VISUAL LOGICAL RULE INDUCTION BY BRIDGING INDUCTIVE LOGIC PROGRAMMING AND MULTIMODAL LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose *ILP-CoT*, a method that bridges Inductive Logic Programming (ILP) and Multimodal Large Language Models (MLLMs) for visual logical rule induction, which involves both discovering logical facts and inducing logical rules from a small number of unstructured visual inputs. This task still remains challenging when solely relying on ILP, due to the requirement of specified background knowledge and high computational cost, or MLLMs, due to the appearance of perceptual hallucinations. Based on the methodology that let MLLM be the input perceptor and rule structure proposer, meanwhile let ILP system serve as the rule inducer and symbolic verifier, our approach automatically builds ILP tasks with pruned search spaces, and utilizes ILP system to output rules built upon rectified logical facts and formal inductive reasoning. Its effectiveness is verified through challenging logical induction benchmarks, as well as a potential application of our approach, namely text-to-image customized generation with rule induction. Our code and data will be open-sourced upon formal release.

1 INTRODUCTION

Although remarkable progress has been made in improving the deductive reasoning abilities of AI systems (OpenAI, 2024b; Guo et al., 2025), inductive reasoning from raw data, which *challenges more on perceiving and understanding complex raw inputs than long-chain deductive reasoning*, remains a significant challenge. To pursue this direction, we study the task of visual logical rule induction, in which the target is to utilize a small number of unstructured visual instances to automatically identify and ground symbolic concepts and then inducing possible logical rules indicated by the instances. This reasoning task involves the dual challenges of both input perception and logical induction. On the one hand, the model needs to extract abstract and transferable symbolic concepts from input instances; on the other hand, utilizing limited instances, it must accurately infer the underlying logical relationships or rules.

Traditionally, the above task can be solved by a two-step pipeline. First, a preprocessing step is performed by perception tools to transform the raw inputs into structured symbolic data. Afterwards, an external Inductive Logic Programming (ILP) module (Muggleton & De Raedt, 1994; Cropper & Dumančić, 2022) is introduced for logical rule induction. ILP systems are formal logical reasoning systems with strong advantages in terms of interpretability and verifiability. By inductively learning from a finite set of facts and background knowledge, ILP is able to produce logically transparent and auditable rules. From a theoretical perspective, the rules output by ILP can be formally verified, a feature that is particularly important in high-risk scenarios or applications with stringent correctness requirements. However, ILP also faces fundamental challenges, such as relying on structured input data and potential inefficiency in large-scale data settings. In response to these challenges, recent work has begun to explore neuro-symbolic methods that integrate deep learning with symbolic logical reasoning (Manhaeve et al., 2018; Evans et al., 2021; Dai & Muggleton, 2021; Cunnington et al., 2023; Shindo et al., 2023; 2024; Wüst et al., 2024). These approaches attempt to use neural networks for perception and representation learning, then employ ILP or other inductive program synthesis strategy for rule induction. Even though these approaches significantly enlarge the applicability in real applications, utilizing ILP usually requires to design logical background knowledge by human experts, which is a fundamental obstacle in handling challenging inductive reasoning problems.

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.

With the rise of Multimodal Large Language Models (MLLMs) (OpenAI, 2024a; Liu et al., 2023a; Wang et al., 2023), researchers have begun to explore the application of these powerful models to textual and visual reasoning tasks. Due to training on massive datasets, MLLMs already exhibit astonishing performance in handling multimodal inputs, extracting symbolic representations, and mining rich semantic information, making them promising candidates for addressing visual logical rule induction. However, MLLMs still face multiple bottlenecks in perception and reasoning (Zhang et al., 2023), including hallucination phenomena, highly opaque reasoning processes, and a lack of verifiable logical chains. We discover that these bottlenecks still limit the ability of MLLMs to directly solve visual logical rule induction, even when guided by Chain-of-Thought (CoT) reasoning (Wei et al., 2022).

To tackle the above challenges, we propose a hybrid method, *ILP-CoT*, to bring the best of both worlds. Our approach integrates the ILP system into the CoT reasoning process of MLLMs in a "plug-and-play" manner, forming a fully interpretable reasoning pipeline from start to finish without additional training. Leverage the strong cross-modal perception and symbol extraction capabilities, our approach let MLLM be the input perceptor and rule structure proposer, which automatically builds ILP tasks with pruned search spaces. Subsequently, our approach let ILP system serve as the rule inducer and symbolic verifier. It dynamically invoke the ILP system to perform formal rule induction, yielding explainable and verifiable rules with rectified logical facts. This division of labor separates the complex symbolic grounding process and reduces the size of the rule hypothesis space, letting ILP focus exclusively on the more compact and structured symbolic data. This approach not only reduces the risk of hallucination during MLLM-driven inference, but also utilizes formal verification from ILP to ensure the accuracy and consistency of the rules.

We introduce visual logical rule induction benchmarks that remain challenging to current MLLMs to systematically verify the efficacy of our approach (Stammer et al., 2021; Helff et al., 2023; Vedantam et al., 2021). Furthermore, we propose a potential application, namely text-to-image customized generation with rule induction. We show that our approach enables to induce the latent logical rules from the training images, which can be utilized by downstream pre-trained text-to-image generation models to further generate images following user preferences.

2 Related Work

Avoiding hallucination is a fundamental challenge for Large Language Models (LLMs) (Dasgupta et al., 2022; Saparov & He, 2022). A widely adopted strategy is Retrieval-Augmented Generation (RAG) (Lewis et al., 2020), which utilizes retrieval in external knowledge bases to avoid generating ungrounded contents. Although RAG is effective in reducing factual errors, it is invalid for rectifying fallacious reasoning processes. Furthermore, the requirement of accessing an external knowledge base is somehow limited for solving general reasoning tasks. Recently, there have been attempts to use formal programming code (Gao et al., 2023; Li et al., 2023; Chae et al., 2024; Ling et al., 2024) or logical rules (Xu et al., 2024) as LLM-generated intermediate content during CoT reasoning. These studies justify that formalizing the reasoning states can improve the accuracy of the reasoning chain without using external tools. However, it is still challenging to conduct fully reliable reasoning based on this mechanism. On the other hand, the idea of integrating external formal reasoning systems with LLMs has been explored in various reasoning tasks. Some research proposed to transform natural languages into code and further execute them using external symbolic solvers (Wu et al., 2022; He-Yueya et al., 2023; Lyu et al., 2023; Pan et al., 2023a; Ye et al., 2024; Jiang et al., 2024). A major issue is that formalizing natural language into executable code is itself a difficult task, which is also a key challenge that we try to tackle in our work. In complex reasoning tasks with long reasoning chains, such as solving mathematical challenges, formal reasoning systems are treated as the sledgehammer to integrate with LLMs (Trinh et al., 2024). Unlike existing approaches that focus on deductive reasoning tasks (Pan et al., 2023b; Olausson et al., 2023), our work focuses on inductive reasoning. The significant difference lies in that inductive reasoning usually does not challenge the ability to do long-step reasoning but rather the ability to perceive and understand the input. This makes inductive reasoning more challenging for MLLMs due to the difficulty in perceiving complex multimodal inputs. We also note that this makes our contribution in parallel with multimodal deductive reasoning methods (Wang et al., 2022; Madaan et al., 2023; Gao et al., 2024; Mondal et al., 2024). Research on breaking the perceptual limitations of MLLMs has also received great attention. Existing approaches utilize scene graph knowledge (Mitra et al., 2023) or visual

prompts (Wu et al., 2024), while formal methods have not received significant attention in MLLMs. The closest idea comes from visual question answering, in which textual LLMs are integrated with visual perception models to perform visual reasoning tasks (Hsu et al., 2024; Kamali et al., 2024). Purely textual LLMs rely on external visual processing models to perceive the input, while the target of our work is to conduct visual logical rule induction based on the internal perceptual ability of MLLMs without using external tools to take the responsibility of perception.

