BEYOND PREDICTION: A COMPONENT-LEVEL STUDY OF CONCEPT RECOVERY IN IN-CONTEXT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context learning (ICL) enables language models to make accurate predictions from demonstrations, but prediction success alone does not reveal the extent to which the underlying concept has been recovered. Existing evaluations typically assess recovery at the level of the whole concept, making it difficult to identify which components are recovered and to separate model limitations from ambiguity in the demonstrations. We introduce a **component-level framework for concept recovery** that decomposes concepts into their defining components and uses controlled structural interventions to evaluate both component-level and whole-concept recovery under identifiable demonstrations. Across four instruction-tuned LLMs and three composition forms, we find that models often achieve high prediction accuracy despite poor whole-concept recovery, with systematic over-specification of relevant attributes and level constraints. Individual structural components are generally insufficient for whole-concept recovery, whereas jointly specifying complementary components, particularly relevant attributes and their level constraints, substantially improves recovery. However, this improvement is fragile: a single plausible but unnecessary level constraint can substantially disrupt recovery. Finally, joint supervision of recovery and implicit prediction substantially improves recovery while preserving prediction performance, with the learned capability transferring across composition forms. Overall, our component-level analysis provides a more diagnostic view of what models recover from demonstrations than the existing holistic view.

1 INTRODUCTION

Large language models (LLMs) have achieved strong performance across a wide range of tasks, from academic reasoning (Guo et al., 2025; Singh et al., 2025) to coding (El-Kishky et al., 2025; Comanici et al., 2025) and scientific applications (Boiko et al., 2023; Gottweis et al., 2026). Yet, successful behavior alone does not necessarily reveal whether a model has recovered the underlying structure that supports its predictions. Understanding what models infer from task-specific evidence therefore remains an important challenge for LLM interpretability (Luo & Specia, 2024; Singh et al., 2024; Huben et al., 2024; Chen et al., 2025).

In-context learning (ICL) provides a natural setting to study this question. By conditioning on a small set of labeled demonstrations, LLMs can adapt their predictions to the task specified by the demonstrations on subsequent queries (Brown et al., 2020; Min et al., 2022). A natural interpretation is that the demonstrations provide information about an underlying task concept that can be inferred and applied to new queries. This raises a more direct question: *when ICL succeeds, to what extent does the model recover the underlying concept from the demonstrations?* In particular, accurate query prediction does not necessarily imply faithful recovery of the concept that generates the labels. We study this prediction–recovery gap in controlled ICL settings where demonstrations are generated from predefined target concepts, allowing recovery to be evaluated against explicit ground truth.

Several lines of work have investigated the mechanisms underlying successful ICL (Olsson et al., 2022; Dai et al., 2023; Pan et al., 2023; Agarwal et al., 2024). Related approaches to rule or hypothesis recovery, including MIRAGE (Li et al., 2025) and iterative hypothesis refinement (Qiu et al., 2024), further examine whether models can explicitly infer rules from demonstrations. However, recovery in these settings is typically evaluated at the level of the recovered rule as a whole. Such

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.

holistic evaluation can determine whether a recovered concept is correct, but provides limited information about *which concept-defining components are recovered, missed, or incorrectly included*. It also does not by itself establish whether the demonstrations uniquely identify the target concept, making it difficult to distinguish limitations in model recovery from ambiguity in the evidence. These limitations leave open a more diagnostic question: *what does a model recover when it fails to recover the concept as a whole?*

To address this question, we move beyond holistic evaluation to a **component-level framework for concept recovery**, as illustrated in Figure 1. We represent each target concept through its constituent components and evaluate recovery separately at the component and whole-concept levels. We first ensure that the demonstrations are sufficient to uniquely identify the target concept, separating model-side recovery errors from ambiguity in the evidence. We then use controlled interventions to test whether individual or jointly specified concept-defining components support whole-concept recovery, and whether this recovery remains robust when a plausible but unnecessary structural element is introduced. Finally, we use supervised training to test whether the same structured recovery capability can be learned while retaining implicit prediction.

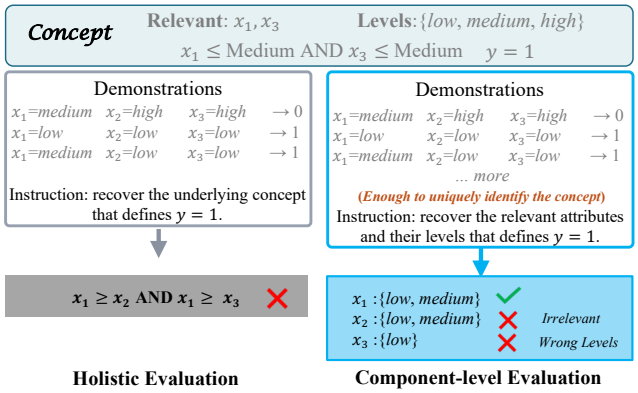

Figure 1: **Holistic versus component-level evaluation of concept recovery.** Given a target concept, we generate demonstrations from the concept and provide them to an LLM for concept recovery. *Holistic evaluation* assesses whether the recovered concept is correct as a whole, without revealing which concept-defining components are recovered correctly or incorrectly. We instead adopt a *component-level view*, using demonstrations sufficient to uniquely identify the target concept and evaluating recovery at the level of individual concept-defining components.

Our experiments reveal several empirical patterns in concept recovery. (1) Models often achieve high query prediction accuracy despite poor whole-concept recovery. Component-level analysis further reveals systematic over-specification: models tend to include irrelevant attributes and overly broad level constraints. (2) Individual structural components are generally insufficient for whole-concept recovery, whereas jointly specifying complementary concept-defining components, particularly the relevant attributes and their level constraints, substantially improves recovery. However, recovery remains fragile: a single plausible but unnecessary level constraint can substantially disrupt it. (3) Structured concept recovery is learnable. Inference-time reasoning alone does not reliably improve recovery, whereas joint training with recovery and implicit-prediction supervision substantially improves recovery while preserving implicit-prediction performance, and the learned capability transfers across composition forms.

We summarize our contributions as follows:

- **A component-level characterization of concept recovery.** We decompose concept recovery into constituent structural components and evaluate which components are correctly recovered, missed, or incorrectly included, going beyond a holistic assessment of whether the recovered concept is correct.
- **A controlled analysis of structural information for recovery.** We use identifiable demonstrations and controlled interventions to characterize how individual structural components and their joint specification support whole-concept recovery, and how plausible but unnecessary information can interfere with recovery.
- **A learnable concept-recovery capability.** We show that structured concept recovery can be learned through supervised training, improving recovery while preserving implicit prediction and transferring across composition forms.

2 Background and Related Work

2.1 Problem Formulation

We consider an in-context learning task governed by an underlying concept c, which specifies the input–output relationship that defines the task. We use "concept" as an abstract term for this latent, task-defining object, which may take different forms depending on the task, such as a rule, function, program, or other structured specification. Given a concept c, demonstrations are generated by applying c to a set of inputs,

$$D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{H}, \qquad y^{(i)} = c(x^{(i)}).$$

A query x^q from the same task is likewise associated with $y^q = c(x^q)$. Given the demonstrations and a query, an ICL model can directly produce the query output $\hat{y}^q = \mathrm{LLM}_{\mathrm{Im}}(D, x^q)$. We refer to this as **implicit prediction**: the model produces the query label without being explicitly required to state the underlying concept.

We can instead evaluate whether the model can **recover the underlying concept** itself from the demonstrations, $\hat{c} = \mathrm{LLM}_{\mathrm{Rec}}(D)$. Thus, prediction uses the demonstrations to answer a new query, whereas concept recovery uses the demonstrations to infer the latent object c that defines the task. This distinction allows us to ask our central question: *when implicit prediction succeeds, to what extent does the model recover the underlying concept?*

2.2 Related Work

Prior work has studied how demonstrations convey task information from several perspectives. Some work examines how models infer hidden tasks or task structure from demonstrations (Xie et al., 2021; Hahn & Goyal, 2023; Pan et al., 2023). Other work studies how properties of demonstrations affect successful prediction, including labels, input distributions, format, similarity, ordering, and retrieval (Zhao et al., 2021; Min et al., 2022; Liu et al., 2022; Lu et al., 2022; Rubin et al., 2022). Related work also considers more explicit forms of task knowledge, including natural-language task descriptions, instructions, and rules (Honovich et al., 2023; Zhu et al., 2023). Another line of work studies how task- or function-level information is induced and represented in ICL (Olsson et al., 2022; Akata et al., 2026). These works motivate viewing ICL as more than direct label prediction, but do not directly address the structured recovery of the underlying task-defining concept considered here. A more detailed discussion of related work is provided in Appendix A.

More directly related to concept recovery are works that explicitly ask models to infer an underlying rule or hypothesis from observations. Qiu et al. (2024) study inductive reasoning through iterative hypothesis refinement, where models generate and refine textual hypotheses before applying them to new instances. MIRAGE (Li et al., 2025) explicitly studies the distinction between rule induction and example inference, showing that models can predict unseen examples correctly without necessarily recovering the correct generalized rule. These studies highlight the distinction between successful prediction and recovery of the underlying rule.

However, these lines of work do not directly address two aspects that are central to our setting. First, they typically evaluate the recovered object as a whole, providing limited information about which concept-defining components are correctly recovered, missed, or incorrectly included, particularly when demonstrations contain plausible but irrelevant structure. Second, they do not explicitly control whether the demonstrations uniquely identify the target concept. When multiple concepts are consistent with the same demonstrations, failure to recover c may reflect ambiguity in the evidence rather than a limitation of the model's recovery capability. We therefore ask a more specific evaluation question: *when D uniquely identifies c, which components of the target concept are recovered, missed, or incorrectly included?*

3 Component-Level Concept Recovery

Our framework addresses the prediction–recovery gap through a **structured view of concept recovery**, as illustrated in Figure 2. First, we represent the underlying concept through its constituent components and evaluate recovery at the component level (Sec.3.1), establishing *what* is being recovered and how recovery errors can be localized. Second, building on this representation, we use controlled interventions on concept-defining information and plausible but unnecessary structural information to characterize which combinations of structural information support recovery (Sec.3.2).

