Symbolic Grounding Reveals Representational Bottlenecks in Abstract Visual Reasoning

Mohit Vaishnav **Tanel Tammet**
Applied Artificial Intelligence Group, Tallinn University of Technology, Estonia 12169
mohit.vaishnav@taltech.ee

Abstract

Vision–language models (VLMs) often fail on abstract visual reasoning benchmarks such as Bongard problems, raising the question of whether the main bottleneck lies in reasoning or representation. We study this on Bongard-LOGO, a synthetic benchmark of abstract concept learning with ground-truth generative programs, by comparing end-to-end VLMs on raw images with large language models (LLMs) given symbolic inputs derived from those images. Using symbolic inputs as a diagnostic probe rather than a practical multimodal architecture, our *Componential–Grammatical (C–G)* paradigm reformulates Bongard-LOGO as a symbolic reasoning task based on LOGO-style action programs or structured descriptions. LLMs achieve large and consistent gains, reaching mid–90s accuracy on Free-form problems, while a strong visual baseline remains near chance under matched task definitions. Ablations on input format, explicit concept prompts, and minimal visual grounding show that these factors matter much less than the shift from pixels to symbolic structure. These results identify representation as a key bottleneck in abstract visual reasoning and show how symbolic input can serve as a controlled diagnostic upper bound.

1 Introduction

Vision–language models (VLMs) have achieved strong results on captioning, retrieval, and recognition from large-scale paired data (Radford et al., 2021; Jia et al., 2021; Ramachandran et al., 2025), yet they often fail on tasks that require abstract reasoning, compositional generalization, or spatial inference (Yuksekgonul et al., 2022; Parascandolo et al., 2025; Li et al., 2025). Analyses of CLIP-style encoders suggest that end-to-end architectures may overemphasize salient objects or early caption tokens while underrepresenting relational and structural information (Fine and Abend, 2025; Kang and Choi, 2023; Campbell et al., 2024). This raises a basic question: is the main bottleneck in current multimodal models their *reasoning* mechanisms, or the *representations* that feed those mechanisms?

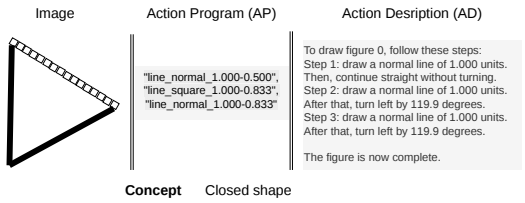


Figure 1: Example Bongard-LOGO instance under the Componential–Grammatical interface. From left to right we show the rendered shape, its action-program (AP) representation, and its action-description (AD) rendering. In the concept-conditioned variant, the prompt additionally provides the high-level concept label, allowing us to compare visual, procedural, and natural-language interfaces for the same underlying example.

We address this question in a controlled synthetic setting: **Bongard-LOGO** (Nie et al., 2020), a benchmark of concept-learning tasks inspired by classic Bongard problems (Bongard, 1970; Foundalis, 2006). Each problem presents positive and negative example images and requires inferring a latent rule that separates them. Figure 1 previews the three interfaces used in our experiments: the rendered image itself, its procedural action program, and its natural-language action description. Unlike conventional benchmarks, Bongard-LOGO emphasizes geometric, relational and topological properties–symmetry, convexity, connectivity–rather than object semantics, and exposes the ground-truth procedural LOGO program that generates each image.

Classic work on Bongard problems argues that solving them requires a *visual language* of structured primitives and relations; Bayesian and causal neuro-symbolic systems define such vocabularies and inference procedures, achieving strong performance on hand-designed sets while remaining in-

Proceedings of the 30th Conference on Computational Natural Language Learning, pages 318–343

terpretable (Depeweg et al., 2024; Grinberg et al., 2023). We follow this line but replace bespoke engines with general-purpose language models as the reasoning backend.

We introduce the **Componential–Grammatical (C–G)** paradigm: a modular interface that decouples perception from reasoning by posing Bongard-LOGO as a symbolic reasoning problem. Instead of operating on pixels, models receive the underlying *action programs* that generated the images or structured textual descriptions derived from them, forming a LOGO-style visual language. A language model then induces a rule and classifies query examples from symbolic input alone. This setup lets us ask: to what extent do recent language models exhibit abstract generalization when supplied with explicit symbolic structure, and how do representation format, concept conditioning, and minimal visual grounding modulate their performance?

Empirically, a strong VLM baseline (Gemini-2.5-Flash) performs near chance (∼50%) on Bongard-LOGO, whereas the same tasks reframed via symbolic input yield substantially higher accuracies (often 60–80% across models, with Phi-4-Reasoning reaching 96.2% on Free-form problems). The main driver of this gap is representational: compositional, interpretable structure enables generalization, while variations in format, concept prompts, and grounding yield smaller, model-dependent effects. Although symbolic input is not a realistic perceptual modality, we treat C–G as a diagnostic upper bound on what current models can do once grounded in an appropriate visual language, rather than as a complete solution to perception.

Our goal is diagnostic rather than architectural. We do not introduce a new end-to-end multimodal system or learned visual front-end; instead, we vary the *representational interface* between perception and reasoning on Bongard-LOGO. By comparing the same family of language models under procedural programs, natural-language descriptions, concept prompts, and structure-breaking perturbations, we isolate how input format and inductive bias shape abstract generalization independently of vision encoder choice.

Concretely, our contributions are threefold. *First*, we present the Componential–Grammatical (C–G) interface for Bongard-LOGO and show that representational format alone can induce 25–30 point gains over a strong visual baseline across 12 models. *Second*, we systematically probe this interface via concept conditioning, minimal-context prompting, grounded variants, and token randomization controls, revealing when and how symbolic structure—rather than surface statistics—drives success. *Third*, we provide a cross-model analysis of capacity and brittleness under these interfaces, positioning C–G as a diagnostic tool for studying representational bottlenecks in multimodal reasoning.

While prior work has shown that specialized symbolic solvers perform well on Bongard problems, it is not obvious that general-purpose pretrained LLMs can induce abstraction rules directly from procedural symbolic traces without domain-specific search or operators. Our experiments show that once grounded in an explicit visual language, modern LLMs can function as effective abstract inference engines.

2 Related Work

Vision–Language Models and Grounding. Modern VLMs such as CLIP and ALIGN learn joint image–text representations and excel at recognition and retrieval (Radford et al., 2021; Jia et al., 2021; Ramachandran et al., 2025), but benchmarks targeting compositional generalization, spatial reasoning, and relational understanding reveal systematic failures (Yuksekgonul et al., 2022; Ma et al., 2023; Hu et al., 2023; Hsieh et al., 2023). These models often prioritize coarse object or token statistics, with limited sensitivity to attribute binding, part–whole structure, or word order, and analyses trace this to architectural and training biases such as overreliance on early caption tokens or spatial salience (Fine and Abend, 2025; Kang and Choi, 2023; Parascandolo et al., 2025; Campbell et al., 2024). Recent work introduces more explicit intermediate structure, via multi-granular alignment between sentence parts and image regions (Le et al., 2025), generative-flow-style chain-of-thought (CoT) reasoning over images (Kang et al., 2025), or program-like plans over symbolic state (Xu et al., 2025). In contrast, we change only the *input format*—from pixels to LOGO-style symbolic programs—and ask how this affects abstract generalization in language models.

Bongard Problems and Visual Abstraction. Bongard problems are classic visual reasoning tasks where a rule must be induced to separate positive from negative examples (Bongard, 1970; Hofstadter, 1979; Foundalis, 2006). They empha-

size geometric and topological concepts such as symmetry, containment, and convexity, and have informed work on analogy and visual concept learning (Gentner and Holyoak, 1997; Gentner and Hoyos, 2017). Bongard-LOGO (Nie et al., 2020) extends this paradigm into a synthetic benchmark of 12,000 procedurally generated problems from ground-truth LOGO programs. Despite their visual simplicity, these tasks are difficult for current models: even advanced systems like GPT-4o and Gemini attain only 50–60% accuracy on abstract Bongard-style sets, far below human performance (Wüst et al., 2024). Several systems treat Bongard problems as inference in an explicit visual language: Depeweg et al. (2024) define a symbolic vocabulary and use Bayesian inference with pragmatic constraints, and Bongard Architecture implements a causal, operator-based neuro-symbolic system that solves nearly all Maksimov–Bongard puzzles while remaining interpretable (Grinberg et al., 2023). Bongard-in-Wonderland shows that frontier foundation models still often fail on simple geometric concepts like spirals (Wüst et al., 2024). Our work follows this tradition in using a structured visual language, but replaces bespoke inference engines with general-purpose LLMs and leverages Bongard-LOGO's generative programs as the underlying representation.

Symbolic Interfaces and Neuro-Symbolic Models. Our approach fits within a broader effort to modularize perception and reasoning through structured interfaces. Componential Analysis (CA) (Vaishnav and Tammet, 2025) uses a VLM to generate natural-language scene descriptions that an LLM then uses for classification or inference; this works well on natural-image Bongard datasets such as Bongard-OpenWorld (Wu et al., 2024), but performs poorly on Bongard-LOGO, where precise relations are hard to verbalize (Peng et al., 2024). Inverse-graphics and shape-grammar approaches similarly emphasize program-like representations of visual structure (Stiny and Gips, 1972; Gips, 1975; Knight and Stiny, 2015; Kulkarni et al., 2015; Yildirim et al., 2020; Kulits et al., 2024). For Bongard-LOGO, probabilistic concept models combined with Sinkhorn distances over symbolic shapes (Song and Yuan, 2024) and ARC-style neuro-symbolic agents that search in task-specific languages (Batorski et al., 2025) illustrate the value of explicit languages as grounding interfaces. Our Componential–Grammatical (C–G) interface instantiates this pattern for LOGO-based Bongard problems with LLMs as the reasoning backend and allows us to evaluate how procedural programs versus descriptive text modulate downstream reasoning performance independently of perceptual noise.

3 Method

Bongard-LOGO as a Diagnostic Benchmark. We adopt the LOGO benchmark (Nie et al., 2020) as a controlled testbed for studying the interface between perception and reasoning. Each of its 12,000 problems consists of six positive and six negative example images, along with a query image to classify. All images are procedurally generated from known LOGO programs, enabling precise control over visual properties. Following the released benchmark splits, we report results for *Free-form* (FF), *Basic* (BD), and two *Human-designed* evaluation splits: HD-Comb and HD-Novel. FF concepts are defined by stroke sequences, BD concepts by reusable shape categories and their combinations, and the two HD splits test transfer to human-designed shapes under combinatorial and novel-shape regimes. Unless otherwise noted, HD denotes the mean of HD-Comb and HD-Novel.