3 METHOD

ILP-CoT follows two connected steps: it turns images into logical facts and uses those facts to induce a rule. The MLLM proposes what to observe and which rule structures are worth searching; Metagol instantiates and tests those structures. Errors on labeled support examples can lead to revisions of capture tokens, image facts, and rule structures in later rounds; a support-based selector chooses among the executable programs.

3.1 PRELIMINARIES

In an abductive logical rule induction task, a small number of textual or visual instances are provided. Each instance is unstructured and labeled as *positive* or *negative* based on whether it is consistent with a set of latent logical rules describing regularities among its objects. The targets are twofold: (1) transforming the unstructured instances into structured ones by discovering logical facts about the objects, their attributes, and their relations; and (2) inducing the latent logical rules based on the discovered facts. For the visual tasks considered here, we write the labeled images as the *support set* $S = \{(x_i, y_i)\}_{i=1}^{n}$, where $y_i = 1$ denotes a positive example and $y_i = 0$ a negative one. The facts extracted from image x_i form its fact set z_i. We apply the learned rule to new *query images* Q.

Inductive logic programming (ILP) seeks to identify a set of logical rules h that can explain, or more formally *entail*, positive examples E^+ while excluding negative examples E^-, combined with background knowledge B (Muggleton & De Raedt, 1994; Cropper & Dumančić, 2022). Here E^+ and E^- express which instances satisfy the target concept, and B supplies the facts and relations used to explain them. For example, a positive train contributes `eastbound(t)` to E^+, while facts such as `car(t,a)` and `barrel(a)` belong to B and describe that train. The label specifies what the rule should predict; the facts provide the evidence on which the rule operates.

A *fact* is a statement of the form $p(c_1, \ldots, c_m)$ about particular objects, also called *entities*. Each c_i is an *identifier*, a name assigned to an object, and the *predicate* p names a property or relation. Thus, `barrel(a)` states that car a carries barrels, and `behind(b,a)` states that car b follows car a. An identifier names the same object wherever it occurs within one image's facts. A *rule*, or clause, states that a conclusion holds when its conditions hold. For example,

$$\text{eastbound}(T) \leftarrow \text{car}(T, A), \text{barrel}(A)$$

means that a train is Eastbound if it contains a car carrying barrels. Capital letters such as T and A are variables that can match different objects; repeating a variable requires the same object. One or more such rules form a *logic program*, and a program considered during learning is a *hypothesis* h.

To conduct rule induction, ILP systems search the *hypothesis space* $\mathcal{H}$ of possible programs for a hypothesis satisfying the above objective. Rule-structure constraints provide an *inductive bias*: they guide the search toward particular ways of combining facts. As in general machine learning, an appropriate inductive bias can reduce the search space and improve learning efficiency. A *meta-rule* expresses such a structure while leaving its predicate names open. For example, $P(X) \leftarrow Q(X, Y), R(Y)$ specifies how two conditions share an object. Here P, Q, R are predicate placeholders, whereas X, Y are object variables. Replacing the placeholders with `eastbound`, `car`, and `barrel` gives the rule above.

We use Metagol as the ILP system. Given meta-rules, background facts, and positive and negative examples, it searches for programs by filling the predicate placeholders and testing the resulting rules against the examples. Predicate *argument types* describe the kinds of objects a predicate takes, such as a train and a car for `car(T,A)`. A successful Prolog query predicts 1; a failed query predicts 0. In the typed fact evaluator, an explicitly unknown property or an incomplete image record can instead leave a condition undecided, denoted by ?; ordinary Prolog failure is not treated as unknown.

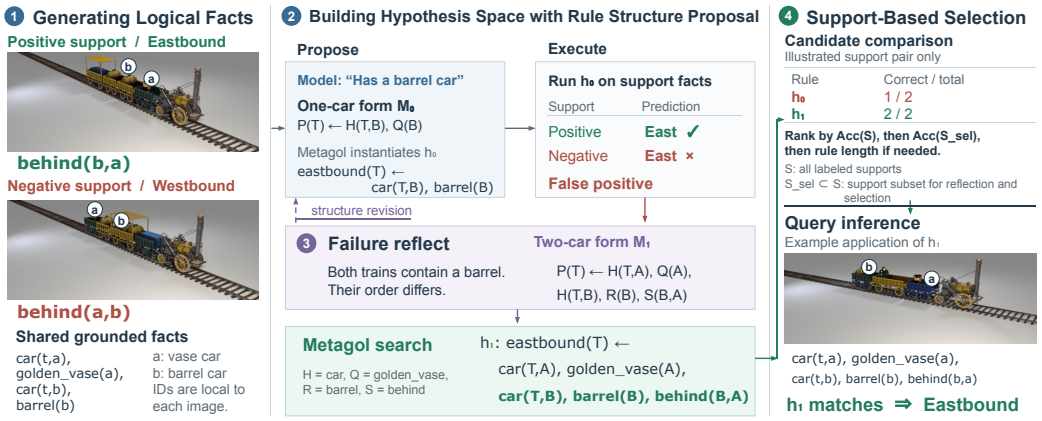

Figure 1: The ILP-CoT workflow. Details are discussed from Sec. 3.2 to Sec. 3.5.

During support evaluation and program selection, ? matches neither a positive nor a negative label and counts as an error. ILP-CoT uses the MLLM to supply the initial facts and rule structures, giving this search a starting point grounded in the input images.

3.2 Generating Logical Facts

We propose a unified procedure for turning images into logical facts that can be checked by ILP. Learning a shared rule requires a property to keep the same meaning across images, even when the model describes it in different words. In each round, the MLLM proposes *capture tokens*: names and descriptions of visual attributes and relations that may matter for the labeled examples. It grounds the current tokens in the images so that a token has a consistent meaning within that round. The token set is not fixed: feedback may add, revise, or remove tokens before the next grounding pass. For the train images in Fig. 1, useful tokens include `golden_vase`, `barrel`, and `behind`.

To connect attributes with relations, the model gives each car an identifier such as a or b and reuses it whenever a fact refers to that car. It assigns these identifiers separately for each image. For the positive and negative trains in Fig. 1, the records include

$$z_+ : \{\text{golden_vase}(a),\ \text{barrel}(b),\ \text{behind}(b,a)\},$$
$$z_- : \{\text{golden_vase}(a),\ \text{barrel}(b),\ \text{behind}(a,b)\}.$$

The model observes which car follows the other; reusing a and b connects this observation to the vase and barrel properties. A rule can then combine the three facts to test the order of the cars carrying these loads.

These per-image fact records form the current round's *support inventory*, retaining the observed properties, their objects, and the relations between them. Shared capture tokens make properties comparable across images in that round, while repeated object identifiers connect the facts that a rule can combine. This gives ILP a consistent basis for testing the same conditions against positive and negative examples without treating the initial tokens or facts as permanently correct.

3.3 Building Hypothesis Space with Rule Structure Proposal

Once logical facts have been extracted, the next step is to construct a hypothesis space that enables efficient rule induction with ILP. We adopt a two-substep approach.