Finally, we use supervised learning to explicitly teach the same structured recovery capability while retaining the original implicit-prediction objective (Sec.3.3). These stages move from *measuring* concept recovery, to *localizing* its structural dependencies, to *learning* the recovery capability itself.

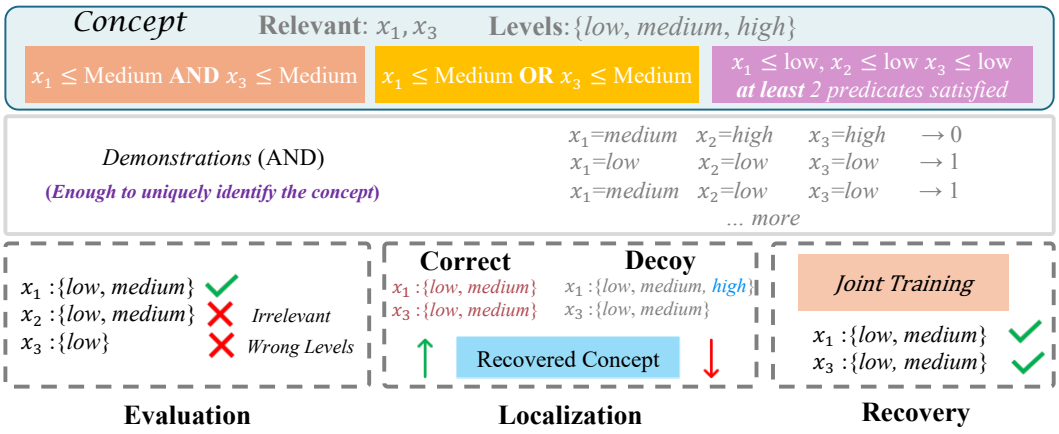

Figure 2: **Overview of our component-level framework for concept recovery.** The top panel illustrates the three composition forms, while the lower panels show the evaluation, localization, and recovery stages. We first decompose recovered concepts into defining components and evaluate at the component level. We then use controlled interventions on structural information to localize their effects on recovery. Finally, we use supervised learning to finetune the model to recover the structured concept while preserving its implicit-prediction capability.

3.1 Structured Concept Representation

From Holistic to Structured Representation. We represent each target concept as a structured specification composed of concept-defining components. This representation makes the underlying concept explicit and provides a common basis for evaluating recovery at both the component and concept levels.

We instantiate our framework in a STAGGER-style controlled concept-learning environment (Schlimmer & Granger Jr, 1986), which provides a suitable setting for our framework because concepts are defined through multiple interacting attributes rather than represented only as an input–output mapping. This structure allows us to study whether a model recovers *which attributes matter*, *what constraints they impose*, and *how these constraints are combined*, while retaining explicit ground truth for each component.

Each concept is generated from K predicate slots and a known compositional form g, of which only a subset may impose effective constraints. Each effective predicate is defined by an input attribute together with a constraint on its possible levels. For an attribute j, we represent its constraint by a set of satisfying levels $L_j \subseteq \mathcal{V}$, where $\mathcal{V} = \{low, medium, high\}$. The compositional form g specifies how these predicates are combined: conjunction (AND) requires all predicates to be satisfied, disjunction (OR) requires at least one, and m-of-K requires at least m predicates to be satisfied. Thus, the concept-specific instantiation can be represented as

$$c_g = \begin{cases} (A, \{L_j\}_{j\in A}), & g \in \{\text{AND}, \text{OR}\}, \\ (A, \{L_j\}_{j\in A}, m), & g = \text{m-of-K}, \end{cases} \tag{1}$$

where $A \subseteq \{1, \ldots, M\}$ denotes the indices of the effective relevant attributes, $\{L_j\}_{j\in A}$ specifies their level constraints, and m specifies the threshold when applicable. We use K to denote the nominal number of predicate slots; some predicate slots may impose no effective constraint, so $|A| \le K$. Although K can provide structural information for recovery, it is not included in c_g because it specifies the nominal predicate capacity rather than an effective constraint on the input–output concept. The composition form g is known and is not itself a recovery target. Concept recovery therefore targets the concept-defining specification c_g.

Component-Level Concept Recovery. The target concept determines the output of each demonstration through its concept-defining components, while each observation is represented over the

full candidate attribute space. Thus, only a subset of the observed attributes belongs to the target concept, whereas the remaining attributes are irrelevant to the output. Across demonstrations, the correspondence between attribute values and outputs provides evidence about which attributes are relevant and what value constraints they impose. Details of concept construction and demonstration generation are provided in Appendix B.

Given the ground-truth concept, we evaluate recovery at the level of its constituent components: the relevant attribute set A, the level constraint L_j for each $j \in A$, and, for m-of-K concepts, the threshold m. This allows us to distinguish components that are correctly recovered from those that are missed or incorrectly specified. This component-level measurement goes beyond whole-concept correctness, allowing us to characterize *what* models recover and *which* parts of the underlying concept remain unresolved.

Identifiability of the Target Concept. To separate recovery errors from ambiguity in the demonstrations, we require the demonstration set D to uniquely identify the target concept within the admissible hypothesis class. Following the version-space formulation, D is concept-identifying when the version space contains no behaviorally distinct hypothesis other than the target concept. We verify this condition at the deployed demonstration budgets by exhaustive enumeration and theoretical analysis; details are provided in Appendix C.

3.2 Localizing Information Dependencies in Concept Recovery

Component-level evaluation localizes recovery errors but does not determine whether explicitly providing the corresponding structural information improves recovery. We therefore use controlled information interventions to isolate how different structural components support concept recovery.

In each intervention, we fix the target concept and its demonstrations and vary only the ground-truth structural information explicitly provided to the recovery model. Let S denote the set of structural information components provided in addition to the demonstrations. In our interventions, S may specify K, A, L, m, or combinations thereof, including a plausible decoy constraint $\tilde{L}$. The recovery model produces $\hat{c}_S^{(i)} = \mathrm{LLM}_{\mathrm{Rec}}\big(D^{(i)}, S^{(i)}\big)$, with $c^{*(i)}$ denotes the corresponding ground-truth target concept for instance i. This formulation allows us to compare recovery under different information states while keeping the underlying task instance and recovery procedure fixed.

Whole-Concept Recovery. We evaluate whether the recovered specification is behaviorally equivalent to the target concept over the full input space:

$$\hat{c}_S^{(i)} \equiv c^{*(i)} \quad \Longleftrightarrow \quad \forall x \in \mathcal{X}_i,\ \hat{c}_S^{(i)}(x) = c^{*(i)}(x), \tag{2}$$

where $\mathcal{X}_i$ denotes the input space of the i-th target concept. We evaluate this equivalence exhaustively when the input space is tractable; otherwise, we use a fixed deterministic subsample of at most 4096 contexts.

The whole-concept recovery rate under structural information S is defined as

$$R(S) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\Big[\hat{c}_S^{(i)} \equiv c^{*(i)}\Big], \tag{3}$$

where N is the number of target concept instances. Thus, $R(S)$ measures whether the recovered specification defines the same input–output concept as the target, complementing component-level metrics that evaluate individual elements of the recovered specification.

Decoy-Level Interference. We next test the robustness of recovery to plausible but unnecessary structural information. For each relevant attribute $j \in A$, we augment its ground-truth level constraint with one additional level $\tilde{\ell}_j \notin L_j$, yielding $\tilde{L}_j = L_j \cup \tilde{\ell}_j$. We compare whole-concept recovery under the correct and augmented structural information, namely $R(A, L)$ and $R(A, L, \tilde{L})$. The target concepts, demonstrations, recovery procedure, and evaluation remain fixed across conditions, so the comparison isolates the effect of adding plausible but unnecessary level information.

3.3 Learning Structured Concept Recovery

The preceding framework defines concept recovery as structured prediction over the concept representation in Eq. (1) and identifies the structural information that can support this recovery. We next

ask whether this recovery capability can be explicitly learned. We therefore supervise the structured concept representation while retaining the original implicit-prediction objective.

Joint Recovery and Prediction Supervision. We treat concept recovery and implicit prediction as complementary objectives. Recovery supervision provides demonstrations and requires the model to explicitly state the underlying concept, including its attributes, level constraints, and, when applicable, threshold m. Implicit-prediction supervision retains the original ICL objective by providing demonstrations and a query and supervising the corresponding output. For each training concept, we construct one recovery example and two balanced prediction examples, one for each label, and jointly optimize them with the standard token-level cross-entropy objective. This objective directly supervises the structured recovery target while preserving the original implicit-prediction objective.

Training and Transfer. We train on m-of-K concepts because this family exercises the full structured specification (A, L, m). We evaluate the trained model on held-out concepts within the training family and on concepts from unseen AND and OR composition forms. This evaluates whether the learned recovery capability transfers beyond the composition form observed during training.

4 Experimental Studies

We first introduce the experimental setup, including the target concepts, evaluated models, and evaluation metrics (Sec.4.1). We then examine concept recovery at the component level and characterize systematic recovery errors (Sec.4.2). Next, we use controlled interventions to test whether constraining individual components or jointly specifying complementary components enables whole-concept recovery, and whether successful recovery is robust to plausible but unnecessary structural information (Sec.4.3). Finally, we evaluate whether structured concept recovery can be learned through supervised training while retaining implicit prediction (Sec.4.4).

4.1 Experimental Setup

Concepts and Data. We instantiate 48 target concepts from the STAGGER-style environment, covering three composition forms (AND, OR, and m-of-K), 2 nominal arities ($K \in \{3, 4\}$), and 8 generator seeds. For each concept, we construct balanced demonstration sets with $H \in \{100, 200\}$ labeled examples and a balanced held-out query set Q containing 32 probes per concept and evaluation condition.