Symbolic Representation via Procedural Grammars. Unlike typical image benchmarks, Bongard-LOGO provides access to the full generative trace for each image. Each figure is constructed from a hierarchy of LOGO-like drawing instructions: a `BongardImage` comprises one or more `OneStrokeShape` objects, each built from `BasicAction` primitives (e.g., `Line`, `Arc`). We encode these structures as compact grammar strings that preserve the sequence of operations, angle information, and shape identifiers. For instance, a triangle might be represented as a sequence of three equal-length line segments with 120-degree turns between them. These symbolic traces form a procedural shape grammar that is both noise-free and compositional, aligning with inverse-graphics views of visual processing (Stiny and Gips, 1972; Knight and Stiny, 2015; Kulkarni et al., 2015; Kulits et al., 2024).

Componential–Grammatical Paradigm (C–G). We introduce the **Componential–Grammatical (C–G)** paradigm, a modular pipeline that separates visual structure recovery from downstream reasoning. C–G operates in two stages:

- **Symbolic Grounding:** Each image in a Bongard-LOGO problem (positives, negatives, and query) is represented as a symbolic input—either a procedural grammar string or a descriptive natural-language version—using its ground-truth program.
- **Textual Reasoning:** A language model receives only the symbolic inputs and must infer a rule and classify the query as positive or negative. No pixel data or visual features are provided.

This setup transforms the task from multimodal perception-plus-reasoning to a purely symbolic inference problem. Crucially, it allows us to isolate the role of input representation in driving model behavior. If LLMs perform well in the symbolic setting but VLMs fail on raw images, this pattern points to a representational or grounding bottleneck rather than a lack of reasoning capacity.

4 Experiments

We design our experiments to answer three questions: (i) do symbolic representations unlock abstract visual reasoning that VLMs fail to exhibit on Bongard-LOGO, (ii) how does the *form* of symbolic input (programs vs. descriptions, with or without concepts and images) modulate performance, and (iii) do models rely on *rule-like structure* rather than exploiting token-frequency or lexical-overlap heuristics?

Models. We evaluate proprietary Gemini models (Google DeepMind, 2024), using Gemini-2.5-Flash as the primary end-to-end visual baseline and including Gemini-3-Flash in the supplementary model-capacity analysis. We also evaluate open-source models including DeepSeek-R1 (8B, 14B, 32B) (Guo et al., 2025), Phi-4 and Phi-4-Reasoning (14B) (Microsoft Research, 2024), Qwen2.5 and Qwen3 (up to 32B) (Yang et al., 2025), Gemma 3 (12B, 27B), and Mistral Magistral. Open models are run locally using the Ollama framework. Refer to Appendix B for more details.

Evaluation Protocol. Each Bongard-LOGO problem contains 6 positive examples, 6 negative examples, and 1 query image (detailed in Appendix A). Models must predict whether the query belongs to the positive or negative class. Accuracy is computed over a fixed 2,000-problem subset containing 500 problems from each reported evaluation split: BD, FF, HD-Comb, and HD-Novel. The subset is drawn uniformly at random from the test split without any performance-based filtering, and all models are evaluated on the same subset for comparability. Open-source models use deterministic decoding (temperature 0.0, no sampling); Gemini models use low-temperature official modes. Outputs are parsed from a constrained JSON format containing a reasoning trace, induced rule, and binary label; unparsable outputs are counted as errors. Exact prompts, model identifiers, and infrastructure details appear in Appendix C,D.

4.1 Main Conditions: From Pixels to Symbols

Our primary comparison contrasts an end-to-end visual baseline with variants of the Componential–Grammatical (C–G) interface, where models operate on symbolic input only.

Visual VLM Baseline. Gemini-2.5-Flash receives all 13 *images* and directly predicts the query label. This represents a standard end-to-end VLM pipeline in which perception and reasoning are tightly coupled.

C–G (Formal Grammar). Models receive only procedural shape grammar strings (as shown in Appendix A.3) derived from the ground-truth LOGO programs for all 13 images, with no pixel input. This condition tests whether explicit, compositional structure alone enables high-accuracy rule induction and classification.

C–G (Natural Language). Instead of grammar strings, models see step-by-step English descriptions of the drawing actions and geometric relations as seen in Appendix A.4. This probes whether recasting the same structure in familiar linguistic form helps or harms abstract reasoning relative to a compact formal syntax.

4.2 Ablations: How Much Structure and Supervision?

We next vary the amount and form of guidance around the symbolic input.

Minimal-Context Grammar. Models receive raw grammar strings plus a brief instruction describing the task, but no explanation of the formalism itself. This few-shot setting asks whether models can infer a novel syntactic interface and the intended rule-induction task from examples alone.

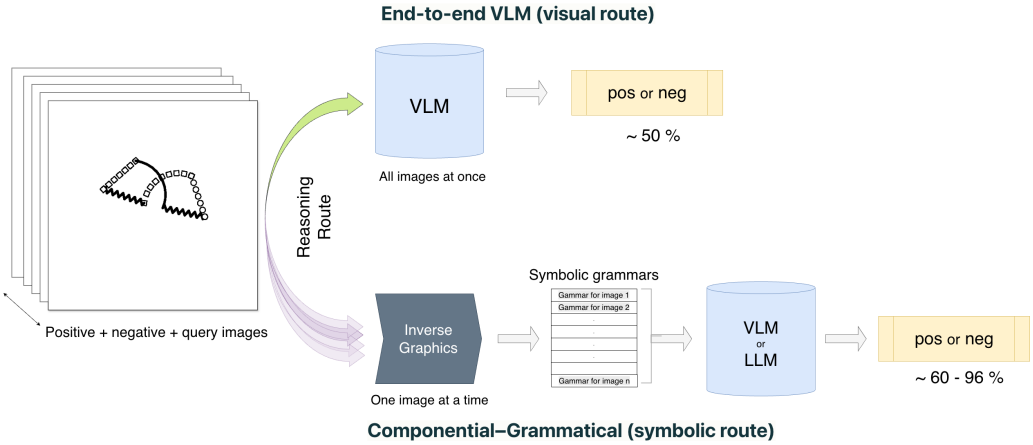


Figure 2: Comparison of visual and symbolic pipelines on Bongard-LOGO. In the *visual route* (top), a Vision–Language Model (VLM) receives images directly and must infer both structure and rules, typically yielding near-chance accuracy. In the *symbolic route* (bottom), problem instances are represented by LOGO-style action programs; a language model then performs rule induction and classification using these symbolic representations, achieving substantially higher performance on the same task.

Concept-Conditioned C–G. The high-level target concept (e.g., "is convex", "contains a triangle" Appendix A.5) is provided in the system prompt alongside the action programs. This converts the task from rule discovery to rule verification and tests whether explicit concept supervision provides additional gains over purely example-based learning.

Grounded C–G. Models receive symbolic programs for all 13 images plus the *rendered query image*. This condition probes whether a single visual anchor, tied to otherwise symbolic input, yields further improvements beyond the purely symbolic setting. Because it requires image input, we restrict this variant to VLMs that accept both text and images (Gemma3:12B, Gemma3:27B, LLaVA-Llama3, and Qwen2.5-VL:32B).

4.3 Structure vs. Surface: Token Randomization Tests

Finally, we test whether models rely on genuine rule learning over symbolic structure rather than superficial token matching.

Randomizing Query Programs. In the first control, we randomly permute the sequence of drawing actions in the *query* program while keeping the support examples, the query image, and its label unchanged. If models primarily exploit token inventory alone, performance should remain similar; if they rely on structured comparisons between query and support examples, accuracy should degrade once the ordered composition of the query is broken.

Shuffling Positive and Negative Sets. In the second control, we randomly reassign the 12 support examples between the positive and negative sets while leaving the query program, query image, and query label untouched. This preserves within-problem token inventories but destroys the original support-set partition. Performance therefore need not fall exactly to 50%: after shuffling, the support sets can still instantiate alternative regularities that partially align with the unchanged query. The diagnostic signal is the drop relative to the unperturbed Action Program condition.

We report these controls only for BD and FF. HD performance is already comparatively low in the unperturbed setting, so the same perturbations would be less informative there.

Together, these perturbation studies test whether high C–G performance depends on relational and compositional structure in the symbolic input, rather than on simple token-frequency heuristics alone.

5 Results

We report findings from systematic evaluation of the Componential–Grammatical (C–G) paradigm across 12 models and multiple input regimes. Our core result is consistent: when models receive structured symbolic input in place of raw perceptual input, performance improves dramatically, especially on conceptually grounded reasoning tasks.

Table 1: Summary of key results across experimental conditions on Bongard-LOGO. The first five rows and the randomization rows are mean accuracy (%) across 12 language models. Rows marked with † are averaged over the matched 4-model VLM-capable subset (Gemma3 12B/27B, LLaVA-Llama3, Qwen2.5-VL 32B) and should be compared only within that subset. 'Base' denotes the minimal-context symbolic baseline. AP = Action Program (formal LOGO-style grammar), AD = Action Description (natural-language description of actions).

Condition	FF	BD	HD
Visual VLM (Gemini-2.5)	50.2	49.8	50.1
C–G (AP)	78.1	70.8	61.0
C–G (AD)	79.3	**73.0**	59.1
C–G + Concept (AP + C)	**79.3**	70.0	**61.1**
Minimal-Context Grammar	77.3	67.7	59.6
Visual baseline (VLM subset)†	62.6	57.1	57.5
Grounded C–G (AP)†	62.4	55.3	58.2
Grounded C–G (AD)†	61.2	55.3	55.8
Category Permutation	70.6	60.1	–
Sequence Permutation	58.0	64.2	–

5.1 Main Effects of Symbolic Grounding

Table 1 reports mean accuracy across three benchmark categories and key experimental conditions. Gemini-2.5-Flash achieves chance-level performance (50–51%) across the board when operating on raw images. In contrast, when provided with symbolic input (Action Programs), models reach up to 96.2% accuracy (Phi-4-Reasoning, Free-form), with average gains of roughly 20–30 points on BD and FF and smaller, but still substantial, gains on HD over the visual baseline.

