control in 14 of the 15 environments that retain scored pairs, with one tie and no reversals, and exceeds the zero latent in 13, with one tie and one reversal (Table 9). Paired on identical evaluation pairs, the matched condition exceeds the mismatched control by 25.9 percentage points $[23.8, 28.1]$, the zero latent by 23.7 points $[21.6, 25.7]$, and the last frame by 28.9 points $[26.8, 31.0]$. Local application at the source reaches 53.4%, 20.4 points above matched accuracy $[18.1, 22.8]$.

Table 8: **Exclusions of sampled pairs.** Categories are mutually exclusive under the screening order of the text.

| Outcome | Pairs | Share |
|---|---|---|
| Source collision | 23,959 | 74.9% |
| Target collision | 815 | 2.5% |
| Source branch reset | 529 | 1.7% |
| Source boundary | 400 | 1.2% |
| Target branch reset | 148 | 0.5% |
| Target boundary | 122 | 0.4% |
| Scored | 6,027 | 18.8% |

**Target-local control.** For a source transition $(x, y)$ and a target transition $(z, w)$ produced by the same command, the target-local condition compares

$$\widehat{w}_{\text{transfer}} = D\big(z, E(x, y)\big) \qquad \text{with} \qquad \widehat{w}_{\text{local}} = D\big(z, E(z, w)\big),$$

using the same native frame format, decoder and candidate successors at the target. Its encoder receives the observed target successor, so it is a reconstruction control, not a transfer method. Target-local accuracy is 54.4%, 21.4 points $[19.3, 23.7]$ above matched codes (crossed $[17.4, 25.7]$), and higher in 13 of the 15 environments with scored pairs (Table 9). The observed target-local-minus-source-local difference is 1.0 percentage point (95% CI $[-0.7, 2.8]$).

**Sensitivity across training runs.** On the same 851 bigfish evaluation pairs, the main checkpoint has source-local, target-local and matched accuracies of 0.691, 0.684 and 0.318. A second training run with the same configuration and training step scores 0.697, 0.693 and 0.478. Local accuracies are similar, whereas cross-state accuracy differs by 16.0 percentage points. This comparison involves two training runs in one environment.

**Uncertainty.** We cluster evaluation pairs by environment and source episode, since consecutive pairs roll forward within one episode, and resample the clusters to obtain the primary interval. The source and target episode partitions are *crossed*, not nested, so we also report a crossed bootstrap that resamples source and target episode labels independently and weights each pair by the product of its two multiplicities (Owen, 2007). Over 1,536 source and 793 target clusters and 4,000 replicates, it gives $[21.9, 30.3]$ points against the primary $[23.8, 28.1]$; the contrast excludes zero under both.