Models. We evaluate four instruction-tuned LLMs: Qwen2.5-72B-Instruct, Llama-3.1-70B-Instruct, Gemma-2-27B-IT, and Mistral-Small-24B-Instruct-2501. All models use publicly released checkpoints and are evaluated under the same prompting protocol with greedy decoding ($T = 0$).

Evaluation Metrics. We evaluate two aspects: *concept application* and *concept recovery*. **Concept Application** measures prediction accuracy on the balanced held-out query set Q. We denote the accuracy of *implicit prediction* by Q_{imp} and that of *explicit prediction*—applying the recovered concept to the same queries—by Q_{exp}. **Concept Recovery** measures whether the recovered concept matches the target concept. We use the whole-concept recovery rate $R(S)$ defined in Eq. (3), and Acc_A, Acc_L, and Acc_m for component-level recovery of relevant attributes, level constraints, and the threshold, respectively. Q_{exp} measures how accurately the recovered concept is applied on held-out queries, whereas $R(S)$ measures whole-concept recovery over the input space and is therefore a stricter measure of whether the underlying concept has been recovered.

Experimental Protocol. All experiments use the same modular prompt templates, with only the relevant prompt components varied across experimental conditions. The prompt templates and inference-time variants are provided in Appendix D.1.

4.2 Component-Level Evaluation of Concept Recovery

We first examine whether accurate implicit prediction is accompanied by faithful explicit concept recovery. We compare the model's implicit predictions with predictions obtained by executing its recovered concept on the same held-out queries. As shown in Table 1, implicit prediction achieves 0.67–0.83 accuracy, whereas the recovered concepts achieve only 0.50–0.63. Thus, models can make substantially more accurate predictions than the concepts they explicitly recover from the same demonstrations.

We next ask what underlies this prediction–recovery gap. Our component-level evaluation shows that the errors are systematic rather than random. Exact recovery of the relevant-attribute set A is uniformly 0.00, while models predict 6.56–7.50 attributes as relevant on average. Since each input contains $M = K + 4 \in \{7, 8\}$ attributes, with $K \in \{3, 4\}$ denoting the nominal number of predicate slots, models tend to include nearly the entire candidate attribute space rather than isolate the attributes that define the target concept. Among correctly identified relevant attributes, level-constraint mismatches are also biased toward *Over* errors, while threshold errors in m-of-K concepts are predominantly *Too High* (Appendix D.2; Appendix D.3). Together, these results indicate a consistent tendency to **over-specify the target concept**: models tend to include additional attributes, satisfying levels, or required predicates.

Table 1: Concept application and recovery under demonstrations alone. Q_{imp} and Q_{exp} denote prediction accuracy on the held-out query set Q using implicit prediction and the recovered concept, respectively. Acc_A, $\text{Acc}_{L|A}$, and Acc_m measure recovery of relevant attributes, level constraints conditional on correctly identified attributes, and the composition threshold, respectively; $|\hat{A}|$ reports the number of predicted relevant attributes.

		Concept Application		Concept Recovery			
Form	Model	Q_{imp}	Q_{exp}	Acc_A	$\lvert\hat{A}\rvert$	$\text{Acc}_{L\lvert A}$	Acc_m
AND	Qwen2.5-72B	0.83	0.63	0.00	7.16	0.70	–
	Llama-3.1-70B	0.73	0.56	0.00	7.31	0.71	–
	Gemma-2-27B	0.72	0.58	0.00	7.47	0.38	–
	Mistral-24B	0.77	0.52	0.00	7.47	0.62	–
OR	Qwen2.5-72B	0.80	0.50	0.00	6.91	0.05	–
	Llama-3.1-70B	0.74	0.51	0.00	7.50	0.15	–
	Gemma-2-27B	0.77	0.50	0.00	7.06	0.05	–
	Mistral-24B	0.69	0.51	0.00	7.47	0.24	–
m-of-K	Qwen2.5-72B	0.73	0.53	0.00	6.56	0.39	0.00
	Llama-3.1-70B	0.71	0.51	0.00	7.06	0.29	0.03
	Gemma-2-27B	0.67	0.51	0.00	6.97	0.21	0.03
	Mistral-24B	0.68	0.51	0.00	7.47	0.32	0.03

A paired comparison on the same held-out queries further tests this dissociation, avoiding differences due to independently sampled test sets. Across models, the recovered rule overturns 0.32–0.39 of predictions that are correct under implicit prediction, while the reverse pattern is consistently less frequent ($c < b$). McNemar's exact test rejects symmetry between the discordant prediction counts for all models ($p < 10^{-4}$). This paired analysis therefore shows that the prediction–recovery gap is not driven simply by different query samples: the recovered rules systematically overturn a substantial fraction of predictions that the models otherwise get correct.

Table 2: Paired comparison of implicit prediction and predictions from the model's stated rule. b counts queries correct under implicit prediction but incorrect under the stated rule, while c counts the reverse pattern. *Cond. Loss* measures the fraction of implicitly correct queries overturned by the stated rule. McNemar's exact test compares the discordant counts. ↓ denotes lower-is-better.

Model	b	c	Cond. Loss ↓	McNemar p
Qwen2.5-72B	84	18	0.33	2.3×10^{-11}
Llama-3.1-70B	76	33	0.32	4.7×10^{-5}
Gemma-2-27B	86	34	0.39	2.3×10^{-6}
Mistral-Small-24B	90	22	0.39	6.0×10^{-11}

We then rule out general difficulty in applying an explicit concept: directly providing the target concept yields substantially higher held-out accuracy (Appendix D.4). We also examine whether the component-level pattern extends beyond the controlled environment. On MONK-1, recovered concepts tend to include more attributes than the ground-truth relevant-attribute set, while analogous attribute-count patterns are observed on Mushroom (Appendix D.5).

Takeaway. Accurate implicit prediction does not imply faithful explicit concept recovery. **Component-level analysis reveals a systematic tendency to over-specify the target concept**, rather than merely failing to recover it at random. This pattern also persists beyond the controlled STAGGER-style environment.

Table 3: Concept recovery under progressively revealed structural information. Each entry reports whole-concept recovery $R(S)$ after providing the indicated ground-truth information S in addition to the demonstrations. $\varnothing$ denotes demonstrations alone; K, A, L, and m denote the corresponding structural components.

Form	Model	$R(\varnothing)$	$R(K)$	$R(A)$	$R(L)$	$R(m)$	$R(A,L)$	$R(A,L,m)$
AND	Qwen2.5-72B	0.06	0.09	0.81	0.00	–	1.00	–
	Llama-3.1-70B	0.00	0.03	0.84	0.03	–	1.00	–
	Gemma-2-27B	0.00	0.00	0.19	0.06	–	0.97	–
	Mistral-24B	0.00	0.00	0.66	0.06	–	1.00	–
OR	Qwen2.5-72B	0.00	0.00	0.00	0.03	–	0.94	–
	Llama-3.1-70B	0.00	0.00	0.00	0.09	–	0.88	–
	Gemma-2-27B	0.00	0.00	0.00	0.00	–	0.81	–
	Mistral-24B	0.00	0.00	0.00	0.00	–	1.00	–
m-of-K	Qwen2.5-72B	0.00	0.00	0.00	0.00	0.00	0.47	0.94
	Llama-3.1-70B	0.00	0.00	0.00	0.00	0.00	0.94	1.00
	Gemma-2-27B	0.00	0.00	0.00	0.00	0.00	0.00	0.81
	Mistral-24B	0.00	0.00	0.00	0.00	0.00	1.00	1.00

4.3 Structural Information Shapes Concept Recovery

The component-level analysis reveals systematic over-specification in the recovered concepts. We therefore next ask whether providing the correct value of an individual structural component is sufficient to correct the corresponding error and recover the full concept. To test this, we keep the target concepts and demonstrations fixed while providing selected ground-truth structural components. Because the main error in A is over-inclusion, we also provide K, the nominal number of predicate slots, to test whether constraining the size of the relevant attribute is sufficient for recovery.

Table 3 compares recovery when individual structural components are provided separately or jointly. Providing K, A, L, or m individually is generally insufficient for whole-concept recovery. The main exception is AND, where providing A alone yields 0.19–0.84 recovery. However, recovery remains near zero for OR and m-of-K. These results show that **identifying a component-level recovery error does not by itself provide sufficient information to recover the whole concept**. We next jointly provide (A, L). This substantially improves recovery across the three concept families. Thus, **jointly specifying complementary defining components enables substantially more complete whole-concept recovery**, while m provides the additional information required by m-of-K.

Having established that (A, L) provides a condition under which recovery can be highly successful, we next test whether this recovery is robust to plausible but unnecessary structural information. We deliberately contaminate the successful specification by augmenting each ground-truth level constraint with one plausible but unnecessary level, while keeping A fixed. This intervention is motivated by the systematic over-specification of level constraints observed in Sec. 4.2. If the model can distinguish concept-defining information from unnecessary but plausible structure, the correct joint specification should remain sufficient despite this single decoy level.

Table 4: Sensitivity of concept recovery to decoy-level information. We compare $R(A, L)$ with recovery after augmenting each level constraint with one plausible but unnecessary level, denoted by $\tilde{L}$.

Form	Model	$R(A,L)$	$R(A,\tilde{L})$
AND	Qwen2.5-72B	1.00	0.97
	Llama-3.1-70B	1.00	0.59
	Gemma-2-27B	0.97	0.31
	Mistral-24B	1.00	0.66
OR	Qwen2.5-72B	0.88	0.19
	Llama-3.1-70B	1.00	0.38
	Gemma-2-27B	0.81	0.00
	Mistral-24B	1.00	0.16
m-of-K	Qwen2.5-72B	0.47	0.00
	Llama-3.1-70B	0.94	0.16
	Gemma-2-27B	0.00	0.00
	Mistral-24B	1.00	0.03

As shown in Table 4, this single decoy-level intervention reduces whole-concept recovery in 11 of 12 model–form conditions. The degradation is especially pronounced for OR and m-of-K. These results show that **successful recovery remains sensitive to plausible but unnecessary structural information**, even when the correct joint specification is already provided.