These results confirm that models' poor performance in the visual setting stems primarily from representational limitations rather than an inability to reason over structured input. The pooled symbolic conditions improve markedly over the visual-only Gemini baseline, and the matched four-model subset shows that adding a single query image does not recover additional gains once the same models already receive symbolic structure. Across conditions, FF is easiest to recover from symbolic input, BD is intermediate, and HD remains the most challenging.

Beyond overall accuracy, we also examine how models treat positive vs. negative examples under the AP interface. Across datasets, accuracy on positives substantially exceeds accuracy on negatives, and a chi-square test confirms a strong dependence of correctness on example class (see Appendix E.7 for details). This indicates that models adopt a liberal decision criterion that favors "positive" classifications: they are more reliable at recognizing configurations that satisfy a learned rule than at rejecting heterogeneous counterexamples.

5.2 Descriptive vs. Procedural Input

When symbolic structure is rendered in natural language, pooled accuracy is higher than AP on FF and BD, while AP retains a modest advantage on HD. Figure 3 makes the direction of that difference more explicit: relative to the minimal-context symbolic baseline, AD yields larger mean gains on FF and BD (+2.0 and +5.3 points), whereas on HD the mean AD shift is slightly negative (-0.5) and AP remains modestly positive (+1.4). This pattern is not driven by a single outlier model: the point clouds show that AD is broadly competitive on the easier FF and BD regimes, whereas AP is slightly more stable on the human-designed regime (Table 1).

5.3 Concept Conditioning and Minimal Guidance

Providing models with the target rule label (AP + C) yields modest but heterogeneous effects. Relative to AP alone, the mean change is +1.2 points on FF, -0.8 on BD, and near zero on HD (+0.1), but Figure 3 shows that these averages mask mixed model-level behavior rather than a universal improvement. The appendix tables clarify why: some larger models show positive average gains under concept conditioning, whereas smaller or more instruction-sensitive models exhibit mixed or even negative deltas (Tables A.4 and A.6). We therefore treat AP + C as a model-dependent prompt variant rather than a uniformly stronger condition. The same pattern appears in the minimal-context setting: a shorter prompt can outperform the full prompt for some models, suggesting that additional guidance sometimes introduces distraction or prompt overload rather than useful supervision.

5.4 Visual Grounding Offers Little Added Benefit

Within the matched VLM-capable subset, adding a single visual anchor in the Grounded C–G setting leaves accuracy essentially unchanged relative to the corresponding symbolic-only baseline and slightly below the same subset's visual baseline on FF and BD (Table 1; see Appendix E.5). The main result is therefore not that query grounding helps a little, but that once symbolic structure is already

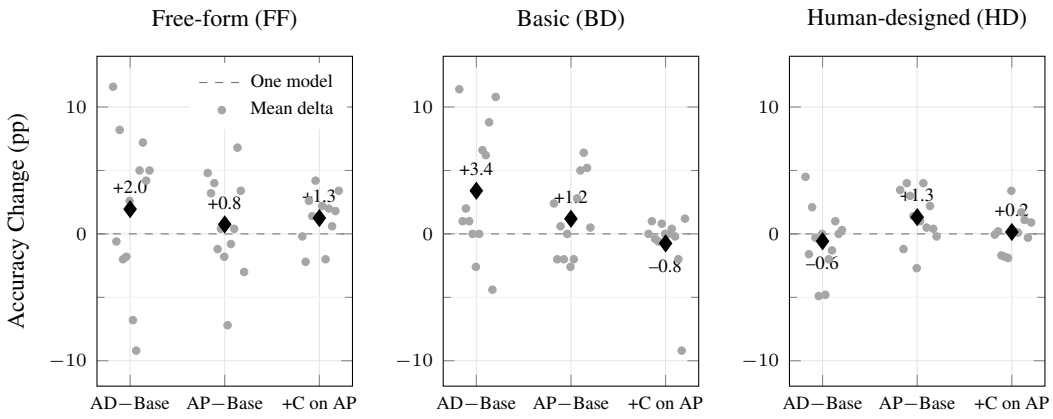


Figure 3: Per-model percentage-point change under symbolic interventions, aggregated from Table A.3. Each gray dot is one model; black diamonds mark mean deltas. 'AD−Base' and 'AP−Base' measure gains over the minimal-context symbolic baseline, while '+C on AP' isolates the incremental change from AP to AP+C. The zero line marks no change. This makes the reviewer-facing pattern explicit: Action Descriptions help most on FF and BD, AP is slightly more stable on HD, and concept conditioning yields modest but heterogeneous model-level effects.

available, a single additional image does not unlock further gains.

5.5 Randomization Controls: Sensitivity to Structure

To test whether models rely on structured rule induction rather than token-level heuristics, we introduce two perturbations in the Action Program regime.

In *Category Permutation*, we randomly reassign support examples between positive and negative sets within each problem, while keeping the query and its gold label fixed. This manipulation preserves token inventories but breaks the original support partition. Performance remains above 50% on both BD and FF, which is expected under this protocol because the unchanged query can still match alternative regularities induced by the shuffled support sets; the relevant diagnostic is the drop relative to the unperturbed AP condition. Under that comparison, FF still drops by about 7 points and BD by about 9 points (Table 1).

In *Sequence Permutation*, we shuffle the order of drawing actions in the query program while leaving the support examples, query image, and gold label intact. BD accuracy drops by about 5 points relative to AP, but FF drops by about 20 points, showing that ordered stroke composition matters most in the Free-form regime.

The two perturbations affect Basic and Free-form tasks in systematically different ways. On Basic (BD) problems, shuffling the assignment of support programs to positive vs. negative sets reduces pooled C–G AP accuracy from 68.8% to 60.1% (–8.7 points), whereas permuting the action sequence in the query program yields a smaller drop to 64.2% (–4.6 points), suggesting that coherent support-set structure is more critical than exact stroke order for these library-based shapes. In contrast, on Free-form (FF) problems, where concepts depend more directly on stroke-level geometry, query sequence permutation is far more damaging: accuracy falls from 78.1% to 58.0% (–20.1 points), compared to 70.6% (–7.5 points) under category permutation (more in Appendix E.6).

6 Discussion

Our results on LOGO reveal a strong dependence of abstract visual reasoning performance on representational format. VLMs operating on raw pixels perform near chance on these Bongard-style tasks, whereas the same underlying reasoning engines, when supplied with structured symbolic input, achieve large gains that, on some categories, approach reported human performance from prior work. This shift, together with the minimal impact of adding a visual anchor and the sharp degradations under structural randomization, highlights a representational bottleneck on Bongard-style tasks: current visual encoders, though successful at object recognition and retrieval, fail to capture the precise geometric and relational structure required for Bongard-style abstraction.

The Componential–Grammatical (C–G)

paradigm isolates this representational gap. Symbolic input acts as a grounding scaffold: it preserves object identity, shape structure, and compositional relations that are typically lost or blurred in visual embeddings. Importantly, these representations are interpretable and amenable to structured manipulation, echoing earlier proposals to treat Bongard problems as inference in a visual language of primitives and relations (Depeweg et al., 2024; Grinberg et al., 2023), and our grounded C–G experiments show that, within the VLM-capable subset, once such a scaffold is in place additional visual evidence offers at best marginal benefits.

We also observe that input format interacts with task complexity. Natural-language descriptions offer gains on simple categorization tasks (e.g., "contains a triangle") but degrade on more abstract tasks that require global structure (e.g., symmetry or convexity). Procedural grammars, by contrast, support more reliable reasoning on complex problems. The perturbation results mirror this pattern: BD accuracy is more sensitive to shuffling which programs belong to the positive vs. negative sets, whereas FF accuracy collapses primarily when we disrupt the ordered composition of strokes in the query program, indicating that different concept classes rely on different facets of symbolic structure. These format-specific effects suggest that there is no universal "best" representation; alignment between representational form and task structure is critical. Model behavior varies as well. Larger models show strong relative gains from symbolic input but are sensitive to verbosity and format. Smaller, instruction-tuned models can sometimes underperform when exposed to over-structured prompts, highlighting brittleness to task framing (more in Appendix E.4). Overall, symbolic inputs help most when they preserve structure without overloading the model with linguistic noise, and the randomization controls indicate that it is precisely this relational and compositional structure—rather than superficial token frequencies—that models rely on, especially for stroke-level Free-form tasks.

These results support a modular view of multimodal AI: instead of relying solely on end-to-end systems, it may be more effective to explicitly separate perception from reasoning, using intermediate symbolic representations as a grounding interface. This design pattern is already common in domains such as formal mathematics and program synthesis, and is increasingly reflected in neurosymbolic visual systems: SBSD models Bongard-LOGO concepts via probabilistic concept models over symbolic shape descriptors (Song and Yuan, 2024), NSA solves ARC tasks by combining neural proposals with search in a domain-specific language (Batorski et al., 2025), agent architectures such as VLAgent plan executable programs from visual input before calling neural modules for execution (Xu et al., 2025), and PRISM explicitly decouples perception and reasoning in VLM pipelines by converting visual information into a textual intermediate representation before downstream inference (Qiao et al., 2024). Our findings extend this pattern to large language models on Bongard-LOGO, suggesting that general-purpose LLMs function effectively as inference engines once grounded in an appropriate visual language. In this sense, C–G should be viewed as a diagnostic upper bound on what current LLMs can do once grounded in an appropriate visual language, rather than as a proposal for solving perception.

Work on compositional visual reasoning in VLMs points to a complementary axis of progress: unrolling reasoning into multi-step procedures. Progressive alignment methods explicitly model multi-granular interactions between language and image regions, and GFlowVLM shows that CoT style multi-step reasoning improves performance on complex visual tasks (Le et al., 2025; Kang et al., 2025). Neurosymbolic agents similarly decompose queries into symbolic subgoals and verify intermediate results (Xu et al., 2025). Our experiments are orthogonal in that, without changing model internals or prompting them for explicit chains of thought, we demonstrate that simply changing the *representation format* — from pixels to LOGO-style programs or descriptions—already unlocks large gains. Together, these lines of work suggest that multi-step reasoning mechanisms and better grounding interfaces are complementary: CoT procedures operate over whatever representations they are given, and our results indicate that providing a suitable visual language is a powerful lever for improving downstream reasoning.