Substep 1: Generating rule structure proposals using the MLLM. The same MLLM that extracts the logical facts is asked to propose plausible rules from these facts and the positive and negative examples. We call these rules *rule structure proposals* because we retain their structures to guide ILP search. We require the model to write each proposal in Prolog, specifying which attributes and relations to combine and how they share objects. Its prior knowledge provides an initial

inductive bias: Metagol starts from these proposed combinations rather than exploring all possible rule structures.

For example, an initial proposal for the trains in Fig. 1 may state that a train is Eastbound if it has a barrel car:

$$h_0 : \quad \text{eastbound}(T) \leftarrow \text{car}(T, B), \text{barrel}(B).$$

This proposal connects a train to one car and tests a property of that car.

Substep 2: Transforming proposals into logical meta-rules. The rule structure proposals are automatically transformed into meta-rules compatible with Metagol. The transformation replaces predicate names with placeholders and any specific object names with variables, preserving shared variables and argument order. For h_0, replacing `eastbound`, `car`, and `barrel` with P, H, and Q gives

$$M_0 : \quad P(T) \leftarrow H(T, B), Q(B).$$

H links the train to a car, and Q tests a property of that same car. The property is now left open: Metagol can fill Q with `barrel` or another property from the extracted facts. The resulting meta-rules define the initial hypothesis space, within which Metagol searches for rules that explain the labeled examples. The proposals therefore supply a useful structural starting point for search, while ILP determines which predicates to use. This retains the model's guidance about how to combine facts in a form that Metagol can search directly.

3.4 Failure Reflection

One meta-rule can yield several programs that fit the images used in search. To help distinguish them, we split the positive and negative support images separately into a fitting set S_{fit} and a selection set S_{sel}. The initial rule proposals and fitting searches use S_{fit}. Missing, wrong, or extra facts in some images can prevent a useful rule from fitting all examples at once. We therefore search on balanced, possibly overlapping subsets of S_{fit}, from small subsets to the full fitting set. A smaller subset can support a useful rule even when faulty facts in other examples would block it. Because this rule may still fail outside its search subset, we test every candidate on all support images and use the errors to guide reflection. This also allows labels from S_{sel} to inform later feedback and candidate repair.

We retain candidate rules with different predictions and select positive and negative examples that several rules misclassify or predict differently. The same MLLM receives the selected images, their labels and facts, the candidate rules and their prediction errors, and a summary of the subsets searched and the number of rules found. We ask it to compare the conditions tested by these rules with the visible differences between positive and negative examples. If a rule wrongly accepts a negative image, the model looks for a condition present in positive images but absent from that image. If it rejects a positive image, the model checks whether its conditions are incorrect or whether a separate clause is needed.

The model turns these comparisons into suggestions about how to combine facts: whether properties should describe the same object, whether conditions on different objects need a connecting relation, or whether different positive examples require alternative clauses. The reflection prompt asks it to check each suggested structure against the shown positive and negative examples. The model also specifies which visual properties need checking to evaluate these suggestions. We use the corresponding capture tokens to ask the model to recheck the support and unlabeled query images and update their facts. The next rule proposal then uses both the structural suggestions and these updated facts.

For example, h_0 misclassifies the negative train because it only requires a barrel car. Both displayed trains contain a vase and barrels, so adding a vase-presence condition would still accept the negative train. Their car positions, however, differ. A visual check would verify the loads and positions: with a naming the vase car and b the barrel car, the positive image supports `behind(b,a)` and the negative image supports `behind(a,b)`. This contrast suggests testing a property on each of two cars and an ordering relation between those same cars. The next proposal can therefore require a barrel car to follow a vase car. Applying the transformation in Substep 2 gives

$$M_1 : \quad P(T) \leftarrow H(T, A), Q(A), H(T, B), R(B), S(B, A).$$

Metagol searches within this new form. For $H = \text{car}$, $Q = \text{golden_vase}$, $R = \text{barrel}$, and $S = \text{behind}$, it can produce

$$h_1 : \quad \text{eastbound}(T) \leftarrow \text{car}(T, A), \text{golden_vase}(A), \text{car}(T, B), \text{barrel}(B), \text{behind}(B, A).$$

In M_1, $Q(A)$ and $R(B)$ test the two car properties, while $S(B, A)$ compares those cars. The feedback thus changes the search from testing one car's property to testing two properties together with a relation. The resulting program h_1 accepts the displayed positive train and rejects the negative one. We evaluate the new programs on all support images and use their errors in the next reflection round, up to two round limit; all executable programs remain available for final selection.

3.5 Support-Based Program Selection

Programs that fit their own support subsets equally well may still disagree on other images. The selector therefore evaluates every program on the same full support set S and selection set S_{sel}. We compute accuracy by dividing the number of correct predictions by the total number of evaluated images. Accuracy on S measures overall agreement with the support labels, while accuracy on S_{sel} gives a separate view of examples excluded from the initial fitting searches.

The selector first compares accuracy on S. For programs with similar full-support accuracy, it then compares their prediction accuracies on S_{sel}. When both scores are similar, it prefers a shorter program, avoiding extra conditions justified only by a small score increase. Balanced accuracy, the average of the correct-prediction rates for the two classes, helps resolve remaining choices while giving both classes equal weight.

The selector is applied after each search round. Its current choice supplies the errors examined in the next reflection round; after the last round, the final choice is applied to each query image. It returns a class prediction for classification tasks or a natural-language description of the learned condition when the task asks for one.

4 Experiments

4.1 Evaluation settings and measures

Benchmarks. The three classification benchmarks use the same few-shot task: given a set of positive and negative images, infer the condition that separates them and classify new test images. They differ in what the images show and what a rule must express.

CLEVR-Hans (Shindo et al., 2024) shows rendered scenes containing cubes, spheres, and cylinders with different colors, sizes, and materials. One of its three concepts requires both a large cube and a large cylinder. In the training images, the cube's gray color can accompany this concept even though color is not part of its rule; the test images vary that color. The task therefore tests whether a model learns the defining objects rather than a coincidental cue.

V-LoL (Helff et al., 2024) instead shows rendered trains whose cars differ in length, wheel count, and loads. Its rules combine car properties, loads, counts, and relations between cars. The learner must identify which combinations of these conditions determine the class of the whole train.

CURI-90 includes 90 few-shot concept tasks uniformly sampled from the original large-scale CURI dataset (Vedantam et al., 2021), with rendered scenes of multiple simple objects. Concepts can combine properties and conditions over several objects, such as requiring all objects to share a color or one object to have a specified color while the remaining objects have a specified shape. Several rules may fit a small support set, so this benchmark tests the combination of visual properties and reasoning under uncertainty.

For classification, the primary outcome is the query accuracy of the *final selected program* for ILP-CoT, or the final query prediction for a baseline. Dataset summaries average task accuracies; V-LoL covers three rules, CLEVR-Hans three concepts, and CURI covers 90 tasks. All percentages are shown in two decimal places.

Table 1: Query accuracy (%) across datasets, reasoning methods, and base models. Bold marks the highest accuracy within each model and dataset. All baseline names ignore the "CoT" suffices.