Takeaway. Individual component constraints are generally insufficient for whole-concept recovery, whereas **jointly specifying complementary defining components enables substantially more**

complete recovery. However, this successful recovery can be disrupted by a single plausible but unnecessary structural element.

4.4 Learning Structured Concept Recovery

Before turning to supervised training, we first ask whether the prediction–recovery gap can be reduced through inference-time reasoning. As shown in Appendix D.6, neither chain-of-thought reasoning (Wei et al., 2022) nor iterative self-refinement (Madaan et al., 2023) consistently improves structured recovery, even when the identities of the relevant attributes are provided. We therefore ask whether this structured recovery capability can instead be learned directly through supervision. For supervised training, we use Qwen2.5-7B-Instruct with LoRA, intentionally using a smaller model than the 72B-scale models used for evaluation. Training details are provided in Appendix D.7.

Table 5: Learning structured concept recovery on m-of-K concepts. Q_{imp} denotes implicit prediction accuracy on held-out queries, while $R(\varnothing)$ measures whole-concept recovery from demonstrations alone. *Recovery only* uses only structured recovery supervision, whereas *Joint* combines recovery and implicit-prediction supervision.

Form	Training	Q_{imp}	$R(\varnothing)$
m-of-K	No training	0.64	0.00
	Recovery only	0.50	0.96
	Joint	0.98	0.97

We train on m-of-K concepts, where recovery requires the full structured specification (A, L, m). We compare three conditions: *No training*, *Recovery only*, which supervises only the structured concept representation, and *Joint*, which combines recovery supervision with the original implicit-prediction objective. For each training concept, we construct one recovery example and two balanced prediction examples, one for each label, and jointly optimize the corresponding token-level cross-entropy objectives.

As shown in Table 5, recovery-only training increases $R(\varnothing)$ from 0.00 to 0.96, but reduces Q_{imp} from 0.64 to chance level. In contrast, joint training achieves both high recovery and high implicit-prediction accuracy, reaching $R(\varnothing) = 0.97$ and $Q_{\text{imp}} = 0.98$. Thus, structured recovery can be learned without sacrificing implicit prediction when the two objectives are jointly supervised.

We further evaluate the performance on composition forms not observed during training. Table 6 shows that recovery-only training retains high recovery on unseen AND and OR concepts, while joint training additionally preserves high implicit-prediction accuracy.

Table 6: Cross-family transfer after training on m-of-K concepts. Q_{imp} denotes implicit prediction accuracy on held-out queries, while $R(\varnothing)$ measures whole-concept recovery from demonstrations alone.

Form	Training	Q_{imp}	$R(\varnothing)$
AND	No training	0.72	0.00
	Recovery only	0.50	0.95
	Joint	0.97	0.94
OR	No training	0.66	0.00
	Recovery only	0.51	0.96
	Joint	0.99	0.98

Takeaway. Supervised recovery training substantially improves whole-concept recovery, but recovery-only supervision disrupts implicit prediction. **Joint recovery and prediction supervision learns structured recovery while preserving prediction capability**, and the learned recovery capability transfers to unseen composition forms.

5 Conclusions

We introduced a component-level framework for concept recovery in in-context learning, moving beyond holistic evaluation by decomposing concepts into defining components and controlling the structural information available for recovery. Our findings show that successful prediction can coexist with poor concept recovery, and that component-level analysis provides a more diagnostic view of what models recover from demonstrations. Although our controlled concept-learning environment enables precise measurement and intervention, it leaves open whether the same patterns hold for more complex and open-ended concepts. Extending component-level recovery to natural-language rules, programs, and other structured task representations, and relating explicit recovery to the internal computations used for prediction, are promising directions for future work.

AI USE STATEMENT

In this work, we used generative AI tools for improving the language and grammar of the manuscript. Additionally, we used generative AI tools for conducting our experiments, as LLMs are the primary subjects of our work. The reported experimental results are the direct outputs of interactions with these models, using structured prompts designed and refined solely by the authors. All research questions, methodologies, and experimental designs were developed by the authors. We have reviewed all AI-assisted work before submission, and we take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, et al. Many-shot in-context learning. *Advances in Neural Information Processing Systems*, 37:76930–76966, 2024.

Elif Akata, Konstantinos Voudouris, Vincent Fortuin, and Eric Schulz. In-context function learning in large language models. *arXiv preprint arXiv:2602.11863*, 2026.

Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, et al. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 4005–4019, 2023.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 1107–1128, 2024.

Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, et al. Competitive programming with large reasoning models. *arXiv preprint arXiv:2502.06807*, 2025.

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. *Advances in neural information processing systems*, 35:30583–30598, 2022.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, et al. Accelerating scientific discovery with co-scientist. *Nature*, pp. 1–3, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Michael Hahn and Navin Goyal. A theory of emergent in-context learning as implicit structure induction. *arXiv preprint arXiv:2303.07971*, 2023.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333, 2023.

Guan Zhe Hong, Bhavya Vasudeva, Vatsal Sharan, Cyrus Rashtchian, Prabhakar Raghavan, and Rina Panigrahy. Latent concept disentanglement in transformer-based language models. In *International Conference on Learning Representations*, volume 2026, pp. 27223–27260, 2026.

Or Honovich, Uri Shaham, Samuel R Bowman, and Omer Levy. Instruction induction: From few examples to natural language task descriptions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1935–1952, 2023.

Robert Huben, Hoagy Cunningham, Logan Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations*, volume 2024, pp. 7827–7845, 2024.

Jiachun Li, Pengfei Cao, Zhuoran Jin, Yubo Chen, Kang Liu, and Jun Zhao. Mirage: Evaluating and explaining inductive reasoning process in language models. In *International Conference on Learning Representations*, volume 2025, pp. 73944–73969, 2025.

Jiaoda Li, Yifan Hou, Mrinmaya Sachan, and Ryan Cotterell. What do language models learn in context? the structured task hypothesis. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12365–12379, 2024.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd workshop on knowledge extraction and integration for deep learning architectures*, pp. 100–114, 2022.

Sheng Liu, Haotian Ye, Lei Xing, and James Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*, 2023.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, 2022.

Haoyan Luo and Lucia Specia. From understanding to utilization: A survey on explainability for large language models. *arXiv preprint arXiv:2401.12874*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 11048–11064, 2022.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8298–8319, 2023.

Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, et al. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. In *International Conference on Learning Representations*, volume 2024, pp. 1452–1481, 2024.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2655–2671, 2022.

Jeffrey C Schlimmer and Richard H Granger Jr. Incremental learning from noisy data. *Machine learning*, 1(3):317–354, 1986.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Chandan Singh, Jeevana Priya Inala, Michel Galley, Rich Caruana, and Jianfeng Gao. Rethinking interpretability in the era of large language models. *arXiv preprint arXiv:2402.01761*, 2024.

Eric Todd, Millicent Li, Arnab Sen Sharma, Aaron Mueller, Byron Wallace, and David Bau. Function vectors in large language models. In *International conference on learning representations*, volume 2024, pp. 17282–17333, 2024.

Xinyi Wang, Wanrong Zhu, Michael Saxon, Mark Steyvers, and William Yang Wang. Large language models are latent variable models: Explaining and finding good demonstrations for incontext learning. *Advances in Neural Information Processing Systems*, 36:15614–15638, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.

Qinyuan Ye, Robin Jia, and Xiang Ren. Function induction and task generalization: An interpretability study with off-by-one addition. In *International Conference on Learning Representations*, volume 2026, pp. 88017–88054, 2026.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pp. 12697–12706. Pmlr, 2021.

Bowen Zheng, Ming Ma, Zhongqiao Lin, and Tianming Yang. Label words as local task vectors in in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 6319–6332, 2026.

Zhaocheng Zhu, Yuan Xue, Xinyun Chen, Denny Zhou, Jian Tang, Dale Schuurmans, and Hanjun Dai. Large language models can learn rules. *arXiv preprint arXiv:2310.07064*, 2023.

A Detailed Comparison with Related Work

This appendix provides a more detailed discussion of the related work summarized in Sec. 2, covering task and structure induction, behavioral studies of ICL, explicit task, rule, and function induction, task and function representations, and explicit concept recovery.

A.1 Task and Structure Induction

A line of work views demonstrations as evidence about an underlying task or latent structure. Xie et al. (2021) interpret ICL as implicit Bayesian inference over a latent task concept shared by the demonstrations, while Hahn & Goyal (2023) study ICL as implicit structure induction and characterize how compositional structure can be inferred from examples. Wang et al. (2023) similarly model LLMs as latent-variable models and find empirical evidence that demonstrations induce latent task information. Pan et al. (2023) further distinguish task recognition from task learning, showing that models can recognize a task from demonstrations, even without ground-truth labels, while separately learning input–label mappings not available from pretraining.

These studies motivate viewing demonstrations as evidence about a latent task rather than merely as examples for direct prediction. Our work adopts this perspective but makes the recovery target explicit: instead of asking whether demonstrations induce latent task information that supports prediction, we ask which components of a predefined task-defining concept are recovered from demonstrations. This enables us to evaluate recovery at the component level and to verify whether the demonstrations uniquely identify the target concept.

A.2 Behavioral Studies of ICL

A large body of work studies ICL from a behavioral perspective, examining which properties of demonstrations affect prediction. Min et al. (2022) show that labels, input distributions, and sequence format contribute differently to ICL performance, while Zhao et al. (2021) demonstrate that label distributions can substantially bias model predictions. Other work studies how the choice and ordering of demonstrations affect performance, including semantic similarity-based example selection, prompt ordering, and learned retrieval of useful examples (Liu et al., 2022; Lu et al., 2022; Rubin et al., 2022). Recent surveys organize these findings across different types of demonstration effects (Dong et al., 2024).

These studies show that the information contained in demonstrations can substantially affect prediction. Our work instead focuses on a different evaluation target: rather than measuring how demonstration properties affect query accuracy, we control the structural information provided for recovery to determine which parts of the underlying concept are recovered.