At the same time, our oracle symbolic interface should not be conflated with natural-image reasoning. In natural-image settings, the intermediate representation itself must be inferred from pixels, typically with noise and partial coverage. Recent representation-first pipelines such as Componential Analysis (Vaishnav and Tammet, 2025) and PRISM (Qiao et al., 2024) point toward this direction by

first constructing textual or structured world models and then reasoning over them; natural-image Bongard benchmarks such as Bongard-OpenWorld and Bongard-HOI provide testbeds for whether such approximate symbolic interfaces transfer beyond synthetic geometry (Wu et al., 2024; Jiang et al., 2022). Our results therefore best support a conditional claim: if a model is given a faithful symbolic abstraction, its downstream reasoning can be much stronger than its raw-pixel performance suggests.

Finally, recent "Bongard in Wonderland" evaluations show that even frontier foundation models (e.g., GPT-4o) often fail on elementary Bongard concepts such as spirals (Wüst et al., 2024). Alongside our symbolic upper bounds and prior visual-language solvers (Depeweg et al., 2024; Grinberg et al., 2023), this convergence of evidence supports a common conclusion: robust abstract visual reasoning on Bongard-style benchmarks is unlikely to emerge from monolithic end-to-end VLMs alone, and instead appears to require explicit representational structure, whether via hand-engineered operators, probabilistic visual languages, or program-like grounding interfaces for LLMs.

7 Limitations

While the C–G approach reveals useful structure–reasoning interactions, it comes with several limitations (expanded in Appendix F). *First*, Bongard-LOGO is a synthetic benchmark tailored to abstract geometry, which makes it ideal for perturbation studies (grounding, randomization) but leaves open how well our findings about symbolic grounding transfer to natural-image domains such as Bongard-OpenWorld or Bongard-HOI, where the intermediate representation must itself be inferred from cluttered scenes (Wu et al., 2024; Jiang et al., 2022; Vaishnav and Tammet, 2025). *Second*, our setup assumes oracle access to ground-truth action programs, effectively removing perception from the loop; in realistic settings such programs would need to be inferred from pixels via inverse graphics or symbolic detectors, and our grounded C–G variants only partially probe this gap, since they still rely on perfect symbolic inputs and add only a single visual anchor. *Third*, because we bypass perception entirely, C–G is diagnostic rather than prescriptive: it shows what current models can do *given* suitable structure, but not how to learn such representations, pointing to the need for learned front-ends that induce approximate programs or scene graphs from raw data. *Fourth*, our model set, though broad, does not systematically cover architectures with explicit geometric inductive biases, so our observations about capacity and brittleness may not generalize uniformly. Finally, even with perfect symbolic input, models still struggle on Human-designed problems involving global structure and multiple abstraction levels, echoing results from ARC and related benchmarks that reasoning over rich concepts remains challenging even for neurosymbolic systems. Inferring symbolic structure from perception, aligning it with downstream tasks, and scaling robust reasoning over such representations remain important directions for future work.

8 Conclusion

We have used Bongard-LOGO as a controlled testbed to study how representation shapes abstract visual reasoning in large language models. Comparing end-to-end VLMs on images to the same reasoning engines operating on ground-truth symbolic programs suggests that, on this benchmark, a large portion of the performance gap is attributable to perceptual representations rather than to a lack of reasoning capacity. The Componential–Grammatical (C–G) interface, together with grounded and randomization variants indicates that explicit symbolic structure is the main driver of gains in our experiments, while additional visual anchoring and surface-level token statistics play only a minor role. These findings support a modular view of multimodal AI in which learned or engineered front-ends extract structured visual languages that general-purpose LLMs can then reason over. Extending this approach beyond synthetic settings and developing robust mechanisms for acquiring such representations from pixels remain key directions for future work.

Acknowledgements

This work was supported by the Estonian Centre of Excellence in Artificial Intelligence (EXAI) under grant TK213, funded by the Estonian Ministry of Education and Research. The authors also acknowledge the High Performance Computing Centre of Tallinn University of Technology (TalTech) for providing the computational resources necessary for this research.

References

Paweł Batorski, Jannik Brinkmann, and Paul Swoboda. 2025. Nsa: Neuro-symbolic arc challenge. *arXiv preprint arXiv:2501.04424*.

Mikhail M Bongard. 1970. *Pattern Recognition*. Hayden Book Company, Rochelle Park, NJ.

Declan Campbell, Sunayana Rane, Tyler Giallanza, C Nicolò De Sabbata, Guillermo Ortiz-Jiménez, Pascal Frossard, and Olga Russakovsky. 2024. Understanding the limits of vision language models through the lens of the binding problem. *Advances in Neural Information Processing Systems*, 37.

Stefan Depeweg, Contantin A Rothkopf, and Frank Jäkel. 2024. Solving bongard problems with a visual language and pragmatic constraints. *Cognitive Science*, 48(5):e13432.

Idan Fine and Omri Abend. 2025. A fine-grained analysis of multi-object representation in clip encoders. *arXiv preprint arXiv:2502.19842*. Submitted to ICLR 2025.

Harry E Foundalis. 2006. Phaeaco: A cognitive architecture inspired by bongard's problems. *Doctoral Dissertation, Indiana University*.

Dedre Gentner and Keith J Holyoak. 1997. Reasoning and learning by analogy. *American Psychologist*, 52(1):32–34.

Dedre Gentner and Christian Hoyos. 2017. Analogy and abstraction. *Topics in Cognitive Science*, 9(3):672–693.

James Gips. 1975. Shape grammars and their uses: Artificial perception, shape generation and computer aesthetics. *PhD thesis, Stanford University*.

Google DeepMind. 2024. Gemini 2.0 flash: Technical report. https://ai.google.dev/gemini-api/docs. Accessed October 2025.

Tetiana Grinberg, Dmitry Grinberg, Grigory Shcherbanskiy, and Lev Cherbanski. 2023. Bongard architecture: Towards scalable and transparent machine reasoning. *AAAI Spring Symposium Series EDGeS Submission*.

Daya Guo and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Douglas R Hofstadter. 1979. *Gödel, Escher, Bach: An Eternal Golden Braid*. Basic Books.

Cheng-Yu Hsieh, Jieyu Zhang, Zixian Ma, Aniruddha Kembhavi, and Ranjay Krishna. 2023. Sugarcrepe: Fixing hackable benchmarks for vision-language compositionality. In *Advances in Neural Information Processing Systems*, volume 36, pages 38957–38977.

Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A Smith. 2023. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20406–20417.

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. 2021. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR.

Huaizu Jiang, Xiaojian Ma, Weili Nie, Zhiding Yu, Yuke Zhu, and Anima Anandkumar. 2022. Bongard-hoi: Benchmarking few-shot visual reasoning for human-object interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19735–19745.

Cheongwoong Kang and Jaesik Choi. 2023. Impact of co-occurrence on factual knowledge of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7770–7791.

Haoqiang Kang, Enna Sachdeva, Piyush Gupta, Sangjae Bae, and Kwonjoon Lee. 2025. Gflowvlm: Enhancing multi-step reasoning in vision-language models with generative flow networks. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3815–3825.

Terry Knight and George Stiny. 2015. Making grammars: From computing with shapes to computing with things. *Design Studies*, 41:8–28.

Peter Kulits, Haiwen Feng, Weiyang Liu, Victoria Abrevaya, and Michael J Black. 2024. Re-thinking inverse graphics with large language models. *arXiv preprint arXiv:2404.15228*.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. 2015. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, volume 28.

Quang-Hung Le, Long Hoang Dang, Ngan Hoang Le, Truyen Tran, and Thao Minh Le. 2025. Progressive multi-granular alignments for grounded reasoning in large vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 4473–4481.

Haoxin Li, Bohao Liu, Hanyang Zhang, and Heng Huang. 2025. Enhancing vision-language compositional understanding via synthetic perturbations. *arXiv preprint arXiv:2503.10825*. CVPR 2025.

Zixian Ma, Jerry Hong, Mustafa Omer Gul, Mona Gandhi, Irena Gao, and Ranjay Krishna. 2023. Crepe: Can vision-language foundation models reason compositionally? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10910–10921.

Microsoft Research. 2024. Introducing phi-4: Microsoft's newest small language model specializing in complex reasoning. https://techcommunity.microsoft.com/blog/azure-ai-foundry-blog/introducing-phi-4.

Weili Nie, Zhiding Yu, Lei Mao, Ankit B Patel, Yuke Zhu, and Anima Anandkumar. 2020. Bongard-logo: A new benchmark for human-level concept learning and reasoning. In *Advances in Neural Information Processing Systems*, volume 33, pages 16468–16480.

Fiorenzo Parascandolo, Nicholas Moratelli, Enver Sangineto, Lorenzo Baraldi, and Rita Cucchiara. 2025. Causal graphical models for vision-language compositional understanding. In *International Conference on Learning Representations (ICLR)*.

Wujian Peng, Sicheng Xie, Zuyao You, Shiyi Lan, and Zuxuan Wu. 2024. Synthesize, diagnose, and optimize: Towards fine-grained vision-language understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26161–26171.

Yuxuan Qiao, Haodong Duan, Xinyu Fang, Junming Yang, Lin Chen, Songyang Zhang, Jiaqi Wang, Dahua Lin, and Kai Chen. 2024. Prism: A framework for decoupling and assessing the capabilities of vlms. In *Advances in Neural Information Processing Systems*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and 1 others. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Rahul Ramachandran, Ali Garjani, Roman Bachmann, Andrei Atanov, Oğuzhan Fatih Kar, and Amir Zamir. 2025. How well does gpt-4o understand vision? evaluating multimodal foundation models on standard computer vision tasks. *arXiv preprint arXiv:2507.01955*.

Ruizhuo Song and Beiming Yuan. 2024. Solving the clustering reasoning problems by modeling a deep-learning-based probabilistic model. *arXiv preprint arXiv:2403.03173*.

George Stiny and James Gips. 1972. Shape grammars and the generative specification of painting and sculpture. In *Information Processing 71: Proceedings of the IFIP Congress 1971*, volume 2, pages 1460–1465. North-Holland.

Mohit Vaishnav and Tanel Tammet. 2025. A cognitive paradigm approach to probe the perception-reasoning interface in vlms. *arXiv preprint arXiv:2501.13620*.

Rujie Wu, Xiaojian Ma, Zhenliang Zhang, Wei Wang, Qing Li, Song-Chun Zhu, and Yizhou Wang. 2024. Bongard-openworld: Few-shot reasoning for free-form visual concepts in the real world. In *International Conference on Learning Representations (ICLR)*.