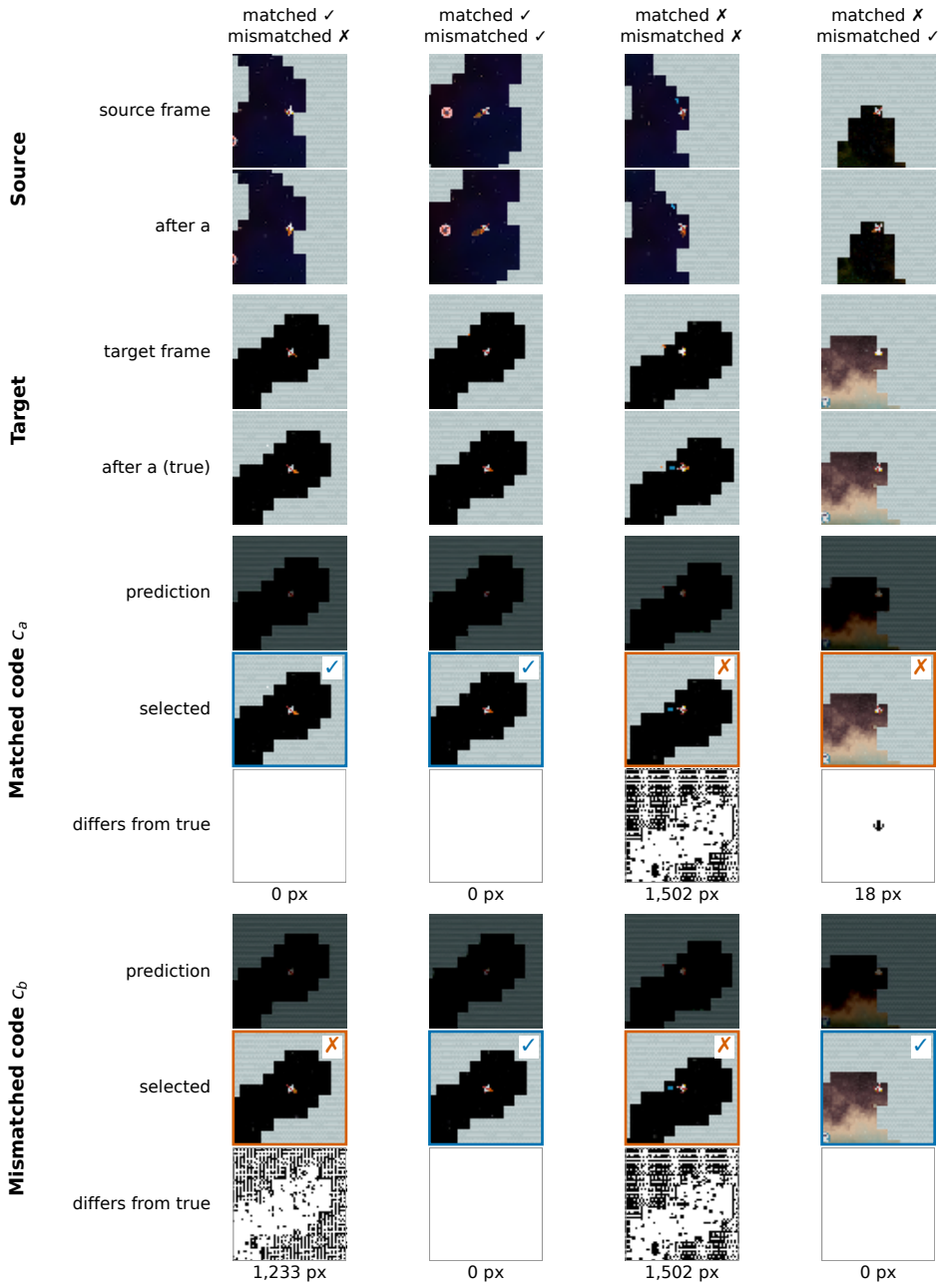



Figure 5: **Source-to-target reuse under the counterfactual scoring rule.** Each column is the first caveflyer evaluation pair of the main sample in one of the four matched/mismatched outcome cells, which hold 27.7, 5.3, 65.3 and 1.8% of the 6,027 scored pairs. The columns illustrate the cells, not their frequency. The rows show the last source and target history frames with their successors under the executed command $a$. For the matched code $c_a$ and the mismatched code $c_b$ (inferred under a different command $b$), they then show the generated *prediction* and the *selected* successor nearest to it under pixel mean squared error. A selected successor is framed blue with a check when it is the executed command's successor and vermillion with a cross otherwise. *Differs from true* marks in black where the selected successor differs from the true one; a white mask means they are equal, not that the prediction reconstructs the frame exactly.

Table 9: **Functional reuse on frozen stage-1 checkpoints, all sixteen Procgen environments.** Every rate is the fraction of that environment's scored evaluation pairs on which the nearest of the fifteen counterfactual successors is the executed command's, scored at the target for Target-local and the transfer conditions and at the source for Source-local and Cycle; the conditions are those of Section 5.1. Rows are ordered by Scored, and Pooled is the accuracy pooled over eligible evaluation pairs. Fruitbot retains no scored pair. In plunder one command contributes every eligible pair, and all seven conditions score zero on those pairs.