A.3 Explicit Task, Rule, and Function Induction

Another line of work explicitly studies whether models can infer task-level descriptions, rules, or functions from demonstrations. Instruction Induction (Honovich et al., 2023) prompts language models to generate natural-language instructions that describe the task underlying a set of input–output examples, and evaluates the induced instructions by their execution on new inputs. Hypotheses-to-Theories (HtT) (Zhu et al., 2023) similarly induces and verifies candidate rules from examples, building a rule library that can be used for subsequent reasoning. These works show that models can make task-level structure explicit rather than relying only on implicit prediction.

Related work studies the same question in more controlled function-learning settings. Garg et al. (2022) show that Transformers trained on synthetic data can learn unseen functions from in-context examples, including linear functions, sparse linear functions, neural networks, and decision trees. Li et al. (2024) further examine competing explanations of ICL and provide evidence that models can compose previously learned task components to implement novel tasks in context. More recent work studies function learning and function induction under controlled function classes and through internal mechanisms (Akata et al., 2026; Ye et al., 2026).

Our setting shares this focus on an underlying structured object, but differs in the evaluation target: rather than evaluating induced descriptions through downstream execution, generalization, or task performance, we evaluate recovery against a target concept and decompose that concept into iden-

tifiable components. This allows us to distinguish components that are correctly recovered, missed, or incorrectly included.

A.4 Task and Function Representations in ICL

A complementary line of work studies how task- or function-level information induced by demonstrations is represented and used inside language models. Induction heads have been identified as an important mechanism for in-context pattern completion (Olsson et al., 2022), while subsequent work shows that demonstrations can give rise to compact task- or function-level representations (Hendel et al., 2023; Liu et al., 2023; Todd et al., 2024). More recent studies examine how structured task information is represented, including function induction, latent concept representations, and local task-specific representations (Hong et al., 2026; Zheng et al., 2026).

These studies provide evidence that task- or function-level information can be encoded in model representations and used to support prediction. However, representation-level evidence does not directly measure behavioral recovery of a predefined concept. In particular, the presence or decodability of task information in hidden states does not reveal which components of the target concept are recovered, whether irrelevant structure is also incorporated, or whether the available demonstrations uniquely identify the target. Our evaluation instead treats concept recovery as an explicit behavioral target and measures it at the level of individual concept-defining components.

A.5 Explicit Concept Recovery and Prediction–Recovery Gaps

The most closely related work explicitly evaluates whether models recover the underlying rule or hypothesis from observed examples. Qiu et al. (2024) study inductive reasoning through iterative hypothesis refinement, where language models propose, select, and refine textual rules before applying them to new instances. By evaluating the intermediate hypotheses and their subsequent application, they identify a gap between rule induction and rule application. MIRAGE (Li et al., 2025) further separates rule induction from example inference in a controlled synthetic setting and shows that models can often infer unseen examples correctly without recovering the corresponding generalized rule. These results establish that successful prediction does not necessarily imply correct recovery of the underlying rule.

Existing evaluations primarily treat the recovered rule or hypothesis as a holistic object, making it difficult to determine which concept-defining components are correctly recovered, missed, or incorrectly included, particularly when observations contain plausible but irrelevant structure. Moreover, they do not explicitly verify whether the demonstrations uniquely identify the target concept, leaving recovery failure potentially confounded with ambiguity in the evidence. We instead construct demonstrations that uniquely identify a target concept and evaluate recovery at the level of explicit structural components. This formulation further enables controlled interventions on defining and irrelevant structural information, as well as supervised learning of recovery.

B Controlled Concept and Demonstration Generation

This section specifies how we instantiate the controlled concepts used in our experiments and generate the corresponding demonstrations and held-out probes. We first describe the STAGGER-style concept environment and the construction of individual target concepts in Sec. B.1. We then describe how demonstrations and held-out probes are sampled from each instantiated concept in Sec. B.2.

B.1 STAGGER-style Concept Environment

We construct our controlled tasks using a STAGGER-style concept-learning environment (Schlimmer & Granger Jr, 1986), where each task is generated from a known concept with a finite and interpretable structure. This provides explicit ground truth for each concept-defining component.

Concept representation. Following Sec. 3.1, we represent each target concept by its task-specific components:

$$c_g = \begin{cases} (A, \{L_j\}_{j \in A}), & g \in \{\text{AND}, \text{OR}\}, \\ (A, \{L_j\}_{j \in A}, m), & g = \text{m-of-K}, \end{cases}$$

where g denotes the known composition form, A is the effective relevant-attribute set, L_j specifies the satisfying levels for attribute j, and m is the composition threshold when applicable.

Attribute information. To provide a fixed set of distractor attributes, each input contains $M = K+4$ ordered attributes, where the nominal arity is $K \in \{3, 4\}$ and thus $M \in \{7, 8\}$. The generator selects K attributes as predicate slots; the remaining four attributes are never selected as predicate slots. Each selected attribute is assigned one predicate, but some predicates may impose no effective constraint. We therefore define A as the set of indices of selected attributes whose predicates affect the realized input–output mapping, so that $|A| \leq K$.

Level constraints. Each attribute takes one of three ordered levels,

$$\mathcal{V} = \{\textit{low}, \textit{medium}, \textit{high}\}.$$

Identifying these levels with $\{0, 1, 2\}$ gives the input space $\mathcal{X} = \{0, 1, 2\}^M$. For each selected attribute, the generator samples an ordinal threshold predicate with an operator in $\{\leq, \geq\}$ and a threshold index in $\{0, 1\}$. The resulting predicate space is

$$\mathcal{Q} = \{(\leq, 0),\ (\leq, 1),\ (\geq, 0),\ (\geq, 1)\}.$$

These predicates correspond to the satisfying level sets

$$(\leq, 0) \mapsto \{\textit{low}\}, \quad (\leq, 1) \mapsto \{\textit{low}, \textit{medium}\}, \quad (\geq, 0) \mapsto \mathcal{V}, \quad (\geq, 1) \mapsto \{\textit{medium}, \textit{high}\}.$$

Thus, A specifies *which dimensions* participate in the realized concept, while L_j specifies *which levels* of each relevant dimension satisfy its predicate.

Concept composition. The K predicates are combined using one of three known composition forms, $g \in \{\text{AND}, \text{OR}, \text{m-of-K}\}$. Under AND, an input is positive if all predicates are satisfied; under OR, it is positive if at least one predicate is satisfied. Under m-of-K, an input is positive if at least m predicates are satisfied, where $m = \left\lceil \frac{K+1}{2} \right\rceil$. Here, m is the *composition threshold*, which determines how many satisfied predicates are required for a positive output and is distinct from the threshold index used within an individual predicate.

Non-effective predicates. The predicate $(\geq, 0)$ corresponds to $x_j \geq \textit{low}$ and is satisfied by every level in $\mathcal{V}$. It therefore imposes no effective constraint on the input. Under OR, such a predicate makes the concept always positive, while under m-of-K, it reduces the effective composition to an $(m-1)$-of-$(K-1)$ rule. We therefore reject concepts containing this predicate under OR and m-of-K to preserve the intended nominal composition form and arity. Under AND, a non-effective conjunct does not change the resulting concept and is therefore allowed. Consequently, the nominal arity K need not equal the effective number of relevant attributes $|A|$.

Example. Consider $K = 3$, giving $M = 7$ attributes. Suppose the generator selects x_1, x_3, x_5 and assigns

$$x_1 \geq \textit{medium}, \qquad x_3 \geq \textit{low}, \qquad x_5 \leq \textit{medium}.$$

Under AND, the induced input–output mapping is

$$c(x) = \mathbb{I}[(x_1 \geq \textit{medium}) \wedge (x_3 \geq \textit{low}) \wedge (x_5 \leq \textit{medium})]\,.$$

Here, the predicate on x_3 is non-effective and therefore does not affect the realized concept. The effective relevant-attribute set is consequently $A = \{1, 5\}$, even though the nominal arity is $K = 3$. The corresponding level constraints are

$$L_1 = \{\textit{medium}, \textit{high}\}, \qquad L_5 = \{\textit{low}, \textit{medium}\}.$$

Realized concepts. Across 8 generator seeds, 2 nominal arities ($K \in \{3, 4\}$), and 3 composition forms, we instantiate 48 target concepts in total, or 16 concepts for each composition form. During concept recovery, the model is not given K; accordingly, the admissible recovery hypothesis class allows different cardinalities of the relevant-attribute set. We characterize this hypothesis class and its identifiability in Appendix C.

B.2 Generating Demonstrations and Held-out Probes

Given an instantiated concept specification c_g, we sample inputs and assign labels using the induced deterministic mapping f_{c_g}:

$$y = f_{c_g}(x), \qquad x \in \mathcal{X}.$$

Demonstration generation. For each target concept, we construct demonstration sets with $H \in \{100, 200\}$ examples. Each set is balanced between positive and negative examples, containing $H/2$ examples from each class, and the resulting examples are randomly shuffled to form

$$D_H = \{(x^{(i)}, y^{(i)})\}_{i=1}^{H}.$$

The same demonstration set is used as the evidence available to the model for both implicit prediction and concept recovery. Because the demonstrations are balanced, chance-level prediction accuracy is 0.50.

Held-out probes. For each concept and evaluation condition, we construct a held-out query set Q containing 32 probes, with 16 positive and 16 negative examples. The probes are generated from the same underlying concept c_g but are kept separate from the demonstrations. We sample probes without replacement from attribute configurations that do not occur in the demonstration set.

Consistency across intervention conditions. For comparisons across information-intervention conditions, we keep both the underlying concept and the demonstration set fixed, and vary only the information provided to the model before concept recovery. This ensures that differences in recovery can be attributed to the manipulated structural information rather than to differences in the sampled demonstrations.

C Demonstration Identifiability and Sufficiency

Before interpreting concept-recovery failures, we first determine whether the demonstrations uniquely identify the target concept. We formalize this question using the version space induced by the admissible hypothesis class. For a realized demonstration stream, we define the smallest prefix that leaves a single behaviorally distinct hypothesis as the *concept-identification threshold*.