	V-LoL				CLEVR-Hans				CURI-90			
Model	Base	Custom	Code	Ours	Base	Custom	Code	Ours	Base	Custom	Code	Ours
Qwen3-VL-32B	52.45	58.97	48.82	**81.25**	50.78	53.13	47.66	**82.55**	60.35	70.60	56.75	**92.19**
MiMo-V2.5	40.22	49.68	58.97	**68.75**	49.74	52.60	50.00	**79.69**	54.43	73.01	54.83	**86.52**
GPT-4o	54.17	53.44	50.00	**67.57**	50.00	52.86	47.66	**69.79**	62.71	76.83	58.84	**83.26**
GPT-5.2	51.18	49.73	46.92	**81.25**	53.39	58.07	48.18	**83.33**	67.50	73.46	69.01	**86.89**

Table 2: Ablation of failure reflection and support-based program selection with Qwen3-VL-32B. Values are query accuracy (%). Uniform-choice columns report expected accuracy over the candidate pool; the other columns report final-program accuracy.

	With failure reflection			Without failure reflection		
Dataset	Support-based	Support-argmax	Uniform choice	Support-based	Support-argmax	Uniform choice
V-LoL	**81.25**	70.83	56.34	62.50	58.33	55.75
CLEVR-Hans	**82.55**	81.25	54.52	79.43	77.60	53.57
CURI-90	**92.19**	82.11	59.07	91.47	81.31	58.58

4.2 Baseline Comparison

The baseline comparisons are conducted with four base MLLMs: Qwen3-VL-32B-Instruct (Bai et al., 2025), MiMo-V2.5 (Xiaomi MiMo Team, 2026), GPT-4o (OpenAI, 2024a), and GPT-5.2 (OpenAI, 2025b). We introduce the following baselines. **Base CoT** shows the model the positive and negative images and asks it to reason in natural language before classifying the test image. **Custom CoT** is a specific baseline which follows the similar grounding-then-inducing paradigm to ILP-CoT, but does not utilize ILP system for induction. **Code CoT** test the native coding agent ability of the base model, which asks the model to write its inferred rule in Prolog and then uses that rule to classify the test image. The implementation details of the baselines are in Sec. A.

The results are illustrated in Tab. 1. It can be seen that ILP-CoT significantly outperforms baselines in all scenarios. This justifies the overall efficacy of our method. Note that Custom CoT usually performs second best. This not only justifies the benefit of our grounding-then-inducing reasoning paradigm, but also justifies the benefit of introducing the ILP system as the formal reasoner and verifier. In addition, the undesirable performance of the Code CoT baseline shows that the native coding ability of MLLMs is not sufficient to tackle complex visual inductive reasoning tasks.

4.3 Ablation Study

Tab. 2 verifies two essential components discussed in Sec. 3.4 and 3.5, which are the failure reflection mechanism and the program selection strategy. With Qwen3-VL-32B, we remove failure reflection or replace support-based program selection. **Support-based** is the default implementation in our approach. **Support-argmax** chooses the program with the highest support score. **Uniform choice** reports the expected test accuracy if an executable candidate were chosen at random.

The results show that both components make significant contributions. Our selector outperforms support-argmax on all three datasets, both with and without reflection. Both methods use the same labeled support examples; the difference is how they rank the candidate programs. Rather than relying on full-support accuracy alone, our selector also checks S_{sel} accuracy and program length. Its advantage over support-argmax suggests that a higher full-support score does not always identify the rule that predicts new images better. Uniform choice performs much worse than either selector, indicating substantial differences in quality among the rules returned by search. Reflection raises uniform-choice accuracy on all three datasets. Since uniform-choice accuracy is the mean test accuracy of the candidate rules, this gain shows that reflection improves the average quality of the candidate pool. Both selectors also achieve higher final-program accuracy with reflection, so the gains occur at both the pool and selected-program levels. Sec. B details the pool mean reported by uniform choice.

Table 3: ILP backend comparison with Qwen3-VL-32B. Values are query accuracy (%). The program search time limit are 60 seconds for V-LoL, 300 seconds for CLEVR-Hans and CURI-90. "No rule output" means that no rule is induced on the benchmark consistently.

Dataset	Metagol (Ours)	Aleph (Srinivasan, 2001)	ILASP (Law et al., 2014)	XHAIL (Ray, 2009)
V-LoL	**81.25**	54.17	No rule output	Out-of-Time
CLEVR-Hans	**82.55**	67.68	Out-of-Time	Out-of-Time
CURI-90	**92.19**	81.6	Out-of-Time	Out-of-Time

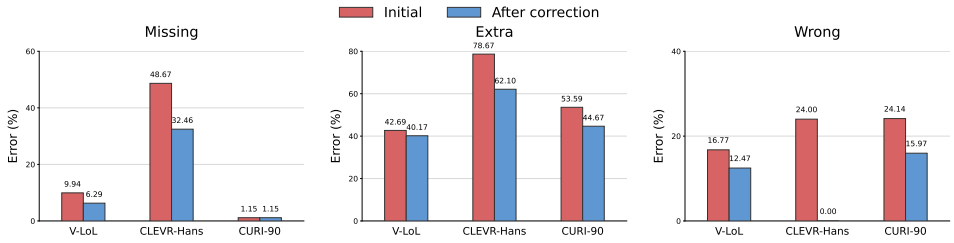

Figure 2: Fact error rates before and after ILP-CoT processing(%). Missing: true, task-relevant facts omitted from the record. Wrong: recorded facts that misdescribe the image. Extra: visually correct facts irrelevant to the target rule.

The comparison for substituting the ILP backend in our method is illustrated in Tab. 3. The results justify that Metagol is a proper choice for our ILP-CoT pipeline, which is based on the structure priors defined by meta-rules. Meta-rules define a focused search space and specify plausible ways to combine visual facts. By reducing irrelevant combinations while retaining promising rule structures, this guidance helps Metagol find accurate programs within the search time limits.

4.4 Visual Fact Errors and Correction

Fig. 2 illustrates the error rates of perception generated by MLLMs, before and after ILP-CoT processing. The results show that the perceptual error consistently decreases, justifying the effectiveness of the formal verification of ILP. On the other hand, we can also observe an interesting fact that the perception errors are actually not eliminated after ILP-CoT processing, even when the induction performance gets significantly better. This justifies the noise tolerance ability of our approach: Correct rules can be induced even when perceptual errors still exist.

4.5 Text-to-Image Customized Generation with Rule Induction

We propose a potential application of our approach. In this task, a small number of images provided by the user are given, including multiple subjects that the user cares about. Furthermore, the images are labeled by the user as "liked" or "disliked", followed by the preferences of the user for some latent regularities among the subjects, which can be represented by logical rules. For justification, we construct a benchmark named **ILP-CoT-Customization**, whose details are reported in Sec. D.

In ILP-CoT-Customization, the model expresses the inferred rule in natural language rather than assigning a class label to a test image. Compared with the classification benchmarks, this dataset features more varied subjects, scenes, and relations. These include spatial arrangements, object attributes, interactions among people or animals, and responses to the environment. Its 22 rule-induction tasks each provide five positive and five negative images. The challenge is to describe the relevant subjects and their attributes or interactions accurately across these diverse scenes, without including incidental details.

For rule induction performance, Tab. 4 compares Direct Predict with ILP-CoT, both using GPT-4o as the base model. Direct Predict asks the model to describe the rule from the positive and negative images directly. We vary the number of positive (P) and negative (N) examples using four settings: 1P1N, 3P3N, 5P1N, and 5P5N. Human evaluations (averaged over two participants), Gemini 2.5 Pro (Comanici et al., 2025), and GPT-5 (OpenAI, 2025a) compare each inferred rule with the reference rule and assign one of four grades. The results show that our approach performs

Table 4: Rule-induction performance on ILP-CoT-Customization. Both Direct Predict and ILP-CoT use GPT-4o as the base model. Each entry gives the proportions of completely correct / mostly correct / partially correct / incorrect rules.