Antje Wüst, Sven Magg, and Stefan Wermter. 2024. Bongard in wonderland: Visual puzzles that still make ai models hallucinate. *arXiv preprint arXiv:2410.19546*.

Yichang Xu, Gaowen Liu, Ramana Rao Kompella, Sihao Hu, Tiansheng Huang, Fatih Ilhan, Selim Furkan Tekin, Zachary Yahn, and Ling Liu. 2025. Language-vision planner and executor for text-to-visual reasoning. *arXiv preprint arXiv:2506.07778*.

An Yang and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Ilker Yildirim, Mario Belledonne, Winrich Freiwald, and Josh Tenenbaum. 2020. Efficient inverse graphics in biological face processing. *Science Advances*, 6(10):eaax5979.

Mert Yuksekgonul, Federico Bianchi, Pratyusha Kalluri, Dan Jurafsky, and James Zou. 2022. When and why vision-language models behave like bags-of-words, and what to do about it? *arXiv preprint arXiv:2210.01936*.

Appendix

A Example Bongard-LOGO Instance

To illustrate the different symbolic views used in our experiments, we present a single Bongard-LOGO problem from the Basic (BD) category. The underlying concept combines an unbalanced trapezoidright triangle configuration with an uneven band of four arcs.

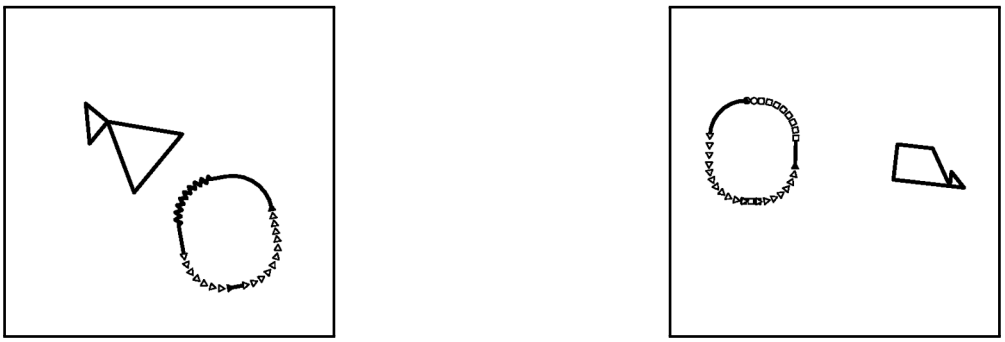
A.1 Concept and Natural-Language Description

This problem is labeled with the high-level Basic concept:

- **Concept UI:** "unbalanced trapezoid right_triangle AND uneven band four arcs"

A.2 Visual Layout

Figure A.1 shows the one positive and one negative example image.



(a) Negative example

(b) Positive example

Figure A.1: Example Basic (BD) Bongard-LOGO problem. Both the images conforming to the "unbalanced trapezoid right_triangle AND uneven band four arcs" concept.

A.3 Procedural Action Programs

Table A.1 shows the LOGO-style action programs for one negative and one positive example. Each token encodes both a primitive type and quantized geometric parameters (e.g., length, radius, angle).

A.4 Natural-Language Action Descriptions

For the C–G (Action Description) condition, each program is rendered as a stepwise English procedure. Below we show the corresponding descriptions for the same negative and positive examples.

Negative example.

To draw figure 1, follow these steps:

Step 1: draw a normal line of 0.300 units. Then, continue straight without turning.

Step 2: draw a normal line of 0.424 units. After that, turn left by 135.0 degrees.

Step 3: draw a normal line of 0.300 units. After that, turn left by 135.0 degrees.

Table A.1: Action programs for one negative and one positive example in the BD instance.

Negative Action Program	Positive Action Program
line_normal_0.300-0.500,	line_normal_1.000-0.500,
line_normal_0.424-0.875,	line_normal_0.283-0.875,
line_normal_0.300-0.875,	line_normal_0.200-0.875,
line_normal_0.800-0.167,	line_normal_0.583-0.086,
line_normal_0.800-0.833,	line_normal_0.500-0.664,
line_normal_0.800-0.833,	line_normal_0.500-0.750,
line_normal_0.200-0.500,	line_square_0.200-0.500,
arc_zigzag_0.500_0.625-0.500,	arc_triangle_0.500_0.625-0.500,
line_normal_0.400-0.500,	line_normal_0.400-0.500,
arc_triangle_0.500_0.625-0.500,	arc_square_0.500_0.625-0.500,
line_normal_0.200-0.500,	line_circle_0.200-0.500,
arc_triangle_0.500_0.625-0.500,	arc_normal_0.500_0.625-0.500,
line_triangle_0.400-0.500,	line_triangle_0.400-0.500,
arc_normal_0.500_0.625-0.500	arc_triangle_0.500_0.625-0.500

Step 4: draw a normal line of 0.800 units. After that, turn right by 119.9 degrees.

Step 5: draw a normal line of 0.800 units. After that, turn left by 119.9 degrees.

Step 6: draw a normal line of 0.800 units. After that, turn left by 119.9 degrees.

Step 7: draw a normal line of 0.200 units. Then, continue straight without turning.

Step 8: draw a zigzag arc with a radius of 0.500 and sweeping 90.0 degrees. Then, continue straight without turning.

Step 9: draw a normal line of 0.400 units. Then, continue straight without turning.

Step 10: draw a triangle arc with a radius of 0.500 and sweeping 90.0 degrees. Then, continue straight without turning.

Step 11: draw a normal line of 0.200 units. Then, continue straight without turning.

Step 12: draw a triangle arc with a radius of 0.500 and sweeping 90.0 degrees. Then, continue straight without turning.

Step 13: draw a triangle line of 0.400 units. Then, continue straight without turning.

Step 14: draw a normal arc with a radius of 0.500 and sweeping 90.0 degrees. Then, continue straight without turning.

The figure is now complete.

Positive example.

To draw figure 1, follow these steps:

Step 1: draw a normal line of 1.000 units. Then, continue straight without turning.

Step 2: draw a normal line of 0.283 units. After that, turn left by 135.0 degrees.

Step 3: draw a normal line of 0.200 units. After that, turn left by 135.0 degrees.

Step 4: draw a normal line of 0.583 units. After that, turn right by 149.0 degrees.

Step 5: draw a normal line of 0.500 units. After that, turn left by 59.0 degrees.

Step 6: draw a normal line of 0.500 units. After that, turn left by 90.0 degrees.

Step 7: draw a square line of 0.200 units. Then, continue straight without turning.

Step 8: draw a triangle arc with a radius of 0.500 and sweeping 90.0 degrees. Then, continue straight without turning.

Step 9: draw a normal line of 0.400 units. Then, continue straight without turning.

Step 10: draw a square arc with a radius of 0.500 and sweeping 90.0 degrees. Then, continue straight without turning.

Step 11: draw a circle line of 0.200 units. Then, continue straight without turning.

Step 12: draw a normal arc with a radius of 0.500 and sweeping 90.0 degrees. Then, continue straight without turning.

Step 13: draw a triangle line of 0.400 units. Then, continue straight without turning.

Step 14: draw a triangle arc with a radius of 0.500 and sweeping 90.0 degrees. Then, continue straight without turning.

The figure is now complete.

A.5 Concept-Conditioned Prompt Fragment

When evaluating the Concept-Conditioned C–G setting, the same instance is annotated with a high-level rule:

> **Concept (system prompt fragment):** "Here is the overall concept behind the positive samples: *unbalanced trapezoid right_triangle AND uneven band four arcs.*"

This example illustrates the three kinds of input used in our experiments for a single Bongard-LOGO problem: raw images (Figure A.1), procedural action programs (Table A.1), and natural-language action descriptions, optionally augmented with an explicit concept label.

B Model Details and Version Information

B.1 Proprietary Models

Table A.2: Proprietary Models Evaluated

Model	Developer	Release	Context Window	Multi-Image
Gemini-2.5-flash-preview	Google	May 2025	1M tokens	Yes
Gemini-3-flash-preview	Google	Dec. 2025	1M tokens	Yes

Access: Google AI Studio APIs. Requires authentication and API key.

B.2 Open-Source Models (via Ollama)

Infrastructure: All open-source models executed locally using Ollama framework on HPC cluster with heterogeneous NVIDIA GPUs (A100 40GB/80GB VRAM).

C Reproducibility and Implementation Details

C.1 Hardware Requirements

Minimum Configuration:

- GPU: NVIDIA RTX 3090 (24GB VRAM) or equivalent
- RAM: 64GB system memory
- Storage: 500GB for models and datasets

C.2 Software Stack

- Ollama v0.9.6 or later (model serving framework)
- Python 3.10+
- PyYAML for configuration parsing
- Requests library for HTTP communication
- Google API Python client for Gemini API access

C.3 Dataset Access

Bongard-LOGO Dataset: Available from original authors at: https://github.com/NVlabs/Bongard-LOGO

Dataset includes:

- 12,000 problem instances (3,600 Free-form, 4,000 Basic, 4,400 Abstract)
- Ground-truth labels and rule annotations
- Action program specifications for all images
- Human-designed shape library with 627 shapes and action programs

C.4 Code Availability

Complete experimental code, including:

- Data loading and preprocessing utilities
- Prompt generation functions (all variants in Appendix D)
- Model interaction wrappers (Ollama + Gemini API)
- Evaluation and result aggregation scripts
- Statistical analysis and visualization code

Code will be released at the public GitHub repository https://github.com/vaishnavmohit/computation-grammatical-symbolic-learning.

API usage for grounded calls. Grounded C–G calls differ from symbolic-only C–G only in that the query image is passed alongside the textual prompt. For Gemini models, we pass the prompt and a PIL image object; for Ollama-served VLMs, we add the image as a base64-encoded field in the system message. This setup ensures that *only* the query receives both symbolic and visual input, while positives and negatives remain purely symbolic, making Grounded C–G a minimal-anchoring ablation on top of the Componential–Grammatical interface.

C.5 Computational Costs

This work required substantial computational resources:

- 2000+ GPU-hours on HPC infrastructure

D Complete Prompts for All Experimental Conditions

This appendix provides the complete, unedited prompts used for each experimental condition. These prompts are essential for reproducibility and can be adapted for similar reasoning tasks.