C.1 Diagnostic Setup

Admissible hypothesis class. Following the concept construction in Appendix B.1, let $\mathcal{H}_{M,K',g}$ denote the set of valid concepts over M attributes with nominal arity K' and known composition form g. Because g is specified to the model, we fix it to the target composition form. In contrast, the target nominal arity K is not provided during concept recovery, so alternative hypotheses may have different nominal arities. We therefore define the admissible hypothesis class as

$$\mathcal{H}_{\cup} = \bigcup_{K'=1}^{M} \mathcal{H}_{M,K',g}.$$

This class contains all valid concepts that could be recovered under the information available to the model.

Behavioral equivalence. Different specifications in $\mathcal{H}_{\cup}$ may nevertheless induce the same input–output mapping. Since such specifications are indistinguishable from demonstrations, we identify hypotheses up to behavioral equivalence:

$$c \equiv c' \quad \Longleftrightarrow \quad c(x) = c'(x) \ \ \forall x \in \mathcal{X}.$$

Let $\overline{\mathcal{H}}_{\cup}$ denote the resulting set of behaviorally distinct equivalence classes.

Version space. Let $D_h = \{(x^{(i)}, y^{(i)})\}_{i=1}^{h}$ denote the first h demonstrations in a realized demonstration stream. The corresponding version space is the set of behaviorally distinct hypotheses consistent with all observed demonstrations:

$$\overline{\mathcal{H}}_{\cup}(D_h) = \left\{ [c] \in \overline{\mathcal{H}}_{\cup} \ : \ c(x^{(i)}) = y^{(i)}, \ \forall i \le h \right\}.$$

A demonstration prefix is *concept-identifying* if $\left|\overline{\mathcal{H}}_{\cup}(D_h)\right| = 1$.

Stream–specific identification threshold. For a target concept c^* and a realized demonstration stream D, we define the concept-identification threshold as

$$H^*_{\text{concept}}(c^*; D) = \min \left\{ h \ : \left|\overline{\mathcal{H}}_{\cup}(D_h)\right| = 1 \right\}. \tag{4}$$

This threshold measures how many demonstrations are required for this particular stream to uniquely identify the target concept within the admissible hypothesis class. It is therefore a property of the realized demonstration stream, rather than an intrinsic property of the target concept alone.

C.2 General Sufficient-Budget Characterization

The version-space formulation also provides a general sufficient condition for concept identification. Although the exact identification threshold can be computed in our finite controlled environment, the following characterization gives a model-independent upper bound for finite hypothesis classes. In particular, it relates the required demonstration budget to the number of behaviorally distinct alternatives and their separation from the target concept.

Behavioral separation. For a target concept c^* and a behaviorally distinct competitor c, define the conditional disagreement rate for output $y \in \{0, 1\}$ as

$$\delta_y(c, c^*) = \Pr_{x \sim \mathcal{X}} \left[c(x) \neq c^*(x) \mid c^*(x) = y \right].$$

We then define their balanced behavioral separation as

$$\delta_{\mathrm{bal}}(c, c^*) = \frac{\delta_0(c, c^*) + \delta_1(c, c^*)}{2}.$$

This quantity measures the average fraction of positive and negative contexts on which the competitor disagrees with the target. The balanced form matches the demonstration construction used in our experiments, which contains equal numbers of positive and negative examples.

Let

$$\delta_{\min}(c^*) = \min_{\substack{[c] \in \overline{\mathcal{H}}_\cup \\ [c] \neq [c^*]}} \delta_{\mathrm{bal}}(c, c^*)$$

denote the minimum balanced behavioral separation between the target and any behaviorally distinct competitor.

Proposition 1 (Sufficient demonstration budget)**.** *Let $\overline{\mathcal{H}}_\cup$ be a finite set of behaviorally distinct hypotheses containing the target concept c^*. Suppose that demonstrations are sampled independently within each output class, with $H/2$ positive and $H/2$ negative examples, and that examples within each output class are sampled uniformly from the corresponding input subset. Then*

$$\Pr\left[\left| \overline{\mathcal{H}}_\cup(D_H) \right| > 1 \right] \leq \left(|\overline{\mathcal{H}}_\cup| - 1 \right) (1 - \delta_{\min})^H. \tag{5}$$

Consequently, for any $\eta \in (0, 1)$, it is sufficient to use

$$H_{\mathrm{suff}}(\eta) = \left\lceil \frac{\log\left(|\overline{\mathcal{H}}_\cup| - 1 \right) + \log(1/\eta)}{-\log(1 - \delta_{\min})} \right\rceil$$

demonstrations to identify the target concept with probability at least $1 - \eta$.

Proof. For a fixed competitor c, the probability that it agrees with the target on all $H/2$ positive demonstrations and all $H/2$ negative demonstrations is

$$(1 - \delta_1(c, c^*))^{H/2} (1 - \delta_0(c, c^*))^{H/2}.$$

By the arithmetic–geometric mean inequality,

$$\sqrt{(1 - \delta_1(c, c^*)) (1 - \delta_0(c, c^*))} \leq 1 - \delta_{\mathrm{bal}}(c, c^*),$$

so the competitor-survival probability is at most

$$(1 - \delta_{\mathrm{bal}}(c, c^*))^H \leq (1 - \delta_{\min})^H.$$

Applying a union bound over the $|\overline{\mathcal{H}}_\cup| - 1$ behaviorally distinct competitors gives Eq. (5). Requiring the resulting upper bound to be at most η and solving for H yields the stated sufficient budget. □

The bound is conservative because it protects against the simultaneous survival of every behaviorally distinct competitor, even though different competitors may be eliminated by overlapping subsets of demonstrations. It therefore provides a general sufficient budget rather than an estimate of the exact identification threshold for a realized demonstration stream.

C.3 Exact Concept Identification

In our controlled environment, the behaviorally distinct hypothesis space $\overline{\mathcal{H}}_{\cup}$ is finite and can be exhaustively enumerated. For each target concept and realized demonstration stream, we evaluate the version space after each demonstration prefix and compute the exact identification threshold H^*_{concept} from Eq. (4). This provides a stream–specific measure of the number of demonstrations required to uniquely identify the target concept, without relying on the conservative sufficient-budget bound.

Table 7 summarizes the resulting thresholds and the size of the remaining version space at several demonstration checkpoints. At $H = 100$, 47 of the 48 realized concept–stream instances are uniquely identified, corresponding to an identification rate of 97.9%; at $H = 200$, all 48 realized concept–stream instances are identified. The single unresolved stream at $H = 100$ is an OR concept with $K = 4$, for which exactly one behaviorally distinct competitor remains. Under the fixed-arity analysis, all 96 concept–budget pairs are identified.

The exact enumeration confirms that the deployed demonstration budgets are sufficient for concept identification in all but one realized stream at $H = 100$, and for all realized streams at $H = 200$. At the same time, the variation in identification thresholds across composition forms and demonstration streams indicates that the amount of evidence required for identification is not uniform across tasks.

Table 7: Exact concept-identification thresholds and version-space contraction over $\overline{\mathcal{H}}_{\cup}$. The single stream that remains unidentified at $H = 100$ (OR, $K = 4$) is excluded from the threshold statistics. For each form–arity pair, *med.* and *p90* report the ranges of the corresponding statistics across the independently realized demonstration streams under $H = 100$ and $H = 200$, while *max* reports the largest observed identification threshold among the identified streams. The last three columns report the median number of additional behaviorally distinct rival hypotheses remaining after 5, 12, and 30 demonstrations.

form	K	med.	p90	max	$h = 5$	$h = 12$	$h = 30$
AND	3	14–16	24–29	43	1,013	24	0
AND	4	15–17	26–30	31	2,091	88	0
M-OF-K	3	25	29–33	37	9,881	251	0
M-OF-K	4	26–32	39–46	56	41,537	660	4
OR	3	16–24	26–35	48	82	2	0
OR	4	31–52	60–72	76	95	3	1

C.4 Query-Level Demonstration Sufficiency

Full concept identification is stronger than what is required to determine the label of an individual query. A version space may contain multiple behaviorally distinct concepts while all remaining concepts agree on the label of a particular input. We therefore define the number of labels supported by the version space for a query x_q as

$$N_q(D_h, x_q) = \left|\left\{c(x_q) : [c] \in \overline{\mathcal{H}}_{\cup}(D_h)\right\}\right|.$$

The query is identified when $N_q(D_h, x_q) = 1$, with exact query-level threshold

$$H^*_{\text{query}}(x_q; D) = \min\left\{h : N_q(D_h, x_q) = 1\right\}.$$

We compute this quantity for the same held-out probes used in ICL evaluation. Each concept–stream instance contains 32 probes, yielding 1,536 probes at each demonstration budget. At $H = 100$, 99.9% of probes are uniquely identified, with only one unresolved probe; at $H = 200$, all probes are identified. Across the twelve form–arity–budget cells, the median H^*_{query} ranges from 7 to 21, compared with 14–52 across the corresponding form–arity cells for concept-level identification.

Restricting the analysis to the 95 concept–stream instances that are identified within the corresponding demonstration budget, 82.4% of evaluation probes become identifiable strictly before the full concept is identified, while the remaining 17.6% become identifiable only when the final behaviorally distinct rival is eliminated. Thus, query-level sufficiency is typically reached earlier than full concept identification, although some queries remain tied to the final concept-level ambiguity.

Over the full input domain, concept-level identification is equivalent to identifying the labels of every possible input:

$$H^*_{\text{concept}} = \sup_{x \in \mathcal{X}} H^*_{\text{query}}(x).$$

The equality follows because a version space is a singleton if and only if all hypotheses in it agree on every $x \in \mathcal{X}$, while any pair of behaviorally distinct hypotheses disagrees on at least one such input.

C.5 Information Structure and Sampling Effects

The exact identification threshold varies across realized demonstration streams even for the same target concept. To distinguish intrinsic structure from sampling effects, we generate additional demonstration streams for each target concept and decompose the variation in H^*_{concept}. Across these streams, between-concept variation accounts for only 4–28% of the variance, with the remaining 72–96% corresponding to variation across sampled demonstration streams. Thus, identification difficulty is primarily a property of the target–stream pair rather than an intrinsic property of the concept alone.