Evaluator	Method	1P1N	3P3N	5P1N	5P5N
Human	Direct Predict	0.11/0.14/0.42/0.33	0.12/0.16/0.26/0.45	0.15/0.18/0.36/0.31	0.14/0.17/0.37/0.32
	ILP-CoT	**0.53/0.21/0.25/0.01**	**0.64/0.21/0.13/0.02**	**0.58/0.24/0.17/0.01**	**0.63/0.27/0.09/0.00**
Gemini 2.5 Pro	Direct Predict	0.17/0.15/0.44/0.23	0.19/0.20/0.26/0.35	0.07/0.20/0.42/0.31	0.13/0.10/0.40/0.37
	ILP-CoT	**0.52/0.13/0.34/0.01**	**0.56/0.24/0.17/0.03**	**0.53/0.20/0.25/0.02**	**0.59/0.28/0.12/0.01**
GPT-5	Direct Predict	0.06/0.09/0.52/0.33	0.04/0.11/0.22/0.63	0.24/0.09/0.39/0.28	0.13/0.18/0.43/0.26
	ILP-CoT	**0.49/0.24/0.27/0.00**	**0.57/0.24/0.17/0.02**	**0.53/0.30/0.16/0.01**	**0.61/0.28/0.11/0.00**

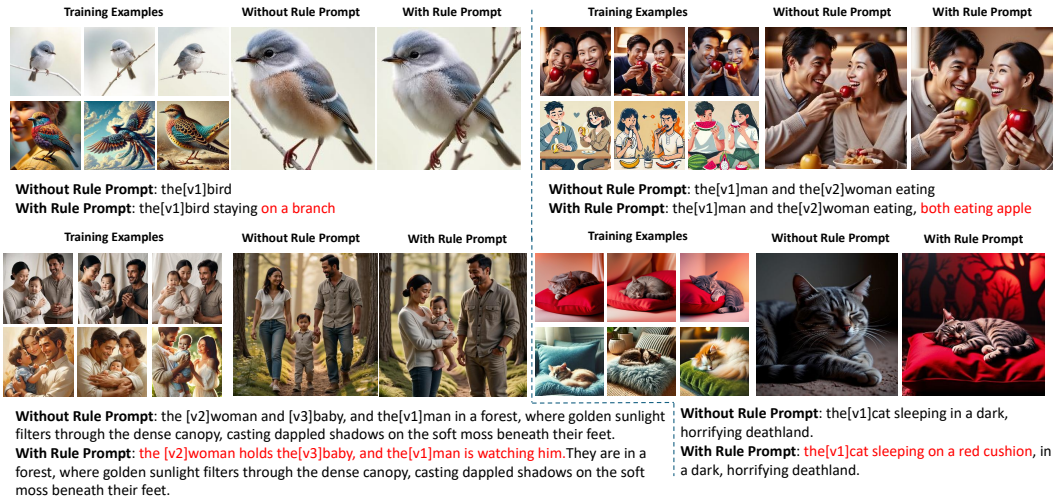

Figure 3: The figure presents four cases of customized image generation, showing training examples (upper: positive, lower: negative) and images generated with or without rule-based prompts. Rules, highlighted in red, ensure relational constraints are preserved in diverse contexts.

consistently better. Furthermore, we provide qualitative results of customized image generation using our approach in Fig. 3 and 4. The details of implementation are reported in Sec. C.

5 Conclusion

In this work, we study the task of visual logical rule induction by bridging inductive logic programming (ILP) and Multimodal Large Language Models (MLLMs). We propose ILP-CoT, a training-free method to integrate ILP into the Chain-of-Thought (CoT) process of MLLMs. The key technical contribution lies in utilizing MLLM to automatically build ILP tasks with pruned search spaces, and utilizing ILP system to output rules built upon rectified logical facts and formal inductive reasoning. We also propose the task of text-to-image customized generation with rule induction as a potential application for our approach.

Limitations and future work. We discuss two main limitations of this work. First, the perceptual hallucinations of MLLMs increase as the number of subjects in an image grows. In such cases, even though our approach can still detect and reject wrong rules, correct rules may not be successfully discovered. Enabling MLLMs to utilize perceptual tools during reflection could be an effective way to address this issue. Second, a strong benefit of ILP methods is that logically rigorous rules can still be generated even when the rule structure is complicated, such as recursion with long chains. This work has not explored such kind of tasks, which can be a meaningful future work.

AI USE STATEMENT

We used generative AI tools to assist with language editing, improving clarity and readability, limited code-related tasks such as debugging and implementation assistance, and generating synthetic image data. We did not use generative AI tools to generate experimental results, fabricate or select citations, or make scientific claims on behalf of the authors. All AI-assisted text was carefully reviewed and revised by the authors, and AI-assisted code was manually inspected and tested for correctness. The authors take full responsibility for the final content of this work, including all text, claims, results, and artifacts produced with the aid of generative AI.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Black Forest Labs. Flux: Transformer-powered flow models at scale. `https://bfl.ai/announcements/24-08-01-bfl`, 2024.

Hyungjoo Chae, Yeonghyeon Kim, Seungone Kim, Kai Tzu-iunn Ong, Beong-woo Kwak, Moohyeon Kim, Seonghwan Kim, Taeyoon Kwon, Jiwan Chung, Youngjae Yu, et al. Language models as compilers: Simulating pseudocode execution improves algorithmic reasoning in language models. *arXiv preprint arXiv:2404.02575*, 2024.

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.

Andrew Cropper and Sebastijan Dumančić. Inductive logic programming at 30: a new introduction. *Journal of Artificial Intelligence Research*, 74:765–850, 2022.

Daniel Cunnington, Mark Law, Jorge Lobo, and Alessandra Russo. Ffnsl: feed-forward neural-symbolic learner. *Machine Learning*, 112(2):515–569, 2023.

Wang-Zhou Dai and Stephen Muggleton. Abductive knowledge induction from raw data. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 1845–1851, 2021.

Ishita Dasgupta, Andrew K Lampinen, Stephanie CY Chan, Antonia Creswell, Dharshan Kumaran, James L McClelland, and Felix Hill. Language models show human-like content effects on reasoning. *arXiv preprint arXiv:2207.07051*, 2(3), 2022.

Richard Evans, Matko Bošnjak, Lars Buesing, Kevin Ellis, David Pfau, Pushmeet Kohli, and Marek Sergot. Making sense of raw input. *Artificial Intelligence*, 299:103521, 2021.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *ICML*, 2023.

Timin Gao, Peixian Chen, Mengdan Zhang, Chaoyou Fu, Yunhang Shen, Yan Zhang, Shengchuan Zhang, Xiawu Zheng, Xing Sun, Liujuan Cao, et al. Cantor: Inspiring multimodal chain-of-thought of mllm. In *ACM MM*, 2024.

Mengmeng Ge, Xu Jia, Takashi Isobe, Xiaomin Li, Qinghe Wang, Jing Mu, Dong Zhou, Li Wang, Huchuan Lu, Lu Tian, et al. Customizing text-to-image generation with inverted interaction. In *ACM MM*, 2024.

Yuchao Gu, Xintao Wang, Jay Zhangjie Wu, Yujun Shi, Yunpeng Chen, Zihan Fan, Wuyou Xiao, Rui Zhao, Shuning Chang, Weijia Wu, Yixiao Ge, Ying Shan, and Mike Zheng Shou. Mix-of-Show: Decentralized low-rank adaptation for multi-concept customization of diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/hash/3340ee1e4a8bad8d32c35721712b4d0a-Abstract-Conference.html`.