| Env | Scored | Source-local | Target-local | Matched | Mismatched | Zero latent | Copy last | Cycle |
|---|---|---|---|---|---|---|---|---|
| caveflyer | 1058 | 0.212 | 0.218 | 0.134 | 0.113 | 0.125 | 0.072 | 0.121 |
| starpilot | 1058 | 0.589 | 0.610 | 0.383 | 0.066 | 0.094 | 0.081 | 0.098 |
| heist | 948 | 0.767 | 0.790 | 0.289 | 0.084 | 0.113 | 0.037 | 0.126 |
| dodgeball | 933 | 0.673 | 0.685 | 0.593 | 0.031 | 0.035 | 0.003 | 0.132 |
| bigfish | 851 | 0.691 | 0.684 | 0.318 | 0.073 | 0.094 | 0.020 | 0.109 |
| bossfight | 606 | 0.112 | 0.132 | 0.078 | 0.058 | 0.076 | 0.036 | 0.111 |
| miner | 178 | 0.893 | 0.803 | 0.781 | 0.017 | 0.157 | 0.000 | 0.331 |
| plunder | 101 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| jumper | 85 | 0.694 | 0.800 | 0.494 | 0.071 | 0.035 | 0.047 | 0.118 |
| maze | 72 | 0.931 | 0.806 | 0.792 | 0.069 | 0.236 | 0.000 | 0.194 |
| ninja | 54 | 0.593 | 0.796 | 0.537 | 0.019 | 0.000 | 0.056 | 0.056 |
| climber | 33 | 0.394 | 0.364 | 0.303 | 0.030 | 0.000 | 0.000 | 0.030 |
| coinrun | 27 | 0.556 | 0.519 | 0.370 | 0.333 | 0.296 | 0.000 | 0.370 |
| leaper | 21 | 0.619 | 0.571 | 0.286 | 0.143 | 0.333 | 0.000 | 0.429 |
| chaser | 2 | 1.000 | 1.000 | 1.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| fruitbot | 0 | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| Pooled | 6027 | 0.534 | 0.544 | 0.330 | 0.070 | 0.093 | 0.041 | 0.123 |

**Cycle closure and reuse.** The round trip returns the observed source successor on $12.3\%$ of evaluation pairs, and the re-encoded code never equals the original ($1.07\%$ of indices agree). Comparing pairs within each environment, those that close the cycle have $5.1$ points $[1.6, 9.1]$ higher reuse accuracy ($2.7$ points $[-0.6, 5.6]$ on the replication sample).

**Relation to the dictionary bound.** Let $V$ denote the whole target-side record (the target history and its fifteen one-step outcomes), and let $L_a(V, c)$ indicate that code $c$ applied at that target fails to select the successor of command $a$. This loss has the form of Proposition 4.2. The sampler pairs instances in collection order and advances the target only after the source passes, so it does not enforce $C \perp V \mid A$, and the observed accuracy is an empirical cross-state evaluation, not a verified bound on fixed-dictionary risk.

### E.2 Source-side return-code controls

**Protocol.** We score four return codes with the same source-side rule that produces the cycle accuracy, changing only the applied code. These are the re-inferred code, a *shuffled* code drawn uniformly from the re-inferred codes of the pairs already scored in that environment, the code inferred for a different command at the target, and a zero latent. The zero-latent control of Table 9 does not answer this question, since it is scored at the target. The shuffled code keeps the environment's code distribution but carries this pair's command only by chance. We sampled $2{,}000$ pairs in each of sixteen environments, scoring $5{,}848$. Ten environments have the fifty scored pairs a shuffled draw needs and supply the $5{,}226$ pairs evaluated under all four return-code conditions.

**Results.** Re-encoding succeeds on $582$ pairs and the shuffled code on $551$, source-return accuracies of $11.1\%$ and $10.5\%$. The paired difference is $0.59$ percentage points, with a source-episode bootstrap interval of $[0.10, 1.07]$ and a crossed-bootstrap interval of $[-0.25, 1.46]$. Whether the interval excludes zero therefore depends on the estimator. Per environment, the interval includes zero everywhere except heist, and the difference is negative in two of the ten environments. The mismatched code scores $6.7\%$ and the zero latent $8.4\%$.

**Code-distribution diagnostics.** Source and re-inferred codes agree on $0.930\%$ of their $41{,}808$ indices, against $0.925\%$ expected from independent draws of each environment's code frequencies. Code agreement differs from functional reuse, since a decoder can ignore part of the code.