The information carried by positive and negative demonstrations also depends on the composition form. For each behaviorally distinct competitor, we record the label of the first demonstration that eliminates it from the version space. For AND concepts, positive examples eliminate 87.5–90.1% of competitors first, whereas for OR concepts, negative examples eliminate 99.3–99.8% first; M-OF-K lies between these cases. This asymmetry reflects the different logical constraints imposed by the three composition forms and helps explain variation in their identification thresholds.

Finally, we verify that balanced demonstration sampling does not itself create identifiability. Replacing the balanced sampler with natural sampling from the uniform input distribution changes the median H^*_{concept} by at most four demonstrations and leaves identification rates unchanged at the deployed budgets. Balanced sampling therefore controls demonstration label frequency without materially altering the evidence required to identify the target concept.

These diagnostics establish the information-side conditions for interpreting the subsequent concept-recovery experiments: the deployed demonstrations uniquely identify the target concept for nearly all realized streams at $H = 100$ and for all evaluated streams at $H = 200$, while the amount and type of evidence required for identification vary systematically across streams and composition forms.

D Additional Results and Experimental Details

D.1 Prompt Templates

We use the STAGGER-style synthetic environment described in Sec. 3.1. We use a modular prompt design with three base templates: *implicit prediction*, *structured concept recovery*, and *concept application*. Unless otherwise noted, the shared components of each template remain unchanged across experimental conditions. The templates below use an AND instance for illustration; the composition-specific instruction is substituted for OR and m-of-K conditions. All examples use the same 200 demonstrations, truncated for display. Here HURT denotes $y = 0$ and HELP denotes $y = 1$.

Implicit Prediction. The implicit-prediction template evaluates whether the model can infer the input–output relationship directly from demonstrations and predict the label of a held-out query. No concept representation is requested from the model.

Implicit Prediction

SYSTEM

You predict whether an action HELPS or HURTS for an item, using ONLY the observed examples. Answer with exactly one word: HELP or HURT.

USER

Observed examples:

```
x1=high x2=low x3=medium x4=high x5=medium x6=low x7=low -> HURT
x1=high x2=high x3=high x4=high x5=medium x6=high x7=low -> HURT
⋮                                  (200 demonstrations total)
x1=high x2=low x3=high x4=low x5=high x6=low x7=high -> HURT
```

Query: `x1=medium x2=low x3=high x4=medium x5=low x6=high x7=high`
Answer (HELP or HURT):

Structured Concept Recovery. The structured-recovery template asks the model to reconstruct a concept from the demonstrations. The template contains two independently varied components: *concept information*, which provides task-defining structural information when available, and *recovery procedure*, which determines how the model reconstructs the concept.

Structured Concept Recovery

SYSTEM

You maintain a RULE predicting HELP vs HURT from an item's features. The rule is a CONJUNCTION: an item HELPS if and only if EVERY listed feature has one of its allowed levels; features you do not list are irrelevant. Levels are only: low, medium, high.
Output the rule as lines exactly of the form `feature: level, level` (allowed levels for HELP). List ONLY features that matter. Output ONLY these lines and nothing else; if no feature matters yet, output the single word NONE. Change the rule as little as possible.
Concept Information. The concept-information slot controls which task-defining components are provided to the model in addition to the demonstrations. The baseline condition provides no additional concept information. The remaining conditions reveal selected components of the target concept.

USER

CURRENT RULE:
NONE

NEW OBSERVATIONS:
`x1=high x2=low x3=medium x4=high x5=medium x6=low x7=low -> HURT`
⋮ *(200 demonstrations total)*
`x1=medium x2=low x3=high x4=medium x5=low x6=high x7=high -> HELP`

{ Concept Information (optional; ∅ by default — nothing is inserted here unless the condition specifies otherwise) }

{ Recovery Procedure (Direct by default — the line below is used as-is; CoT/Iterative replace or wrap it, see below) }
UPDATED RULE (feature: allowed levels for HELP):

Concept Information

$\varnothing$	Nothing is inserted; this is the baseline recovery condition.
K	The prompt specifies that exactly K features are relevant and asks the model to identify which K are relevant. The available features are listed as `x1, x2, ..., x7`.
A	The prompt directly provides the relevant features: `RELEVANT FEATURES (use ONLY these): x1, x2, x4`.
L	The prompt provides the level constraints associated with the target concept, without directly providing the relevant feature identities.
(A, L)	The prompt directly provides both the relevant features and their corresponding level constraints: `x1: low, medium; x2: low, medium; x4: low, medium`.
(A, L, m)	For m-of-K concepts, the prompt additionally provides the composition threshold m.
$(A, L, \tilde{L})$	The prompt provides (A, L) together with one additional plausible but irrelevant level constraint $\tilde{L}$, which does not belong to the target concept.

Recovery Procedure

Direct (default): no change — the base template ends directly at "UPDATED RULE (feature: allowed levels for HELP):".

CoT: the SYSTEM prompt appends *"Reason BRIEFLY (a few sentences) about the observations, THEN on a new line write* `RULE:` *and, on the lines AFTER it, ONE feature per line... Put NO prose, reasons, or other words inside the RULE block."* plus a one-shot example; the USER prompt's closing line becomes *"Reason briefly, then end with the RULE: block:"* in place of "UPDATED RULE:". Only the text after `RULE:` is parsed and scored against c^*.

Iterative (3 turns; Direct is reused verbatim as Turn 1): Turn 1 = the full Direct template above $\rightarrow \hat{c}_1$. Turn 2 (critique, new call): SYSTEM *"You are given a candidate RULE and labeled examples. Check the rule against EVERY example; list which examples it misclassifies and what is inconsistent... Do NOT output a rule — only the critique."*, USER supplies $\hat{c}_1$ + the same 200 demonstrations. Turn 3 (correct, new call): SYSTEM *"You are given a candidate RULE, a critique of it, and the examples. Produce the CORRECTED final rule... Output ONLY lines* `feature: level, level`*... or NONE."*, USER supplies $\hat{c}_1$ + the Turn-2 critique + demonstrations. Only the Turn-3 output is scored.

Concept Application. The concept-application template evaluates whether the model can apply a supplied concept to a held-out query. We vary the representation of the supplied concept and, where applicable, the procedure used to apply it.

Concept Application

SYSTEM

You are given a RULE (lines `feature: allowed levels`). HELP iff EVERY listed feature has one of its allowed levels. Apply it exactly as written; do not second-guess it. Answer with exactly one word: HELP or HURT.

USER

RULE:

{ Concept Representation — see variants below }

Item: `x1=medium x2=low x3=high x4=medium x5=low x6=high x7=high`

Answer (HELP or HURT):

Concept Representation

DSL `x1: low, medium  x2: low, medium  x4: low, medium` (shown in Template 3 above)

Inequality `x1<=medium AND x2<=medium AND x4<=medium` (system swaps to: *"*`fi>=L` *means attribute* i *is at level* L *or higher;* `fi<=L` *means* L *or lower."*)

Natural language *"An item HELPS (else HURTS) if all of the following hold:* x_1 *is low or medium;* x_2 *is low or medium; and* x_4 *is low or medium."*

D.2 Level-Constraint Error Characterization

We further characterize the level-constraint errors underlying the $\mathrm{Acc}_{L|A}$ metric in Table 1. Let A and $\hat{A}$ denote the ground-truth and recovered relevant-attribute sets, and let L_j and $\hat{L}_j$ denote the ground-truth and recovered level constraints for attribute j. We consider only attributes that are correctly identified as relevant, i.e., $j \in A \cap \hat{A}$. Exact matches $\hat{L}_j = L_j$ are counted as correct in the main-table metric; otherwise, the mismatch is categorized by the set relation between $\hat{L}_j$ and L_j.

Specifically, *Over* denotes $\hat{L}_j \supset L_j$, *Under* denotes $\hat{L}_j \subset L_j$, *Partial* denotes overlap without containment, i.e., $\hat{L}_j \cap L_j \neq \varnothing$ with neither set containing the other, and *Disjoint* denotes $\hat{L}_j \cap L_j = \varnothing$. For example, *low* $\rightarrow$ $\{$*low*, *medium*$\}$ is *Over*, whereas $\{$*low*, *medium*$\}$ $\rightarrow$ $\{$*medium*, *high*$\}$ is *Partial*.

Table 8 reports the distribution of mismatches. *Over* is the dominant error type for most model–form pairs, particularly under OR, while *Under* is comparatively rare. Thus, incorrect level recovery is generally biased toward including additional satisfying levels.

Table 8: Distribution of level-constraint mismatches by error type. Values are computed among attributes correctly identified as relevant but with mismatched level constraints. *Over*, *Under*, *Partial*, and *Disjoint* denote the corresponding set relations between recovered and ground-truth level constraints.

Form	Model	Over	Under	Partial	Disjoint
AND	Qwen2.5-72B	0.65	0.25	0.10	0.00
	Llama-3.1-70B	0.58	0.11	0.32	0.00
	Gemma-2-27B	0.76	0.03	0.21	0.00
	Mistral-24B	0.09	0.32	0.59	0.00
OR	Qwen2.5-72B	0.96	0.01	0.00	0.03
	Llama-3.1-70B	0.93	0.00	0.01	0.06
	Gemma-2-27B	0.72	0.00	0.01	0.27
	Mistral-24B	0.63	0.02	0.01	0.33
m-of-K	Qwen2.5-72B	0.59	0.03	0.27	0.11
	Llama-3.1-70B	0.59	0.04	0.17	0.21
	Gemma-2-27B	0.48	0.05	0.24	0.24
	Mistral-24B	0.32	0.06	0.21	0.40

D.3 Threshold Error Characterization

We further characterize the threshold errors underlying the Acc_m metric in Table 1. Let m and $\hat{m}$ denote the ground-truth and recovered thresholds, respectively. We consider only m-of-K concepts with incorrectly recovered thresholds, i.e., $\hat{m} \neq m$. We classify each error by direction: *Too High* denotes $\hat{m} > m$, while *Too Low* denotes $\hat{m} < m$. Table 9 reports the resulting error distribution. Across most models, threshold errors are biased toward overestimating the number of satisfied predicates required for a positive prediction. Qwen, Gemma, and Mistral make almost exclusively *Too High* errors, whereas Llama shows a more balanced pattern.