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

Joy He-Yueya, Gabriel Poesia, Rose E Wang, and Noah D Goodman. Solving math word problems by combining language models with symbolic solvers. *arXiv preprint arXiv:2304.09102*, 2023.

Lukas Helff, Wolfgang Stammer, Hikaru Shindo, Devendra Singh Dhami, and Kristian Kersting. V-lol: A diagnostic dataset for visual logical learning. *arXiv preprint arXiv:2306.07743*, 2023.

Lukas Helff, Wolfgang Stammer, Hikaru Shindo, Devendra Singh Dhami, and Kristian Kersting. V-lol: A diagnostic dataset for visual logical learning. *Journal of Data-centric Machine Learning Research*, 2024. URL https://data.mlr.press/assets/pdf/v02-9.pdf.

Joy Hsu, Jiayuan Mao, Josh Tenenbaum, and Jiajun Wu. What's left? concept grounding with logic-enhanced foundation models. *NeurIPS*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Dongwei Jiang, Marcio Fonseca, and Shay B Cohen. Leanreasoner: Boosting complex logical reasoning with lean. *arXiv preprint arXiv:2403.13312*, 2024.

Danial Kamali, Elham J Barezi, and Parisa Kordjamshidi. Nesycoco: A neuro-symbolic concept composer for compositional generalization. *arXiv preprint arXiv:2412.15588*, 2024.

Nupur Kumari, Bingliang Zhang, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Multi-concept customization of text-to-image diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/papers/Kumari_Multi-Concept_Customization_of_Text-to-Image_Diffusion_CVPR_2023_paper.pdf.

Mark Law, Alessandra Russo, and Krysia Broda. Inductive learning of answer set programs. In *ILP*, 2014.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *NeurIPS*, 2020.

Chengshu Li, Jacky Liang, Andy Zeng, Xinyun Chen, Karol Hausman, Dorsa Sadigh, Sergey Levine, Li Fei-Fei, Fei Xia, and Brian Ichter. Chain of code: Reasoning with a language model-augmented code emulator. *arXiv preprint arXiv:2312.04474*, 2023.

Wang Lin, Jingyuan Chen, Jiaxin Shi, Yichen Zhu, Chen Liang, Junzhong Miao, Tao Jin, Zhou Zhao, Fei Wu, Shuicheng Yan, and Hanwang Zhang. Non-confusing generation of customized concepts in diffusion models. *arXiv preprint arXiv:2405.06914*, 2024. URL https://arxiv.org/abs/2405.06914.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. Deductive verification of chain-of-thought reasoning. In *NeurIPS*, 2024.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023a.

Zhiheng Liu, Yifei Zhang, Yujun Shen, Kecheng Zheng, Kai Zhu, Ruili Feng, Yu Liu, Deli Zhao, Jingren Zhou, and Yang Cao. Cones 2: Customizable image synthesis with multiple subjects. In *NeurIPS*, 2023b.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*, 2023.

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. In *NeurIPS*, 2023.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. In *NeurIPS*, 2018.

Chancharik Mitra, Brandon Huang, Trevor Darrell, and Roei Herzig. Compositional chain-of-thought prompting for large multimodal models. *arXiv preprint arXiv:2311.17076*, 2023.

Debjyoti Mondal, Suraj Modi, Subhadarshi Panda, Rituraj Singh, and Godawari Sudhakar Rao. Kam-cot: Knowledge augmented multimodal chain-of-thoughts reasoning. In *AAAI*, 2024.

Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *EMNLP*, 2023.

OpenAI. Hello GPT-4o, 2024a. URL https://openai.com/index/hello-gpt-4o/.

OpenAI. Learning to reason with llms, 2024b. URL https://openai.com/index/learning-to-reason-with-llms/.

OpenAI. Introducing gpt-5, 2025a. URL https://openai.com/index/introducing-gpt-5/.

OpenAI. Introducing gpt-5.2, 2025b. URL https://openai.com/index/introducing-gpt-5-2/.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*, 2023a.

Yuhao Pan, Yu Zhang, Xintao Li, Yingqi Li, Wei Lu, Hongyu Li, Zhi Li, Bo Wang, Wei Wei, and Zhiqiang Yang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*, 2023b. URL https://arxiv.org/abs/2305.12295.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Oliver Ray. Nonmonotonic abductive inductive learning. *Journal of Applied Logic*, 7(3):329–340, 2009.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Rombach_High-Resolution_Image_Synthesis_With_Latent_Diffusion_Models_CVPR_2022_paper.html.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, 2023.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022.

Qingyu Shi, Lu Qi, Jianzong Wu, Jinbin Bai, Jingbo Wang, Yunhai Tong, Xiangtai Li, and Ming-Husan Yang. Relationbooth: Towards relation-aware customized object generation. *arXiv preprint arXiv:2410.23280*, 2024.

Hikaru Shindo, Viktor Pfanschilling, Devendra Singh Dhami, and Kristian Kersting. α ilp: thinking visual scenes as differentiable logic programs. *Machine Learning*, 112(5):1465–1497, 2023.

Hikaru Shindo, Viktor Pfanschilling, Devendra Singh Dhami, and Kristian Kersting. Learning differentiable logic programs for abstract visual reasoning. *Machine Learning*, 113(11):8533–8584, 2024.

Ashwin Srinivasan. The aleph manual. 2001.

Wolfgang Stammer, Patrick Schramowski, and Kristian Kersting. Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3619–3629, June 2021.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Ramakrishna Vedantam, Arthur Szlam, Maximillian Nickel, Ari Morcos, and Brenden M Lake. Curi: A benchmark for productive concept learning under uncertainty. In *International Conference on Machine Learning*, pp. 10519–10529. PMLR, 2021.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

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. In *NeurIPS*, 2022.

Junda Wu, Zhehao Zhang, Yu Xia, Xintong Li, Zhaoyang Xia, Aaron Chang, Tong Yu, Sungchul Kim, Ryan A Rossi, Ruiyi Zhang, et al. Visual prompting in multimodal large language models: A survey. *arXiv preprint arXiv:2409.15310*, 2024.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *NeurIPS*, 2022.

Antonia Wüst, Wolfgang Stammer, Quentin Delfosse, Devendra Singh Dhami, and Kristian Kersting. Pix2code: Learning to compose neural visual concepts as programs. *arXiv preprint arXiv:2402.08280*, 2024.

Xiaomi MiMo Team. Mimo-v2.5. `https://huggingface.co/collections/XiaomiMiMo/mimo-v25`, 2026.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. Faithful logical reasoning via symbolic chain-of-thought. *arXiv preprint arXiv:2405.18357*, 2024.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Satlm: Satisfiability-aided language models using declarative prompting. In *NeurIPS*, 2024.

Xulu Zhang, Xiao-Yong Wei, Wengyu Zhang, Jinlin Wu, Zhaoxiang Zhang, Zhen Lei, and Qing Li. A survey on personalized content synthesis with diffusion models. *arXiv preprint arXiv:2405.05538*, 2024. URL `https://arxiv.org/abs/2405.05538v1`.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023.

A Baseline Comparison Details

The three baselines receive the positive and negative example images, their labels, and the test image. They differ in how the model is instructed to infer and apply a rule. Each baseline produces the final class prediction through the multimodal model.