D.1 Baseline: End-to-End VLM Visual Reasoning

Listing A.1: VLM Visual Reasoning Prompt

```
SYSTEM_PROMPT = """
You are an expert visual analyst, working on Bongard Logo problems containing synthetic images with
    abstract shapes. You are presented with {n + m + 1} images:
   - The first {n} images are labeled 'cat_2' (positive examples).
   - The next {m} images are labeled 'cat_1' (negative examples).
   - The last image is the 'Test Image'.

   Your task is to analyze the visual features of all images and then classify the 'Test Image'
       based on the features it shares with 'cat_1' or 'cat_2'.

   **Your Task:**

   1. **Analyze Positive and Negative Images:**
      * Carefully compare the positive (cat_2) and negative (cat_1) images.
      * **Describe the visual characteristics of *each* category (cat_1 and cat_2) in detail.**
          Focus on shapes, lines, orientations, spatial relationships, and quantities. Break down
          complex shapes into simpler components. Describe shapes as if explaining them to someone
          who can't see them.
      * The goal is to implicitly understand the pattern distinguishing cat_2 from cat_1, even if
          you don't explicitly state it as a single rule.

   2. **Analyze the Test Image:**
      * Describe the test image's visual features in detail, using the same level of detail as in
          step 1.

   3. **Classify the Test Image:**
      * Based on your analysis of the positive, negative, and test images, determine whether the
          test image belongs to cat_2 (positive) or cat_1 (negative).
      * **Provide a clear justification for your classification.** Explain *why* the test image's
          features align more closely with the positive examples or the negative examples,
          referencing your detailed descriptions from steps 1 and 2.

   **Output Format:**

   * **Analysis of Positive and Negative Examples:** (Detailed descriptions of cat_1 and cat_2
       characteristics)
   * **Test Image Analysis:** (Detailed description of the test image's features)
   * **Test Image Classification:** (Either "cat_2 (Positive)" or "cat_1 (Negative)", followed by
       your justification)

   **Begin Analysis:**
   **Output Format (JSON):**
   }
       "Test Image": "(Detailed description of the test image's features)",
       "Rule": "(List of features identified as consistently present in cat_2 category)",
       "Analysis": "(Detailed descriptions of cat_1 and cat_2 characteristics)",
       "Conclusion": "(cat_1 or cat_2)"
   }

   Respond in the JSON format provided, and avoid unnecessary explanations or text outside of the
       JSON structure.

   **Begin Analysis of All Images:**
"""
```

D.2 Componential-Grammatical (C-G): Action Programs

Listing A.2: C-G Action Program Prompt

```
SYSTEM_PROMPT = """
**Objective:** Solve a Bongard-style visual reasoning puzzle by identifying the rule that
    distinguishes a "positive" set of programs from a "negative" set.

      **Input Context:**
```

```
        - You will receive human-readable, step-by-step instructions that describe how to construct
            a set of abstract shapes. These are called "action descriptions".
        - You will NOT be shown any images. Your task is to reason directly from these textual
            descriptions.

        **Your Core Task:**
        1. Analyze the provided "action descriptions" for the positive, negative, and test sets.
        2. Infer the final geometric properties of the shapes from these descriptions.
        3. Identify the abstract rule that defines the positive set.
        4. **Crucially, your reasoning must focus on the final geometric properties of the shapes,
            not the specific steps used to construct them.**

        Here is the overall concept behind the positive samples: {concept}

        **Your Task and Required Output:**
        Respond **only** with a single JSON object. Do not include any text or explanations outside
            of the JSON structure.

        The JSON object must have the following keys:
        }
            "Analysis": "A brief analysis of the distinguishing characteristics you observed in the
                'pos' samples that make them distinct from 'neg' samples.",
            "Rule": "A clear and concise rule that defines the positive samples.",
            "Test Image": "A summary of the test sample's properties, as inferred from its
                description, and how it fits or doesn't fit the derived rule.",
            "Conclusion": "Your final categorization of the test image. The value must be either
                'pos' or 'neg'."
        }
"""
```

D.3 Concept-Conditioned C–G

We add a single declarative sentence specifying the target concept, keeping the rest of the prompt identical across experiments to isolate the effect of explicit concept conditioning. Concept-Conditioned C–G uses the same prompt as in Listing A.2, with one additional line inserted before the "Your Task and Required Output" block:

Here is the overall concept behind the positive samples: {*concept*}

D.4 Natural Language Procedures (Ablation)

Listing A.3: Natural Language Action Description Prompt

```
SYSTEM_PROMPT = """
**Objective:** Solve a Bongard-style visual reasoning puzzle by identifying the rule that
    distinguishes a "positive" set of programs from a "negative" set.

        **Input Context:**
        - You will receive human-readable, step-by-step instructions that describe how to construct
            a set of abstract shapes. These are called "action descriptions".
        - You will NOT be shown any images. Your task is to reason directly from these textual
            descriptions.

        **Your Core Task:**
        1. Analyze the provided "action descriptions" for the positive, negative, and test sets.
        2. Infer the final geometric properties of the shapes from these descriptions.
        3. Identify the abstract rule that defines the positive set.
        4. **Crucially, your reasoning must focus on the final geometric properties of the shapes,
            not the specific steps used to construct them.**

        **Your Task and Required Output:**
        Respond **only** with a single JSON object. Do not include any text or explanations outside
            of the JSON structure.

        The JSON object must have the following keys:
        {
            "Analysis": "A brief analysis of the distinguishing characteristics you observed in the
                'pos' samples that make them distinct from 'neg' samples.",
```

```
            "Rule": "A clear and concise rule that defines the positive samples.",
            "Test Image": "A summary of the test sample's properties, as inferred from its
                description, and how it fits or doesn't fit the derived rule.",
            "Conclusion": "Your final categorization of the test image. The value must be either
                'pos' or 'neg'."
        }
"""
```

D.5 Minimal-Context Baseline (Ablation)

Listing A.4: Minimal-Context Prompt

```
SYSTEM_PROMPT = """
**Your Task and Required Output:**
       Analyze the provided samples and respond *only* with a single JSON object. Do not include
           any additional text, explanations, or markdown formatting outside of the JSON structure.

       The JSON object must have the following keys:
       }
           "Analysis": "A brief analysis of the distinguishing characteristics you observed in the
               'pos' samples that make them distinct from 'neg' samples.",
           "Rule": "A clear and concise rule that defines the positive samples.",
           "Test Image": "A summary of the test image's properties as they relate to the derived
               rule.",
           "Conclusion": "Your final categorization of the test image. The value must be either
               'pos' or 'neg'."
       }
"""
```

D.6 Grounded Componential–Grammatical (Ablation)

In the Grounded C–G setting, models receive the same symbolic inputs as in the main C–G conditions, but the *query* also includes its rendered image. The goal is to test whether a single visual anchor, paired with otherwise symbolic input, provides additional benefits once an explicit visual language is available.

Concretely:

- For Gemini models, we call the official API with a text prompt and a PNG image object as a second content element.
- For open VLMs served via Ollama (e.g., Gemma3, LLaVA-LLaMA3, Qwen2.5-VL), we send a system message containing the text prompt and attach a single image encoded as base64.

In both cases, only the query image is passed; positive and negative examples are provided in symbolic form (action programs), and the model must output a JSON object with an analysis, an induced rule, a description of the test image, and a binary conclusion (pos/neg).

Grounded base (visual-only baseline). For the purely visual "base" context, the system prompt is:

Listing A.5: Minimal-Context Prompt

```
SYSTEM_PROMPT = """
    **Your Task and Required Output:**
    Analyze the provided samples and respond *only* with a single JSON object. Do not include any
        additional text, explanations, or markdown formatting outside of the JSON structure.

    The JSON object must have the following keys:
    }
        "Analysis": "A brief analysis of the distinguishing characteristics you observed in the
            'pos' samples that make them distinct from 'neg' samples.",
        "Rule": "A clear and concise rule that defines the positive samples.",
        "Test Image": "A summary of the test image's attributes, referencing the provided png
            image.",
        "Conclusion": "Your final categorization of the test image. The value must be either 'pos'
            or 'neg'."
    }
    """
```

Grounded C–G with Action Descriptions. For the grounded Action Description condition, the model sees symbolic descriptions for all examples and a *paired* description+image for the query:

Listing A.6: Minimal-Context Prompt

```
SYSTEM_PROMPT = """
    **Persona:** You are an expert visual reasoning system specializing in analyzing abstract
        patterns.

    **Objective:** Your primary goal is to solve a Bongard-style visual reasoning puzzle. You will
        be given abstract descriptions (action programs) used to generate synthetic images and a
        reference image. You must identify the underlying rule that distinguishes a "positive" set
        from a "negative" set.

    **Crucially, for the test sample, you will be given both its action program and the final
        rendered image.** Use this pair to understand the connection between the symbolic action
        programs and their visual generation. Apply this understanding to decipher the rule
        governing the positive and negative sets, for which you only have action programs. These
        action programs are the steps to draw the abstract shape on the canvas.

    **Your Task and Required Output:**
    Analyze the provided samples and respond *only* with a single JSON object. Do not include any
        additional text, explanations, or markdown formatting outside of the JSON structure.

    The JSON object must have the following keys:
    {
        "Analysis": "A brief analysis of the distinguishing characteristics you observed in the
            'pos' samples that make them distinct from 'neg' samples.",
        "Rule": "A clear and concise distinguishing rule that defines the positive samples.",
        "Test Image": "A summary of the test image's attributes, referencing the provided png
            image.",
        "Conclusion": "Your final categorization of the test image. The value must be either 'pos'
            or 'neg'."
    }
    """
```

Grounded C–G with Action Programs. For the grounded Action Program condition, the model receives LOGO-style programs for all examples and a program+image pair for the query, with an extended explanation of the program format:

Listing A.7: Minimal-Context Prompt

```
SYSTEM_PROMPT = """
    **Persona:** You are an expert visual reasoning system specializing in analyzing abstract
        patterns.

    **Objective:** Your primary goal is to solve a Bongard-style visual reasoning puzzle. You will
        be given abstract descriptions (action programs) used to generate synthetic images. You must
        identify the underlying rule that distinguishes a "positive" set from a "negative" set.

    **Input Format and Generative Context:**
    You will receive 13 sets of abstract figure descriptions enclosed in [...]. These descriptions
        are the **source code** (called "action programs") that programmatically generate the
        images. Understanding the generation process is key.

    * **The Generation Pipeline:**
        1. **Stage 1: Base Shape Definition:** Shapes are first defined with basic geometry using
            degrees for angles.
            * `set of base actions`: A list of primitives like `line_LENGTH` or `arc_RADIUS_ANGLE`.
            * `turn angles`: A sequence of turns like `L90.0--R45.0...`, specified in degrees.
        2. **Stage 2: Object Creation (`BasicAction`):** The system reads the base geometry and
            creates programmable `BasicAction` objects. At this stage, a visual stroke style
            (`line_type` like "zigzag" or "triangle") is added.
        3. **Stage 3: Final Serialization (The Input You See):** The `BasicAction` objects are
            serialized into the final action program strings. All values are **normalized** into a
            [0, 1] range.
            * **Line String:** `line_TYPE_LENGTH-TURNANGLE`
            * **Arc String:** `arc_TYPE_ARCANGLE_ARCRADIUS-TURNANGLE`
```

```
    * **Program Structure Hierarchy:**
        The final action program is a nested list: `BongardProblem` -> `BongardImage` ->
            `OneStrokeShape` -> `BasicAction`.

    * **Data Sets You Will Analyze:**
        1. **Positive Samples (`pos`):** Action programs that all conform to a common abstract rule.
        2. **Negative Samples (`neg`):** Action programs that do not conform to this rule.
        3. **Test Sample:** An action program to be categorized, accompanied by its rendered image
            to provide visual context.

    * **Crucial Interpretive Note:** While the action programs detail *how* a figure is constructed,
         your task is to reason about the final, abstract geometric properties of the resulting
         figure. The construction method is just one of several possible ways to draw the same shape.

    **Your Task and Required Output:**
    Analyze the provided samples and respond *only* with a single JSON object. Do not include any
         additional text, explanations, or markdown formatting outside of the JSON structure.

    The JSON object must have the following keys:
    }
        "Analysis": "A brief analysis of the distinguishing characteristics you observed in the
            'pos' samples that make them distinct from 'neg' samples.",
        "Rule": "A clear and concise distinguishing rule that defines the positive samples.",
        "Test Image": "A summary of the test image's attributes, referencing the provided png
            image.",
        "Conclusion": "Your final categorization of the test image. The value must be either 'pos'
            or 'neg'."
    }
    """
```

D.7 User Prompt

We used the same user prompt for all our experiments.

Listing A.8: Natural Language Procedure Prompt

```
USER_PROMPT = """
POSITIVE SET (6 descriptions):
[Description 1]
[Description 2]
[Description 3]
[Description 4]
[Description 5]
[Description 6]

NEGATIVE SET (6 descriptions):
[Description 7]
[Description 8]
[Description 9]
[Description 10]
[Description 11]
[Description 12]

QUERY (1 description):
[Description 13]

Classify the query as 'positive' or 'negative'. Respond with JSON
only.
"""
```

E Extended Results Tables

E.1 Complete Results by Model and Condition

Table A.3 reports full C–G performance on the Bongard-LOGO benchmark across all 12 models and experimental conditions. For each model, we include Action Description, Action Program, and minimal-context Base (Ablation) conditions, broken down by Basic Shapes (BD), Free-form (FF), Human-designed Combined (HD Comb), and Human-designed Novel (HD Novel).

Table A.3: C-G paradigm performance on Bongard-LOGO. Results show accuracy (%) for Action Desc. and Action Prog. across problem categories with concept (C) and without concept (NC) guidance. BD = Basic Shapes, FF = Free-form, HD = Human-designed. Action Desc. = Action description and Action Prog. = Action program.

Model	Experiment	BD		FF		HD Comb		HD Novel	
		C	NC	C	NC	C	NC	C	NC
DeepSeek-R1									
8b	Action Desc.	67.2	65.0	67.4	63.0	54.8	56.2	54.2	54.6
	Action Prog.	56.0	56.0	56.0	56.2	53.4	55.0	55.2	53.8
	Baseline		53.0		51.4		48.8		53.0
14b	Action Desc.	73.2	71.6	76.2	74.0	56.4	57.0	58.6	57.6
	Action Prog.	69.6	68.6	75.6	77.8	56.8	57.8	59.0	57.6
	Baseline		71.8		74.6		59.2		58.6
32b	Action Desc.	66.6	69.2	84.2	82.8	59.2	59.6	56.8	58.6
	Action Prog.	67.4	67.8	81.2	78.6	59.4	57.8	59.2	64.2
	Baseline		66.2		74.6		55.8		58.2
Gemma3									
12b	Action Desc.	66.8	66.4	73.4	69.2	58.4	56.6	64.2	65.8
	Action Prog.	62.8	63.4	71.4	70.0	63.0	65.4	62.4	63.6
	Baseline		66.6		71.2		60.8		62.2
27b	Action Desc.	73.0	68.6	84.6	81.0	60.0	57.4	62.2	62.6
	Action Prog.	69.4	68.6	87.4	83.2	64.4	66.0	64.4	66.6
	Baseline		67.6		82.8		63.0		66.8
Magistral:24b	Action Desc.	78.8	72.8	83.4	84.8	58.6	60.4	66.8	63.2
	Action Prog.	72.8	72.8	81.6	80.4	57.8	60.0	67.2	58.2
	Baseline		72.6		82.2		59.4		64.2
Phi4:14b	Action Desc.	75.4	73.6	91.8	90.2	55.6	55.0	59.8	58.0
	Action Prog.	71.6	71.6	92.0	89.8	60.2	62.2	64.8	62.4
	Baseline		70.2		97.0		61.2		61.4
Phi4-Reasoning:14b	Action Desc.	79.6	79.0	90.0	87.8	54.4	51.8	59.8	58.8
	Action Prog.	75.6	75.2	94.2	96.2	60.0	59.4	62.8	63.2
	Baseline		70.2		97.0		55.6		59.0
Qwen2.5:32b	Action Desc.	76.8	75.8	81.2	80.8	61.2	60.2	63.0	61.4
	Action Prog.	74.4	74.6	78.2	76.2	63.6	62.2	65.0	63.0
	Baseline		71.4		75.8		63.4		60.8
Qwen2.5vl:32b	Action Desc.	70.0	71.6	73.8	72.8	57.8	59.6	61.6	59.4
	Action Prog.	67.2	69.2	73.0	72.4	62.6	60.2	61.0	61.2
	Baseline		60.2		65.6		60.8		56.2
Qwen 3									
30b	Action Desc.	75.0	75.2	78.0	78.2	58.0	56.8	60.6	63.2
	Action Prog.	75.6	84.8	79.2	77.4	59.4	59.4	60.8	61.4
	Baseline		65.4		74.0		58.6		61.4
32b	Action Desc.	78.4	86.7	81.6	86.8	61.6	58.8	60.6	64.6
	Action Prog.	77.6	76.4	82.2	78.8	61.8	61.0	62.4	61.4
	Baseline		77.4		81.8		59.4		63.4

E.2 Effect of Concept Conditioning by Model

Table A.3 details the impact of concept conditioning for each model, representation, and category. These numbers underlie the aggregated trends: concept conditioning has small, model- and category-dependent effects, with Action Programs often showing slightly more stable behavior than Action Descriptions in the Human-designed regime.

Concept conditioning yields only small and model-dependent effects overall. Across all regimes, the mean improvement from adding the concept label is +0.2 percentage points for Action Programs and +0.6 pp for Action Descriptions, neither consistently favouring one format. The largest category-level effect for AP is a modest +1.3 pp on Free-form problems, while Human-designed (Abstract) problems show minimal average change (+0.2 pp for AP). These small averages conceal high variability across models (see Table A.4).

E.3 Effect of Symbolic Representation Format

Natural-language descriptions provide strong inductive bias for the easier FF and BD regimes but fail to generalize to Human-designed abstractions. In contrast, action programs yield smaller but consistent gains, motivating their use as the primary concept representation. Overall, format choice has a modest impact relative to the much larger gap between symbolic and visual-only input.

E.4 Effect of Model Capacity

We observe that the effect of symbolic input and concept conditioning varies across models (Table A.4). While some smaller and mid-sized models show clear gains over their visual baselines, others see little change or even slight decreases, and the larger models in our analysis tend to exhibit modest but more consistently positive improvements. This pattern is compatible with the view that structured input can act as an inductive bias that reduces representational burden, but its benefits are not uniform across capacity tiers.

Table A.4: Mean accuracy improvement (%) over baseline across all regimes, stratified by model. The effect of symbolic input and concept conditioning is heterogeneous: larger models show slightly more consistent positive averages, while smaller and instruction-sensitive models exhibit mixed gains.

Model	AD	AP	AD + C	AP + C
DeepSeek-R1:14b	-0.7	-0.3	+0.4	-0.5
DeepSeek-R1:32b	+3.6	+3.2	+2.8	+2.9
DeepSeek-R1:8b	+8.0	+3.6	+9.2	+3.5
Gemma3:12b	-0.4	+0.7	+0.8	0.0
Gemma3:27b	-2.9	+0.8	-0.4	+1.1
Magistral	0.0	-2.5	+1.6	-0.5
Phi4	-4.1	-1.8	-2.7	-1.2
Phi4-Reasoning	-1.7	+2.5	0.0	+2.2
Qwen2.5:32b	+2.2	+1.6	+3.2	+2.9
Qwen2.5vl:32b	+4.5	+4.4	+4.5	+4.6
Qwen3:30b	0.0	+2.4	-0.5	+0.4
Qwen3:32b	+4.1	-0.7	+0.4	+0.9

To analyze how symbolic input interacts with model capacity, we partition models into *smaller* and *larger* capacity tiers. This grouping is based on a combination of parameter scale, reasoning-oriented pretraining, and baseline performance on Bongard-LOGO. The grouping is fixed across all experiments and comparisons.

To further analyze the interaction between model capacity and symbolic input, we group models into smaller and larger tiers and average accuracy improvements within each group. This grouping is based on a combination of parameter scale as shown in Table A.5. Accuracy is presented in Table A.6. Larger models in this dataset show slightly higher mean gains from symbolic input and concept conditioning than the smaller group, although variance across individual models remains substantial. These exploratory results suggest that the benefits of symbolic input depend on both capacity and pretraining, rather than uniformly favoring lower-capacity models.