Table 9: Distribution of threshold errors by error direction. Values are computed among m-of-K concepts with incorrectly recovered composition thresholds. *Too High* denotes $\hat{m} > m$, while *Too Low* denotes $\hat{m} < m$.

Form	Model	Too High	Too Low
m-of-K	Qwen2.5-72B	1.00	0.00
	Llama-3.1-70B	0.50	0.50
	Gemma-2-27B	1.00	0.00
	Mistral-24B	1.00	0.00

D.4 Concept Application Control

We evaluate model performance when the target concept is provided directly, without requiring concept recovery. Table 10 reports held-out query accuracy across the three concept families and multiple concept representations. Models generally achieve substantially higher accuracy when the concept is directly provided than when it must be recovered from demonstrations. The deterministic executor achieves 1.00 throughout, confirming that the provided concepts are executable under the evaluation procedure.

This control suggests that the prediction–recovery gap is not simply attributable to difficulty in applying an explicitly specified concept. Rather, models can generally apply a provided concept with high accuracy, while recovering the underlying concept from demonstrations remains substantially more difficult.

D.5 Additional Benchmark Datasets

We further examine whether the component-level over-inclusion pattern extends beyond the controlled STAGGER-style environment. We consider two complementary settings: MONK-1, which has a specified target concept and relevant and irrelevant attributes [1], and Mushroom, a real-world

[1] https://archive.ics.uci.edu/dataset/70/monk

Table 10: Held-out query accuracy on Q when the target concept c is directly provided in different representations. *DSL*, *Inequality*, and *Natural Language* express the same concept in symbolic, inequality-based, and natural-language forms, respectively. *DSL + CoT* additionally provides a reasoning trace, while *Executor* applies the concept using a deterministic symbolic executor.

Form	Model	DSL	Inequality	Natural	DSL + CoT	Executor
AND	Qwen2.5-72B	0.94	0.97	0.88	0.98	1.00
	Llama-3.1-70B	0.99	0.96	1.00	0.98	1.00
	Gemma-2-27B	0.90	0.82	0.87	0.98	1.00
	Mistral-24B	0.80	0.81	0.91	1.00	1.00
OR	Qwen2.5-72B	0.97	0.99	0.99	0.99	1.00
	Llama-3.1-70B	0.76	0.95	0.94	1.00	1.00
	Gemma-2-27B	0.94	0.87	0.92	0.81	1.00
	Mistral-24B	0.64	0.97	0.93	1.00	1.00
m-of-K	Qwen2.5-72B	0.79	0.87	0.81	0.98	1.00
	Llama-3.1-70B	0.74	0.77	0.80	0.98	1.00
	Gemma-2-27B	0.83	0.83	0.78	0.86	1.00
	Mistral-24B	0.68	0.73	0.76	0.99	1.00

categorical classification dataset without a dataset-official ground-truth concept [2]. MONK-1 therefore permits both behavioral and feature-level recovery evaluation, whereas Mushroom provides an external setting for examining attribute selection without assuming an official ground-truth concept.

As shown in Table 11, recovered concepts tend to include more attributes than the available or documented relevant-attribute reference on both datasets, except for Llama on MONK-1. This pattern is consistent with the over-inclusion observed in the controlled setting.

Table 11: Concept application and recovery on additional benchmark datasets. Q_{imp} denotes direct prediction from the demonstrations, while Q_{exp} denotes prediction obtained by executing the recovered specification on held-out queries. Since the query sets are balanced, 0.50 corresponds to random guessing. The remaining columns report attribute-level recovery. A-Precision and A-Recall measure precision and recall of the recovered relevant-attribute set on MONK-1; Ref. $|A|$ denotes the documented reference feature count and $|\hat{A}|$ the average number of recovered attributes.

		Concept Application		Concept Recovery			
Dataset	Model	Q_{imp}	Q_{exp}	A-Precision	A-Recall	Ref. $\lvert A\rvert$	$\lvert\hat{A}\rvert$
MONK-1	Qwen2.5-72B	0.81	0.52	0.43	0.67	3	4.6
	Llama-3.1-70B	0.81	0.49	0.70	0.40	3	1.8
	Gemma-2-27B	0.65	0.52	0.39	0.67	3	5.0
	Mistral-24B	0.74	0.48	0.59	0.73	3	3.8
MUSHROOM	Qwen2.5-72B	0.96	0.67	–	–	4	7.8
	Llama-3.1-70B	0.73	0.56	–	–	4	4.4
	Gemma-2-27B	0.76	0.46	–	–	4	12.0
	Mistral-24B	0.78	0.50	–	–	4	13.4

D.6 Inference-Time Reasoning and Self-Refinement

We further examine whether inference-time reasoning can improve recovered-concept prediction when the available structural information is incomplete. In particular, we consider two commonly used inference-time interventions: chain-of-thought reasoning (Wei et al., 2022) and iterative self-refinement (Madaan et al., 2023). These interventions leave the demonstrations and recovery procedure unchanged and modify only the inference process used to produce the recovered concept.

We evaluate the resulting recovered concepts by applying them to the same held-out query set Q and reporting Q_{exp}. We consider two information settings. In the first, the model receives demonstrations alone and must recover the concept without additional structural information. In the second, we provide the identities of the ground-truth relevant attributes A, while withholding their level

[2] https://archive.ics.uci.edu/dataset/73/mushroom

constraints and the composition rule. This setting tests whether inference-time reasoning can make effective use of partial structural information.

For chain-of-thought reasoning, the model is prompted to reason about the demonstrations and explicitly derive the concept before producing the recovered specification. For iterative self-refinement, the model first produces a recovered concept, then revisits its prediction and revises the specification based on its own intermediate result. We use the same evaluation protocol and held-out queries across all inference conditions.

Table 12 summarizes the results. Neither chain-of-thought reasoning nor iterative self-refinement consistently improves recovered-concept prediction over direct recovery. This remains the case when the identities of the relevant attributes are provided: although some model–form combinations improve, others remain unchanged or degrade. Thus, the tested inference-time interventions do not reliably compensate for incomplete structural information.

Overall, these results provide a complementary sanity check on the structural-information analysis in Sec. 4.3. The recovery gains observed when complementary structural components are jointly specified are not reproduced simply by adding inference-time reasoning or self-refinement.

Table 12: Inference-time interventions for recovered-concept prediction. Q_{exp} denotes prediction accuracy on held-out queries after executing the recovered concept; $Q_{\text{exp}}(A)$ additionally provides the identities of the ground-truth relevant attributes, without their level constraints or composition rule. Superscripts CoT and Iter denote chain-of-thought reasoning and iterative self-refinement, respectively.

Form	Model	Q_{exp}	$Q_{\text{exp}}^{\text{CoT}}$	$Q_{\text{exp}}^{\text{Iter}}$	$Q_{\text{exp}}(A)$	$Q_{\text{exp}}^{\text{CoT}}(A)$	$Q_{\text{exp}}^{\text{Iter}}(A)$
AND	Qwen2.5-72B	0.63	0.54	0.56	0.95	0.81	0.83
	Llama-3.1-70B	0.56	0.59	0.55	0.96	0.83	0.94
	Gemma-2-27B	0.58	0.52	0.54	0.64	0.61	0.65
	Mistral-24B	0.52	0.52	0.51	0.83	0.74	0.68
OR	Qwen2.5-72B	0.50	0.53	0.51	0.52	0.65	0.53
	Llama-3.1-70B	0.51	0.56	0.52	0.54	0.66	0.56
	Gemma-2-27B	0.50	0.48	0.50	0.49	0.43	0.49
	Mistral-24B	0.51	0.56	0.51	0.52	0.57	0.56
m-of-K	Qwen2.5-72B	0.53	0.53	0.54	0.54	0.50	0.53
	Llama-3.1-70B	0.51	0.52	0.51	0.50	0.50	0.52
	Gemma-2-27B	0.51	0.49	0.51	0.47	0.44	0.50
	Mistral-24B	0.51	0.47	0.51	0.47	0.41	0.47

D.7 Training Details

This appendix describes the supervised training procedure used for the results in Sec. 4.4. The supervised models are trained on m-of-K concepts; AND and OR concepts are used only for cross-family transfer evaluation.

Training Data. We generate training concepts using the same procedure as in the main experiments, with $K \in \{3, 4\}$ and demonstration budgets $H \in \{100, 200\}$. Each training concept specifies its relevant attributes, level constraints, and composition threshold m. We construct $3,000$ training concepts. For *recovery-only* training, each concept contributes one structured-recovery example mapping demonstrations to the target concept, yielding $3,000$ training examples. For *joint* training, each concept additionally contributes two balanced implicit-prediction examples, one for each label, yielding $9,000$ examples in total. Recovery and prediction examples are interleaved and jointly optimized.

Evaluation Split. To reduce the possibility of memorizing specific relevant-attribute combinations, held-out evaluation concepts use relevant-attribute combinations that are disjoint from those used for training. The evaluation set contains 570 concepts, including 270 with $K = 3$ and 300 with $K = 4$.

Training Objective. Both structured recovery and implicit prediction are trained with the standard autoregressive next-token cross-entropy objective. We use completion-only masking: prompt tokens are excluded from the loss, which is computed only over the target completion. Recovery

examples target the structured concept specification, whereas implicit-prediction examples target the corresponding binary label.

Model and Optimization. We fine-tune Qwen2.5-7B-Instruct with LoRA while keeping the backbone parameters frozen. LoRA uses rank $r = 16$, scaling factor $\alpha = 32$, and dropout 0.05, introducing approximately 40M trainable parameters. We use AdamW with a learning rate of 1×10^{-4}, cosine learning-rate decay, and a 3% warmup ratio. Training uses bf16 precision.