Direct CoT. The model examines the positive and negative examples, reasons in natural language about the condition that distinguishes them, and decides whether the test image satisfies that condition. The prompt leaves the organization of this reasoning to the model: it does not prescribe separate attribute-extraction, relation-extraction, and rule-induction stages. This baseline measures what the model can infer directly from the labeled images with a general reasoning instruction.

Custom CoT. This baseline breaks visual rule induction into explicit reasoning steps:

1. **Capture-token generation.** Propose the visual attributes and relations that may help explain the positive and negative examples. These capture tokens specify what to look for in the images.
2. **Attribute identification.** Describe the relevant objects and their properties in each example, guided by the proposed tokens. Each property is associated with the object it describes.
3. **Relation identification.** Describe how the relevant objects are related in each example, including which objects participate in each relation.
4. **Rule induction and application.** Compare these observations across the positive and negative examples, express the inferred condition in natural language, and use it to classify the test image.

The intermediate observations help the model organize the visual evidence before drawing a conclusion. The model itself performs the final induction and prediction; the additional structure comes from the sequence of reasoning instructions.

Code CoT. The prompt explicitly suggests using inductive logic programming (ILP) to learn rules from the positive and negative examples and asks the model to express them as Prolog code. This supplies prior guidance about how to approach rule induction. The model identifies the relevant objects, attributes, and relations in the examples and writes predicates and clauses that represent its proposed explanation. It then uses this model-written rule to reason about the test image and returns a class prediction. The multimodal model supplies both the rule and the prediction. In ILP-CoT, by comparison, Metagol searches for concrete programs from the visual facts and proposed meta-rules, and the support-based selector chooses the program used for prediction.

Accuracy. For task t with query set Q_t, $\mathrm{Acc}_t = |Q_t|^{-1} \sum_{q \in Q_t} \mathbf{1}[\hat{y}_q = y_q]$, where y_q is the reference class and $\hat{y}_q$ is the predicted class. Each test image contributes one correct or incorrect prediction. The dataset result is the average of task accuracies, $|T|^{-1} \sum_{t \in T} \mathrm{Acc}_t$, reported as a percentage. For ILP-CoT, these are the predictions of the selected program; for a baseline, they are the model's final answers.

B Support-based program selection

B.1 Candidate evaluation

Let $\mathcal{C}$ contain the executable programs retained from the search rounds. The full support set is $S = S_{\mathrm{fit}} \cup S_{\mathrm{sel}}$. Initial rule proposals and fitting searches use S_{fit}, while S_{sel} contains the positive and negative examples set aside from those initial searches. As described in the method, errors on the full support set can inform later reflection.

For each program h and support image i, we execute h on the image's fact record to obtain $\hat{y}_i(h) \in \{0, 1, ?\}$. For either $D = S$ or $D = S_{\mathrm{sel}}$, accuracy is

$$a_D(h) = \frac{1}{|D|} \sum_{i \in D} \mathbf{1}[\hat{y}_i(h) = y_i],$$

where $y_i = 1$ denotes a positive example and $y_i = 0$ a negative example. An undetermined prediction ? contributes no correct prediction and remains in the denominator. All candidates are thus evaluated on the same images, even when they were learned from different fitting subsets.

For the remaining comparisons, let $r_D^+(h)$ and $r_D^-(h)$ be the fractions of positive and negative examples classified correctly. Balanced accuracy is $b_D(h) = (r_D^+(h) + r_D^-(h))/2$, so both classes receive equal weight. Program length $L(h)$ counts predicate occurrences in the heads and bodies of its clauses. For example, a clause with one head and three body predicates contributes four to $L(h)$.

B.2 Selection order

The selector compares the scores in the order used in the main method:

1. **Full-support accuracy.** Find the highest $a_S(h)$ in $\mathcal{C}$ and retain the programs with accuracy close to that score. This compares rules against all support images, rather than just the subset that produced each rule.
2. **Selection-set accuracy.** Among the remaining programs, find the highest $a_{S_{\text{sel}}}(h)$ and retain those with similar accuracy on S_{sel}. The best score for this step is computed within the candidates that passed the first comparison.
3. **Program length.** Choose a program with the smallest $L(h)$ among those retained by both comparisons. A small score increase therefore need not justify adding more conditions to a rule.
4. **Remaining choices.** If several programs have the same minimum length, compare balanced accuracy on S_{sel} first and on S second. If these also agree, prefer the program with the larger $\min(r_S^+(h), r_S^-(h))$, so the less accurately predicted class is better covered. Exact ties retain the previous selected program when possible; otherwise a fixed ordering of program identifiers resolves the choice.

Here a program identifier is a deterministic hash of its serialized rule representation; it is used only for the final exact tie. The selector runs after each search round. Its chosen program supplies prediction errors for reflection, and the final round's choice produces the query predictions.

The two accuracy comparisons preserve rules that explain the examples similarly before rule length is considered. This is useful when a longer rule gains little accuracy by adding conditions to imperfect visual facts. Full-support accuracy checks agreement beyond each rule's own fitting subset, and selection-set accuracy compares rules on examples excluded from the initial fitting searches.

B.3 Selection ablations and candidate-pool accuracy

Table 2 compares three selection strategies, each with and without failure reflection. With reflection, prediction errors guide revisions to the visual facts and rule proposals. Without reflection, we remove these feedback-driven revisions while retaining visual fact extraction, rule proposals, ILP search, and program selection. The three selection strategies share the same final candidate pool and differ as follows.

Support-based selection. We use the sequence described above: compare full-support accuracy, compare selection-set accuracy among the remaining programs, and then prefer shorter programs. The selected program supplies the final query predictions.

Support-argmax. We choose a program with the highest full-support accuracy $a_S(h)$ and use its query predictions. This removes the second accuracy comparison on S_{sel} and the subsequent choice of a shorter program among programs with similar accuracy. A shorter program cannot replace a more accurate one under this criterion. The comparison therefore tests whether maximizing agreement with the support examples is sufficient for selecting a rule that generalizes.

Uniform choice. We assign equal probability to every executable program in the final candidate pool and report the expected query accuracy:

$$\overline{a}_Q = \frac{1}{|\mathcal{C}|} \sum_{h \in \mathcal{C}} a_Q(h),$$

where $a_Q(h)$ is the query accuracy of candidate h. We compute this expectation by averaging all candidate accuracies, rather than by drawing a single program. We first average within each task and then across tasks. This calculation does not rerun reflection with randomly selected programs; it measures the quality of the final pool when no program is preferred over another.

Thus, the uniform-choice columns measure average candidate quality, whereas the other columns measure the accuracy of the chosen program. Comparing reflection with no reflection within each strategy shows how the corresponding result changes when feedback-driven revision is included. Reflection raises the pool mean on all three datasets, indicating that its benefit extends to the average quality of the available rules.

C Text-to-Image Customization with Rule Induction

To provide a potential application of our visual induction method, we introduce the task of text-to-image customization with rule induction. Most text-to-image customized generation approaches (Ruiz et al., 2023; Zhang et al., 2024) focus on subject customization. Although several research studies investigate multi-subject customization (Kumari et al., 2023; Liu et al., 2023b; Lin et al., 2024; Gu et al., 2023), they do not explicitly induce semantic relationships among subjects from training examples for test-time generation. Recently, there have been attempts to introduce relational constraints in the customization task (Ge et al., 2024; Shi et al., 2024). These studies focus on improving the control ability of pre-defined constraints instead of inducing rules from data. In the rule-based customization task, the instances are labeled as positive and negative ones, potentially by the users, indicating a latent rule to be induced. After fine-tuned customization, the testing-stage generation should follow both the subjects and rules. We design a straightforward baseline for this task in which the proposed ILP-CoT approach is used for rule induction. In the experiments, we show that the baseline method achieves desirable performance in common generation tasks, while the room for improvement is also large, indicating future research in this task.