Table A.5: Model grouping used for capacity-based analysis. Models are grouped by approximate parameter scale and reasoning capacity.

Capacity Tier	Models
Smaller	Phi-4-Reasoning, Phi-4, Magistral, Gemma3:12B, DeepSeek-R1:8B, DeepSeek-R1:14B
Larger	Qwen3:32B, Qwen3:30B, Gemma3:27B, DeepSeek-R1:32B, Qwen2.5vl:32B, Qwen2.5:32B

Table A.6: Mean accuracy improvement (%) over baseline, averaged across model size groups. In this dataset, larger models show slightly higher mean gains from symbolic input and concept conditioning than the smaller group.

Model Group	AD	AP	AD + Concept	AP + Concept
Smaller Models	$+0.2 \pm 3.7$	$+0.4 \pm 2.2$	$+1.6 \pm 3.7$	$+0.6 \pm 1.7$
Larger Models	$+1.9 \pm 2.6$	$+2.0 \pm 1.6$	$+1.7 \pm 1.9$	$+2.1 \pm 1.5$

E.5 Effect of Grounded C–G with Query Image Anchors

The Grounded C–G condition augments symbolic input with a single visual anchor: the rendered query image. This condition is only applicable to models that support image inputs. We therefore restrict this analysis to VLM-capable models: Gemma3 12B, Gemma3 27B, LLaVA-Llama3, and Qwen2.5-VL 32B.

In all cases, the model receives (i) the action programs or action descriptions for the six positive and six negative examples, and (ii) the corresponding symbolic representation plus rendered image for the query. The task and evaluation protocol are identical to the main experiments (Section 5).

Table A.7 reports per-model performance across Bongard categories for the two symbolic formats, and Table A.8 summarizes mean accuracy across the four VLMs, aggregating Human-designed Combined and Novel (HD Comb, HD Novel) into a single HD score. Note that, unlike Table 1, which averages over 12 models, the statistics here are computed over this smaller matched VLM subset and should be compared only within that subset.

For ease of comparison with the main results (Table 1), we also report mean accuracy and standard deviation across the four VLMs, averaging HD Comb and HD Novel into a single Human-designed score:

Overall, the grounded setting yields only modest changes relative to the corresponding symbolic-only baselines for this matched VLM subset, and Action Programs retain a small advantage on Free-form and Human-designed categories. This reinforces our main conclusion that symbolic structure, rather than limited visual anchoring, is the primary source of gains on Bongard-LOGO.

E.6 Randomization Experiments

To probe whether high C–G performance reflects genuine rule learning over symbolic structure rather than exploitation of superficial token statistics, we conduct two perturbation experiments in the Action Program setting:

- **Categories Shuffle**: for each Bongard-LOGO problem, we randomly permute the assignment of the 12 support examples to the positive and negative sets while leaving the query program, query image, and gold query label unchanged. This preserves marginal token distributions within a problem but destroys the coherent support-set partition that defines the concept.

- **Test Sequence Shuffle**: for each problem, we randomly shuffle the order of drawing actions in the *query* program at inference time, while leaving the support examples, query image, and gold query label unchanged. This maintains the multiset of action tokens but disrupts the compositional structure of the query.

Both perturbations use the same prompts and decoding settings as the main Action Program condition (Section 4). We report them only on Basic (BD) and Free-form (FF), since the lower unperturbed HD

Table A.7: Grounded C–G performance for VLM-capable models. Each entry is accuracy (%) on Bongard-LOGO categories when symbolic input (Action Description or Action Program) is augmented with the rendered query image. BD = Basic, FF = Free-form, HD Comb/HD Novel = Human-designed shape subsets.

Model	Experiment	BD	FF	HD Comb	HD Novel
Gemma3:12B	Action Desc	55.0	56.2	56.8	56.0
	Action Prog	56.31	58.8	57.2	60.12
Gemma3:27B	Action Desc	56.2	69.2	57.6	61.0
	Action Prog	59.0	70.4	65.0	63.4
LLaVA-Llama3	Action Desc	49.6	50.1	50.0	50.1
	Action Prog	50.2	49.0	49.2	50.2
Qwen2.5VL:32B	Action Desc	60.4	69.2	57.8	57.4
	Action Prog	55.8	71.2	59.6	61.0

Table A.8: Grounded C–G: mean accuracy (%) $\pm$ standard deviation across VLM-capable models (N=4). HD is the average of HD Comb and HD Novel. These numbers are not directly comparable to the 12-model pooled rows in Table 1.

Experiment	BD	FF	HD
Action Description	55.3 ± 4.5	61.2 ± 9.6	55.8 ± 4.0
Action Program	55.3 ± 3.7	62.4 ± 10.6	58.2 ± 6.1
Baseline	57.1 ± 5.5	62.6 ± 11.2	57.5 ± 5.4

baseline would make the controls less diagnostic there. Table A.9 reports per-model accuracies on BD and FF, and Table A.10 summarizes mean accuracy across the nine models included in this analysis.

Overall, Categories Shuffle and Test Sequence Shuffle both reduce accuracy relative to the unperturbed Action Program setting, but they need not drive performance exactly to 50% because the shuffled support sets can still instantiate alternative rules that partially align with the unchanged query. The relevant signal is the relative degradation: Categories Shuffle primarily damages coherent support-set reasoning, whereas Test Sequence Shuffle particularly harms Free-form performance by disrupting the ordered structure of the query. These effects support the interpretation that models rely on the relational and compositional structure of the symbolic programs, especially for Free-form tasks, rather than on simple surface-level token matching.

E.7 Class Asymmetry in Rule Induction

Beyond overall accuracy, we analyze how models treat positive vs. negative examples under the Action Program (AP) interface. Aggregating across models, we find a pronounced asymmetry: on Basic (BD) problems, accuracy on positive examples is 91.2% compared to 53.4% on negatives (a +37.8 point gap), and on Free-form (FF) problems, positives reach 98.5% vs. 61.4% for negatives (+37.1 points). A chi-square test of correctness $\times$ class yields $\chi^2 \approx 1156.3$ for BD and $\chi^2 \approx 1504.2$ for FF ($p \ll 0.001$ in both cases), indicating that correctness is strongly dependent on whether an item belongs to the positive or negative set. Consistent with this, the pooled predicted labels are skewed toward "positive" (e.g., 4,799 predicted positives vs. 2,201 negatives on FF), suggesting that models default to classifying borderline or ambiguous cases as positive rather than negative.

F Limitations

F.1 Synthetic Benchmark Limitations

While Bongard-LOGO provides a controlled environment for hypothesis testing, it is fundamentally limited:

- Simple geometric shapes with programmatic rules (not learned from data)
- Absence of real-world noise, occlusion, variation, and clutter

Table A.9: Performance of C–G (Action Program) under two randomization controls. **Categories Shuffle** randomly permutes the assignment of support examples to positive vs. negative sets within each problem, while leaving the query and its gold label unchanged. **Test Sequence Shuffle** randomly shuffles the order of drawing actions in the query program at inference time, leaving the support examples, query image, and gold label unchanged. Values show accuracy (%) on Basic (BD) and Free-form (FF) problems for each model.

Model	Shuffle Type	BD	FF
DeepSeek-R1:14B	Categories	63.0	63.0
	Test Sequence	61.6	52.2
DeepSeek-R1:32B	Categories	58.8	73.0
	Test Sequence	68.2	68.8
DeepSeek-R1:8B	Categories	54.3	54.0
	Test Sequence	54.6	51.4
Gemma3:12B	Categories	57.6	63.2
	Test Sequence	64.7	54.2
Gemma3:27B	Categories	58.6	76.6
	Test Sequence	65.0	55.4
Magistral	Categories	64.2	70.9
	Test Sequence	66.4	61.4
Phi-4	Categories	61.4	81.2
	Test Sequence	60.0	53.4
Phi-4-Reasoning	Categories	57.8	86.8
	Test Sequence	65.2	52.6
Qwen2.5:32B	Categories	63.8	54.6
	Test Sequence	71.6	67.8

Table A.10: Mean accuracy (%) $\pm$ standard deviation across the nine models in Table A.9 for each randomization control. BD = Basic shapes, FF = Free-form shapes.

Shuffle Type	BD	FF
Categories Shuffle	59.9 ± 3.3	69.3 ± 11.4
Test Sequence Shuffle	64.1 ± 4.9	57.5 ± 6.8

- Binary classification rather than open-ended reasoning
- Formal, unambiguous rules rather than statistical regularities

Results on this benchmark should not be interpreted as evidence of general visual reasoning capability. Evaluation on natural-image Bongard datasets (e.g., Bongard-OpenWorld with VLM–LLM interfaces) and other diverse benchmarks is essential for establishing broader generalization.

F.2 Ground-Truth Privilege and Practical Applicability

Our approach assumes access to perfect symbolic specifications—the ground-truth action programs used to generate each image. This represents a significant privilege that is unavailable in most real-world scenarios:

- Most domains lack formal generative specifications
- Obtaining symbolic descriptions requires either domain-specific engineering, learned perceptual systems, or human annotation

We caution against over-interpreting the success of the C-G approach as evidence that human-level visual reasoning is imminent. Rather, it illustrates what becomes possible when perception is perfectly decoupled from reasoning.

F.3 Model Coverage Gaps

Our evaluation covered recent reasoning-capable models but excluded:

- Vision-language models in their symbolic reasoning mode (architectural constraints limited multi-image symbolic reasoning)
- Older and specialized models (e.g., earlier GPT/Claude variants, mathematical reasoners) were not evaluated systematically.
- Models from certain organizations (OpenAI, Anthropic APIs were not used)
- Smaller models (<7B parameters) were not systematically evaluated

Results on a broader model portfolio would strengthen generality claims and establish baseline expectations for future work.

F.4 Environmental and Computational Impact

This research required:

- €1000 in API costs to run frontier models (financial barrier to reproducibility)
- 2000+ GPU-hours (computational barrier)
- Significant energy consumption (environmental impact)

These barriers limit reproducibility and raise equity concerns.

F.5 Incomplete Analysis

Several aspects of this work merit deeper investigation:

- Human-designed performance ($\approx$60–66%) remains substantially lower than the BD and FF regimes despite perfect symbolic input.
- Robustness to symbolic input corruption (e.g., noise in programs or descriptions) has not been evaluated.

Further work is needed to analyze failure modes in output parsing and to characterize computational efficiency and scaling behavior.