To enable the automated generation of new, rule-compliant images featuring specific roles, we introduce a mechanism that associates each main role with a unique special token. Concretely, each role is labeled with a token in the format `[v0]`, `[v1]`, `[v2]`, and so on. We employ LoRA (Hu et al., 2021) to fine-tune FLUX (Black Forest Labs, 2024)—specifically adjusting the linear layers in single and double streams as well as the CLIP model (Radford et al., 2021)—so that each special token is mapped to its corresponding role.

ILP-CoT-Customization contains 22 rule-induction tasks and seven separate generation tasks. For each generation task, we first learn its rule from three positive and three negative examples. We then use a scene governed by the same rule but with different details. The scene initially violates the rule, and the generation task is to correct it into a rule-consistent scene. Both prompt conditions use the same task setting and generation setup; the only difference is whether the prompt includes the learned rule to guide the correction.

Semantic segmentation for role isolation. A semantic segmentation model is first used to segment the original image according to its main roles (e.g., a dog or a cat). After segmentation, each patch associated with a main role is paired with its corresponding special token. This pairing allows us to apply LoRA-based fine-tuning on FLUX, wherein we minimize the MSE loss to disentangle the visual features of each special token from those of the other roles. Through this process, each special token becomes distinctly representative of a particular entity.

Rule and token integration. Once the model is fine-tuned, we combine the induced rules with special tokens to generate customized images that satisfy both the learned constraints and the newly introduced narrative details. For example, suppose the two main roles are labeled as `[v0]dog` and `[v1]cat`, and we have a rule stating that `[v0]dog` has golden fur and plays with `[v1]cat`. We can then prompt the MLLM to produce a story-like description—for instance, one that portrays a

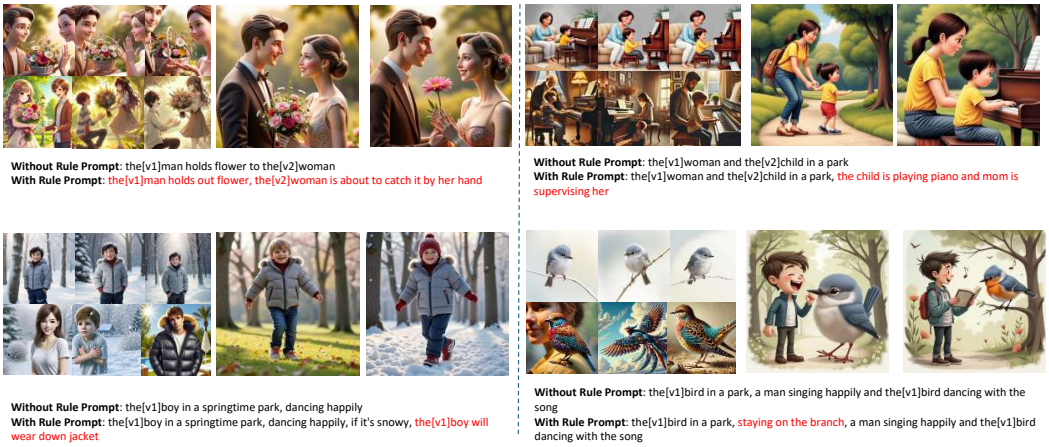

Figure 4: The figure presents four more cases of customized image generation, showing training examples (top: positive, bottom: negative) and images generated with or without rule-based prompts. Rules, highlighted in red, ensure relational constraints are preserved in diverse contexts.

dog and a cat in a moonlit forest beside twisted, ancient trees and a solitary, small flower. We subsequently replace all references to the dog and cat in this description with `[v0]dog` and `[v1]cat`, respectively, and place the rules at the beginning of the description as constraints. This approach is designed to guide the generated image to (1) accurately reflect the roles associated with each token, (2) comply with the rule regarding the dog's golden fur and its interaction with the cat, and (3) integrate the newly described context from the MLLM-generated story.

Rule adherence and visual fidelity. By explicitly linking each main role to a token and restricting the model's understanding of that role via LoRA fine-tuning, we aim to make the final synthesized images respect the rules learned during the ILP phase while preserving key visual characteristics of the original roles. This mechanism helps reduce unwanted alterations (e.g., changing a dog's color or form) and allows us to integrate new contexts or story elements—such as environmental changes—while retaining the rule conditions in the prompt. The goal is to maintain both fidelity to the original subjects and consistency with the high-level narrative details specified through the MLLMs.

Rule-guided prompting strategy. Our generation process begins with a user-generated prompt that describes a scenario, objects, or attributes of interest. Each object mentioned in the prompt is tagged with a special token, denoted as `[v#]`, which was introduced during training to maintain a binding between the textual description and its corresponding visual representation. To merge the user prompt with a learned rule, we prepend or append a concise rule-based statement to the prompt.

For the paired comparison, we keep the scene description and special tokens unchanged. The rule-guided prompt adds the learned rule; the other prompt omits it. Fig. 3 and 4 show qualitative generation results.

D ILP-CoT-Customization Dataset

Dataset generation and composition. We generated 29 different rule-based tasks using images produced by Stable Diffusion (Rombach et al., 2022), designed to evaluate ILP-CoT's ability to abduce visual rules. The dataset consists of:

- **22 induction tasks:** Each task contains 10 images, including 5 positive and 5 negative examples. Each image represents a complete rule independently. These tasks are used for learning visual ILP tasks.

- **7 generation tasks:** Each task contains 3 positive and 3 negative examples. The main subjects in all positive examples maintain the same appearance and style. These tasks are designed for customization based on rules.

The 29 tasks encompass a diverse range of rule-based relationships, including relationships between a single primary subject and a theme, relationships between multiple primary subjects, relationships between a single primary subject and background characters, and relationships involving multiple primary subjects and background characters. The dataset further includes:

- **Spatial relation tasks:** These tasks focus on relative spatial positioning, such as left/right, above/below, etc.
- **Attribute association tasks:** These tasks require the model to capture associations between attributes (e.g., color, category) and objects, such as "The cat likes the golden dog".
- **Role interaction tasks:** For example, "The mother is holding the child, and the father is watching the child," requiring the model to understand interactions between roles.
- **Environmental response tasks:** Such as "The sunflower faces the sun", testing whether the model can infer how objects respond to environmental changes.

Positive and negative example generation strategy. In the dataset generation process, a unique predefined rule is used to determine positive examples. For negative examples, we randomly select one or more conditions from the rule and invert them, ensuring that at least one condition is violated. This approach guarantees that positive examples strictly follow the predefined rule, while negative examples systematically deviate from it, thereby providing a challenging and diverse dataset for rule induction.

Evaluating criterion. Each task is evaluated under four different data settings: 1 positive + 1 negative example; 3 positive + 3 negative examples; 5 positive + 1 negative example; and 5 positive + 5 negative examples. Rules are categorized into four levels of accuracy: Correct, Mostly Correct, Partially Correct, and Incorrect. Two human evaluators and two AI evaluators were invited to assess the generated rules. The evaluation process involved:

1. The evaluator is presented the groud-truth rule for each task.
2. Evaluators analyzing whether the generated rule precisely describes all positive examples while excluding negative ones.
3. Evaluators scoring the rule independently, without additional hints.

Each task was repeated five times, and the final scores were averaged.